Advancing Drone Delivery: A Lightweight Machine Learning Model for Safe Landing Zone Prediction

Wells Ling

Received June 26, 2025
Accepted September 21, 2025
Electronic access November 11, 2025

Last-mile delivery for consumer packages has always been the most challenging phase of shipping due to being the most time-consuming, least cost-effective, and least environmentally friendly part of delivery. Drone-based delivery offers a promising solution but faces a critical challenge in identifying safe landing zones (SLZs) in dynamic environments. This study aims to develop a lightweight machine learning model to predict SLZs in real-time using publicly available aerial datasets. A U-Net model with a MobileNetV2 encoder was designed to run on lightweight systems, first segmenting out areas deemed safe before predicting the best using distance transformations on contiguous regions. Despite the promising validation performance, there were observed significant drops in test performance. The paper also includes potential implications and areas for future studies to improve upon.

Keywords: Safe Landing Zone (SLZ), semantic segmentation, Intersection over Union, MobileNetV2, U-Net, real-time, drone delivery, TensorFlow, augmentation

Introduction

Package delivery plays a key role in modern society, as it drives commerce and enables convenience. As e-commerce grows, so does the demand for faster, more efficient shipping solutions. Last mile delivery for consumer packages, the final step in the supply chain, has always been the most challenging due to being the most time-consuming, least cost effective, and least environmentally friendly part of shipping. In the last few years, last-mile delivery has accounted for up to half of all total shipping cost and roughly 30% of total logistics sector emissions, a disproportionate impact relative to the distance it covers [1,2]. Drone-based delivery, a concept popularized after Amazon revealed its Prime Air drone program on CBS in 2013, are an innovative solution to the problem by being easily scalable, more environmentally friendly, and more potentially more efficient with less delay [3].

However, despite these advantages, drone-based delivery systems face a critical challenge when identifying safe landing zones (SLZs) in dynamic environments. To address this, most current systems use fiducial marker-based landing for guidance, where drones rely on visual tags such as ArUco markers placed on the ground to identify their delivery targets4. Many leaders in drone delivery, such as Amazon or Wing, currently use this approach in their state-of-the-art systems: Amazons MK27-2 drone, its flagship model from 2022 to 2024, used large 2.5-foot-square ArUco mats during trial deliveries. Additionally, past research has also used various other sensors such as GPS, 9DoF IMU, and barometer readings in addition to ArUco markers for precision landing [4]. While effective in controlled testing environments, this approach presents significant scalability and accessibility challenges. Requiring predetermined markers limits real-world deployment flexibility and may pose accessibility issues, especially in areas where such markers may be tampered with or unable to deploy. Such issues and shortcomings add unwanted variability to a necessarily robust system, and would ultimately undermine the autonomous potential of drone systems.

With recent advancements in computer vision and machine learning, particularly in image segmentation techniques, semantic segmentation, the technique of assigning pixels to different object classes, has become increasingly accurate and efficient with the introduction of U-Net architectures, a specialized form of fully convolutional networks (FCNs) originally developed for biomedical image segmentation [5]. U-Nets combine high-resolution spatial information from the encoder path with contextual information from the decoder path through skip connections, preserving fine-grained details while capturing broader image structure [6]. This architecture allows models to perform precise pixel classification, even with relatively small training datasets. It is ideal for applications where annotated data is limited and/or model complexity is a constraint.

Beyond U-Nets, more specialized semantic segmentation models have been developed for more specific scenarios. A recent example includes KDP-Net, which is designed for autonomous emergency UAV landing, able to achieve processing speeds of 38.79 fps and >65% IoU on testing data. The IoU (Intersection over Union), is a standard evaluation metric for image segmentation that quantifies the overlap between the predicted

$$IoU = \frac{TP}{(TP + FP + FN)}$$

Fig. 1 Mathematical expression of IoU expressed with values from the confusion matrix

segmentation and the total unique area segmented between the prediction and ground truth. Another application came in real time natural disaster surveillance systems, with models like DeepLabV3+ achieving >10fps processing speeds and 90% mIoU. These highly specialized semantic segmentation frameworks have furthered real time segmentation possibilities. This paper aims to leverage recent advancements in AI image segmentation to design and develop a real-time safe landing zone prediction algorithm for drone delivery systems and demonstrate the feasibility of running the algorithm on a drones onboard computer. This studys focus is on the lightweight machine learning algorithm to predict SLZs. It will not tap into the hardware aspect of drone landing (i.e. controlling and landing a physical drone). Furthermore, due to hardware constraints, both from our experimental circumstances and the environment the proposed system inhabits, the final system will be tested on a Raspberry Pi 4B, which may limit the size and complexity of the completed model due to the lack of a GPU.

