CS-433 : Project 2 Road Segmentation

Aurélien Ferlay

***Abstract*—This project centers on road segmentation, a critical task for applications such as traffic planning, monitoring, and autonomous driving. We present the use of machine learning algorithms to segment roads, utilizing a dataset of real-world images to analyze road segment characteristics. Our evaluation demonstrates excellent performance in recognizing road patterns, showcasing the models' capability to accurately differentiate road areas from non-road regions.**

***Index Terms*—Segmentation, Machine Learning, Road Detection.**

I. Introduction

This project explores the implementation of road segmentation using satellite imagery obtained from Google Maps, aiming to classify image segments into two categories: **road** and **background.** This paper details our approach to selecting and optimizing neural network models tailored for pixel-level classification tasks.

II. Data Processing

A. Data Analysis

The training dataset consists of 100 satellite images, each with dimensions of 400 × 400 pixels and three color channels (red, green, and blue - RGB). Each image is paired with a corresponding ground-truth mask, a 400 × 400 grayscale image where pixel labels classify areas as either "road" (white) or "background" (black).

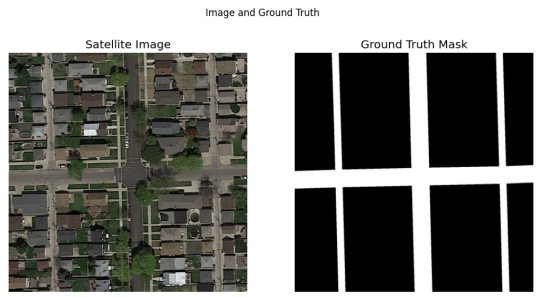


Fig. 1: The figure displays a satellite training image on the left, showing the original RGB data, and its corresponding ground-truth mask on the right, a grayscale image where white pixels represent "road" and black pixels indicate "background."

a) Data Characteristics: An analysis of pixel labels revealed a significant class imbalance:

- **Road pixels:** 18.68%
- **Background pixels:** 81.32%

This imbalance necessitates the use of evaluation metrics that balance precision and recall. As such, the F1-score was selected as the primary metric for assessing model performance.

B. Data Augmentation

a) Motivation for Augmentation: Our initial dataset of 100 satellite images lacked the diversity necessary for training a robust neural network. Neural networks require extensive datasets to generalize well to unseen scenarios. To overcome this limitation, we implemented data augmentation techniques to artificially expand the dataset, introducing variability in road appearance, orientation, and scale.

b) Augmentation Techniques: We applied the following transformations, tailored to the unique characteristics of satellite imagery and road segmentation:

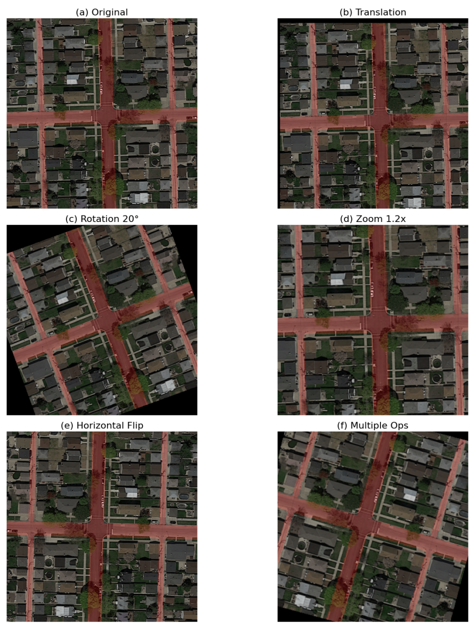


Fig. 2: Examples of data augmentation techniques applied to a satellite image and its corresponding mask (shown in red as an overlay). (a) Original image with mask overlay, (b) Translated image, (c) Rotated image (20°), (d) Zoomed image (1.2x zoom), (e) Horizontally flipped image, and (f) Image with multiple combined transformations (rotation, translation, zoom, and flip). These transformations illustrate the variety of augmentations used to enhance model robustness.

To enhance the robustness of our model and improve its ability to generalize, we implemented a comprehensive data augmentation pipeline. The following transformations were applied to the original dataset:

- **Rotation:** Random rotations were applied with angles uniformly selected between $-30°$ and $30°$. This helps the model handle images where roads appear at different orientations.

