CAB420 Machine Learning Report
Semantic Segmentation of Aerial Imagery Using RGB and Elevation Data

Aron Bakes, n11405384[1], Deegan Marks, n11548444[2], Jordan Geltch-Robb, n11427515[3*]

June 10, 2025

[*1] Aron Bakes, [2] Deegan Marks and [3] Jordan Geltch-Robb are with the Faculty of Engineering, Queensland University of Technology (QUT), Australia. This report is in partial fulfilment of CAB420 unit requirements and was submitted on **June 10, 2025**. aron.bakes@connect.qut.edu.au, deegan.marks@connect.qut.edu.au, jordan.geltchrobb@connect.qut.edu.au

Executive Summary

This project focused on enhancing semantic segmentation of aerial imagery, a critical task for urban planning, environmental monitoring and autonomous navigation. Our core objective was to determine if integrating elevation and slope data could significantly improve segmentation accuracy compared to only RGB. To achieve this, we compare three distinct machine learning models, the convolutional U-Net, the state-of-the-art SegFormer, and a traditional logistic regression pipeline, evaluating each on both RGB-only and multimodal datasets.

Our approach utilised the DroneDeploy benchmark dataset, providing aerial RGB imagery, detailed elevation maps and pixel-level semantic labels. The dataset categorises every pixel into one of six classes: *Building*, *Clutter*, *Vegetation*, *Water*, *Background*, and *Car*. To harness this data, we preprocessed the elevation to create a slope mask by calculating gradients. This resulted in two distinct input configurations for our models: a 3-channel RGB-only input and an enhanced 5-channel input combining RGB with both elevation and slope data. This allowed for a direct comparison of spatial information's impact. Notably, our deep learning models were implemented from scratch in TensorFlow/Keras without relying on pre-trained weights. This deliberate approach was chosen because the shift from a 3-channel RGB input to a 5-channel RGB + Elevation + Slope input fundamentally alters the network's initial layers, making direct use of existing pre-trained backbones (which are typically designed for 3-channel images) incompatible without significant architectural modifications or complex data fusion strategies, which were beyond the scope of this comparative study. To address severe class imbalance, particularly the dominance of 'Background' and 'Vegetation' and under-representation of *Car* and Water, we implemented a targeted chip selection strategy, prioritising samples with higher minority class proportions and excluding images with $\geq 95\%$ *Background* distribution. Furthermore, a modified self-CutMix strategy was applied specifically for the *Water* class, this led to a substantial 50% relative gain in *Water* class IoU.

Our quantitative results, using mean Intersection over Union (mIoU) as the primary metric , demonstrated consistent improvements across all models with the addition of elevation and slope data. SegFormer, with RGB+Elevation+Slope input, achieved the highest mIoU of 0.4364. U-Net also showed improvement, from 0.3985 with RGB alone to 0.4228 with elevation and slope. Even the logistic regression pipeline saw gains, improving from 0.1976 to 0.2648. A deeper look at per-class IoU revealed remarkable gains for the 'Building' class after incorporating elevation data, with U-Net improving by 39.93% IoU and SegFormer by an impressive 117.29% IoU.

In terms of computational efficiency, the classical logistic regression model was the fastest, with training completed in approximately 10 minutes and nearly instant inference. U-Net trained in under 4 hours, while SegFormer required over 9 hours due to its transformer architecture and larger parameter count. Despite slower training, SegFormer exhibited the lowest inference latency. All deep models were trained on an NVIDIA A100 GPU with 40GB VRAM and 83.5GB system RAM via Google Colab Pro+.

Direct comparison to existing benchmarks is challenging as our models were trained entirely from scratch without pre-trained weights, unlike most state-of-the-art benchmarks. Existing UNet variants achieve around 0.20 higher mIoU, with validation IoU scores reaching the mid-60s. Similarly, alternate SegFormer benchmarks also perform approximately 0.20 better, in the 60-70% mIoU range. These benchmarks highlight that state-of-the-art models trained with pre-trained backbones commonly exceed 60% mIoU, reinforcing the performance ceiling our from-scratch models aim to achieve.

In conclusion, the addition of elevation and slope data consistently improved semantic segmentation performance across all models. SegFormer was the most accurate overall (0.4364 mIoU) , U-Net offered a strong baseline providing a good balance of speed and accuracy , and even logistic regression performed well with spatial features and CRF post-processing. This study confirms the value of multimodal inputs for aerial imagery semantic segmentation. For future work, we recommend cleaning label masks, introducing a *Road* class, and investigating adaptive fusion strategies to combine RGB and elevation features more effectively. This will pave the way for leveraging pre-trained backbones in future research for potentially higher performance ceilings.

Contents

1 Project Motivation and Objective

Semantic segmentation is a fundamental task in computer vision that assigns a class label to each pixel in an image. It enables detailed scene understanding and is widely applied in domains that require spatial precision and contextual awareness. While significant progress has been made using RGB imagery, incorporating elevation data introduces an additional dimension of spatial context that can enhance performance, particularly in topographically complex environments.

This project investigates how semantic segmentation of aerial images can be improved by fusing RGB and elevation data, evaluating the effectiveness of this approach across different model types. Elevation maps provide valuable structural cues that help distinguish between visually similar but functionally different features, such as rooftops versus roads.

We focus on real-world applications that include urban planning, civil engineering, environmental monitoring, and autonomous navigation in vehicles and aerial drones. For example, semantic segmentation can assist in land-use classification, flood zone mapping, post-disaster infrastructure assessment, and remote surveying in hard-to-reach areas.

Our objective is to quantify the contribution of elevation information to segmentation accuracy and to assess trade-offs between model complexity, computational cost, and performance when using multimodal input data. To achieve this, we evaluated a diverse set of models ranging from lightweight baselines to state-of-the-art architectures.

2 Related Work

Semantic segmentation has received significant attention in computer vision due to its ability to perform dense, structured prediction. In aerial imagery, it enables pixel-wise classification of land use, infrastructure, and vegetation, which is essential for domains such as disaster response, urban planning, and autonomous navigation.

Convolutional Architectures

The U-Net architecture [1], initially proposed for biomedical imaging, remains of the most popular models for semantic segmentation. U-Net is a symmetric encoder-decoder with skip connections between matching encoder and decoder blocks. This enables precise localisation, the models ability to identify boundaries and position of objects within an image. However, U-Net's standard architecture can struggle to capture long range dependencies and global context.

Several U-Net inspired variants have emerged. U-Net++ [2] introduces nested and dense skip connections to improve multi-scale feature aggregation, enhancing performance on complex structures. PSPNet [3] uses pyramid pooling to aggregate context at multiple spatial scales, making it better for large scale scene prediction, at the cost of high memory usage and . It is effective for large-scale scene understanding, but its high memory usage and longer inference time can limit real-time applications. DeepLabv3+ [4] incorporates atrous spatial pyramid pooling (ASPP) to capture multi-scale context, but is sensitive to changes in resolution. These CNN-based models form a strong baseline, but their struggle with global context motivated the exploration of transformer architectures.

Transformer-Based Architectures

Transformers have recently gained traction for image segmentation, Vision Transformers (ViT) [5] model global attention across patches but require large datasets and struggle with small objects due to coarse patch size. SegFormer [6] addresses this issue by avoiding positional embeddings, making it more adaptable to varying input resolution. SegFormer also uses efficient Mix-FFN (Feed-Forward Network) layers which use depthwise convolutions for better efficiency, although SegFormer still require substantial compute for training.

Lightweight and Edge-Aware Models

For applictions requiring compact, low-latency architectures such as UAVs or drones, lightweight models have been developed. ESPNet [7] and Fast-SCNN [8] reduce computational demands through lightweight

convolutional backbones, making them suitable when resources are limited. However, their reduced capacity limits performance on complex scenes compared to deeper networks. As a computationally lightweight method, we implemented a classical machine learning pipeline combining logistic regression with Dense Conditional Random Fields (CRF) and Simple Linear Iterative Clustering (SLIC) superpixels [9, 10]. This approach, while far simpler than deep learning models, provides insight into the effect of spatial priors and post-processing.