In this project, data is first collected and used to train a U-Net model to segment aerial imagery. The algorithm then takes the segmentation models output and performs a distance transform to find the largest contiguous area within a designated safe region before inscribing an ellipse to mark the best SLZ in that area. The segmentation model is evaluated based on its accuracy and mean IoU for the safe region for both the validation and test sets. The model is then loaded onto a Raspberry Pi 4B to test its ability to perform in real time. Through these procedures, the goal is to develop a lightweight machine learning model capable of predicting SLZs in real-time.

Methodologies

Experimental Design

This study aims to create a model that can empirically satisfy these conditions, validating its performance under static and real-time deployment conditions. To start, data to train the semantic segmentation model must be gathered and assessed for the right characteristics, which are vertical or low-oblique angled low-altitude aerial imagery. As described above in the introduction, the framework chosen for this model will be the U-Net infrastructure. More sophisticated state-of-the-art models such as DeepLabV3 and specialized semantic segmentation

	General Accuracy	Mean IoU for SLZ
Validation	80–90%	>70%
Test	75–85%	>60%

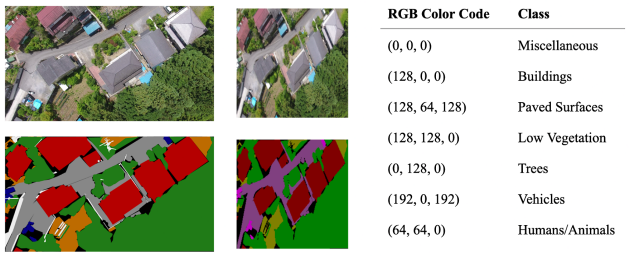
models were initially considered; however, they were scrapped for being easily overfitted, too complex, and computationally draining for typical drone onboard systems. In order to test the final model, the proper minimum metrics had to be determined. Recent work on the DroneDeploy dataset using DeepLabv3+ reported a validation mIoU near 70% and a test mIoU of 52.5%[7]. Another study achieved a 75.15% overall accuracy with a non-state-of-the-art U-Net model on urban UAV imagery from the UAVid dataset[8]. During experimental deployment, the goal of this proposed system will be to perform inferences at least once every 1 second, with sub-second latency. Similar studies with lighter models have shown 200-500 ms latency with 2-5 inference frames per second on comparable hardware, making these target metrics reasonable[9]. This rate is likely sufficient for the task of SLZ prediction in this paper. However, for real world deployment, these target metrics would need to be higher, especially if operating in dynamic environments.

Data Collection & Preprocessing

Aerial segmentation datasets were first gathered from Kaggle, Papers With Code, and GitHub. Four publicly available datasets which fit the application's needs were chosen: the UAVid dataset, the Semantic Drone dataset (SDD), the Swiss Drone and Okutama Drone datasets (SDOD), and the ISPRS Potsdam dataset[10–13]. All datasets provided high-resolution imagery, typically captured from a vertical or low-oblique angled perspective. This viewpoint is essential for aerial segmentation tasks as it ensures spatial consistency across scenes. All data, excluding those from the ISPRS Potsdam dataset, were captured using low-altitude drones, offering a closer, more ground-level view than satellite imagery. This lower perspective provides a clearer view of smaller objects like people, cars, and pathways, essential for ensuring a safe landing. Each image was paired with its corresponding pixel-level masks which classified the images features into one of 7 classes. The image data spans a variety of urban and semi-urban environments, including residential areas, streetscapes, and mixed-use outdoor spaces. Importantly, the datasets share consistent label classes and labeling quality.

