Semantic Segmentation of Aerial Imagery

H S Jayanth
231IT024, Information Technology
National Institute of Technology Karnataka
Surathkal, India
hsjayanth.231it024@nitk.edu.in

Shivam Kumar A
231IT068, Information Technology
National Institute of Technology Karnataka
Surathkal, India
shivamkumara.231it068@nitk.edu.in

***Abstract*—This project introduces a U-Net-based algorithm for efficient semantic segmentation of aerial imagery to accurately classify each pixel with its corresponding object class. The model exploits the encoder-decoder structure of the U-Net architecture, providing effective feature extraction and spatial localization. It is suitable for the segmentation of high-resolution aerial imagery. Our approach takes particular preprocessing techniques like image normalization, contrast enhancement, and tiling into account due to the aerial data challenges, for instance, with different object scales, and complex backgrounds. Data augmentation strategies are adopted to provide model robustness in diverse terrains and conditions. Additionally, techniques for further boosting model accuracy by boundary-focused approaches, especially custom loss functions for improved boundary localization and better ability to separate classes with close positions, are used. The post-processing methods include morphological operations and conditional random fields, which further reduce noise and enhance coherence in the segmentation output. The fine-grained, labeled maps produced by our system have potential applications in urban planning, environmental monitoring, disaster response, etc. Thus, this research has demonstrated the adequacy of U-Net for domain-specific aerial imagery segmentation as it produces high accuracy pixel-wise classifications for real-world application-driven scenarios.**

***Index Terms*—semantic segmentation, unet, aerial imagery**

I. Introduction

Semantic segmentation is that important computer vision task of classifying each pixel in an image relative to predefined categories. Contrary to this traditional method of image classification, where only one label is assigned to the entire image, semantic segmentation focuses more on the spatial structure and content by labeling every pixel individually. This pixel-wise classification enables the model to outline complex scenes at high granularity. It is very useful in applications where the precise localization of different objects in an image is required.

Semantic segmentation finds quite widespread applications where detailed scene understanding is involved. In autonomous driving, it helps vehicles identify and distinguish roads, pedestrians, cars, and even other important elements in real-time, further enhancing navigation safety. Semantic segmentation is essential in medical imaging to identify and delineate tissues, organs, or pathological regions, which can be critical for diagnostics and treatment planning. Applications also include looking for classification of land use through aerial imagery analysis for urban planning or environmental monitoring and in industrial settings for quality control and defect detection.

On the other hand, semantic segmentation informs about locations and boundaries of objects with fine-grained detail - an important contribution towards progress in automation, diagnostics, and intelligent decision-making. Thus, it is also an important tool in research and practical deployments.

