U-Net Semantic Segmentation of Aerial Photographs

Grayson Martin
Harvard College
CS 2831
graysonmartin@college.harvard.edu

Abstract

Semantic segmentation of aerial imagery is a critical tool for applications such as environmental monitoring, urban planning, and disaster assessment. In this project, I employed the U-Net architecture attempting a variety of enhancements to improve segmentation accuracy on a set of sattelite images of Mumbai. Seven models were trained and evaluated, each integrating specific changes to the base model in loss function, encoder depth, dropout regularization, and addition of attention mechanisms. Metrics including IoU, Dice, Precision, and Recall were used to assess model performance across six classes: vegetation, built-up areas, informal settlements, impervious surfaces, barren land, and water. A small "unclassified" class is also considered.

1. Introduction

Aerial photography has a wide variety of uses, including geology, archaeology, disaster assessment, and environmental monitoring [8]. The technology improved and became more widely utilized for military purposes starting in World War I, and has since gone on to be used for tasks such as identifying different vegetation types and detecting diseased and damaged vegetation and counting how many missiles, planes, and other military hardware adversaries have and where it is located [2]. Semantic segmentation, or pixel-wise classification, is useful in this context. Creating a mask that classifies all regions of an aerial image allows for monitoring of environmental conditions, foreign objects, and changing conditions over time. While this project explores semantic segmentation of aerial images, the technology is useful for a broad number of tasks in biology, robotics, agriculture, sports analysis, and more.

1.1. Semantic Segmentation Performance Metrics

Performance metrics for semantic segmentation can be thought of in terms of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) classifications for each pixel.

The most common performance metric for semantic segmentation is the Jaccard Score, also known as *Intersection over Union (IoU)*.

$$\text{IoU} = \frac{TP}{TP + FP + FN} \tag{1}$$

Intuitively, this score represents the amount of overlap between prediction and ground truth across the image.

Another frequently used metric for semantic segmentation is the Dice Score, also known as the *F1 Score*. This metric is formulated from two related metrics, *Precision* and *Recall*. The F1 Score is formulated

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1 Score} = 2\left(\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}\right) \tag{4}$$

Intuitively, Precision can be interpreted as the proportion of predicted positive pixels that are correctly segmented, Recall as the proportion of ground truth pixels that are correctly segmented, and F1 as the harmonic mean of both.

1.2. Transposed Convolution

Transposed convolution, also known as *upconvolution*, is a method of upsampling similar to downsampling with convolution. Between each input pixel, zeros are inserted to increase the size of the feature map before convolution with the kernel. An example of transposed convolution is given in Figure 1.

1.3. Dataset Description

The dataset explored in this project is the Manually Annotated High Resolution Satellite Image Dataset of Mumbai for Semantic Segmentation [1]. The dataset was created from high-resolution, true-color satellite imagery of

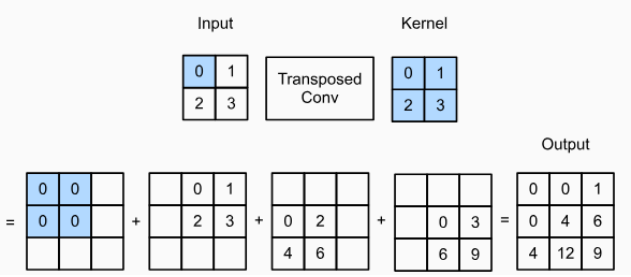

Figure 1. An Example Upconvolution with a 2×2 Input, 2×2 Kernel, and Stride of 1[12]

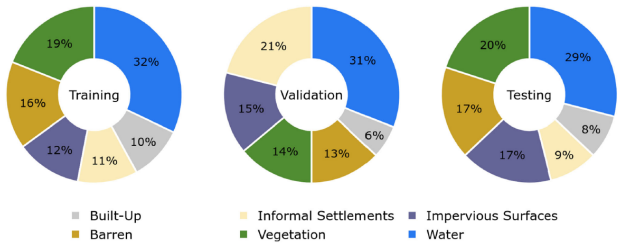

Figure 2. Distribution of Labels Across Patches[4]

Pleiades-1A acquired on March 15, 2017 over Mumbai. There are six classifications: vegetation, built-up, informal settlements, impervious surfaces (roads/highways, streets, parking lots, road-like areas around buildings, etc.), barren, water, and a small number of unclassified pixels. The exact pixel distribution for the training set is given in table 1.3.

There are a total of 110 600×600 images with manually labelled segmentation masks, split into 120×120 patches with 50% horizontal and vertical overlap for training, validation, and testing. The authors provide information about class distribution in their paper exploring semantic segmentation of the dataset, shown in Figure 2.