To maximize the amount of raw training and validation data, the UAVid, Semantic Drone Dataset, and Swiss Drone and Okutama Drone Datasets were selected for training, with the ISPRS Potsdam dataset designated as the testing set. Despite the ISPRS Potsdam dataset being collected through satellite imagery, not through drone imagery, it can be processed to simulate the viewpoint of an UAV for most classes. Since the viewpoint is from so far up, smaller classes such as humans/animals will likely not be

Dataset	Raw images	Initial Resolution	Role
UAVid	400	3840×2160	Train + Validation
SDD	260	6000×4000	Train + Validation
SDOD	191	4608×3456	Train + Validation
ISPRS	37	6000×6000	Testing



RGB Color Code	Class
(0, 0, 0)	Miscellaneous
(128, 0, 0)	Buildings
(128, 64, 128)	Paved Surfaces
(128, 128, 0)	Low Vegetation
(0, 128, 0)	Trees
(192, 0, 192)	Vehicles
(64, 64, 0)	Humans/Animals

Fig. 2 Data from the Swiss Drone and Okutama Datasets before and after preprocessing to standardize training, validation, and testing data

captured, leading to inaccurate measurements of those classes. However, as there are a limited amount of publicly available aerial datasets in general which fit the criteria, this compromise is workable.

The chosen training sets are merged into one unified dataset, as doing so shows improved results for generalization [14]. Standardization was necessary before model training since each dataset had its own resolution, labeling system, and class definitions. Googles Colab Python environment was used throughout this project to install and run OpenCV and TensorFlow for model training.

With Google Colab as the primary development environment, a preprocessing pipeline was created in its Python environment, which leveraged OpenCV for image manipulation. All image and mask pairs were then resized to 256×256 pixels using the cv2.resize() function to lower processing times of the completed model. During this resizing step, interpolation from downsampling the masks introduced color distortions in the segmentation mask, especially pronounced along the borders between classes. This resulted in discolored interpolation artifacts that no longer matched any valid class.

To address this issue of non-class colors, an algorithm was created to map each invalid color to the closest predefined class color based on perceived visual similarity. A KD-Treebased nearest neighbor search was ultimately adopted over other methods. Each mask pixel was checked against the list of valid RGB class colors, and if a pixels color did not match any known class exactly, it was reassigned to the nearest valid color based on Euclidean distance in RGB space. This would ensure that all pixels are reassigned to valid class labels while preserving the integrity of class boundaries. The processed data were then loaded into Colab, which underwent more processing. A custom data loader was implemented to facilitate efficient training and avoid

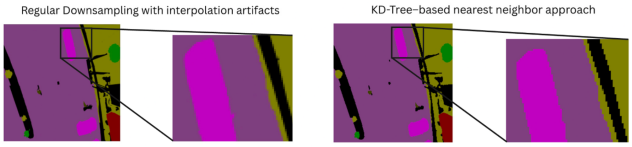


Fig. 3 Interpolation artifacts via downsampling v.s. KD-Tree based reclassification displayed here, with clear pixel borders using KD-Tree compared to regular downsampling

memory overload. This loader dynamically generated batches of images and masks, performing real-time augmentations and yielding NumPy arrays for training. Due to the smaller relative size of the model, generalization may become a bottleneck for performance. One way to resolve this is through augmentations. When done correctly, augmentations increase dataset diversity and the data pool, resulting in less potential overfitting.

For each batch, the loader generated went through one of three augmentation pathways: no augmentation, geometric augmentation, or photometric augmentation. Geometric augmentations, including flips, rotations, and distortions, simulated variations in drone orientation, perspective, and flight stability. Photometric augmentations, such as blurring, brightness, or contrast, simulate changes in lighting, atmospheric conditions, and camera sensor responses [15].

Rather than applying augmentations to the dataset ahead of time, he loader would load them dynamically during training enabled a virtually infinite number of image variations. This approach significantly improved the model's exposure to unseen conditions and reduced the risk of memorization, offering better generalization without increasing dataset size or storage demands.

Model Architecture

A U-Net architecture was implemented with a MobileNetV2 encoder to perform semantic segmentation of aerial imagery with high accuracy and computational efficiency. MobileNetV2, a lightweight convolutional neural network pretrained on ImageNet and optimized for mobile and embedded vision applications, was the encoder [16]. Compared to other semantic segmentation encoders, including ResNet-50, EfficientNet, and many others, MobileNetV2 has 3.4M parameters compared to the 11M and 5.3M of ResNet-50 and EfficientNet respectively [17–19]. This configuration was chosen for its balance between performance and computational efficiency, critical for deployment on resource-constrained edge devices like a drones onboard computer.