Multimodal Fusion

The fusion of RGB imagery with auxilliary modalities like elevation has shown promise in improving semantic segmentation performance by providing complementary spatial information. Early fusion techniques, such as simple channel concatenation are straightforward to implement and can provide benefits [11, 12]. FusionNet [13] explores late fusion of LiDAR and RGB data, and LMFNet [14], a lightweight framework that uses a self-attention mechanism, accepts any number of channel inputs, allowing for easier scalability to new data inputs. Adaptive or attention-based fusion methods [15, 16] also look promising, but are typically more computationally intensive.

Model Selection Rationale

Our selection of U-Net, SegFormer and the Logistic Regression + CRF/SLIC aims to investigate a diverse range of machine learning techniques in terms of architecture and computation. U-Net serves as a reliable convolutional baseline with modest resource requirements. SegFormer, representing transformer-based models, allow us to explore the benefits of long-range dependencies for multi-scale understanding, although more computationally demanding. Finally, the classical logistic regression baseline provides a lightweight, interpretable alternative, offering insights into how traditional methods perform with multi-modal data when computational resources are limited. This diverse range allows us to investigate not just absolute performance, but also the trade-offs in different architectural types, as well as available time and computational investment, providing a comprehensive comparison.

Ethical Considerations

While semantic segmentation offers powerful tools for automation and analysis, its societal impacts must be acknowledged. In safety critical applications like autonomous driving, incorrect predictions could result in critical or fatal errors. In aerial surveillance, model outputs may be misused for persistent monitoring or tracking, raising ethical and privacy concerns [17]. Furthermore, performance on underrepresented regions or classes may degrade due to inherent bias in training data [18]. These risks highlight the importance of transparency, bias mitigation, and ensuring robustness in both design and deployment, aligning with broader AI ethics guidelines [19].

3 Data

The dataset used in this project is the DroneDeploy Segmentation Benchmark.[1], which offers high-resolution aerial scenes captured from drones. Each scene include a corresponding RGB image, elevation map, and semantic segmentation mask of the ground truth.

Each scene has a ground resolution of approximately 10 cm per pixel and includes:

- **RGB Image**: A three-channel `.tif` image.
- **Elevation Map**: A single-channel floating-point `.tif` map representing elevation in metres above sea level.
- **Semantic Label Map**: A pixel-wise annotated `.png` file using unique RGB values.

An example from the dataset is shown in Figure 1.

[1]Dataset and benchmark code: `https://github.com/dronedeploy/dd-ml-segmentation-benchmark`

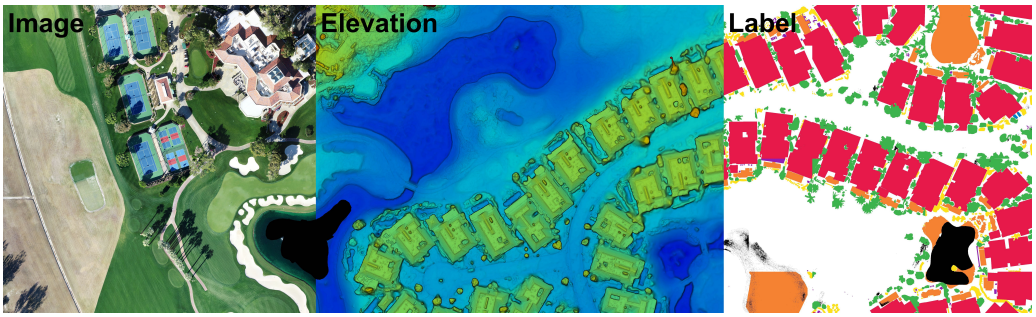


Figure 1: Example from the dataset: RGB image (left), elevation map (middle), semantic label map (right).

Semantic Labels

The semantic labels are encoded using specific RGB values, as detailed in Table 1.

Table 1: Semantic segmentation class colour map and relative frequency in the training set.

Label	Class	Colour	Code	RGB Values	Train Freq. (%)
0	Building	Red		(230, 25, 75)	11.99
1	Clutter/debris	Purple		(145, 30, 180)	4.68
2	Vegetation	Green		(60, 180, 75)	10.68
3	Water	Orange		(245, 130, 48)	6.83
4	Background	White		(255, 255, 255)	64.60
5	Car	Blue		(0, 130, 200)	1.23
6	Ignore	Magenta		(255, 0, 255)	–

Data Cleaning

The dataset exhibits a substantial class imbalance, the *Background* and *Vegetation* classes dominate the pixel distribution, as shown in Table 1. Aerial datasets often suffer from noisy labels and inconsistent class definitions, which undermine learning, especially in underrepresented classes [20].

Several annotation inconsistencies were identified in the dataset, likely due to multiple annotators. Common issues included *Car* mislabelled as *Clutter* or left unlabelled as *Background*. Figure 2 shows a vehicle inconsistently annotated.

These inconsistencies reflect both annotation mistakes and subjectivity in class definitions, particularly for *Clutter*. In one scene, a *Building* rooftop is mislabelled as a Car, and multiple parked *Cars* are labelled as *Buildings*. While most of the dataset was retained, this specific scene was excluded from training and placed in the test set.

It is important to note that other labelling inconsistencies may still be present in the training data, potentially affecting model generalisation. We opted against manual relabeling to maintain reproducibility with the original benchmark.

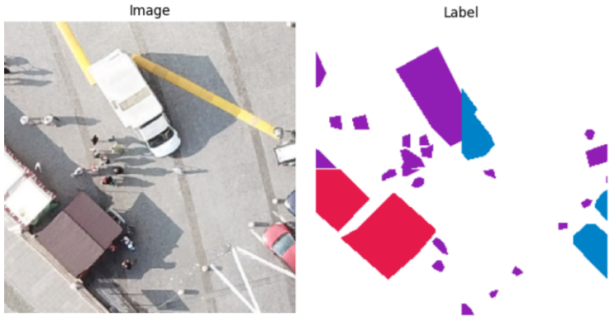


Figure 2: Examples of annotation inconsistencies. A vehicle in the centre is partially mislabelled as clutter ■ and partially as car ■, likely due to differences between annotators.

Preparation and Splitting

To manage high-resolution imagery and irregular scene boundaries, each image was divided into 256×256 pixel tiles. Tiles containing any *Ignore* pixels were excluded. The 55 original scenes were split into training, validation, and test subsets using a 35-10-10 ratio at the scene level, ensuring no spatial overlap and avoiding data leakage. A fixed random seed was used for reproducibility.

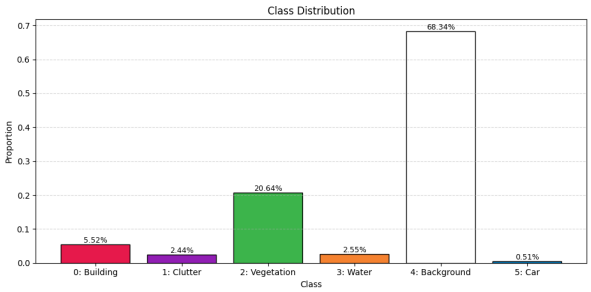
Training tiles were generated using a stride of 128 px to increase dataset diversity, while validation and test tiles used a stride of 256 px (no overlapping) to prevent redundancy during evaluation.

During chipping, class distributions and entropy were recorded for each tile and saved to a metadata `.csv` file which was loaded as a `pandas` dataframe during training. To ensure class diversity, a greedy selection algorithm was used to assign 10 scenes each to validation and test sets based on entropy and class coverage. The remaining 35 scenes formed the training set.

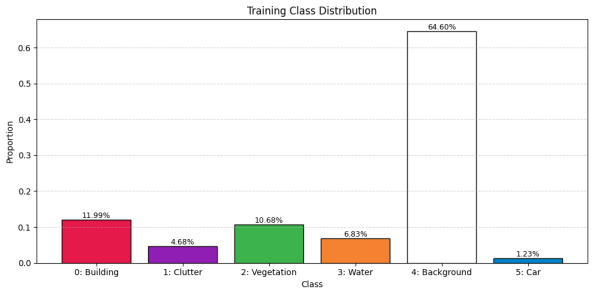
The dataset, as shown in Figure 3a, exhibits a substantial class imbalance. To address this severe class imbalance in the training set (Figure 3b), a subset of chips was specifically selected:

- All tiles containing any *Car* or *Water* pixels were retained.
- Tiles containing only a single class were excluded, as pure *Water* tiles consistently underperformed.
- Tiles with $\geq 95\%$ *Background* pixels were also excluded.