2. Methods

Ronneberger, Fischer, and Brox proposed the first U-Net architecture to segment cells in microscopic images, shown in Figure 3. The symmetric model consists of a contracting path, or encoder, on the left half and an expanding path, or decoder, on the right half. In the encoder, each level applies a 3×3 convolution with ReLU activation before a 2×2 max pooling operation is applied with stride 2 which reduces spatial dimensions by half. By doing this, the encoder is learning increasingly abstract representations of the original image while downsampling for computational efficiency. The decoder starts with the lowest spatial resolution, most abstract features and performs a series of 2×2 upconvolutions which doubles the spatial dimensions. Importantly, a skip connection from the encoder of equivalent spatial dimension to each decoder layer is included to preserve spatial information lost during downsampling. The

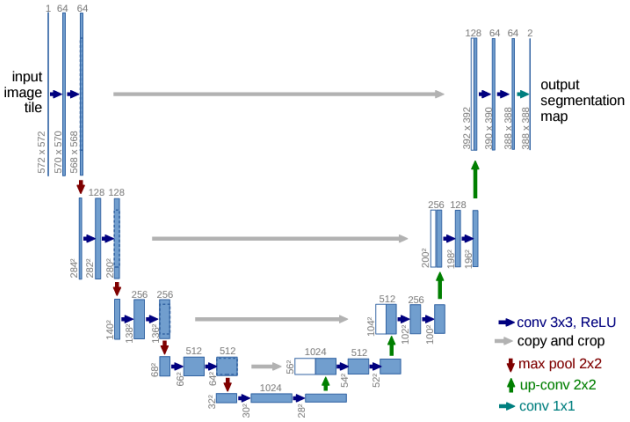

Figure 3. The Original U-Net Architecture[9]

final output layer performs a 1×1 convolution that reduces the number of channels to the number of classes. In this project, I explore the usage of the U-Net architecture for the semantic segmentation of aerial images.

2.1. Loss Function

As with most classification networks, the output of our semantic segmentation model at each pixel is a vector of probabilities $\mathbf{p}$, where each probability for class k is given by the softmax of the activations of each class input,

$$\mathbf{p}_k(x, y) = \frac{\exp\{a_k(x, y)\}}{\sum_{k'=1}^{|\mathbf{k}|} \exp\{a_{k'}(x, y)\}} \tag{5}$$

where $a_k(x, y)$ is the activation of class k at pixel (x, y) and $|k|$ is the number of classes; in our case, $|\mathbf{k}| = 6$. Then, $\mathbf{p}_k(x, y)$ can be interpreted as the probability that pixel (x, y) belongs to class k.

The most common loss function for semantic segmentation is pixel-wise cross-entropy, defined as

$$\mathcal{L}_{ce} = \sum_{(x,y)\in I} \log(\mathbf{p}(x, y)) \cdot \mathbf{y}(x, y) \tag{6}$$

where $\mathbf{y}(x, y)$ is a one-hot encoded vector of the true class of pixel (x, y).

Real-world semantic segmentation datasets are often class-imbalanced, leading to issues with basic cross-entropy loss wherein the network is biased toward majority classes. To combat this, class-specific weights are often introduced to the loss function, often derived from the original data statistics [3]. Such a cost-sensitive loss function can be seen in the original proposal where the authors introduce a weighting function for each pixel that both balanced the classes and emphasized learning separation borders between cells [9]. Here, the loss function is of the form

$$\mathcal{L}_{wce} = \sum_{(x,y)\in I} w(x, y) \log(\mathbf{p}(x, y)) \cdot \mathbf{y}(x, y) \tag{7}$$

Class	Informal Settlements	Built-Up	Impervious Surfaces	Vegetation	Barren	Water	Unclassified
# pixels	12921604	11358632	13436578	22423411	18735038	37789523	120366

Table 1. Training Patch Pixel Distribution

where w is a pre-computed function of (x, y).

2.2. Data Augmentation

The image patches are first upsampled to a resolution of 128×128 to ensure they are compatible with the chosen network architecture, which involves a series of convolutional and pooling operations. This particular size is important because the network utilizes skip-connections between the encoder and decoder layers, and it includes five sequential downsampling stages. Each downsampling stage reduces both the height and width of the input by a factor of two, and since $2^5 = 32$, the input dimensions must be multiples of 32 to avoid boundary issues or the need for cropping. By setting the image patches to 128×128, we ensure smooth downsampling at every stage, maintaining feature alignment between the encoder and decoder pathways for effective information transfer.

To further enhance the training process and reduce the risk of overfitting, at the start of each training epoch, the original image patch and its associated mask are randomly rotated by a multiple of $90°$. This approach adds rotational invariance to the model's learned features and expands the effective size of the training dataset, helping the network generalize better to novel samples and preventing it from simply memorizing the training images.

2.3. Model Selection

As U-Nets are common for semantic segmentation masks, pre-trained networks are available. In this project, I use the Segmentation Models Pytorch library[6], which contains various pre-trained machine learning models for segmentation. The EfficientNet-B0[11] network pre-trained on the ImageNet dataset[5] was chosen as the encoder for this project because of its relatively small size (4M parameters), limiting the potential for overfitting and allowing for training with limited computational resources.