In general, the encoder, or contracting path, is responsible for the extraction of high-level features from the input image while progressively reducing its spatial resolution. The MobileNetV2 encoder performs depthwise separable convolutions and inverted

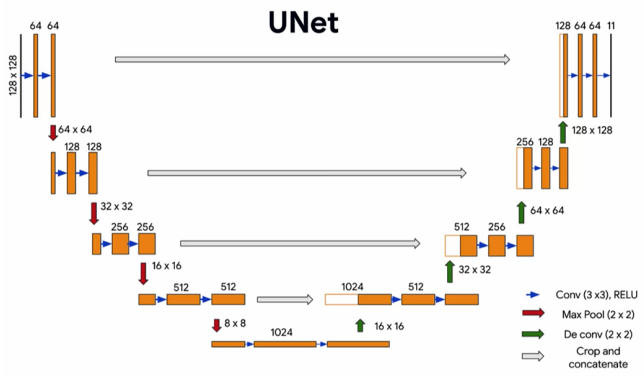


Fig. 4 Diagram of the U-Net model with encoder and decoder layers to perform image segmentation

residual blocks, allowing the model to capture both fine and general visual features with fewer parameters compared to traditional CNNs. As the image data passes through successive convolutional layers and downsampling stages through strided convolutions operations, the model begins generalize upon increasingly abstract features[19]. To further adapt the model for aerial segmentation, the final 30 layers of the encoder were unfrozen and fine-tuned during training. Models trained with frozen encoders converged more quickly but plateaued at lower accuracy, while full fine-tuning yielded the best trade-off between accuracy and generalization[20]. The decoder, or expansive path, complements the encoder by gradually reconstructing the spatial resolution of the segmentation map with upsampling operations. This model uses Conv2DTranspose layers to upsample feature maps and integrate the corresponding feature maps from the encoder via skip connections. These skip connections reintroduce spatial detail which may have been lost during downsampling back into the image, enabling the network to make fine-grained predictions. Each upsampling stage includes a concatenation with encoder features, followed by convolutional layers with ReLU activation, L2 regularization (λ = 1e-3), dropout layers (ranging from 0.1 to 0.4) to combat overfitting, and batch normalization for improved convergence. When dropout was removed, the model exhibited overfitting, with validation accuracy diverging after 100 epochs[21]. Eliminating L2 regularization similarly led to less stable training and weaker generalization [22]. The final output layer uses a 1×1 convolution with softmax activation to produce per-pixel class probabilities across all defined segmentation classes.

To optimize training, several regularization and scheduling mechanisms were implemented. Early stopping was applied to monitor the validation loss, stopping training when no significant improvement was observed after 25 epochs to prevent overfitting. An exponential learning rate scheduler was used to gradually decrease the learning rate over time, balancing initial rapid convergence with later fine-tuning of segmentation details. The decay rate was set to 0.9, with decay steps dependent on the dataset size and batch configuration. This scheduling allowed the model to make large parameter updates early in training and fine-tune more cautiously in later stages. Because the model performed segmentation on multiple classes, categorical cross-entropy loss and the AdamW optimizer was used during training. Replacing AdamW with Adam resulted in slightly faster initial convergence but lower IoU on minority classes[23]. AdamW helps preserve model generalization by decoupling weight decay from the gradient update, enhancing its ability to learn fine object boundaries.

Upon completion of training, the model was exported in Keras format and converted to TensorFlow Lite (TFLite) to prepare it for edge deployment. Post-training 8bit integer (int8) quantization was then applied for further optimization. Prior studies have demonstrated that post-training quantization can reduce model size by up to 75% while maintaining similar metrics on edge devices[24,25]. This conversion replaced the initial 32-bit floating-point weights and activations with fixed-point representations, significantly reducing the model's memory footprint and enabling reduced RAM usage and faster loading times. In addition to size reduction, int8 quantization leverages optimized integer arithmetic routines present in many mobile CPUs and edge accelerators, resulting in up to 4x performance gains during inference compared to full-precision models[26]. This makes int8 quantization especially suitable for real-time semantic segmentation applications running on lightweight platforms like a drones onboard computer.