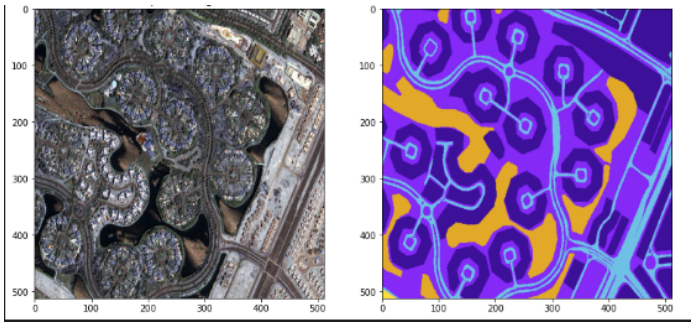

Fig. 1. Semantic Segmentation Example

II. Literature Review

This has been of immense growth in changing the paradigm of dense pixel-level predictions in complex scenes. Basic pioneering work, such as Fully Convolutional Networks for Semantic Segmentation by Jonathan Long, Evan Shelhamer, and Trevor Darrell in 2015, made CNNs applicable for pixel-wise labeling by replacing fully connected layers with convolutional layers. Ever since, the architectures have moved considerably forward, and applications like U-Net by Olaf Ronneberger et al. (2015) and DeepLab by Liang-Chieh Chen et al. (2017), which proposed novel ways to retain spatial detail at affordable processing costs in high-resolution images, have changed the trend to utilize multi-scale context to increase the accuracy of segmentation in densely populated and intricate scenes. However, class imbalance is the primary challenge in this regard. In low-sample classes, the predictions tend to be biased in favor of more frequent classes. Researchers addressed this issue by adopting class-weighted loss functions and sampling methods along with specialized loss functions in a way that tries to ensure a balanced representation across all classes. Another significant problem is that of precise boundary localization. This calls for accurate edge detection, and methods like Boundary-Aware Segmentation Network (BASNet) of Qin et al. (2019) have boosted this task by incor-

porating edge-aware loss functions and hybrid segmentation-boundary detection models to better capture object boundaries.

Handling objects of varying sizes in an image has led to multi-scale feature extraction techniques. Spatial Pyramid Pooling, which was first introduced in the SPP-Net by He et al. in 2014 and then applied to Deeplabv3 + by Chen et al. in 2018 allows models to see the object at multiple scales without necessarily having to pay this computational cost of subdivision. In response to this, lightweight architectures like MobileNetV3 (Howard et al., 2019) and EfficientNet (Tan and Le, 2019) attempt to make reductions in the computational burden by making real-time deployment feasible.

Another area of focus is domain adaptation and generalization. This is particularly important when such models trained on synthetic or controlled datasets need to robustly work in real-world conditions. Techniques like adversarial training, such as in Domain-Adversarial Neural Networks (Ganin et al., 2016), as well as self-supervised learning approaches have been invaluable for improving model adaptability across domains, reducing dependence on large labeled datasets, and helping improve robustness in varied environments.

Semantic segmentation is one of those fields of research that vibrantly continues to be active today. Researchers continue to push the frontiers of accuracy, computational efficiency, and robustness by designing architectures and loss functions dedicated to specific application needs and even broader, more general requirements of deployment. More methods are continued to be researched in efforts to attain superior performance across multiple, challenging visual domains.

A. Challenges

- One of the primary challenges in image segmentation lies in the different levels of granularity with which computers interpret images, as the classification must be performed for each pixel. This pixel-wise classification results in detailed but computationally intense analyses, which can be particularly demanding for high-resolution images.
- In pixel-wise analysis, overlapping patches are often processed multiple times, leading to redundant information. This not only makes the model less efficient but also increases computational load. Furthermore, the training process is often lengthy and resource-intensive, requiring substantial computational power and time, which can be a significant drawback in real-world applications, especially where quick turnaround is necessary.

B. Resolution

- To reduce redundancy and computational load, patch-based training can be employed. Instead of analyzing the entire image at once, the model can focus on smaller, non-overlapping patches, which reduces data overlap and minimizes redundant processing. This approach is particularly helpful in handling high-resolution images more efficiently.
- Another approach is to optimize the model's structure by reducing the number of parameters and fine-tuning hyperparameters to balance accuracy and efficiency. By carefully selecting model depth, filter sizes, and training parameters, the model can become more lightweight, maintaining a high level of performance while reducing computational requirements and making real-time or resource-constrained deployments more feasible.

III. Project Goals

A. Preprocessing

Design a preprocessing algorithm for the data, and try out a variety of models on a small dataset and record the outcomes.

B. Model Training

Design a final model that is trained on the full dataset, and is designed to optimize accuracy while also constraining the model size.

IV. Project Methodologies

A. Preprocessing Methodolgy

For this goal, we will be familiarizing ourselves with the dataset, and performing preprocessing on the dataset including patchifying and using color masks, before passing it on the model. We will also be researching the model architecture, trying out different architectures and hyperparameters, and run different models on a small toy dataset, and record the outcomes.

B. Model Training Methodology

For this goal, we will be finalizing the model based on the results and the differences. We will be looking at how to implement this model on a bigger dataset, and then train the model on a bigger dataset, which includes preprocessing before. We will also be recording the results of the model and documenting the process. Main focus will be on increasing the accuracy, and analyzing our final results.