This filtering reduced the training set to 18,400 tiles out of 34,805 (52.87% retained).



(a) Overall pixel-wise class distribution across the entire dataset (before filtering).



(b) Pixel-wise class distribution across the filtered training data (including overlapping chips due to 128 *px* stride). The selection process aimed to improve representation of minority classes.

Figure 3: Comparison of class distributions. (a) shows the original dataset imbalance, and (b) illustrates the resulting distribution after applying a targeted filtering strategy to the training set.

Data Augmentation

Training batches included a combination of geometric and photometric augmentations designed to improve generalisation while preserving label integrity.

- **Geometric Augmentations:** Random horizontal and vertical flips were applied independently (50% probability each). Random rotations by 0°, 90°, 180°, or 270° were applied with equal probability (resulting in a 75% chance of actual rotation). These discrete rotations avoid interpolation that could corrupt pixel-wise label masks.
- **Photometric Augmentations:** Applied with lower probability, these included random brightness (25%), contrast (25%), saturation (25%), and hue shifts (20%) to simulate different lighting and camera conditions.

In addition to these standard augmentations, a targeted CutMix was applied to duplicate *Water* regions (Figure 4) during training. This increased the diversity of *Water* textures and spatial contexts, resulting in a relative IoU gain of approximately 50% for the (4% to 6%). Augmentations were applied consistently across RGB images, label, elevation and slope masks.

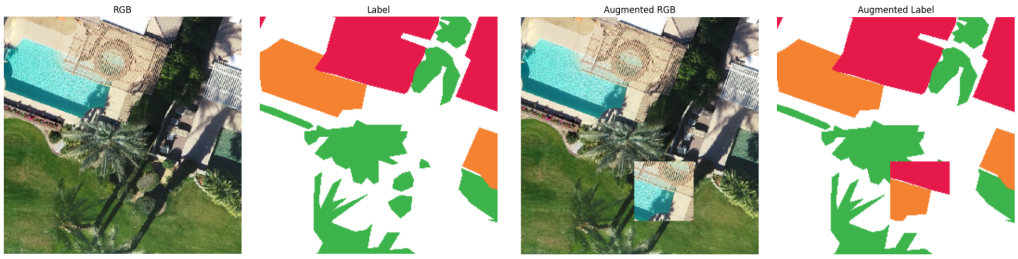


Figure 4: Examples of CutMix augmentation applied to training chips. A *Water* ■ region is duplicated within the RGB image (centre-right) and corresponding label mask (far-right), increasing training diversity.

Elevation Processing

Elevation masks represent height in metres above sea level. To ensure consistency across scenes with varying drone capture altitudes while preserving meaningful local structure, per-scene standardisation (mean subtraction and standard deviation division) was applied. This approach retains local elevation gradients and reduces sensitivity to extreme values.

Inspired by [12] slope masks were calculated to compliment elevation data, using a 3×3 kernel:

$$\theta = \tan^{-1}\left(\sqrt{\left(\frac{dz}{dx}\right)^2 + \left(\frac{dz}{dy}\right)^2}\right) \tag{1}$$

Where $\frac{dz}{dx}$ and $\frac{dz}{dy}$ denote the elevation's rate of change in horizontal and vertical directions. Slope maps emphasise object boundaries and vertical features, aiding in distinguishing adjacent classes. Figure 5 shows an example chip with all four input modalities.

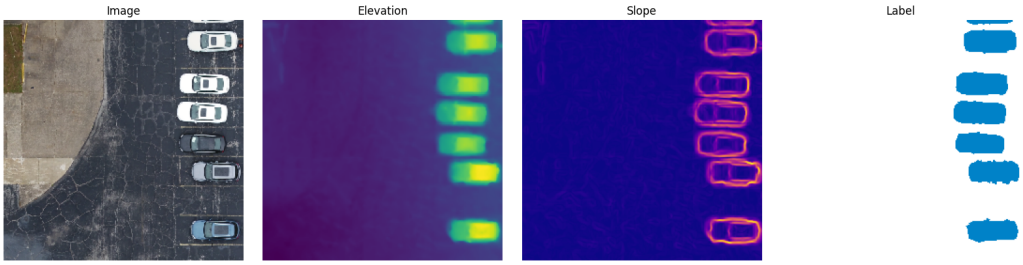


Figure 5: Example chip showing all four input modalities. From left to right: RGB image, standardised elevation, calculated slope, and ground truth segmentation. Elevation highlights structural height differences, while slope accentuates object edges and vertical boundaries.

4 Methodology

4.1 Overview

This project investigates three distinct approaches to semantic segmentation of aerial imagery: a convolutional neural network (U-Net), a transformer-based network (SegFormer), and a classical machine learning pipeline incorporating logistic regression, CRF post-processing, and SLIC superpixels.

U-Net is a widely adopted encoder–decoder architecture, initially proposed for biomedical image segmentation [1]. Its symmetric skip connections help preserve spatial context, making it well suited to high-resolution aerial segmentation tasks.

SegFormer is a transformer-based model that uses hierarchical attention to model long-range dependencies across an image [6]. While transformers have achieved state-of-the-art results in natural image benchmarks, their applicability to aerial imagery remains less explored, particularly in the absence of extensive pretraining.

The classical approach employs a logistic regression classifier trained on pixel-level features, followed by Dense Conditional Random Fields (CRFs) for spatial smoothing and Simple Linear Iterative Clustering (SLIC) superpixels to improve boundary coherence. Though simple, this approach serves as a useful non-deep learning baseline.

4.2 Input Modalities

Across all models, we evaluated performance under two input configurations: RGB only (3 channels) and RGB combined with elevation and slope (5 channels). Prior work has shown that incorporating depth or elevation cues can improve scene understanding and object boundary delineation [12, 21], particularly in cases where objects differ in height but not colour, such as roads and rooftops.

This setup allows us to assess the influence of auxiliary spatial information, specifically elevation and slope on semantic segmentation performance. We perform intra-model comparisons (i.e., RGB vs. RGB + Elev/slope) to isolate the benefits of these additional modalities, and inter-model comparisons to evaluate which architectures best leverage this information.

4.3 Loss Function Design and Class Imbalance Handling

Focal Loss [22] and Dice Loss are widely used to address class imbalance in segmentation. While Categorical Cross-Entropy (CCE) performs well on dominant classes, it can drown out rare class signals. Our deep learning models use a composite loss function inspired by DeepLab [4], combining CCE, Dice, and Focal Loss to improve stability and rare-class performance:

$$\mathcal{L}_{\text{Total}} = \lambda_1 \cdot \mathcal{L}_{\text{CCE}} + \lambda_2 \cdot \mathcal{L}_{\text{Dice}} + \lambda_3 \cdot \mathcal{L}_{\text{Focal}} \tag{2}$$

Loss weights $(\lambda_1, \lambda_2, \lambda_3)$ and class weights (α_c) were tuned separately per model:

- **U-Net:** $\lambda = [0.25,\ 1.5,\ 2.25], \quad \alpha_c = [6.95,\ 3.3,\ 0.3,\ 12.5,\ 4.0,\ 2.6]$
- **SegFormer:** $\lambda = [0.15,\ 1.0,\ 1.35], \quad \alpha_c = [1.13,\ 1.025,\ 0.95,\ 1.39,\ 1.1,\ 1.05]$

Class weights are ordered by class index: [Building, Clutter, Vegetation, Water, Background, Car].

Dice Loss: Measures overlap between predicted and true regions, improving sensitivity to small or fragmented classes:

$$\mathcal{L}_{\text{Dice}} = 1 - \frac{2TP_c}{2TP_c + FP_c + FN_c} \tag{3}$$

Focal Loss: Down-weights easy examples to focus learning on hard-to-classify pixels. We use $\gamma = 5$:

$$\mathcal{L}_{\text{Focal}} = -\sum_{c=1}^{C} \alpha_c (1 - p_c)^{\gamma} y_c \log(p_c) \tag{4}$$

4.4 U-Net

U-Net is a popular encoder-decoder architecture originally developed for biomedical image segmentation [1]. The U-Net architecture consists of a symmetric encoder–decoder structure with skip connections between corresponding layers. The encoder progressively reduces spatial resolution while increasing feature depth, capturing contextual information. The decoder upsamples these features and fuses them with high-resolution features from the encoder, enabling precise localisation. Our implementation includes batch normalisation, spatial dropout and L2 kernel regularisation.