The final portion of the SLZ algorithm included a system to determine the best possible SLZ for the drone. In a paper published by Gonzlez-Trejo et al., the authors propose a system where they perform distance transformations to identify the center of the largest inscribed polygon outside of areas with people[27]. Their main goal was to prevent injury to humans during an emergency drone landing, regardless of the drones integrity. However, this papers targeted application requires the drone to take minimal damage while landing, therefore adding additional restrictions and building upon their existing system.

This system first isolates paved areas, which are determined to be safe based on the assumption that such areas are generally flat and obstruction-free, from the segmentation mask by color, being RGB(128, 64, 128). Using connected components analysis in OpenCV, the algorithm identifies the largest contiguous region within this class. This region is assumed to be the most viable SLZ candidate, as it theoretically provides the most open area for a drone to land. Within the predicted paved areas, a distance transform is applied to locate the point farthest from the regions boundary, which serves as the center for an inscribed ellipse, which indicates the best SLZ prediction. Finally, the ellipse's eccentricity is dynamically adjusted using the sources pitch above the ground to minimize the perspective distortion which occurs when observing from an oblique angle. This cor-

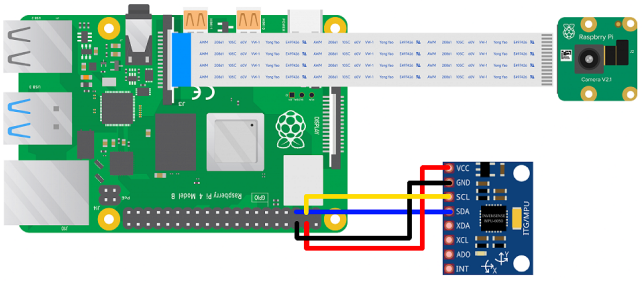


Fig. 5 Hardware diagram of connections, Camera and MPU6050 Sensor on the Raspberry Pi

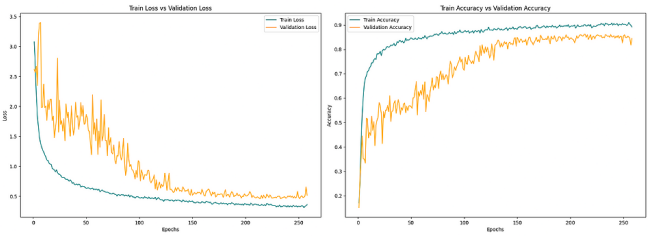


Fig. 6 Train/Validation accuracy and loss graphs from training, with test and validation metrics plateauing and converging

rection more accurately represents and account for the true size of the predicted SLZ from the drones perspective.

Deployment

Upon completion of training and quantization, the optimized entire SLZ prediction algorithm was deployed on a Raspberry Pi 4B equipped with a Pi-Camera and an MPU6050 inertial measurement unit, the sensor providing the systems pitch for the prediction ellipses eccentricity. This embedded system provided a fully self-contained environment for real-time testing of the segmentation model. The Pi-Camera and the MPU6050 were interfaced with the Raspberry Pi through the IC protocol, and a virtual environment was established to manage dependencies. Essential Python libraries were required to interface with the sensors, including opencv-python for image processing, numpy for numerical operations, tflite-runtime for model inference, and mpu6050-raspberrypi. Additional system-level packages like libatlas-base-dev, libhdf5-dev, libcamera-apps, and v4l-utils were required for full hardware compatibility. From here, a custom Python script was written to capture frames from the Pi-Camera and process them through the segmentation model in real time. The final semantic segmentation model stopped training at epoch 233 since it was trained with early stopping, which monitors the validation loss. Validation loss, a metric on how well a machine learning model performs on the validation dataset, should be as low as possible. A high validation loss indicates the model may be overfitting, while a low validation

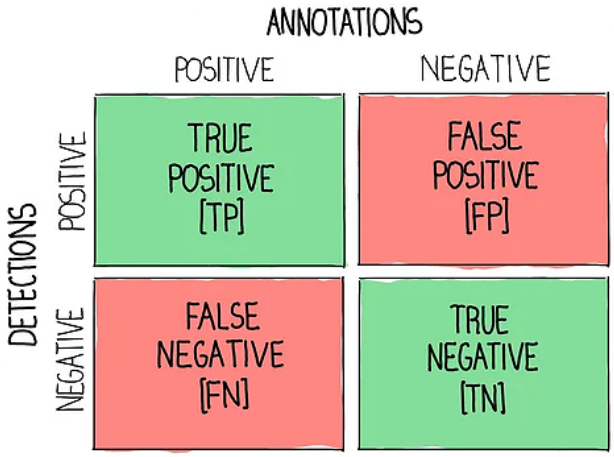


Fig. 7 Confusion Matrix showing different categories pixels can be classified into, with TP being the only category for correct pixel classification

loss indicates the model can generalize from its training data. The models lowest validation loss was 0.4610, with a final training accuracy of 86.00