V. Dataset Used

The main dataset used is the following:
Link to Dataset
The dataset consists of eight folders namely "Tile 1 "to "Tile 8 ", each of which contains images and masks.
First, since we were trying out what model configuration to use, we only used the images from the "Tile 8 "folder within the dataset. Later, the model was trained on the entire dataset.

VI. Implementation of Preprocessing

A. Preprocessing

Here are the techniques that we applied to the images before they were passed on to the model:

1) Patchifying: The images are of varying sizes, and the size of the images are too big for the model to run properly. Hence, the images are patched. This includes creating multiple images from a single image by cropping multiple sections of the image into images of size 256*256. This not only makes the images more suitable for training and testing as they are of a smaller size, but also increases the number of images for training and testing, thus increasing the accuracy of the model. This is also a form of data augmentation as the model only focuses on a small part of the image, thus also reducing overfitting.

2) Scaling: Bigger values such as [0,255] usually do not perform well under Neural Networks as they do not converge quickly, sometimes not at all. Input functions like sigmoid and tanh are also sensitive to the scale, so the best method is to normalize/scale all the color values. Scaling also regularizes the model, making sure that all values contribute better to the final output.

3) Randomizing: Since the preprocessing includes patchifying, patches of the same image are bound to appear together. This can induce bias while the model learns, as similar looking images tell the model that the images look a certain way, which is not the case in general. Hence, randomizing the order of the patched and normalized patches helps the model learn better and avoid the bias induced by the order of data.

4) Converting Masks Pixels to Numbers: The mask pixels are encoded in RGB, one value for each of the class. Now the model wants to treat each pixel as a categorical classification problem so now the numbers are encoded in the numerical format. This helps application of layers like Softmax easier.

B. Implementation

We tried mainly three models on the smaller dataset with slightly different architectures to see which of them would fit the model the best. We mainly used Tensorflow Keras to implement this. The model architectures and hyperparameter details are given in the section following this, along with other details.

C. Hyperparameters and Custom Functions

Apart from the model architectures, here are some common things found in all the experiments:

1) Custom Loss Functions: Custom loss function, that is specific to the field of semantic segmentation, was used. This loss function is a combination of dice loss (with smoothness 1e-6) and focal loss ($\alpha = 0.25, \gamma = 2$)

2) Train-Test Split: The training and test split of the data was 0.8 and 0.2 respectively.

3) Custom Metric: Custom metric, that is specific to the field of semantic segmentation, was used along with accuracy, which is a standard metric. The custom metric was Jaccard coefficient, which measures how similar the predicted and the actual outputs are. The higher the Jaccard coefficient is, the better the model performs.

4) Batch Size: A batch size of 32 was used.

5) Optimizer: ADAM optimizer was used with default parameters available on Tensorflow (those are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$).

D. Model Architectures

Here, nClasses refers to the number of classes.
All layers have a ReLU activation function.

1) Model 1: Input Layer - Conv2D (16 filters, 3x3) - Conv2D (16 filters, 3x3) - MaxPooling2D (2x2) - Conv2D (32 filters, 3x3) - Conv2D (32 filters, 3x3) - MaxPooling2D (2x2) - Conv2D (64 filters, 3x3) - Conv2D (64 filters, 3x3) - MaxPooling2D (2x2) - Conv2D (128 filters, 3x3) - Conv2D (128 filters, 3x3) - MaxPooling2D (2x2) - Conv2D (256 filters, 3x3) - Conv2D (256 filters, 3x3) - Conv2DTranspose (128 filters, 2x2, strides 2) - Concatenate - Conv2D (128 filters, 3x3) - Conv2D (128 filters, 3x3) - Conv2DTranspose (64 filters, 2x2, strides 2) - Concatenate - Conv2D (64 filters, 3x3) - Conv2D (64 filters, 3x3) - Conv2DTranspose (32 filters, 2x2, strides 2) - Concatenate - Conv2D (32 filters, 3x3) - Conv2D (32 filters, 3x3) - Conv2DTranspose (16 filters, 2x2, strides 2) - Concatenate - Conv2D (16 filters, 3x3) - Conv2D (16 filters, 3x3) - Conv2D (nClasses filters, 1x1, softmax activation)