- **Encoder:** Comprises four downsampling blocks, each consisting of two 3×3 convolutional layers with ReLU activations, followed by a 2×2 max pooling layer. Feature dimensions double at each level, starting from 64.
- **Bottleneck:** The lowest level includes two 3×3 convolutions (with 1024 filters), followed by dropout ($p = 0.5$) to prevent overfitting.
- **Decoder:** Mirrors the encoder structure. Each block begins with a 2×2 transposed convolution (upsampling), followed by concatenation with the corresponding encoder output, then two 3×3 convolutions with ReLU activations.
- **Output Layer:** A 1×1 convolution projects the final feature map to a 6-channel output, representing the class logits for each pixel. A softmax activation is applied to obtain per-pixel class probabilities.

Training used the Adam optimiser (learning rate 5.2×10^{-4}), early stopping, and label smoothing ($\epsilon = 0.075$). We tested a range of dropout rates, smoothing values, and learning rates, though extensive hyperparameter tuning was limited due to long training times. To mitigate class imbalance, we experimented with dynamic class weighting, CutMix augmentation, and adaptive sampling strategies.

L2 regularisation was applied to the U-Net architecture to improve generalisation. We also implemented a dynamic chip sampler and a custom callback for updating class weights during training. However, the chip sampler significantly increased training time and was replaced with CutMix, which was more effective and introduced no overhead. Although the dynamic class weight updater callback didn't introduce excessive overhead, it was found to be largely ineffective and disabled.

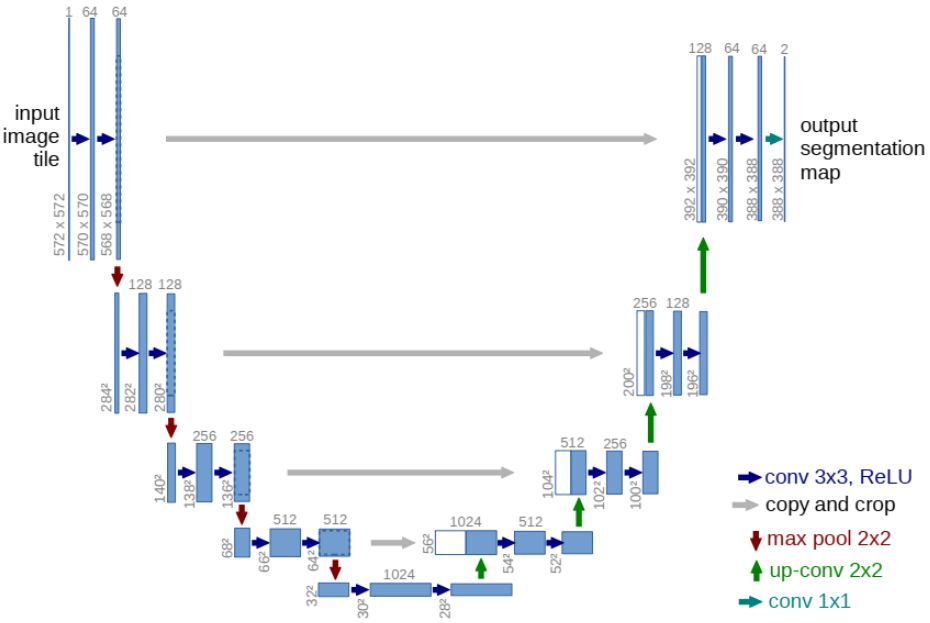


Figure 6: U-Net architecture (example for 32×32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. Adapted from Ronneberger et al. [1].

4.5 SegFormer (Transformer-Based Model)

SegFormer, introduced by Xie et al. [6], is a transformer-based segmentation model designed for efficient long-range dependency modelling and multi-scale feature extraction. We implemented the SegFormer-B1 variant from scratch in TensorFlow/Keras, adapting it for high-resolution aerial imagery (256×256) and 6-class segmentation.

As illustrated in Figure 7, the encoder consists of four hierarchical stages, each beginning with overlapping patch embeddings and followed by two transformer blocks. The use of **patch embeddings**, rather than traditional convolutional layers, enables the model to efficiently capture both local and global features across different spatial scales. Embedding dimensions increase across stages as [64, 128, 320, 512], with attention heads [1, 2, 5, 8], and spatial resolution reduces by factors of [8, 4, 2, 1]. These multi-scale feature maps help to capture both fine details and larger structural context, which is essential for accurately segmenting complex features in aerial imagery. The transformer blocks, designed to capture long-range dependencies, are crucial for understanding large-scale spatial relationships, making SegFormer particularly effective for high-resolution images. Output shapes range from $64 \times 64 \times 64$ to $8 \times 8 \times 512$, reflecting the model's ability to extract increasingly abstract features at multiple scales.

Rather than using positional encodings, SegFormer encodes spatial structure through **depthwise convolutions in the Mix-FFN blocks**. This design choice allows the model to effectively process images with varying spatial scales, helping maintain both accuracy and efficiency in segmentation tasks. These features are projected to a 6-dimensional embedding per class, upsampled to a common 64×64 resolution, concatenated, and fused via a lightweight convolutional module. A custom ResizeLayer restores spatial resolution to 256×256 using bilinear interpolation, followed by a softmax layer for per-pixel classification.

The full model comprises approximately 13.15M parameters in the encoder and 527k in the decoder. The lightweight decoder design contributes to SegFormer's efficiency while maintaining high segmentation performance. Training was performed without pretrained weights, using the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$) and a warmup-based learning rate schedule [23]:

$$\text{lrate} = d^{-0.5} \cdot \min(\text{step}^{-0.5}, \text{step} \cdot \text{warmup}^{-1.5}) \tag{5}$$

where $d_{\text{model}} = 256$ and warmup = 6900 steps (6 epochs). To improve generalisation, label smoothing ($\epsilon = 0.05$) was applied, and a composite loss function (CCE, Dice, Focal) was used with class weights tuned for imbalance. SegFormer was trained for 150 epochs with a batch size of 16 and a maximum runtime of 120 minutes per run.

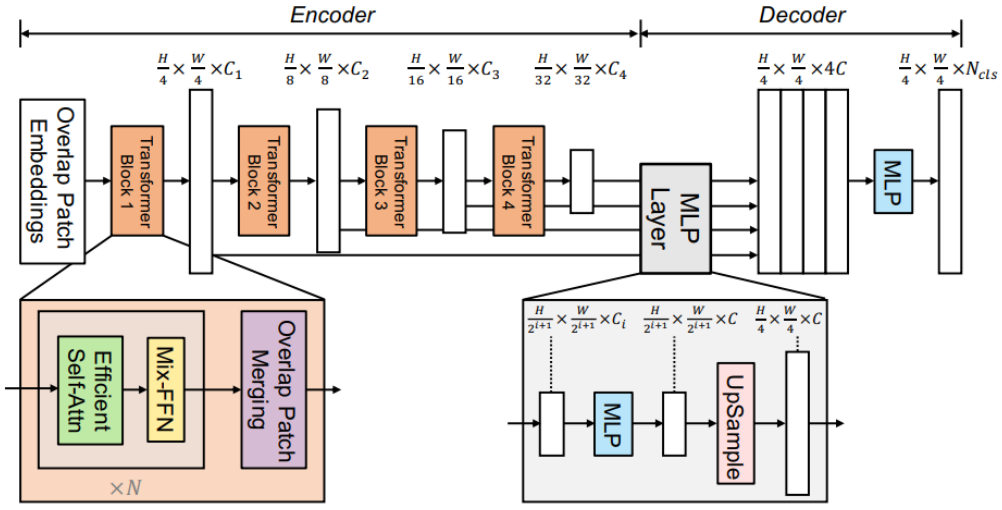


Figure 7: SegFormer-B1 architecture overview. The encoder consists of four hierarchical stages, each with overlapping patch embeddings followed by two transformer blocks. Feature maps are progressively downsampled while increasing in depth. The decoder upsamples and fuses multi-scale features to produce per-pixel class predictions. Positional encoding is handled via depthwise convolutions in the Mix-FFN blocks. Adapted from Xie et al. [6].