Additionally, the Mean Intersection over Union (IoU) for paved surfaces was calculated, as this class directly impacts the accuracy of Safe Landing Zone (SLZ) predictions. First, predicted pixels are classified with accordance to a confusion matrix which either labels them true positives, false positives, or false negatives. The IoU is then calculated by dividing the true positives by the combined sum. By averaging the IoU for paved surfaces across a dataset, it quantifies how accurate the predicted segmentation is. When calculated for validation set, the models Mean IoU for the paved surfaces class was 82.80%, satisfying the initial goal of being above 70%. The test data was sourced from the ISPRS Potsdam Dataset to test the model beyond the validation data. Since this dataset was not created via drone photography, but rather with satellite imagery, the dataset had to be modified for the task. To do so, a Python script was written to randomly 6x zoom in on five separate regions, thereby mimicking the altitude a drone would have. When evaluated against this testing dataset, the models metrics dropped drastically to 59.27% test accuracy and 46.91% mean IoU. The other classs mean IoU are summarized in Figure 8. Some classes show significant decreases in mean IoU scores because these classes are not densely represented in the test data. For instance, the human and animal classes show ”nan” values due to the nature of the satellite imagery used to create the test set, which did not capture these classes at all. It is important to note that this data was taken before compression and quantization from the Keras to the TFLite format, as compression and quantization may further decrease the metrics. During deployment on the Raspberry Pi, the number of inferences per second ranged from

Class	Misc.	Buildings	Paved Surfaces	Low Vegetation	Trees	Vehicles	Humans/Animals
mIoU	.0555	.5488	.4961	.2246	.2441	.4436	nan

Fig. 8 Complete mIoU data for all segmented classes on tf keras model pre-compression

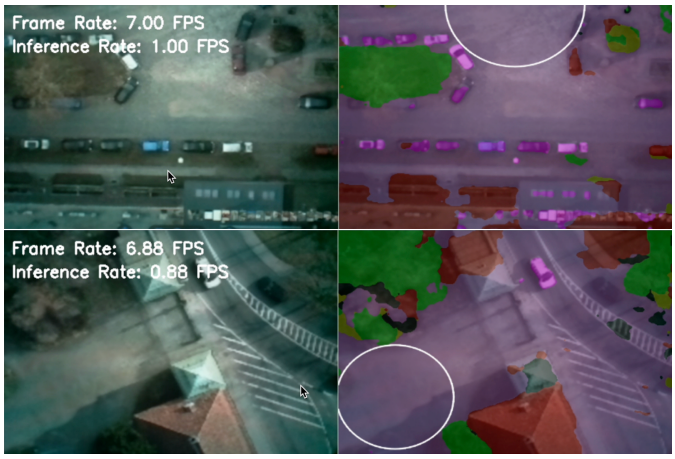


Fig. 9 Figures 9 & 10. Sample predictions of data from the test set showcasing SLZ predictions

0.85 to 1.05, with an average latency of around one second. The latency can generally be broken down into these approximate timeframes: 50-70 ms for image capture, 100-140 ms for preprocessing, 600-700 ms for model inference, and 150-200 ms for post processing. These measurements were collected under controlled conditions using a strong and stable internet connection, ensuring that network instability did not affect performance

Discussion

In the end, the goal of a lightweight, U-Net based, machine learning model which runs on the relatively constrained Raspberry Pi 4B system was realized. The models performance validation set, regarding accuracy and Mean Intersection over Union (IoU), met the thresholds established during the planning phase; however, its performance on the test set did not meet these benchmarks. While suboptimal, the observed mean IoU of 46.91% remains acceptable for early-stage semantic segmentation efforts, particularly in real-world aerial contexts [8]. Therefore, current results indicate a functional proof of concept which will require further refinement before real-world deployment.