- **Translation:** Images were shifted horizontally and vertically by random offsets between -10 and 10 pixels to simulate different viewpoints.
- **Zoom:** Random zoom factors were applied, ranging between $0.8\times$ (zoom out) and $1.2\times$ (zoom in). Cropping or padding was used to maintain the original image dimensions.
- **Flipping:** Random horizontal and vertical flips were performed, accounting for possible symmetry in satellite imagery.
- **Combination of Transformations:** With a probability of 50%, some images underwent a combination of rotation, translation, zoom, and flipping. This approach further increased diversity in the dataset by simulating complex real-world variations.

The augmented dataset significantly increased the number of images, with each original image generating 5 new augmented samples. This resulted in a total dataset size 6 times larger than the original.

C. Dataset Preparation

To train and evaluate our models, we prepared two distinct datasets based on the original 400×400 satellite images. Each dataset was tailored to meet the requirements of the models designed for different input sizes.

1) Augmented 400x400 Dataset: The first dataset consists of the 400×400 augmented images. Starting with the original 100 images, we applied the previously described augmentation techniques to generate 600 images of size 400×400. This dataset is used to train models specifically designed to process 400×400 inputs, preserving the resolution and spatial context of the original images.

2) 256x256 Patch Dataset with Overlapping: The second dataset was created by extracting patches from the augmented 400×400 images. Each 400×400 image was divided into four overlapping patches of size 256×256, ensuring complete coverage of the original image. The overlapping regions guarantee that no information is lost during the patching process.

This process resulted in a dataset of $4 \times 600 = 2400$ patches of size 256×256, along with their corresponding ground-truth masks. This dataset was used to train models designed to process smaller input sizes, allowing for faster training and reduced memory usage.

D. Data Normalization and Splitting

a) Normalization: To ensure numerical stability and consistency during training, all image pixel values were normalized to the range $[0, 1]$. Ground-truth masks were binarized with a threshold of 0.5, ensuring that pixel values were either 0 (background) or 1 (road).

b) Dataset Splitting: Both datasets were split into training, validation, and test sets:

- For the 400×400 dataset:
  - **Training set:** 80% of the 600 images (480 images).
  - **Validation set:** 20% of the 600 images (120 images).
- For the 256×256 patch dataset:
  - **Training set:** 80% of the 2400 patches (1920 patches).
  - **Validation set:** 20% of the 2400 patches (480 patches).

For both datasets, a separate test set consisting of 50 images of size 608×608 (provided on the AICrowd platform) was used for model evaluation. Patches were generated from these test images during inference.

E. Model-Specific Dataset Usage

a) Models for 400×400 Inputs: Models designed for 400×400 inputs were trained using the augmented 400×400 dataset. This allowed the models to leverage the full spatial context of the original images, providing better performance on tasks requiring high spatial resolution.

b) Models for 256×256 Inputs: Models designed for 256×256 inputs were trained using the 256×256 patch dataset. The smaller input size enabled faster training and lower memory requirements, making this approach suitable for scenarios with computational constraints.

F. Summary of Dataset Preparation

- Two datasets were prepared: 400×400 augmented images and 256×256 overlapping patches.
- The 400×400 dataset was used for full-resolution models, while the patch dataset was used for smaller input models.
- Both datasets were split into training (80%) and validation (20%) subsets, with a separate test set of 608×608 images for evaluation.

III. Models and Training

A. Model Architectures and Configurations

To address the task of road segmentation, we designed and trained U-Net architectures with different input sizes and configurations. Our approach involved tailoring the model to the input resolution, balancing computational efficiency with segmentation performance.

1) Configurations: Two types of input sizes were used for model training: 256×256 and 400×400. For each input size, different configurations of batch size, number of filters, and layers were tested.

- **256x256 Input Models:** We used batch sizes of 32 with initial filters of 16 or 32. These configurations allowed faster training on a CPU while maintaining sufficient capacity for feature extraction.
- **400x400 Input Models:** Models with 400×400 inputs were trained using a batch size of 16, with initial filters of 16 or 32. These configurations were computationally more demanding due to the larger input size but captured more global context.

2) Regularization and Skip Connections: For all models, careful attention was paid to the selection of hyperparameters to balance performance and computational efficiency:

- **Regularization:** L2 regularization with a factor of 0.01 was applied to all convolutional layers. This penalizes large weights in the model, helping prevent overfitting and ensuring smoother generalization to unseen data.

- **Dropout:** Dropout rates ranged from 0.1 to 0.3 across layers. Dropout was progressively increased in deeper layers, reflecting their greater susceptibility to overfitting due to a higher number of parameters.
- **Skip Connections:** Skip connections were implemented in all models to directly transfer spatial information from the encoder to the decoder.