2.4. Optimizer and Learning Rate

For fine-tuning the U-Net architecture, I used the Adam optimizer with an initial learning rate of 1×10^{-4} as suggested by Dabra[4]. Adam is particularly well-suited for this application due to its ability to adaptively adjust the learning rates of individual parameters. The chosen learning rate of 1×10^{-4} strikes a balance between making substantial progress during training and maintaining stability, thereby preventing the optimizer from overshooting minima.

To further encourage smooth training integrate a learning rate scheduler using `LambdaLR`, defined by the function

$$\lambda(\text{epoch}) = 0.1^{\frac{\text{epoch}}{40}} [4] \tag{8}$$

This scheduler systematically reduces the learning rate as training progresses, implementing an exponential decay strategy. Such a decay is beneficial for fine-tuning as it allows the model to make large updates during the initial phases of training when significant adjustments are needed, and smaller, more precise updates in later stages to refine the learned features. This gradual reduction in the learning rate helps in escaping local minima and promotes the stabilization of the training process, leading to improved convergence and enhanced generalization of the U-Net model. Collectively, the combination of the Adam optimizer with a carefully scheduled learning rate ensures that the U-Net efficiently learns robust feature representations, resulting in accurate and reliable semantic segmentation.

2.5. Early Stopping

To mitigate the risks of overfitting and the excessive consumption of computational resources during model training, we incorporate an early stopping mechanism in our training pipeline. Early stopping serves as a regularization technique by monitoring the model's performance on a separate validation dataset and halting the training process when no significant improvement is observed over a predefined number of epochs. Specifically, in my implementation, we track the validation loss at each training epoch and terminate the training if the validation loss does not decrease for five consecutive epochs.

3. Results

3.1. Model 1

The initial model is trained exactly as specified in section 2. This serves as a strong foundation for comparison with the other models to better understand the impact of their respective changes.

3.2. Model 2

This model introduces dice loss alongside cross-entropy loss to better address class imbalance by directly optimizing for overlap between predicted and ground truth masks. This combined loss function improves segmentation for underrepresented classes, potentially making it more robust.

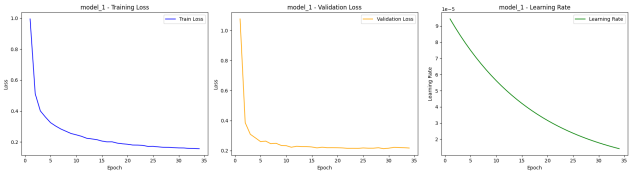

Figure 4. Training Info for Model 1. All other training figures appear similar and are omitted, except for Model 7.

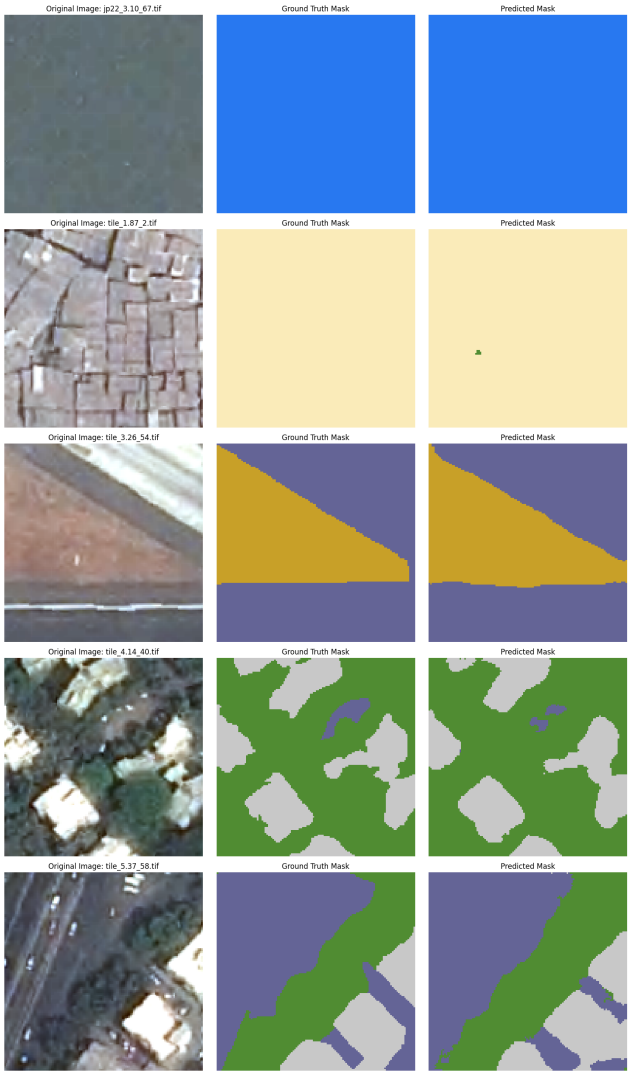

Figure 5. Model 1 Output for Interesting Patches

3.3. Model 3

To further handle class imbalance, class-weighted cross-entropy loss is used with the dice loss. The weights are computed based on pixel frequency, ensuring that rare classes contribute more to the total loss. The weights provided as $w(x, y)$ in equation 7 were calculated

$$w_k = \frac{1}{\text{class } k \text{ pixel count}/\text{total pixel count}} \tag{9}$$