4.6 Non-Deep Learning Baseline (Logistic Regression with CRF and Superpixels)

Our non-deep learning baseline combines classical feature engineering, logistic regression, and structured post-processing using CRF and superpixels. This provides a lightweight, interpretable benchmark for semantic segmentation.

Feature Extraction Each 256×256 image chip is converted into a dense per-pixel feature vector using raw values, local statistics, and geometric cues:

- **RGB values**: Raw intensities, along with 3×3 window means and standard deviations to encode local colour context.
- **Grayscale texture (LBP)**: Local Binary Pattern features [24] for capturing fine texture.
- **Pixel coordinates**: Normalised $(x/w, y/h)$ spatial location of each pixel.

When using the full "RGB + Elevation + Slope" feature set, we also include:

- **Elevation and slope**: Normalised DSM values, local slope statistics, and gradient-based features encoding terrain variation.

The resulting features form a flattened $(H \cdot W, D)$ matrix for each chip, which serves as input to the classifier.

Model Training We employ `scikit-learn`'s `SGDClassifier` configured with the `log_loss` objective to approximate logistic regression. Training is performed incrementally using `partial_fit` to accommodate memory constraints. Feature vectors are standardised using a `StandardScaler` fit online during the training pass. Class imbalance is addressed using class weights computed via the inverse frequency of labels in the training set.

Post-Processing with CRF and Superpixels To enforce spatial consistency, we apply a Dense Conditional Random Field (CRF) as post-processing on the predicted probabilities. Prior to CRF inference, each chip is segmented into 500 superpixels using the SLIC algorithm [10], which groups neighbouring pixels with similar colour and texture. These superpixels are used to guide CRF pairwise terms, reducing noisy predictions and improving class boundaries.

5 Evaluation and Discussion

This section presents a comprehensive evaluation of the three models used in our study: U-Net, SegFormer, and a logistic regression model with CRF and SLIC-based superpixel smoothing. Each model is assessed under two input configurations: RGB-only and RGB combined with elevation and slope (5-channel input). This allows us to isolate the contribution of elevation data and compare performance gains across different architectures.

We analyse each model's segmentation performance in terms of macro and per-class metrics, and we also compare their training and inference times to evaluate computational efficiency.

The primary metric for segmentation performance was the **mean Intersection-over-Union (mIoU)**.

$$\text{IoU}_c = \frac{TP_c}{TP_c + FP_c + FN_c}, \quad \text{mIoU} = \frac{1}{C}\sum_{c=1}^{C} \text{IoU}_c \tag{5}$$

It is preferred over pixel accuracy for imbalanced datasets, as it equally weights each class by evaluating overlap between predicted and ground truth regions, and ignores true negatives. To complement mIoU, macro-averaged precision, recall, and F1-score were also used to comprehensively assess performance across all classes.

5.1 Training and Inference Time

Training and inference characteristics were recorded for each model using both RGB-only and RGB + elevation + slope inputs. All deep learning models were trained on a single NVIDIA A100 GPU. The logistic regression model was trained on CPU using scikit-learn's 'SGDClassifier' in an online fashion.

Both U-Net and SegFormer demonstrated distinct training dynamics. U-Net exhibited smooth convergence for both input modalities, with the addition of elevation and slope data leading to faster decrease in validation loss and fewer epochs (43 vs. 78) to converge. In contrast, SegFormer required more epochs (124 for RGB-only, 150 for RGB + Elev/Slope). While SegFormer with RGB-only inputs showed stability, the RGB + Elev/Slope configuration exhibited constant oscillation in its validation loss, potentially indicating a need for further training or hyperparameter tuning. The observed instability is likely due to the Transformer-based learning rate schedule adapted from [23]. The warmup stage may have terminated too early, and/or the inverse square root decay stage may have been too agressive for our dataset, causing the validation loss to fluctuate. Although stability did gradually improve throughout the training session, the multimodal input significantly extended SegFormer's training time (09:59:49 vs. 03:30:44), a trade-off justified by its superior generalisation.

The logistic regression + CRF/SLIC model was exceptionally fast (00:00:06 for RGB, 00:00:07 for RGB + Elev/Slope) due to its non-iterative nature, though this simplicity limited its segmentation performance.

Table 2 summarises average step times, inference latency, and total training time. Elevation and slope inputs introduced minor computational overhead for deep models (U-Net step time: 191.0 ms to 265.0 ms; SegFormer step time: 56.0 ms to 72.0 ms). When comparing inference latency, SegFormer generally exhibited lower times (7.3 ms RGB, 7.6 ms RGB + Elev/Slope) than U-Net (10.2 ms RGB, 23.9 ms RGB + Elev/Slope), making it more practical for real-time applications despite its longer training. These increases did not significantly impact overall runtime or feasibility.

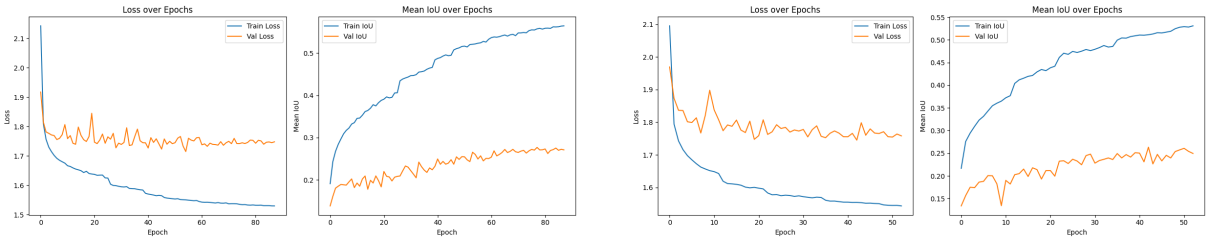
Deep learning models were trained using Google Colab Pro+ with 40GB VRAM, 83.5 GB RAM, and 256 GB disk space. This environment allowed for full-batch training without memory bottlenecks, crucial for large aerial image tiles. This setup highlights the importance of scalable hardware for high-resolution remote sensing applications.

Table 2: Training and inference statistics by model and input configuration.[2]

Model	Input	Batch Size	Step Time (ms)	Inference Time (ms)	Epochs	Train Time (h:m:s)
U-Net	RGB	64	191.0	10.2	78	02:11:02
U-Net	RGB + Elev/Slope	64	265.0	23.9	43	03:46:38
SegFormer	RGB	16	56.0	7.3	124	03:30:44
SegFormer	RGB + Elev/Slope	16	72.0	7.6	150	09:59:49
LogReg + CRF/SLIC	RGB	–	–	2.1	1	00:00:06
LogReg + CRF/SLIC	RGB + Elev/Slope	–	–	2.4	1	00:00:07

Figure 9 presents the training and validation loss curves for each model. U-Net converged smoothly but exhibited minor overfitting near the final epochs. SegFormer required more epochs to converge but maintained stability throughout. The LogReg baseline, being non-iterative, does not have a loss curve.

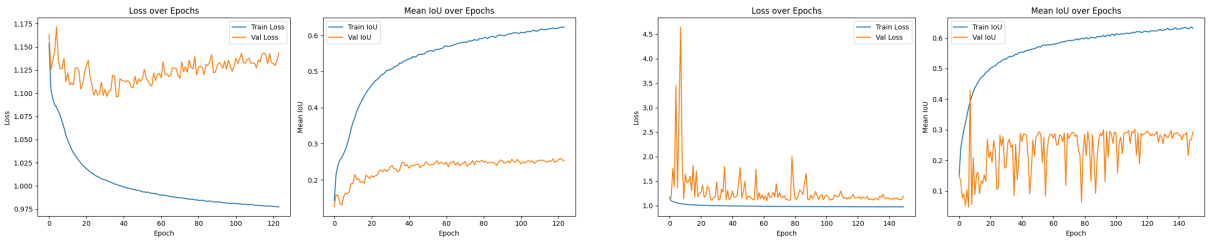
[2]Deep models were trained on an NVIDIA A100 via Google Colab Pro+ with 40GB GPU memory and 83.5GB system RAM.



(a) U-Net (RGB only)

(b) U-Net (RGB + Elev/Slope)

Figure 8: Training and validation loss curves for U-Net using both input modalities, RGB-only input (left) and RGB with elevation and slope input (right).



(a) SegFormer (RGB only)

(b) SegFormer (RGB + Elev/Slope)

Figure 9: Training and validation loss curves for SegFormer across Both Input Modalities.