2) Model 2: Input Layer - Conv2D (16 filters, 3x3) - Dropout (0.2) - Conv2D (16 filters, 3x3) - MaxPooling2D (2x2) - Conv2D (32 filters, 3x3) - Dropout (0.2) - Conv2D (32 filters, 3x3) - MaxPooling2D (2x2) - Conv2D (64 filters, 3x3) - Dropout (0.2) - Conv2D (64 filters, 3x3) - MaxPooling2D (2x2) - Conv2D (128 filters, 3x3) - Dropout (0.2) - Conv2D (128 filters, 3x3) - MaxPooling2D (2x2) - Conv2D (256 filters, 3x3) - Dropout (0.3) - Conv2D (256 filters, 3x3) - Conv2DTranspose (128 filters, 2x2, strides 2) - Concatenate - Conv2D (128 filters, 3x3) - Dropout (0.2) - Conv2D (128 filters, 3x3) - Conv2DTranspose (64 filters, 2x2, strides 2) - Concatenate - Conv2D (64 filters, 3x3) - Dropout (0.2) - Conv2D (64 filters, 3x3) - Conv2DTranspose (32 filters, 2x2, strides 2) - Concatenate - Conv2D (32 filters, 3x3) - Dropout (0.2) - Conv2D (32 filters, 3x3) - Conv2DTranspose (16 filters, 2x2, strides 2) - Concatenate - Conv2D (16 filters, 3x3) - Dropout (0.2) - Conv2D (16 filters, 3x3) - Conv2D (nClasses filters, 1x1, softmax activation)

3) Model 3: Input Layer - Conv2D (16 filters, 3x3) - Conv2D (16 filters, 3x3) - AveragePooling2D (2x2) - Conv2D (32 filters, 3x3) - Conv2D (32 filters, 3x3) - AveragePooling2D (2x2) - Conv2D (64 filters, 3x3) - Conv2D (64 filters, 3x3) - AveragePooling2D (2x2) - Conv2D (128 filters, 3x3) - Conv2D (128 filters, 3x3) - AveragePooling2D (2x2) - Conv2D (256 filters, 3x3) - Conv2D (256 filters, 3x3) - Conv2DTranspose (128 filters, 2x2, strides 2) - Concatenate - Conv2D (128 filters, 3x3) - Conv2D (128 filters, 3x3) - Conv2DTranspose (64 filters, 2x2, strides 2) - Concatenate - Conv2D (64 filters, 3x3) - Conv2D (64 filters, 3x3) - Conv2DTranspose (32 filters, 2x2, strides 2) - Concatenate - Conv2D (32 filters, 3x3) - Conv2D (32 filters, 3x3) -

Conv2DTranspose (16 filters, 2x2, strides 2) - Concatenate - Conv2D (16 filters, 3x3) - Conv2D (16 filters, 3x3) - Conv2D (nClasses filters, 1x1, softmax activation)

E. Training

The models were trained for 50 epochs on P100 GPUs available on Kaggle.

F. Results

The performance of the three models was evaluated using Mean Intersection over Union (Mean IoU), a common metric in semantic segmentation that measures the overlap between predicted and ground-truth regions. Each model was trained for 50 epochs, and the results are summarized in the table.

Model Used	Mean IoU
Model 1	0.671
Model 2	0.579
Model 3	0.652

TABLE I
RESULTS OF THE EXPERIMENT

G. Observations

From the table, it is evident that Model 1 achieved the highest Mean IoU of 0.671, while Model 2 performed the lowest at 0.579, likely due to the added dropout layers, which seemed to induce greater bias. Model 3, which utilized average pooling instead of max pooling, performed moderately well with a Mean IoU of 0.652, though it slightly underperformed compared to Model 1.

The graphical results show the progression of key metrics, including loss and Mean IoU, for all three models over the training epochs. Visual inspection of model outputs at a Mean IoU around 65% indicates that the segmentation quality aligns with expected outcomes, demonstrating clear boundaries for most regions of interest in the images.