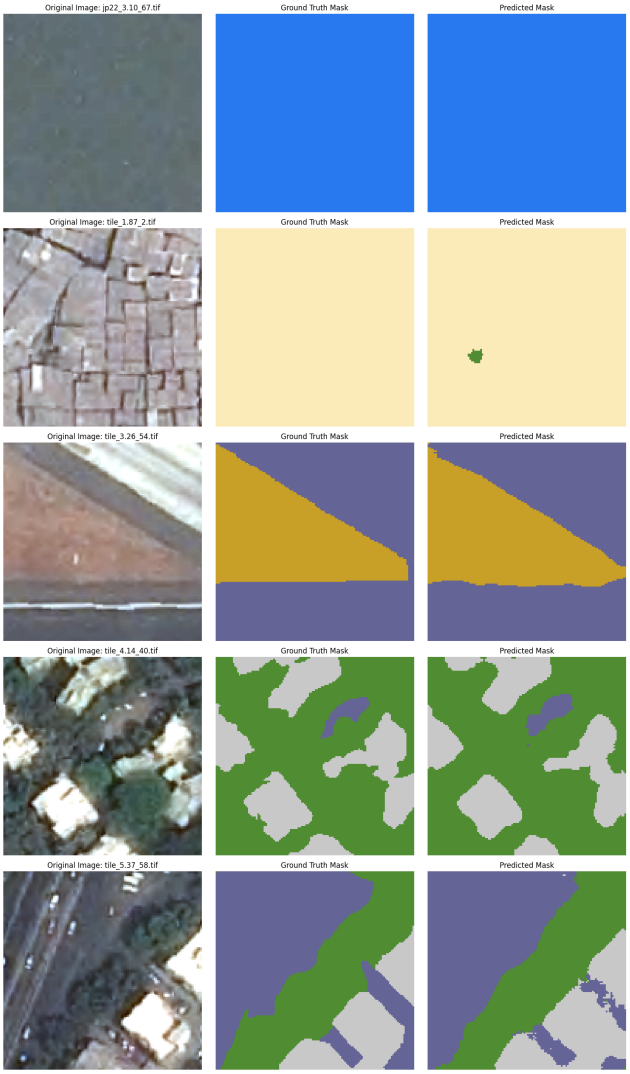

Figure 6. Model 2 Output for Interesting Patches

and then normalized so that all weights sum to 1. Best results were achieved by manually scaling down the weight of the unclassified pixels before normalization as it became too dominant otherwise. The scaling factor used was 0.001. Notice that unlike the original paper, this weighting function only depends on class and not location.

3.4. Model 4

A shallower U-Net architecture is implemented by reducing the encoder depth to 4 and customizing the decoder channels to [256, 128, 64, 32]. This modification reduces computational overhead, making the model faster while maintaining reasonable performance on smaller input patches (128x128).