5.2 Performance of Logistic Regression with CRF and Superpixels

As highlighted in the previous section, the logistic regression model with CRF and superpixel post-processing offers extremely fast training and inference. The traditional pipeline implemented using `SGDClassifier`, was more efficient in training compared to deep learning models, completing multiple epochs in under 10 minutes per partial fit. CRF post-processing improved label consistency by enforcing local spatial and appearance coherence, while SLIC-based superpixel smoothing further refined object boundaries.

However, this approach struggled to generalise to rare or small-scale classes such as *Clutter* and *Car*, which consistently showed the lowest IoU scores across all models. This aligns with the class imbalance in the dataset and the model's limited capacity to learn high-level features.

While surprisingly accurate on *Background* and *Building*, the logistic regression approach lacked robustness. Its inability to capture hierarchical or multiscale context limited its performance, although the lightweight training pipeline remains attractive in resource-constrained environments.

5.3 Results and Analysis

Our experimental results, presented in Tables 3 and 4, clearly demonstrate the impact of incorporating elevation and slope data on segmentation performance.

Table 3: mIoU and macro metrics for each model and input configuration.

Model	**Input**	**mIoU**	**F1 Score**	**Precision**	**Recall**
U-Net	RGB	0.3985	0.5239	0.5489	0.5282
U-Net	RGB + Elev/Slope	0.4228	0.5509	0.5785	0.5334
SegFormer	RGB	0.3771	0.5063	0.5426	0.5037
SegFormer	RGB + Elev/Slope	**0.4364**	0.5639	0.6161	0.5302
LogReg + CRF/SLIC	RGB	0.1976	0.2827	0.2907	0.3793
LogReg + CRF/SLIC	RGB + Elev/Slope	0.2648	0.3471	0.3273	0.4356

Table 4: Per-class IoU scores for each model and input configuration.

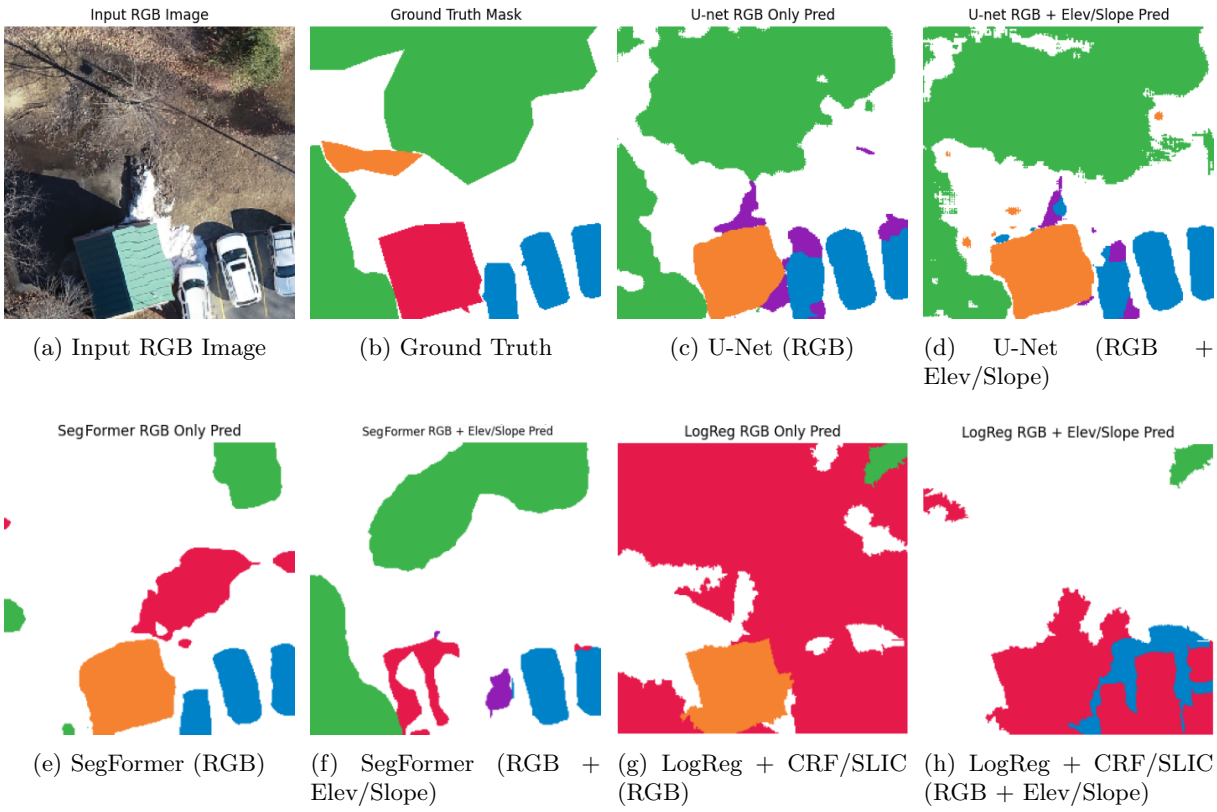
Model	Input	Building	Clutter	Vegetation	Water	Background	Car
U-Net	RGB	0.3658	0.1320	0.5942	0.0521	0.7046	0.5426
U-Net	RGB + Elev/Slope	0.5119	0.1645	0.6143	0.0617	0.7317	0.4525
SegFormer	RGB	0.3135	0.1104	0.5946	0.0877	0.6819	0.4748
SegFormer	RGB + Elev/Slope	**0.6812**	0.1620	0.5085	0.0787	0.7234	0.4645
LogReg + CRF/SLIC	RGB	0.6356	0.0000	0.2319	0.0943	0.2161	0.0076
LogReg + CRF/SLIC	RGB + Elev/Slope	0.7745	0.0000	0.1333	0.1811	0.4991	0.0009

Overall Performance and Impact of Elevation Data

SegFormer, with RGB + Elev/Slope input, achieved the highest overall performance, demonstrating an mIoU of **0.4364**, coupled with strong F1 (0.5639) and Precision (0.6161) scores. This configuration particularly excelled in segmenting structured classes such as *Building* (0.6812 IoU), *Car* (0.4645 IoU), and *Background* (0.7234 IoU).

The inclusion of elevation and slope consistently improved performance across all models. U-Net's mIoU increased from 0.3985 to 0.4228, a relative gain of 6.1%. SegFormer saw its mIoU increase from 0.3771 to 0.4364, a relative improvement of 15.7%. Notably, the *Building* class saw remarkable improvements across all models: U-Net's IoU for Building increased by 39.93% (0.3658 to 0.5119) and SegFormer's by an impressive 117.29% (0.3135 to 0.6812). This significant gain highlights the utility of height information for man-made structures, where distinct elevation changes provide strong discriminative cues.

The classical Logistic Regression pipeline, while exhibiting the lowest absolute mIoU (0.1976 RGB-only, 0.2648 RGB + Elev/Slope), demonstrated notable value. This lightweight approach saw the largest proportional mIoU increase with elevation and slope data (from 0.1976 to 0.2648), underscoring the substantial benefit of spatial features even non-deep learning models. Its efficiency and simplicity make it appealing for prototyping or resource-limited environments. With RGB-only input, the LogReg model tended to over-predict *Background* as *Building*, using all five channels significantly increased *Background* IoU to (0.2161 to 0.4991), a relative gain of 130.96%, indicating elevation helped resolve this confusion.



