Semantic Segmentation of Urban Drone Imagery

University of Milan-Bicocca

Andrea Palmieri
a.palmieri13@campus.unimib.it
Matricula 921785

Abstract

This paper presents a comparative study of semantic segmentation techniques applied to urban drone imagery using the *Semantic Drone Dataset*. The study focuses on evaluating the performance of three encoder architectures—*EfficientNetB5*, *ResNet34*, and *MobileNetV2*—integrated within the *DeepLabv3+* framework. The primary objective is to assess the feasibility of these models for real-time deployment on drones, particularly for tasks like autonomous navigation and infrastructure monitoring in urban environments. The results indicate that while *EfficientNetB5* delivers the highest segmentation accuracy, *MobileNetV2* offers a compelling balance between performance and computational efficiency, making it well-suited for use on resource-constrained platforms such as drones.

1 Introduction

1.1 Background

Semantic segmentation is a critical task in the field of computer vision, especially when applied to drone imagery for urban environments. This technique is essential for enabling drones to autonomously navigate complex urban areas, where accurate identification of various objects like roads, buildings, people, and vegetation is vital. These capabilities are particularly relevant in scenarios such as drone deliveries, where precise obstacle detection and route planning are required, and in infrastructure inspection, where identifying structural elements like roofs or walls is crucial.

Recent advancements in deep learning, particularly with convolutional neural networks (CNNs), have significantly improved the accuracy of image segmentation tasks. Architectures like DeepLabv3+ leverage multi-scale contextual information to achieve state-of-the-art performance. However, deploying these models on drones, which have limited onboard computational resources, presents a unique set of challenges. Efficient model designs are required to maintain high segmentation accuracy while ensuring real-time processing capabilities on resource-constrained devices.

1.2 Objectives

The primary objective of this study is to evaluate the suitability of various encoder architectures for real-time semantic segmentation of urban areas. Specifically, this study compares the performance of *EfficientNetB5*, *ResNet34*, and *MobileNetV2* in terms of segmentation accuracy, computational efficiency, and their feasibility for real-time deployment on drones. By analyzing these models, the study aims to identify the most effective solutions for enabling autonomous drones to navigate urban environments safely and efficiently, with potential applications in areas such as drone deliveries, urban planning, and infrastructure inspection.

2 Dataset

2.1 Dataset Description

The dataset used is the *Semantic Drone Dataset*, a collection of high-resolution aerial imagery captured from a bird's-eye perspective of urban scenes. The images were captured using a resolution of 24 megapixels, resulting in images 6000x4000 pixels. A sample of the images and masks is shown in Figure [**??**]. The imagery in the dataset was captured at varying altitudes, ranging from 5 to 30 meters above ground level. This diversity in altitude provides a rich set of perspectives, allowing us to train models that can adapt to different flight heights and scenarios. 400 public images are available for training, while the

author reserves 200 images as a private test set. This dataset offers a unique opportunity to develop models capable of improving the capability of autonomous drones in navigating safely in urban settings. More information on this dataset is available at `http://dronedataset.icg.tugraz.at`.

2.2 Data Analysis and Preprocessing

2.3 Class Frequencies

The dataset contains 23 classes, and their frequencies are shown in Table **??**. As expected for an urban area imagery dataset, there is significant class imbalance which is expected to impact the performance of the models, with less frequent classes presenting more challenges for accurate detection. The influence of this class imbalance on model performance will be analyzed in the *Comparison of Per-Class IoU* section of the report.

Class	Frequency	%
paved-area	3.62×10^9	37.74
grass	1.92×10^9	19.98
roof	7.07×10^8	7.36
gravel	7.02×10^8	7.31
vegetation	6.82×10^8	7.10
obstacle	3.40×10^8	3.54
dirt	3.07×10^8	3.20
wall	2.58×10^8	2.69
water	2.12×10^8	2.21
tree	1.97×10^8	2.05
bald-tree	1.28×10^8	1.33
person	1.01×10^8	1.05
fence	9.20×10^7	0.96
car	7.55×10^7	0.79
rocks	6.90×10^7	0.72
pool	6.14×10^7	0.64
window	5.37×10^7	0.56
ar-marker	2.18×10^7	0.23
unlabeled	2.09×10^7	0.22
bicycle	2.07×10^7	0.22
fence-pole	5.10×10^6	0.05
door	3.01×10^6	0.03
dog	1.37×10^6	0.01
Total	9.60×10^9	**100**

Table 1: Class frequencies and percentages.