Further optimization will be necessary before this system can be deployed on delivery drones. Several factors likely contributed to the models reduced generalization and quality of predictions on new, unseen data, reflected in the drop in mean IoU scores. Regarding the other mIoU scores for other classes, the lower scores for the miscellaneous and humans/animals category are due to smaller number of testing images with enough pixels classified into those classes compared to the other classes.

Class	mIoU without Augmentations	mIoU with Augmentations	Change
Miscellaneous	6.70%	5.55%	-1.15%
Buildings	50.87%	54.88%	+4.01%
Paved Surfaces	47.85%	49.61%	+1.76%
Low Vegetation	19.92%	22.46%	+2.54%
Trees	26.24%	24.41%	-1.83%
Vehicles	33.09%	44.36%	+11.27%
Humans/Animals	nan	nan	nan

Fig. 10 Class mIoU comparisons for models trained without augmentations vs with augmentations

The most likely issues include overfitting, overlapping semantic features between visually similar classes, and a relatively shallow training dataset that may not have sufficiently captured the diversity of urban conditions. There were observed drops in mIoU in scenarios with many visual occlusions (e.g., harsh shadows), semantically ambiguous regions (e.g., grass vs. shrubs, grey rooftops vs. pavement), and generally, areas where fine-grained class boundaries were not easily distinguishable. Additionally, hardware limitations due to running on a Raspberry Pi without any GPUs constrained the models architectural complexity. These factors combined likely produced the conditions for reduced mean IoU scores.

Additionally, the definition for SLZs could be improved. For static testing or slow real world scenarios, the current definitions and other minimum metrics such as latency or processing speed may be sufficient, In dynamic, real-world scenarios, the movement of people, vehicles, and other objects could pose issues to the SLZs exclusively being paved surfaces. Therefore, a more general definition potentially encompassing some elements of other classes may be a topic for future studies to improve.

However, despite these limitations with the segmentation model, the SLZ prediction algorithm helped compensate for segmentation inaccuracies by focusing on the largest, continuous landing zones, filtering out many of the smaller inaccuracies with the segmentation model. As a result, even with models middling segmentation performance, the system demonstrated reasonable SLZ predictions, suggesting the underlying concept has practical merit. Additionally, current augmentations have generally improved model performance. With augmentations, the model generally has more diverse training data to learn from, leading to better generalization. The validation loss, a metric for determining error in the validation set, is a general indicator of generalization. With the augmented model, the validation loss was 0.5185, compared to the augmented models validation loss of 0.4610. Future studies could focus on and improve upon these current flaws. To combat overfitting, augmentation strategies could be further refined and expanded, incorporating transformations that target generalization under lighting variation, occlusion, and noise. Another solution potential solution may come from access to more raw training data,

as a total of 851 raw training files is considered small for the relatively complex MobileNetV2 encoder and therefore more susceptible to overfitting. Simplifying the class structures, such as merging similar classes together, may help reduce class confusion and improve label consistency. Additionally, if additional aerial imagery datasets satisfy the requirements stated in the methodologies, data diversity would also lead to improved generalization capabilities. Hardware upgrades, such as migrating to an NVIDIA Jetson Orin Nano or other systems with modern GPUs, would enable deploying deeper models with higher capacity. If compression remains necessary, more sophisticated quantization or pruning strategies could help preserve model accuracy while retaining efficiency. These listed improvements, along with others, have the potential to turn this system into a robust, accurate, and scalable method to predict the best SLZ for delivery drones.

References

1 H. Borsenberger, P. Cremer, D. De Donder, D. Joram and S. Lécou, *The Role of the Postal and Delivery Sector in a Digital Age*, 2014.

2 M. Levrey, A. Ntemou, A. Kiousi, I. Fergadiotou, M. Kampa, D. Rizopoulos, B. Magoutas, A. Tsirimpa and P. Georgakis, *Transport Transitions: Advancing Sustainable and Inclusive Mobility*, 2025, pp. 536–542.

3 60 Minutes Overtime Staff, *Amazon unveils futuristic plan: delivery by drone*, CBS News (60 Minutes), 2013, https://www.cbsnews.com/news/amazon-unveils-futuristic-plan-delivery-by-drone/.