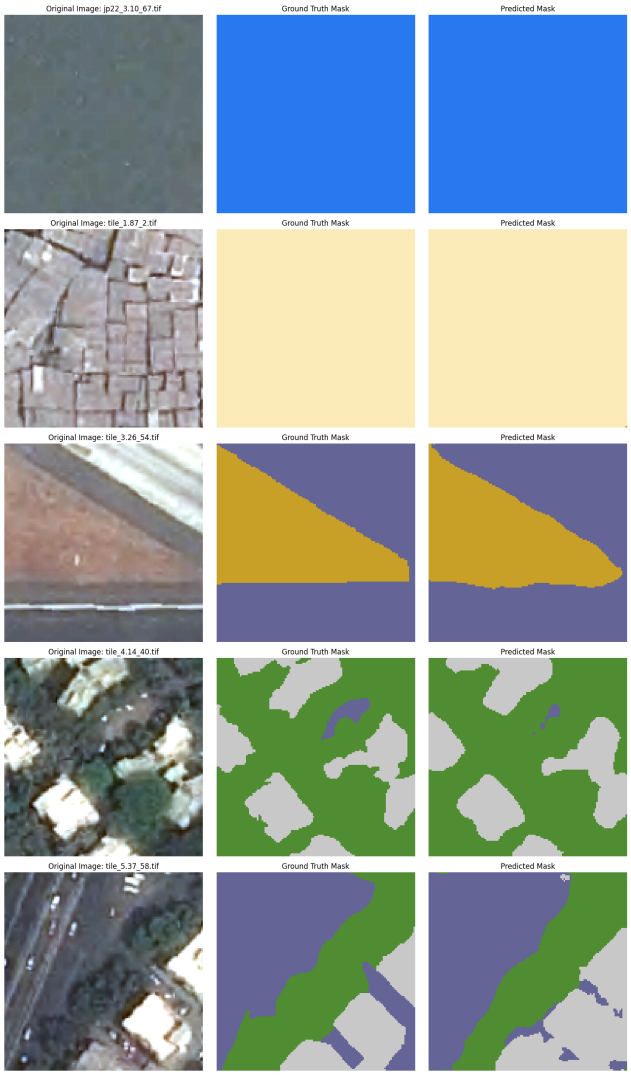

Figure 7. Model 3 Output for Interesting Patches

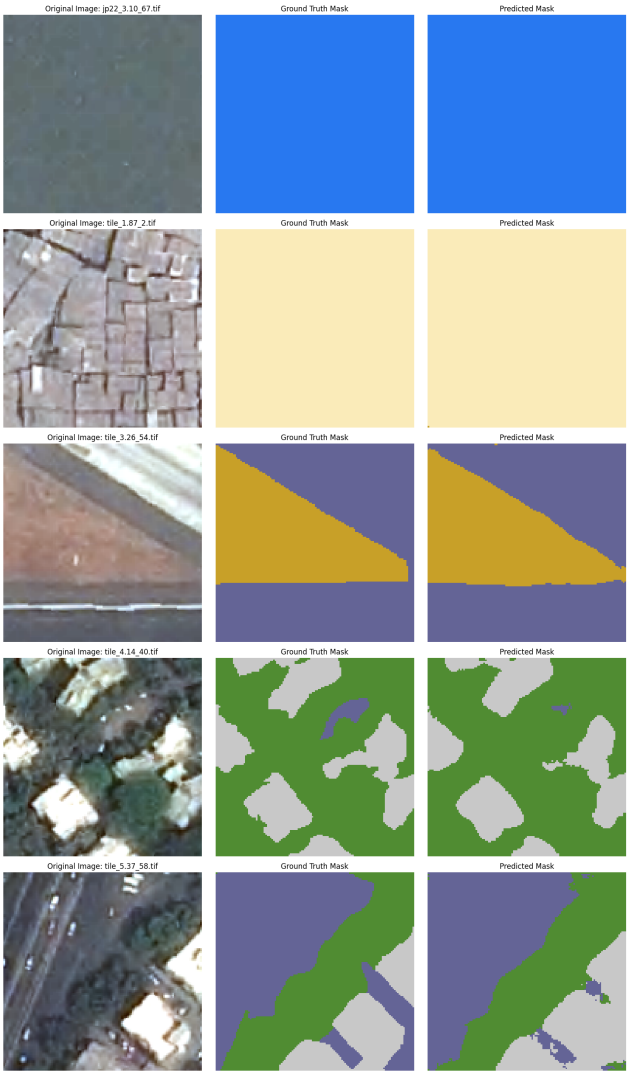

Figure 8. Model 4 Output for Interesting Patches

3.5. Model 5

This version incorporates dropout regularization (with probability 0.2) within the decoder to improve generalization and reduce overfitting. By introducing stochastic regularization, the model learns more robust features, especially useful for smaller or noisy datasets like this one.

3.6. Model 6

Attention mechanisms are added with Squeeze-and-Excitation (SCSE) blocks in the decoder, as outlined in Roy, Navab, and Wachinger [10]. SCSE blocks recalibrate feature responses by adaptively weighting each channel through a squeeze operation (global average pooling) followed by an excitation step using fully connected layers and sigmoid activation. This mechanism allows the network to emphasize important features while suppressing irrelevant ones, enhancing the model's focus on key regions of the feature maps.

3.7. Model 7

This model integrates key ideas from previous versions, including SCSE attention blocks, class-weighted loss, and dice loss, while introducing several unique training features. The loss function combines a weighted focal loss,

$$\mathcal{L}_{wfl} = \sum_{(x,y)\in I} \sum_{c=1}^{|\mathbf{k}|} w_c \cdot (1-\mathbf{p}_c(x,y))^{\gamma} \cdot \log(\mathbf{p}_c(x,y)) \cdot \mathbf{y}_c(x,y) \tag{10}$$

where γ is the focusing parameter which controls the down-weighting of well-classified pixels, designed to handle class imbalance and prioritize difficult samples with dice loss for improved segmentation overlap. Additionally, the model