2.3.1 Tiling

The preprocessing of the dataset involved several steps to prepare the images for training. Initially, images were loaded and resized by rounding them down to the nearest multiple of the desired tile size. Various tile sizes were utilized, including 512, 768, 1000, and 2000 pixels. The number of tiles for each image and the total number of images for each tile size is shown in Table [**??**]. After creating the tiles of different sizes, all tiles were further resized to 256x256 pixels to ensure consistency and computational efficiency during training. This tiling strategy allowed the model to process the images effectively by breaking them down into manageable segments, enhancing its ability to generalize across different regions of the images. The resized 256x256 pixels tiles were saved on disk, enabling faster data loading and processing during training, rather than performing tiling on-the-fly at each iteration.

Tile Size (px)	Tiles/image	Total tiles
512 x 512	77	30,800
768 x 768	35	14,000
1,000 x 1,000	24	9,600
2,000 x 2,000	6	2,400

Table 2: Number of tiles per image and total tiles for tile size.

2.3.2 Image Augmentations

Each image underwent a series of transformations during the *__getitem__* function of the custom Dataset class. We leveraged the Albumentations library to perform these augmentations and by performing them during each iteration the dataset variability and training robustness was increased. The same set of transformations, with minor differences, was applied to both training and validation/test images to maintain consistency.

Augmentations included horizontal and vertical flips, brightness and contrast adjustments, and blur effects like Gaussian and motion blur. Color adjustments, including changes to hue, saturation, and value, were used to simulate different lighting conditions. CLAHE was applied to enhance contrast, and normalization ensured consistent pixel values across images.

The application of these augmentations improved the model's robustness by diversifying the training data, enhancing its ability to accurately segment diverse aerial images.

3 Methodology

3.1 Network Architecture: DeepLabv3+

In this study, we adopt the DeepLabv3+ architecture [**?**] for the task of semantic segmentation, due to its demonstrated ability to capture both global and local contextual information, which is crucial for generating precise segmentation maps. DeepLabv3+ enhances its predecessor, DeepLabv3 [**?**], by introducing a decoder module that refines the resolution of the segmentation outputs, particularly at the boundaries of objects. This refinement is especially important in applications where accurately delineating the contours of objects, such as roads, buildings, and vegetation in urban drone imagery, is required.

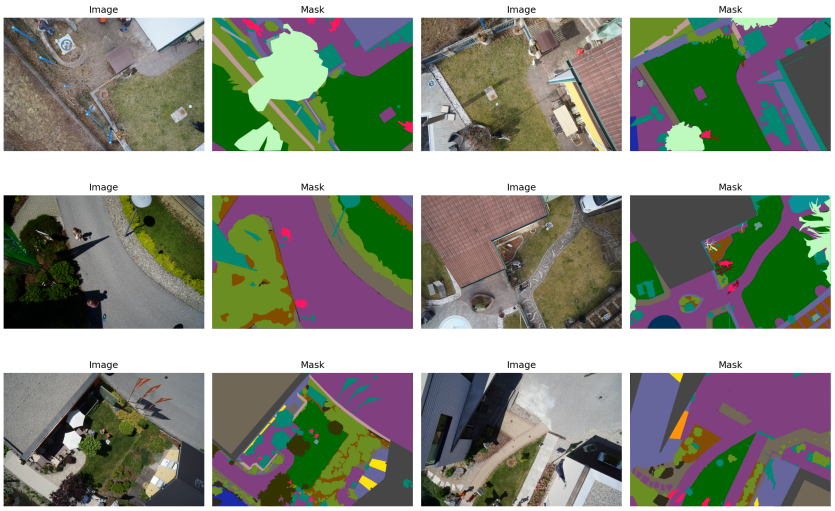


Figure 1: Sample images and respective masks of the Semantic Drone Dataset.

A key innovation in DeepLabv3+ is the employment of atrous (or dilated) convolutions. *Atrous convolutions* allow the model to maintain high spatial resolution while expanding the receptive field, enabling it to capture multi-scale contextual information without increasing the number of parameters. This approach is further enhanced by the *Atrous Spatial Pyramid Pooling (ASPP)* module, which applies atrous convolutions at multiple rates, thereby improving the model's ability to recognize objects at varying scales and capture fine details across different regions of the image.