3) Training Hyperparameters: The following hyperparameters were used consistently across all experiments to ensure fair comparisons:

- **Input Shape:** Dynamically set to the size of the training images.
- **Learning Rate:** A learning rate of 1×10^{-4} was chosen for the Adam optimizer to ensure stable convergence.
- **Batch Size:** Models with 256×256 inputs used a batch size of 32, while models with 400×400 inputs used a batch size of 16 for large inputs).
- **Number of Layers:** The number of encoder-decoder layers was set to 4, providing a balance between model depth and computational requirements.
- **Epochs:** Training was capped at 100 epochs to allow sufficient training time for convergence.
- **Initial Filters:** The number of initial filters in the first convolutional block was set to 16 or 32 and scaled by powers of 2 in deeper layers to capture increasingly abstract features.

4) Activation Functions Used: In this U-Net architecture, the *ReLU* (Rectified Linear Unit) activation function is employed in the convolutional layers to introduce non-linearity while maintaining computational efficiency, effectively mitigating the vanishing gradient problem. Additionally, the output layer uses a *sigmoid* activation function to map predictions to probabilities within the range [0, 1], enabling binary pixel-wise classification.

5) Model Training and Callbacks: To optimize training and ensure model generalization, we implemented two key callbacks during training:

- **Model Checkpointing:** A `ModelCheckpoint` callback was used to save the best-performing model based on validation loss. This ensured that the final saved model represented the configuration with the lowest validation error, minimizing overfitting.
- **Early Stopping:** To avoid overfitting and reduce unnecessary training time, an `EarlyStopping` callback with a patience parameter of 5 epochs was employed.

These callbacks ensured efficient use of computational resources while maintaining model quality.

- **Model Checkpointing:** Prevented the risk of selecting overfit models by consistently saving the optimal weights.
- **Early Stopping:** Reduced training duration, particularly for larger input sizes such as 400×400, by stopping training early when further epochs showed diminishing returns.

By integrating these callbacks into the training process, we achieved a balance between model performance and computational efficiency, making our approach well-suited for CPU-limited environments.

B. Loss Functions

For training our models, we implemented a custom loss function tailored to handle the class imbalance inherent in the dataset and optimize segmentation performance. The loss functions used are described below:

a) Dice Loss: The Dice Loss is specifically designed for binary segmentation tasks and evaluates the overlap between the predicted and ground truth masks. It is particularly effective for imbalanced datasets, as it directly optimizes the intersection-over-union (IoU) metric [1]. The formula for Dice Loss is:

$$\text{Dice Loss} = 1 - \frac{2 \cdot |P \cap G|}{|P| + |G|}$$

where P and G represent the predicted and ground truth masks, respectively.

b) Focal Loss: To address class imbalance, we used Focal Loss. This loss focuses the training process on harder examples by down-weighting the contribution of well-classified pixels. [2] The formula for Focal Loss is:

$$\text{Focal Loss} = -\alpha \cdot (1 - p_t)^{\gamma} \cdot \log(p_t)$$

where p_t is the predicted probability for the true class, γ is a focusing parameter (set to 2.0) and α is a balancing parameter (set to 0.25).

c) Combined Loss (Dice + Focal Loss): The Combined Loss function used in this project combines the strengths of Dice Loss and Focal Loss.

d) Loss Function for All Models: For both the 400×400 and 256×256 models, we consistently used the Combined Loss function.

The following table summarizes the configurations of the four trained models:

TABLE I: Summary of Model Configurations

Model Name	Input Size	Batch Size	Initial Filters
256B32F16	256×256	32	16
256B32F32	256×256	32	32
400B16F16	400×400	16	16
400B16F32	400×400	16	32

IV. Thresholding Method

We applied a thresholding method to binarize the U-Net probability outputs. A threshold T was optimized on the validation set by maximizing the F1 score, ensuring a balance between precision and recall for the final segmentation maps.

V. Results

A. Model Performance

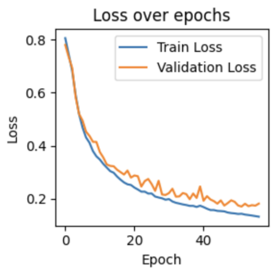
We trained and evaluated four U-Net models on the datasets prepared as described in Section 3. Each model was trained with different input sizes, batch sizes, and regularization strengths. Table II summarizes the key performance metrics for the models, including the F1-scores on the training, validation, and AI Crowd test sets.

TABLE II: Performance of different U-Net models on road segmentation.