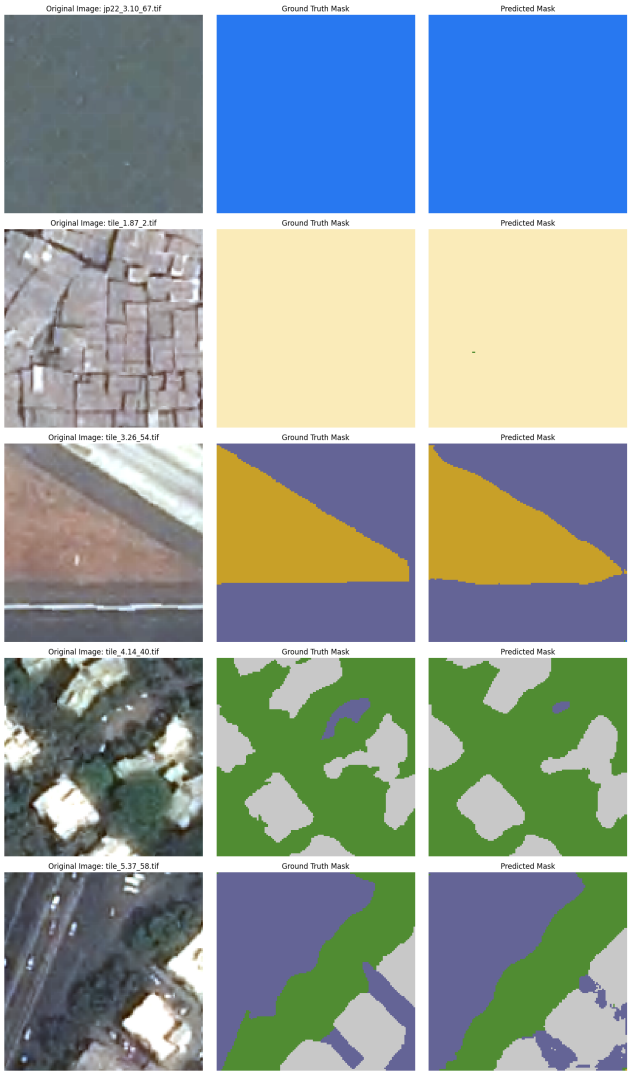

Figure 9. Model 5 Output for Interesting Patches

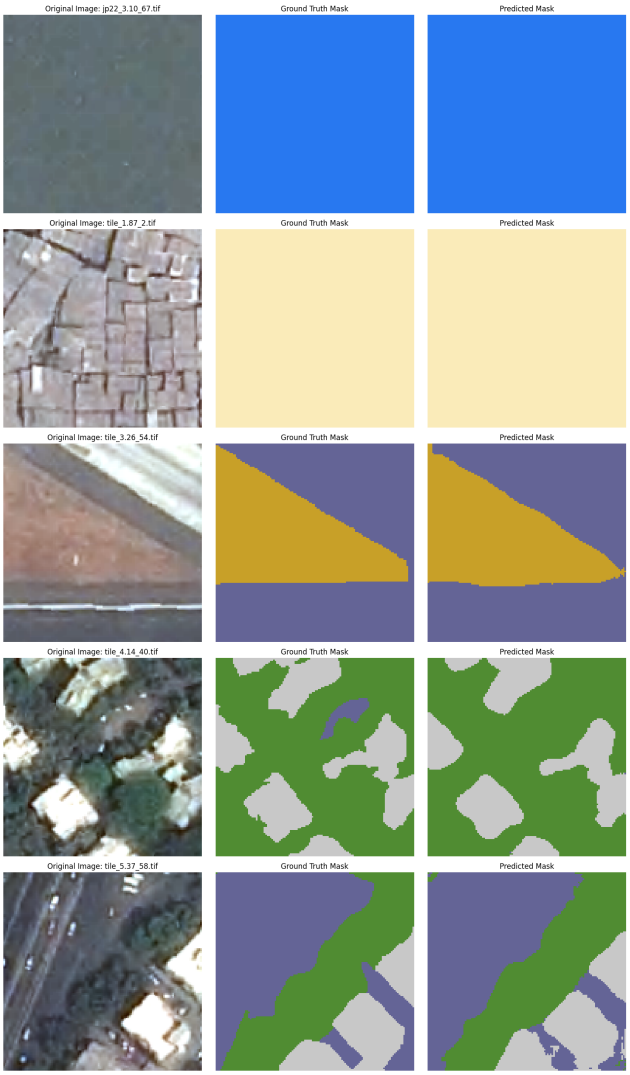

Figure 10. Model 6 Output for Interesting Patches

employs the `AdamW` optimizer, which includes decoupled weight decay for better regularization. A cosine annealing scheduler with warm restarts is used, enabling cyclical learning rate adjustments that help the model escape local minima and refine learning in later stages.

3.8. Model Comparison

Based on the results, Model 1 stands out as the best-performing model overall. It consistently achieves the highest Dice and IoU scores for critical classes, while maintaining strong performance across other classes. Its superior performance is reflected in Figure 13, where it leads in Dice, Jaccard, Precision, and Recall. This balanced and reliable performance highlights the model's ability to capture both fine and coarse details, making it robust for various segmentation tasks. Model 4 closely follows, excelling particularly in Class 1 and Class 3, where it outperforms other models.