The *encoder-decoder structure* of DeepLabv3+ plays a pivotal role in its performance. The encoder extracts deep features from the input image, which are then processed by the ASPP module before being fed into the decoder. The decoder subsequently refines the segmentation map, leading to sharper and more accurate predictions.

3.2 Encoders

For the DeepLabv3+ backbone of our network architecture, three different encoders were selected: EfficientNetB5, ResNet34, and MobileNetV2. Each encoder was chosen to balance various aspects of performance and computational efficiency to suit different operational needs. A brief overview of the number of trainable parameters for these encoders is provided in Table [**??**].

Encoder	# Parameters
MobileNetV2	3.4M
ResNet34	21.8M
EfficientNet-B5	28.3M

Table 3: Number of trainable parameters

3.2.1 EfficientNetB5

EfficientNetB5 is a highly efficient convolutional neural network that uses compound scaling to balance depth, width, and resolution, optimizing both accuracy and computational efficiency [**?**]. Its architecture incorporates mobile inverted bottleneck convolution (MBConv) layers, enhancing feature extraction while keeping the model lightweight.

3.2.2 ResNet34

ResNet34 is a deeper residual network that offers a strong balance between depth and computational efficiency. Its architecture incorporates residual connections to overcome the vanishing gradient problem, allowing for effective training of deeper models [**?**]. ResNet34's increased depth enables it to capture complex features and intricate details, making it well-suited for comprehensive segmentation tasks, including those involving detailed urban imagery.

3.2.3 MobileNetV2

MobileNetV2 is designed specifically for environments where computational resources are constrained, such as mobile and edge devices [**?**]. It employs depthwise separable convolutions and an inverted residual structure with linear bottlenecks to reduce model size and computational demands while maintaining high accuracy. MobileNetV2 is particularly advantageous for real-time applications on drones, where efficiency and rapid processing are critical.

3.3 Training

During the training process, we utilize the *CrossEntropyLoss* as the loss function, which is well-suited for multi-class segmentation tasks.

The *AdamW* optimizer is selected due to its enhanced generalization capabilities and better performance in preventing overfitting compared to the standard Adam optimizer. The learning rate is adjusted using a *Cosine Annealing* strategy, with the learning rate being periodically reset every 25 epochs. This approach helps in escaping local minima and facilitates a more thorough exploration of the loss landscape. For each encoder and tiles dimension configuration, the training procedure involves monitoring the model's performance on a validation set at the end of each epoch. The model with the lowest validation loss is saved as a best model checkpoint for that configuration. Finally, the best models are evaluated on the test set.

3.4 Evaluation Metrics

For the evaluation of the trained models, several metrics are computed to comprehensively assess their performance. These metrics include:

- **Weighted Mean Dice:** This metric is calculated based on the class frequencies in the validation set to account for class imbalance. The Dice coefficient measures the overlap between the predicted and ground truth segmentation masks.
- **Weighted Mean Intersection over Union (IoU):** Similar to the weighted mean Dice, the weighted mean IoU is calculated with weights corresponding to the class frequencies in the validation set. The IoU measures the ratio of the intersection between the predicted and ground truth segmentation masks to their union, offering a more stringent evaluation of the model's performance.
- **Accuracy:** Overall accuracy is computed as the ratio of correctly classified pixels to the total number of pixels.
- **Per-Class IoU:** A per-class IoU is computed to analyze the model's strengths and weaknesses on the different classes.

4 Results

4.1 Comparison of Accuracy, mean IoU and mean Dice

The performance analysis across different encoders and tile dimensions, shown in Table **??**, reveals that **all models performed best with tiles of 512x512 pixels.**

Among the models, *EfficientNet-B5* delivered the highest scores across all metrics. At 512x512 pixels, EfficientNet-B5 recorded an accuracy of 93.32%, an mIoU of 88.87%, and a mean Dice coefficient of 93.24%. These results indicate that EfficientNet-B5 is highly effective at capturing detailed features, making it particularly suitable for tasks requiring high precision.