(a) Input RGB Image (b) Ground Truth (c) U-Net (RGB) (d) U-Net (RGB + Elev/Slope)

(e) SegFormer (RGB) (f) SegFormer (RGB + Elev/Slope) (g) LogReg + CRF/SLIC (RGB) (h) LogReg + CRF/SLIC (RGB + Elev/Slope)

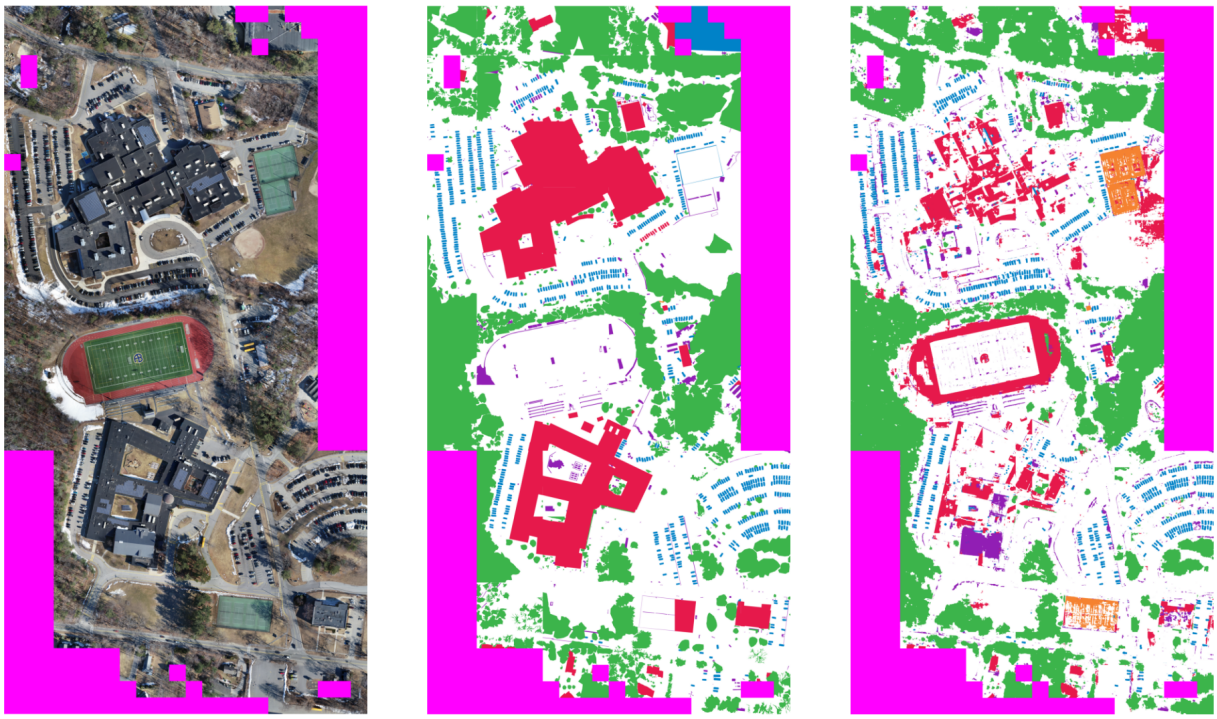
Figure 10: Comparison of segmentation predictions for a test chip. Top row: input RGB and ground truth label, followed by U-Net predictions. Bottom row: SegFormer and LogReg + CRF/SLIC predictions.

Qualitative Analysis and Specific Class Challenges

Qualitative predictions (Figure 10) show U-Net producing clean edges, while SegFormer achieved competitive results, though it struggled slightly more with *Vegetation* than U-Net. Both deep models improved with elevation, especially in recognising buildings. However, *Clutter* and *Water* remained consistently challenging classes, often confused with *Background* or *Vegetation* due to ambiguous boundaries, label noise, and weak visual distinctiveness.

Clutter proved consistently difficult due to its vague definition and overlap with other classes. *Car* was sometimes mislabelled as *Clutter*, or missed entirely and assigned to *Background*. *Building* was occasionally misclassified as *Background*, particularly when flat rooftops resembled roads. The absence of a dedicated *Road* class limited the models' ability to distinguish man-made surfaces. While elevation and slope offered marginal help by highlighting height changes, they were often insufficient in dense, flat urban areas.

Water was the most challenging class, frequently confused with *Background* and *Vegetation* due to visual ambiguity. Its wide variation, spanning rivers, lakes, and swimming pools, complicates abstraction under a single label. For instance, Figure 11 shows a failure case: tennis courts misclassified as *Water*. This is likely due to their blue-green tones and rectangular shape, which models may have learned to associate with swimming pools.



(a) RGB Image (b) Ground Truth (c) Model Prediction

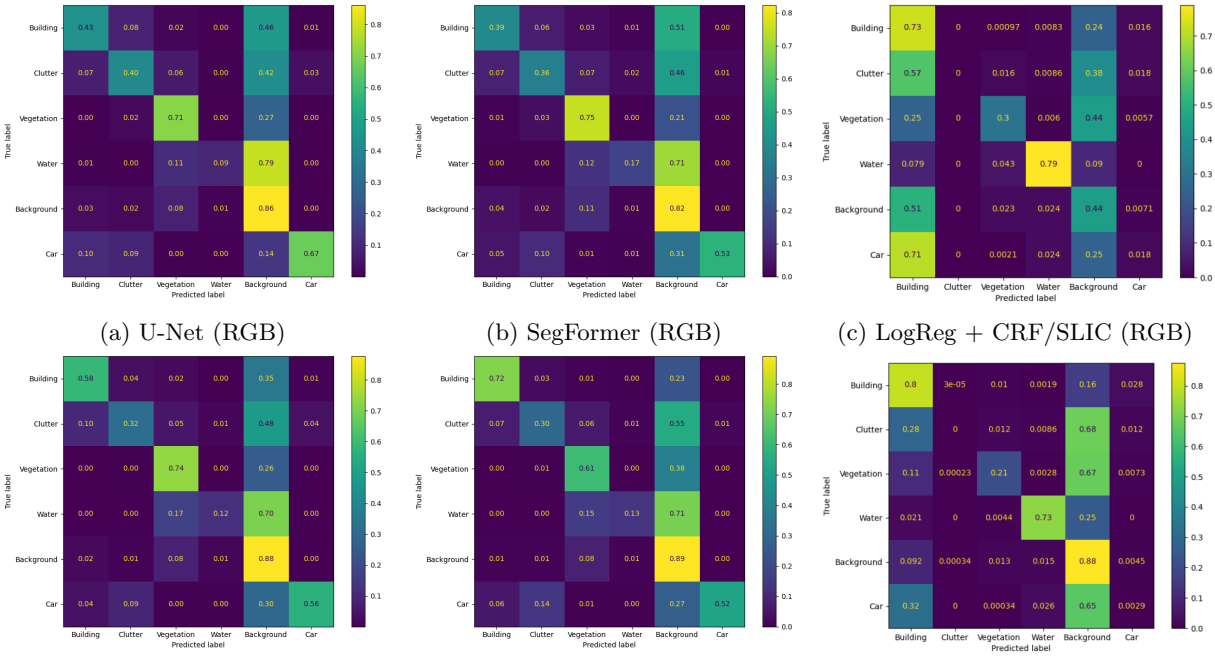
Figure 11: Comparison of RGB input, ground truth labels, and model prediction. U-Net predicts tennis courts (true label: Background □) as Water ■. Source: 25f1c24f30_EB81FE6E2BOPENPIPELINE.

Confusion Matrices and Model Trade-offs

Figure 12 shows the confusion matrices for all three models under both RGB-only and RGB + Elevation/Slope inputs. All models performed well on dominant classes like *Background* and *Building*, though confusion persisted for smaller or ambiguous classes.

SegFormer achieved the highest mIoU, macro F1, and precision scores, but required more training time and GPU memory. U-Net was faster and more efficient, while still performing strongly on *Car*, *Vegetation*, and *Background*. The Logistic Regression + CRF/SLIC pipeline performed worst in absolute terms, yet saw consistent gains across all classes with elevation. While struggling with fine detail, its fast training and inference (under 10 seconds) make it attractive for edge applications or rapid prototyping with limited resources.

These results show that elevation is most useful in tasks focused on man-made structures, such as cadastral mapping or urban planning, where accurately identifying buildings is critical. Here, height data offers clear benefits. Conversely, natural or ambiguous classes like *Clutter* and *Water* remain harder to improve even with additional spatial data, indicating a need for better annotation and class definitions in future datasets.



(a) U-Net (RGB) (b) SegFormer (RGB) (c) LogReg + CRF/SLIC (RGB)

(d) U-Net (RGB + Elev/Slope) (e) SegFormer (RGB + Elev/Slope) (f) LogReg + CRF/SLIC (RGB + Elev/Slope)

Figure 12: Confusion matrices for each model using (top row) RGB-only and (bottom row) RGB + Elevation/Slope inputs.

5.3.1 Comparison to Benchmarks

Comparing our results directly to existing benchmarks is challenging, as our models were trained entirely from scratch without using pre-trained weights, unlike most state-of-the-art implementations.