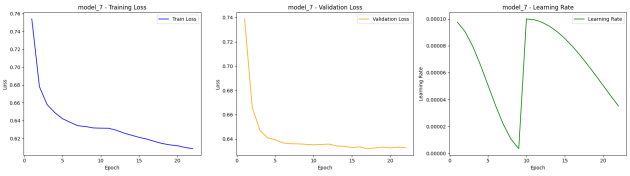

Figure 11. Training Info for Model 7

The optimized decoder channel structure allows Model 4 to extract and refine features effectively, making it highly competitive.

Model 6 also demonstrates strong performance, particularly in Class 3 and Class 5, where its inclusion of SCSE attention blocks helps improve feature focus and segmentation quality. While it falls slightly behind Models 1 and 4 in certain classes, it remains a reliable model with strong average Dice and IoU scores. On the other hand, Model 2

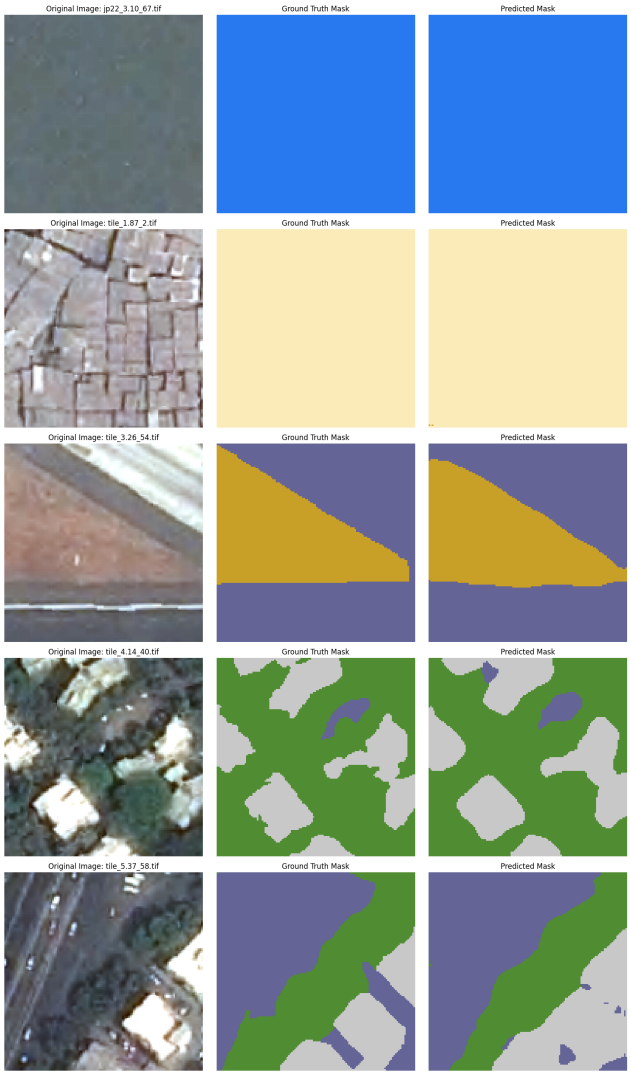

Figure 12. Model 7 Output for Interesting Patches

and Model 7 underperform.

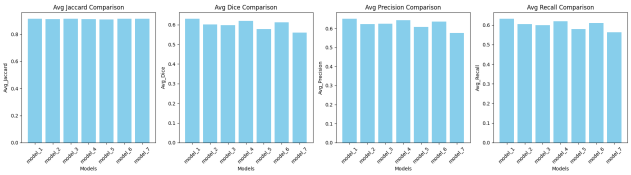

Figure 13. Overall Comparison

4. Conclusion

This work highlights the strengths and trade-offs of various modifications to the U-Net architecture for aerial image segmentation. Model 1, despite being the most "basic," was the most balanced and highest-performing model across all evaluation metrics, maintaining strong Dice and IoU scores