However, it is important to note that **MobileNetV2's performance was remarkably close to that of EfficientNet-B5**, despite its lighter architecture of 3.4 millions trainable parameters against EfficientNet-B5's 28.3 millions. At the same tile size of 512x512 pixels, MobileNetV2 achieved an accuracy of 92.89%, an mIoU of 88.04%, and a mean Dice coefficient of 92.65%. These scores are only about 1% lower than those of EfficientNet-B5, highlighting MobileNetV2's efficiency and possible use on devices with limited computing power such as drones.

ResNet34 also performed well, particularly at the 512x512 pixel tile size, with an accuracy of 93.27%, an mIoU of 88.64%, and a mean Dice coefficient of 93.05%. While ResNet34's performance was strong, it did not significantly surpass that of MobileNetV2, further emphasizing how well MobileNetV2 balances performance and efficiency.

Encoder	Tiles	Accuracy	mIoU	mDice
ResNet34	**512**	**93.27**	**88.64**	**93.05**
ResNet34	768	93.17	88.36	92.94
ResNet34	1000	90.97	84.89	90.73
ResNet34	2000	88.90	81.36	88.56
EfficientNet-B5	**512**	**93.32**	**88.87**	**93.24**
EfficientNet-B5	768	92.63	87.64	92.50
EfficientNet-B5	1000	91.53	85.70	91.27
EfficientNet-B5	2000	90.79	84.35	90.58
MobileNetV2	**512**	**92.89**	**88.04**	**92.65**
MobileNetV2	768	91.90	86.32	91.61
MobileNetV2	1000	90.89	84.65	90.58
MobileNetV2	2000	89.14	81.74	88.79

Table 4: Final performance metrics for different encoders and tiles dimensions in pixel. Accuracy, weighted mean IoU (mIoU) and weighted mean Dice (mDice) are expressed as percentages %.

4.2 Comparison of Per-class IoU

The per-class Intersection over Union (IoU) analysis provides a detailed comparison of the three best-performing network configurations on 512x512 pixel tiles. The results, presented in Table [**??**], highlight the strengths and limitations of each model across various semantic categories..

EfficientNet-B5 consistently outperformed the other models in most classes, achieving the highest IoU in 13 out of the 23 classes. ResNet34 also performed well, particularly in classes such as 'dirt' and 'pool,' where it marginally surpassed EfficientNet-B5.

MobileNetV2 delivered remarkably close results to the other two models, despite being the smallest model with only 3.4 million parameters. In many classes, the IoU difference between

MobileNetV2 and the best-performing model was minimal, often within 1%. This *balance* between model size and performance underscores MobileNetV2's potential for deployment in resource-constrained environments, such as drones, where both computational efficiency and high segmentation accuracy are essential.

Class	EffNet-B5	ResNet34	MobNetV2
paved-area	**95.25%**	95.13%	94.53%
grass	93.89%	94.10%	**94.66%**
roof	82.88%	**83.82%**	77.18%
gravel	80.14%	**82.19%**	81.39%
vegetation	**70.00%**	68.57%	66.36%
obstacle	64.72%	**64.76%**	60.49%
dirt	53.32%	**55.03%**	53.61%
wall	**62.27%**	57.67%	56.80%
water	50.58%	48.66%	**51.94%**
tree	**46.20%**	45.46%	40.69%
bald-tree	28.88%	**31.88%**	28.88%
person	**64.78%**	62.29%	56.56%
fence	**37.49%**	35.98%	33.22%
car	**29.90%**	26.39%	29.53%
rocks	**34.42%**	31.64%	31.92%
pool	27.99%	**28.95%**	24.49%
window	26.09%	**26.87%**	22.78%
ar-marker	**24.80%**	20.93%	21.04%
unlabeled	0.35%	**0.54%**	0.29%
bicycle	**18.98%**	16.86%	17.48%
fence-pole	**5.01%**	4.81%	4.06%
door	**1.01%**	0.55%	0.46%
dog	3.47%	3.10%	**3.48%**

Table 5: Per class IoU comparison for EfficientNet-B5, ResNet34, and MobileNetV2. Values are expressed as percentages.

4.3 Qualitative Results

Figure [**??**] provides a qualitative comparison of the segmentation performance of EfficientNet-B5, ResNet34, and MobileNetV2 using 512x512 pixel tiles. Each figure features a 2x3 grid layout that includes the original image, ground truth mask, predicted mask, model confidence map, overlaid mask on the original image, and a difference mask highlighting regions of incorrect predictions.