The results suggest that the exclusion of dropout layers helps maintain model performance by reducing bias, especially in this specific application where segmentation boundaries are critical. Additionally, max pooling proves to be more effective than average pooling in enhancing segmentation quality, consistent with prior research in the field. Consequently, Model 1 stands out as the most promising candidate for further refinement and potential deployment.

H. Inference

After running each model for 50 epochs and testing on these models, we found that:

- A model with no dropout layers works out better than the ones with a dropout layer, as dropout layers are causing a lot of bias during training. At the current level of model architecture, which is considerably huge, making the model architecture even bigger and compensating it with dropout is not exactly feasible.
- A model with average pooling works in a similar way as one with max pooling, however, it is slightly inferior. Since generally, segmentation models use Max Pooling and are considered better, we will be using it.
- Mean IoU at values close to 65% provide images that look similar to the outputs visually.

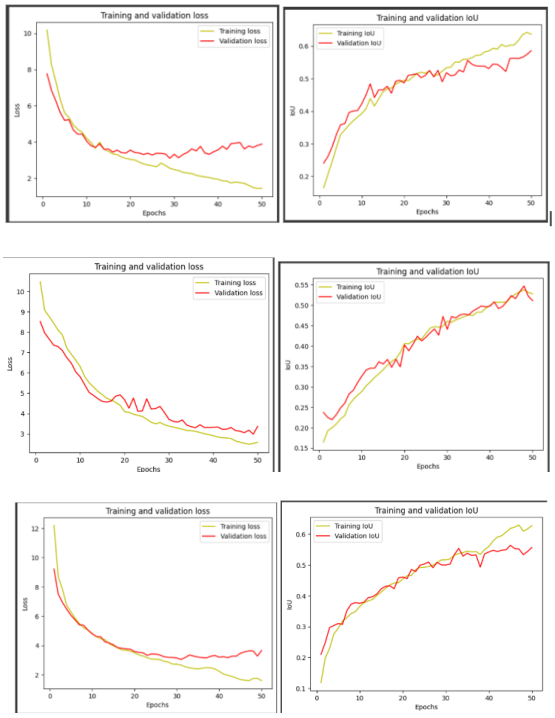

Fig. 2. Metrics of Models 1, 2 and 3.

VII. IMPLEMENTATION OF MODEL TRAINING

Based on the results from the implementation of the previous goal, we picked our best performing model, but we also did a couple of minor changes. One, we changed our activation functions from ReLU to LeakyReLU and that helped the accuracy. This model was trained over the complete dataset.

Here, nClasses refers to the number of classes.
All layers have a LeakyReLU activation function.

Input Layer - Conv2D (16 filters, 3x3) - Conv2D (16 filters, 3x3) - MaxPooling2D (2x2) - Conv2D (32 filters, 3x3) - Conv2D (32 filters, 3x3) - MaxPooling2D (2x2) - Conv2D (64 filters, 3x3) - Conv2D (64 filters, 3x3) - MaxPooling2D (2x2) - Conv2D (128 filters, 3x3) - Conv2D (128 filters, 3x3) - MaxPooling2D (2x2) - Conv2D (256 filters, 3x3) - Conv2D (256 filters, 3x3) - Conv2DTranspose (128 filters, 2x2, strides 2) - Concatenate - Conv2D (128 filters, 3x3) - Conv2D (128 filters, 3x3) - Conv2DTranspose (64 filters, 2x2, strides 2) - Concatenate - Conv2D (64 filters, 3x3) - Conv2D (64 filters, 3x3) - Conv2DTranspose (32 filters, 2x2, strides 2) - Concatenate - Conv2D (32 filters, 3x3) - Conv2D (32 filters, 3x3) - Conv2DTranspose (16 filters, 2x2, strides 2) - Concatenate - Conv2D (16 filters, 3x3) - Conv2D (16 filters, 3x3) - Conv2D (nClasses filters, 1x1, softmax activation)