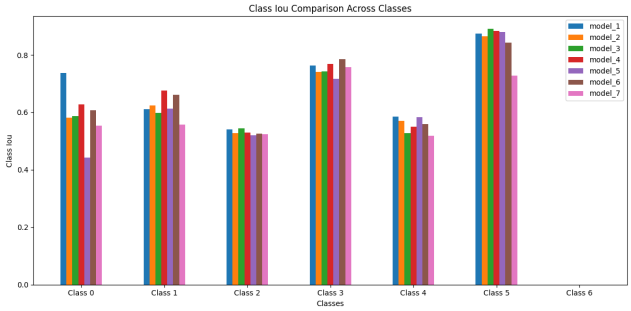

Figure 14. Class-Wise IoU Comparison

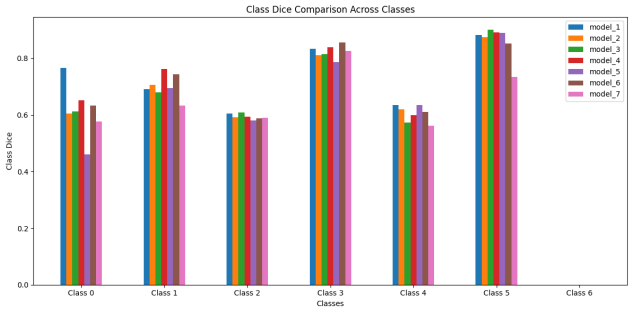

Figure 15. Class-Wise Dice Comparison

for critical classes like water and vegetation. However, targeted improvements in Model 4 (reduced encoder depth and customized decoder channels) demonstrate that computational efficiency can be achieved without substantial loss in accuracy, particularly for well-structured classes like built-up areas. Similarly, Model 6 showcases the value of attention mechanisms in refining feature focus for complex regions, improving segmentation in challenging classes like informal settlements.

While progress is evident, limitations remain, including the inability to capture fine-detailed boundaries and significant class imbalances. A simple improvement may be to use an ensamble of U-Nets [7]. For example, combining the general capabilities of model 1 with the class-wise strengths of models 4 and 6. More radically different potential improvements are laid out in section 4.1.

4.1. Future Work

While this project showed promising results, improvements can certainly be made. The models seemed to struggle with fine-detailed class boundaries, for example, a jagged settlement bordering a patch of vegetation. To enhance the precision of these boundaries, future work could explore incorporating higher-resolution input data, which would provide more detailed information for the model to learn from. Additionally, implementing multi-scale feature extraction techniques, such as feature pyramids, could help the model capture both global and local context more effectively. Another promising approach is the integration of

Conditional Random Fields as a post-processing step to refine segmentation edges by considering spatial dependencies and contextual relationships between pixels. Employing these methods could lead to smoother and more accurate delineations of classes with intricate boundaries and improve overall segmentation performance.

Class imbalance proved to be an issue throughout the project, with classes like water dominating the optimization problem. While class-balancing was attempted through loss function weighting, more methods exist to address this challenge. Future work could investigate advanced strategies such as Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples for underrepresented classes, thereby increasing their presence in the training dataset. Another potential approach is the use of data augmentation techniques such as geometric transformations or color jittering to artificially enhance the diversity and quantity of these classes. Exploring these methods could lead to a more balanced training process and improve the model's ability to accurately segment all classes.

I chose a relatively basic encoder given time and compute constraints, but more advanced encoders such as ResNet or Vision Transformers (ViT) could be integrated to enhance feature extraction capabilities. Vision Transformers, which leverage self-attention mechanisms to capture long-range dependencies within the image, might enable the model to better understand complex spatial relationships, leading to more accurate semantic segmentation.

Finally, it would be useful to explore other state-of-the-art models for semantic segmentation beyond the U-Net. Architectures such as DeepLabv3+, PSPNet (Pyramid Scene Parsing Network), and Mask R-CNN offer alternative approaches that incorporate advanced techniques like atrous convolutions, pyramid pooling modules, and instance segmentation capabilities. For example, DeepLabv3+ utilizes atrous spatial pyramid pooling to capture multi-scale contextual information, which can improve the segmentation of objects at different sizes. PSPNet's pyramid pooling module effectively aggregates global and local context, enhancing the model's ability to understand complex scenes. Mask R-CNN extends the capabilities of object detection frameworks to perform instance segmentation, allowing for more precise delineation of objects within an image.

References

[1] Ayush Dabra. Manually annotated high resolution satellite image dataset of mumbai for semantic segmentation, 2023. Available: https://data.mendeley.com/datasets/xj2v49zt26/1. 1

[2] P. Baumann. History of remote sensing, aerial photography. http://employees.oneonta.edu/baumanpr/geosat2/rs%20history%20i/rs-history-part-1.htm. Accessed: Dec. 8, 2024. 1

[3] G. Csurka, R. Volpi, and B. Chidlovskii. Semantic image segmentation: Two decades of research, 2023. 2

[4] A. Dabra and V. Kumar. Evaluating green cover and open spaces in informal settlements of mumbai using deep learning. *Neural Computing and Applications*, Feb. 2023. 2, 3

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 3

[6] P. Iakubovskii. Segmentation models pytorch. https://github.com/qubvel/segmentation_models.pytorch, 2019. 3

[7] D. Marmanis, J. D. Wegner, S. Galliani, K. Schindler, M. Datcu, and U. Stilla. Semantic segmentation of aerial images with an ensemble of cnns. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-3:473–480, 2016. 7

[8] National Air and Space Museum. The beginnings and basics of aerial photography. https://airandspace.si.edu/stories/editorial/beginnings-and-basics-aerial-photography. Accessed: Dec. 8, 2024. 1

[9] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 2

[10] A. G. Roy, N. Navab, and C. Wachinger. Recalibrating fully convolutional networks with spatial and channel 'squeeze excitation' blocks, 2018. 5

[11] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020. 3

[12] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023. https://D2L.ai. 2