4 V. R. F. Miranda, A. M. C. Rezende, T. L. Rocha, H. Azpúrua, L. C. A. Pimenta and G. M. Freitas, *Journal of Control, Automation and Electrical Systems*, 2022, **33**, 141–155.

5 O. Ronneberger, P. Fischer and T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, 2015, https://arxiv.org/pdf/1505.04597.

6 N. Klingler, *U-Net: a comprehensive guide to its architecture and applications*, 2024, https://viso.ai/deep-learning/u-net-a-comprehensive-guide-to-its-architecture-and-applications/, Viso.ai.

7 M. R. Heffels and J. Vanschoren, *Aerial imagery pixel-level segmentation*, 2020, https://arxiv.org/abs/2012.02024.

8 A. Majidizadeh, H. Hasani and M. Jafari, *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2023, **X-4/W1-2022**, 451–457.

9 T. Q. Khoi, N. A. Quang and N. K. Hieu, IOP Conference Series: Materials Science and Engineering, 2021, p. 012033.

10 Y. Lyu, G. Vosselman, G. S. Xia, A. Yilmaz and M. Y. Yang, *ISPRS Journal of Photogrammetry and Remote Sensing*, 2020, **165**, 108–119.

11 ICG, *ICG-DroneDataset*, 2019, http://dronedataset.icg.tugraz.at.

12 S. Speth, A. Gonçalves, B. Rigault, S. Suzuki, M. Bouazizi, Y. Matsuo and H. Prendinger, *Journal of Field Robotics*, 2022, **39**, 1–29.

13 Papers with Code, *Papers with Code - ISPRS Potsdam Dataset*, 2022, https://paperswithcode.com/dataset/isprs-potsdam.

14 M. Howe, B. Repasky and T. Payne, *Effective utilisation of multiple open-source datasets to improve generalisation performance of point cloud segmentation models*, 2022, https://arxiv.org/abs/2211.15877.

15 A. Tavera, E. Arnaudo, C. Masone and B. Caputo, *Augmentation invariance and adaptive sampling in semantic segmentation of agricultural aerial images*, 2022, https://arxiv.org/abs/2204.07969.

16 K. Ryselis, T. Blažauskas, R. Damaševičius and R. Maskeliūnas, *Sensors*, 2022, **22**, 6354.

17 M. Tan and Q. V. Le, *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*, 2019, https://arxiv.org/abs/1905.11946.

18 K. He, X. Zhang, S. Ren and J. Sun, *Deep Residual Learning for Image Recognition*, 2015, https://arxiv.org/abs/1512.03385.

19 M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, *MobileNetV2: Inverted Residuals and Linear Bottlenecks*, 2018, https://arxiv.org/abs/1801.04381.

20 C. Teuber, A. Archit and C. Pape, *Parameter Efficient Fine-Tuning of Segment Anything Model*, 2025, https://arxiv.org/abs/2502.00418v1, v1.

21 O. Harris, M. Andrews, P. Allen and A. Tripathi, *World Journal of Advanced Engineering Technology and Sciences*, 2024, **13**, 836–853.

22 ArXiv, *Rethinking Conventional Wisdom in Machine Learning: From Generalization to Scaling*, 2016, https://arxiv.org/html/2409.15156v2.

23 V. Parmar, N. Bhatia, S. Negi and M. Suri, *Exploration of Optimized Semantic Segmentation Architectures for Edge-Deployment on Drones*, 2020, https://doi.org/10.48550/arXiv.2007.02839.

24 B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam and D. Kalenichenko, *Quantization and training of neural networks for efficient integer-arithmetic-only inference*, 2017, https://arxiv.org/abs/1712.05877.

25 L. Wu, K. Huang, H. Shen and L. Gao, *A foreground-background parallel compression with residual encoding for surveillance video*, 2020, https://arxiv.org/abs/2001.06590.

26 H. Ahn, T. Chen, N. Alnaasan, A. Shafi, M. Abduljabbar, H. Subramoni and D. K. Panda, *Performance characterization of using quantization for DNN inference on edge devices: extended version*, 2023, https://arxiv.org/abs/2303.05016.

27 J. Gonzalez-Trejo, D. Mercado-Ravell, I. Becerra and R. Murrieta-Cid, *IEEE Robotics and Automation Letters*, 2021, **6**, 7901–7908.