A. Final Model

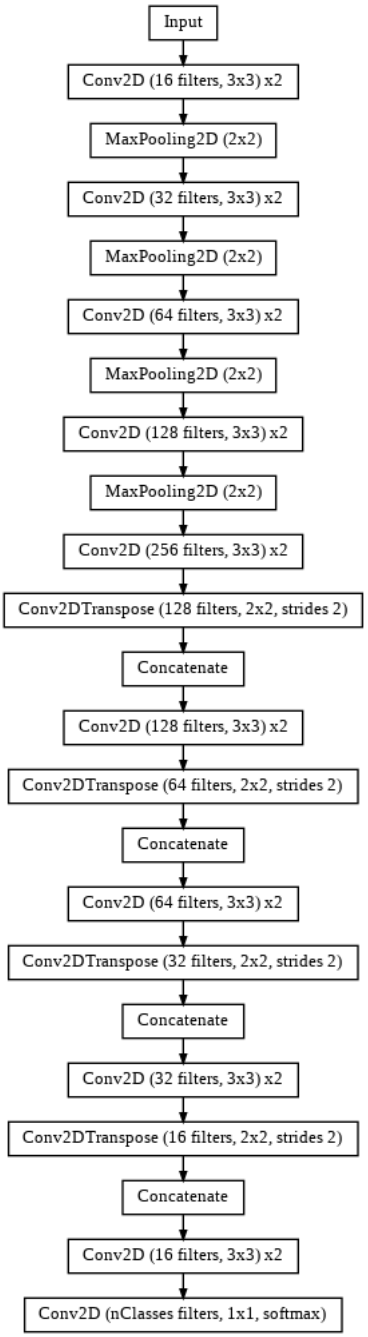

Fig. 3. Final Model Architecture

B. Hyperparameters and Custom Functions

Apart from the model architectures, here are some common things found in all the experiments:

1) Custom Loss Functions: Custom loss function, that is specific to the field of semantic segmentation, was used. This loss function is a combination of dice loss (with smoothness 1e-6) and focal loss ($\alpha = 0.25, \gamma = 2$)

2) Train-Test Split: The training and test split of the data was 0.8 and 0.2 respectively.

3) Custom Metric: Custom metric, that is specific to the field of semantic segmentation, was used along with accuracy, which is a standard metric. The custom metric was Jaccard coefficient, which measures how similar the predicted and the actual outputs are. The higher the Jaccard coefficient is, the better the model performs.

4) Batch Size: A batch size of 32 was used.

5) Optimizer: ADAM optimizer was used with default parameters available on Tensorflow (those are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$).

C. Training

This model was trained for 120 epochs on P100 GPUs available on Kaggle.

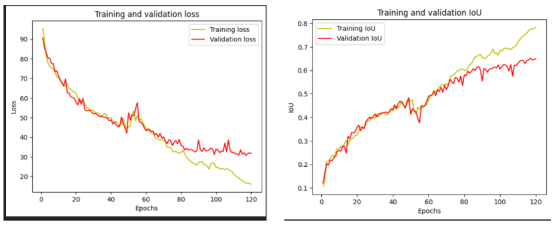

Fig. 4. Loss and metrics during training.

D. Results

The final model boasts an accuracy of 80-85%. Here are some of the images and the prediction of those images using the model we trained:

E. Inference

We experimentally analyzed different variations of U-Net-based models for semantic segmentation of aerial images and found that there are a couple of architectural choices that significantly affect performance. First, we noticed that dropping the dropout layer gives slightly better results since the dropout introduces unnecessary bias for an architecture like ours. We can find slightly better segmentation results if max pooling is used instead of average pooling. The final model, which was fine-tuned with the LeakyReLU activations and trained on the full dataset, is shown to achieve an accuracy of around 80-85% while having a very high degree of visual similarity pertaining to ground-truth segmentation maps. Therefore, the optimized variant of U-Net is considered quite relevant for the task at hand and appears to precisely identify and outline features in aerial images.