Both EfficientNet-B5 and MobileNetV2 exhibit a high level of accuracy in their predicted masks. The difference masks show that most misclassifications occur along object boundaries, which are notoriously difficult areas for segmentation models. Additionally, as shown in Figure [**??**], the accuracy around object edges tends to decrease with increasing tile dimensions. This decline in performance could be attributed to the small size of objects within the classes, especially when viewed from a high altitude. Furthermore, resizing larger tiles to 256x256 pixels may result in a greater loss of information, amplifying this issue.

Despite its smaller size, MobileNetV2 delivers performance that is on par with its more complex counterparts, highlighting its efficiency and suitability for resource-constrained environments, such as deployment on drones.

5 Discussion on Drone Deployment

Deploying deep learning models for real-time semantic segmentation on drones presents significant challenges, primarily due to the limited computational resources available on-board. Although MobileNetV2 is designed for efficiency on edge devices, its performance may still be constrained by the restricted processing power and battery capacity of drones.

During testing, MobileNetV2 achieved approximately 41 frames per second (fps) on a Nvidia RTX 3060 Ti GPU. However, this performance would likely decrease significantly when deployed on a drone, which could limit its effectiveness in real-time segmentation tasks. *Several strategies can be considered to mitigate these limitations.*

5.1 Remote Image Processing

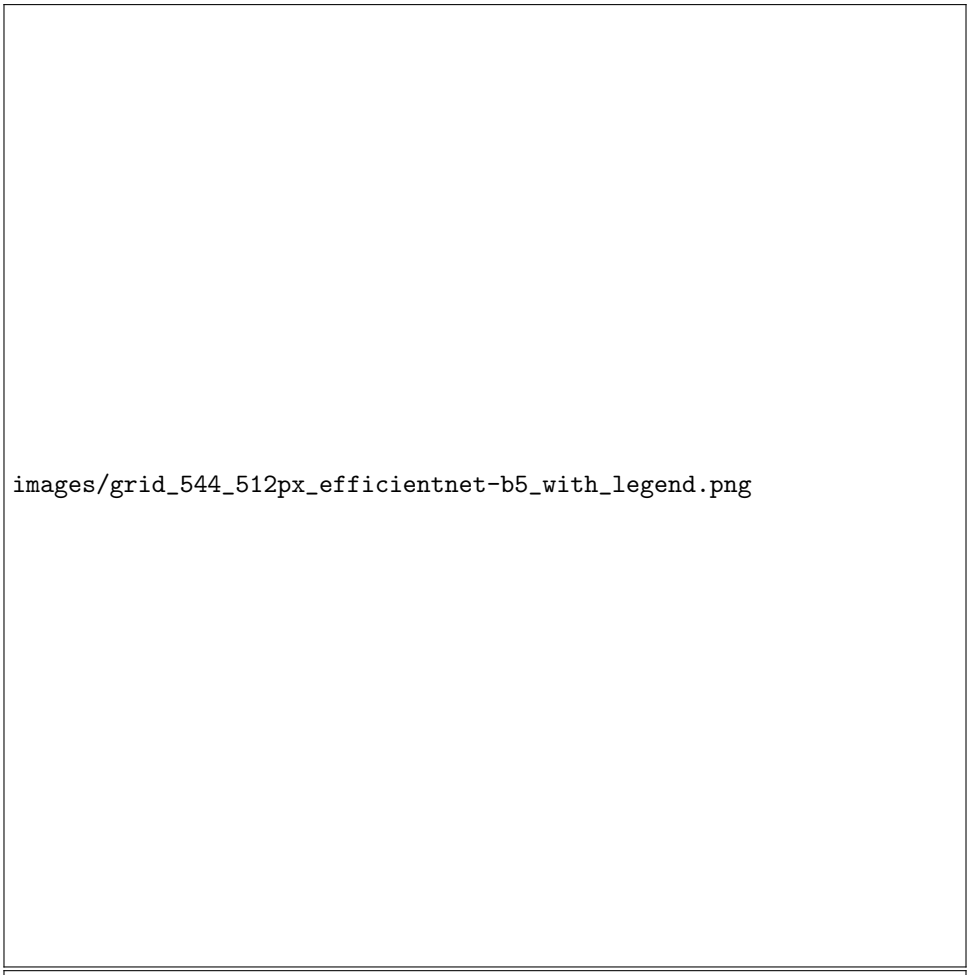
One approach is to offload the image processing tasks to a nearby ground station or cloud server. By transmitting raw image data to a more powerful computing system, the drone's computational burden can be reduced. This approach takes advantage of external resources to perform segmentation and then sends the processed results back to the drone. While effective, this method relies on a stable communication link and may introduce latency.