We first examine existing benchmarks summarised in Table 5 from Heffels et al. [25], which also utilised the DroneDeploy dataset. UNet variants with pre-trained backbones (e.g., InceptionResnetV2) achieve validation mIoU scores around 0.63–0.64. Our best U-Net model, with RGB + Elev/Slope inputs, achieved an mIoU of 0.4228 (Table 3). This represents a substantial performance gap of approximately 0.20 mIoU points, primarily attributable to the benefit of pre-training in the benchmark models. This benchmark primarily focuses on UNet and other CNN architectures and does not provide results for SegFormer.

To contextualise SegFormer's performance, we refer to an alternate benchmark (Table **??**) from a paper by Spasev et al. [26] that investigates SegFormer for aerial segmentation, a similar domain to our own. This benchmark includes results for various SegFormer variants, as well as other models like U-Net and DeepLabV3+, showing mIoU scores predominantly in the 60-70% range (e.g., SegFormer-B0 (tta) achieving 66.91% mIoU, and U-Net achieving 67.22% mIoU). Our best SegFormer model, with RGB + Elev/Slope, achieved an mIoU of 0.4364 (Table 3), and our U-Net RGB + Elev/Slope achieved 0.4228 mIoU. When compared to this alternate benchmark, our models consistently show a performance difference of roughly 0.20 to 0.25 mIoU points.

Table 5: IoU scores for different model and backbone combinations on the DroneDeploy dataset [25].

Model	Backbone	Train IoU	Val IoU
UNet	EfficientNetB3	0.77	0.64
	InceptionResnetV2	0.80	0.63
	MobileNetV2	0.74	0.59
FPN	**EfficientNetB3**	**0.81**	**0.65**
	InceptionResnetV2	**0.81**	**0.65**
	MobileNetV2	0.76	0.61
LinkNet	EfficientNetB3	0.76	0.61
	InceptionResnetV2	0.78	0.63
	MobileNetV2	0.71	0.54
PSPNet	EfficientNetB3	0.71	0.56
	InceptionResnetV2	0.74	0.58
	MobileNetV2	0.68	0.52

Table 6: Selected alternate benchmark mIoU scores for aerial segmentation [26].

Model Variant	mIoU (%)
U-Net	67.22
DeepLabV3+	67.72
UNetFormer	67.8
SegFormer-B0	66.19
SegFormer-B0 (tta)	66.91
SegFormer-B5	69.55
SegFormer-B5 (tta)	**70.23**

6 Conclusion and Future Works

This project successfully explored the impact of adding elevation and slope data to RGB images for semantic segmentation of aerial scenes, evaluating U-Net, SegFormer, and a logistic regression baseline.

Our findings consistently show that auxiliary spatial inputs (elevation and slope) significantly improved semantic segmentation performance across all models, particularly for building-like structures and height-differentiated surfaces. SegFormer achieved the highest overall mIoU of 0.4364, excelling in segmenting structured classes like Building (0.6812 IoU). The Building class saw remarkable improvements with elevation data across deep models, with SegFormer's IoU increasing by an impressive 117.29%. The classical Logistic Regression pipeline, despite lower absolute mIoU, demonstrated the largest proportional gain with elevation and slope data (from 0.1976 to 0.2648). This highlights the substantial benefit of spatial features even for simpler models, proving useful for prototyping or resource-limited environments. Elevation data also helped resolve the Logistic Regression model's tendency to over-predict Background as Building, resulting in a 130.96% relative gain in Background IoU.

Overall, our results demonstrate the significant value of multimodal learning in remote sensing. SegFormer emerged as the most accurate overall model, while U-Net offered a practical balance between performance and efficiency.

For future work, to further improve segmentation quality and robustness, we recommend:

- Introducing a dedicated *Road* class to reduce confusion between *Building* and *Background*.
- Refining ambiguous categories like *Clutter* to reduce annotation noise.
- Investigating advanced data fusion techniques, such as modality-specific encoders, to leverage pre-trained backbones more effectively with multimodal inputs, as current architectures designed for 3-channel RGB are incompatible with 5-channel RGB+Elevation+Slope without significant architectural modifications.
- Training models on "harder" chips (e.g., those dominated by *Water* or rare classes).

This work serves as a reproducible foundation for future research into efficient, robust semantic segmentation of aerial data.

References

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

[2] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pp. 3–11, Springer, 2018.

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

[4] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, pp. 801–818, Springer, 2018.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[6] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[7] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, "Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," *ECCV*, 2018.

[8] R. P. Poudel, S. Liwicki, and R. Cipolla, "Fast-scnn: Fast semantic segmentation network," in *BMVC*, 2019.

[9] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *Advances in Neural Information Processing Systems*, vol. 24, pp. 109–117, 2011.

[10] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[11] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European Conference on Computer Vision (ECCV)*, pp. 345–360, Springer, 2014.

[12] D. Vanderstukken, O. Debeir, B. Cornelis, and S. Jodogne, "Solar panel segmentation on aerial images using color and elevation information," in *ESANN 2025: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 283–288, 2025.

[13] A. Valada, J. Vertens, A. Milan, and W. Burgard, "Adapnet: Adaptive semantic segmentation in adverse environmental conditions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4640–4647, IEEE, 2017.

[14] T. Wang, G. Chen, X. Zhang, C. Liu, J. Wang, X. Tan, W. Zhou, and C. He, "Lmfnet: Lightweight multimodal fusion network for high-resolution remote sensing image segmentation," *Pattern Recognition*, vol. 164, p. 111579, 2025.

[15] Y. Zhang, W. Wang, and L. Lin, "Fusion-aware transformer for multimodal semantic segmentation," in *CVPR Workshops*, 2020.

[16] A. Eitel, J. Richter, L. Schönherr, and M. Strube, "Multimodal segmentation: a survey," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

[17] P. L. Renard, P. Renard, and D. S. J. C., "Privacy concerns in remote sensing: a review,"

[18] A. Torralba and A. A. Efros, "An unbiased look at dataset bias," in *2011 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1521–1528, IEEE, 2011.

[19] A. Jobin, M. Ienca, and E. Vayena, "The global landscape of ai ethics guidelines," *Nature Machine Intelligence*, vol. 1, no. 9, pp. 389–399, 2019.

[20] J. Liu, Z. Wang, and K. Cheng, "An improved algorithm for semantic segmentation of remote sensing images based on deeplabv3+," 2019.

[21] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European Conference on Computer Vision (ECCV)*, pp. 345–360, Springer, 2014.

[22] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *arXiv preprint arXiv:1708.02002*, 2017.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008, 2017.

[24] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[25] M. R. Heffels and J. Vanschoren, "Aerial imagery pixel-level segmentation," *arXiv preprint arXiv:2012.02024*, 2020.

[26] V. Spasev, I. Dimitrovski, I. Chorbev, and I. Kitanovski, "Semantic segmentation of unmanned aerial vehicle remote sensing images using segformer," *arXiv preprint arXiv:2310.10924*, October 2024.

Appendix

A Member Contributions

Each group member contributed equally to this project, with a contribution of approximately 33.33%. Below is a breakdown of individual responsibilities and achievements, as agreed upon by all members.

Aron Bakes (n11405384) – 33.33%

Aron led the development of core pipeline components, writing the majority of the data preprocessing and augmentation logic—including chipping, per-tile standardisation, and data streaming. He implemented the full U-Net model and training pipeline, optimised model evaluation procedures, and supported SegFormer integration. Aron also managed group coordination and documentation, drafted most sections of the final report, and handled LaTeX typesetting and formatting.

Deegan Marks (n11548444) – 33.33%

Deegan contributed heavily to SegFormer implementation and debugging, working closely with Aron to troubleshoot input formatting, model compatibility, and training stability. He also researched related work, helped refine performance comparisons, and contributed to writing sections of the report related to deep learning models. Deegan played an active role in experimenting with hyperparameter configurations and evaluating segmentation output.

Jordan Geltch-Robb (n11427515) – 33.33%

Jordan led development of the non-deep learning baseline, implementing logistic regression with CRF and SLIC smoothing. He conducted extensive testing across input configurations and helped shape the overall experimental design. Jordan also contributed to coding key components on his local environment and collaborated with Aron on model evaluation strategies. Additionally, he took responsibility for producing the video presentation, scripting and editing a complete 5-minute summary of the project.