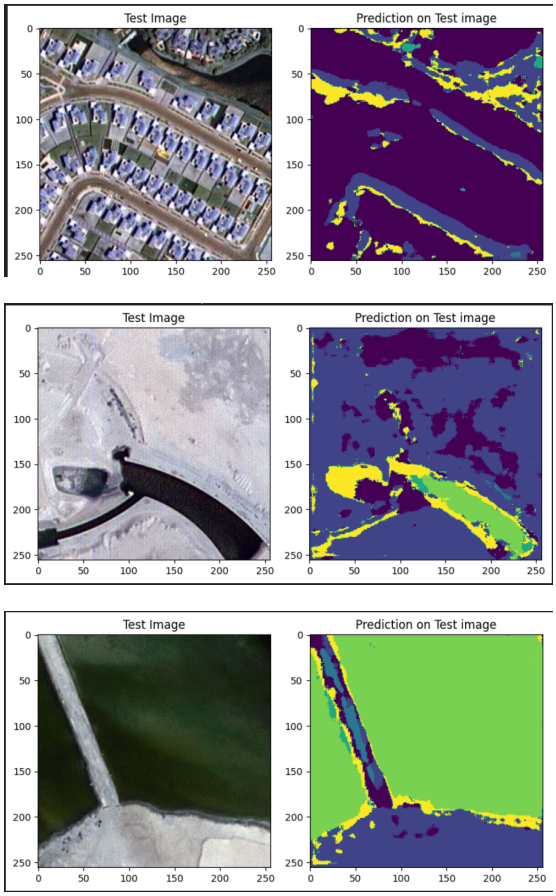

Fig. 5. Some test images and their prediction.

VIII. Code and Experimental Setup

The code can be found here: Semantic Segmentation Code This contains our model training, but for prediction purposes, there is a trained model and test images given.

IX. Conclusion

The results of this study prove that appropriate configurations can be optimized to achieve high accuracy in aerial imagery pixel-wise semantic segmentation based on U-Net architectures. Among key changes - selection of a suitable pooling method, activation function, and regulation of hyperparameters, most importantly - a better performance of models depends on these modifications. Its performance shows how deep learning can be practically deployed in areas that eventually require high-precision image segmentation, like urban planning and land cover mapping to any type of disaster management.

X. Future Scope

Future work could explore several avenues to further enhance segmentation performance. One approach would be incorporating attention mechanisms, such as the Attention U-Net, to help the model focus on relevant spatial information more effectively. Additionally, experimenting with multi-scale feature extraction methods, such as the use of ResNet or DenseNet backbones, might capture finer details and improve segmentation accuracy. Lastly, deploying the model in real-time applications would benefit from optimization techniques to reduce computational complexity, allowing for faster inference and deployment in resource-constrained environments.

Acknowledgment

We thank the National Institute of Technology Karnataka for supporting this research and providing the necessary resources.

References

[1] B. U. Mahmud and G. Y. Hong, "Semantic Image Segmentation using CNN (Convolutional Neural Network) based Technique," 2022 IEEE World Conference on Applied Intelligence and Computing (AIC), Sonbhadra, India, 2022, pp. 210-214, doi: 10.1109/AIC55036.2022.9848977. Semantic Image Segmentation using CNN (Convolutional Neural Network) based Technique — IEEE Conference Publication

[2] Olaf Ronneberger, Philipp Fischer, & Thomas Brox. (2015), U-Net: Convolutional Networks for Biomedical Image Segmentation [1505.04597] U-Net: Convolutional Networks for Biomedical Image Segmentation

[3] Ansari, Maaz & Bhosale, Surendra & Choudhary, Archana. (2023). Semantic Segmentation using Convolutional Neural Networks. 10. 31-34. (PDF) Semantic Segmentation using Convolutional Neural Networks

[4] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 3431-3440, doi: 10.1109/CVPR.2015.7298965. Fully Convolutional Networks for Semantic Segmentation — IEEE Conference Publication

[5] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, "Pyramid Scene Parsing Network," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 6230-6239, doi: 10.1109/CVPR.2017.660. Pyramid Scene Parsing Network