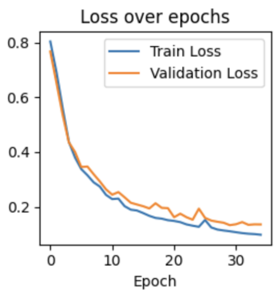
Model Name	L2 Reg	T F1	V F1	AI Crowd F1
256B32F16	1e-6	0.912	0.868	0.806
256B32F32	1e-6	0.934	0.889	0.832
400B16F16	1e-6	0.867	0.821	0.805
400B16F32	1e-6	0.957	0.896	0.851

B. Training Loss Analysis

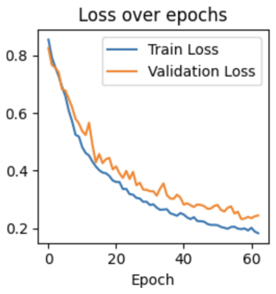
To better understand the training dynamics of the models, we plotted the training and validation loss curves for each model over the epochs. Figure 3 shows these curves for the four models.



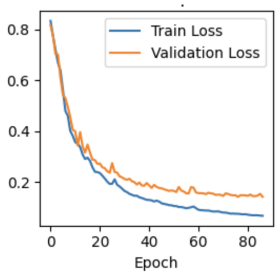
(a) 256B32F16 Loss Curve



(b) 256B32F32 Loss Curve



(c) 400B16F16 Loss Curve



(d) 400B16F32 Loss Curve

Fig. 3: Training and validation loss curves for the four U-Net models. Each subplot corresponds to a specific model configuration. The X-axis represents the epochs, while the Y-axis shows the loss.

a) Analysis of Loss Trends: - For all models, the training loss decreased steadily over the epochs, indicating proper optimization. - Validation loss closely followed the training loss, suggesting minimal overfitting due to the use of L2 regularization and dropout. - Models trained with larger filter sizes (e.g., 32 vs. 16) achieved slightly lower validation loss but at the cost of longer training times.

C. Discussion

- **Smaller Models:** The models with 256x256 input images and 16 filters per layer (e.g., 256B32F16) trained faster but slightly underperformed on the AI Crowd test set compared to models with 32 filters. - **Larger Models:** Models trained with 400x400 input images and larger filters (e.g., 400B16F32) achieved competitive performance but required significantly longer training times. - Regularization techniques such as dropout and L2 penalties proved effective in preventing overfitting, as indicated by the consistent gap between training and validation F1-scores.

VI. Improvements

To enhance performance, increasing the number of filters in the U-Net layers (e.g., from 16-32 to 64-128) could allow the model to capture more complex features, improving segmentation quality. Additionally, expanding the dataset with more diverse augmentations, such as varying lighting conditions or introducing noise, could improve the model's robustness to unseen scenarios.

Further optimization could focus on hyperparameter tuning, including exploring different learning rates, dropout rates, and L2 regularization factors to achieve a better balance between overfitting and underfitting. Automated techniques like grid search or Bayesian optimization could help identify the best combinations of these hyperparameters.

VII. Ethics and Risks

Road segmentation offers significant societal benefits but poses ethical challenges. **Bias** may arise if the dataset lacks diversity in road types or regions, leading to unequal performance. **Privacy concerns** must also be addressed when using satellite imagery to avoid exposing sensitive information. Lastly, **errors** in segmentation could have critical implications, particularly in autonomous driving. Mitigation strategies include curating diverse datasets, ensuring compliance with privacy standards, and auditing models for fairness and accountability.

VIII. Conclusion

In this project, we explored road segmentation using neural networks, comparing models trained on datasets with varying input sizes and augmentation techniques. The combination of advanced loss functions, regularization, and data augmentation demonstrated the potential for accurate segmentation even with limited computational resources. The best performance was achieved with the model trained on 400x400 images using a batch size of 16, initial filters of 32, regularization and dropout, which obtained an F1-score of 0.851 on the AICrowd test set. Despite these promising results, further improvements in model architecture, data diversity, and computational efficiency are necessary to enhance performance and scalability. This work highlights the importance of robust preprocessing, careful hyperparameter tuning, and ethical considerations in machine learning applications for real-world challenges.

References

[1] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Proceedings of the Deep Learning in Medical Image Analysis Workshop (DLMIA)*, ser. Lecture Notes in Computer Science, vol. 10553. Quebec City, Canada: Springer, 2017, pp. 240–248.

[2] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018. [Online]. Available: https://arxiv.org/abs/1708.02002