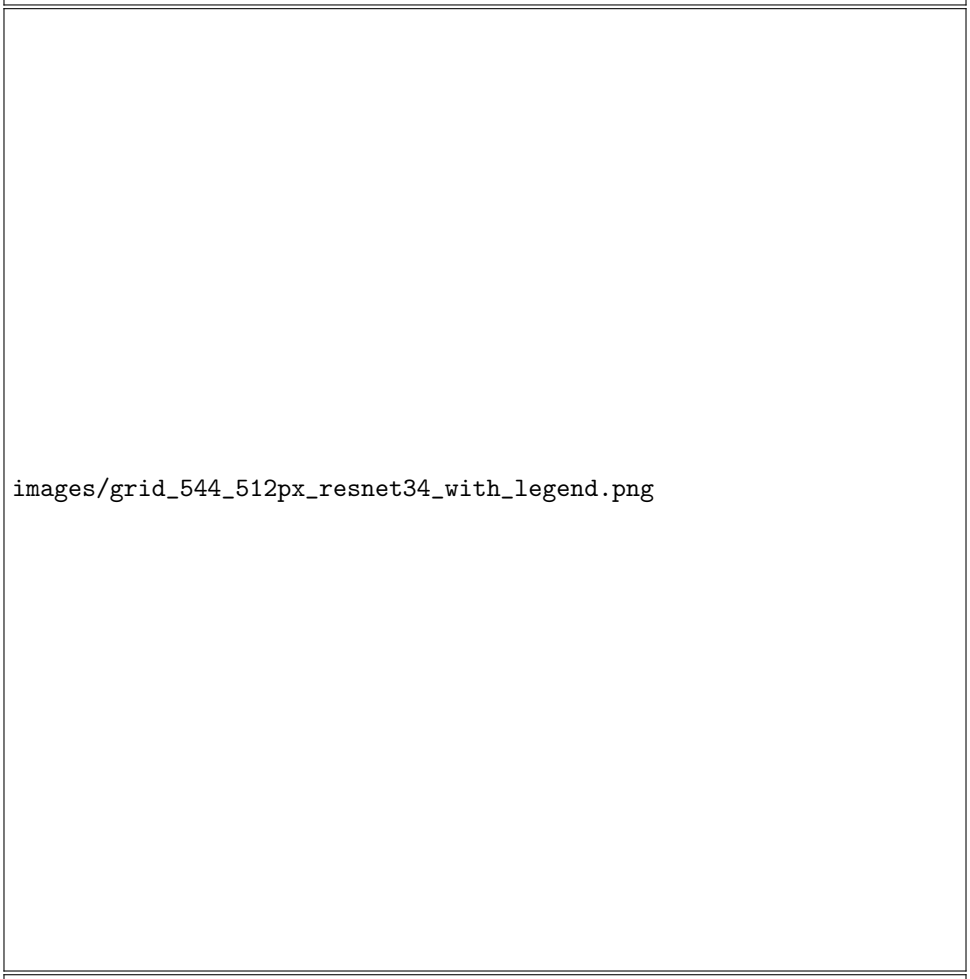
5.2 Frame Rate Reduction

Another viable strategy is to decrease the frame rate of the drone's camera. Since the objects of interest in an urban area are typically static or change slowly, processing fewer frames per second can significantly lower the computational demand. This trade-off can help conserve battery life and maintain operational efficiency while still achieving accurate segmentation results.

5.3 Model Quantization

Model quantization is an effective technique to address the computational challenges of deploying neural networks on drones. This process involves converting a model's floating-point weights and activations into lower precision formats, such as integers, which reduces both the model's size and its computational requirements [**?**]. Quantization enhances the model's efficiency, making it more suitable for real-time applications on resource-constrained devices like drones. In addition to


images/grid_544_512px_efficientnet-b5_with_legend.png


images/grid_544_512px_resnet34_with_legend.png

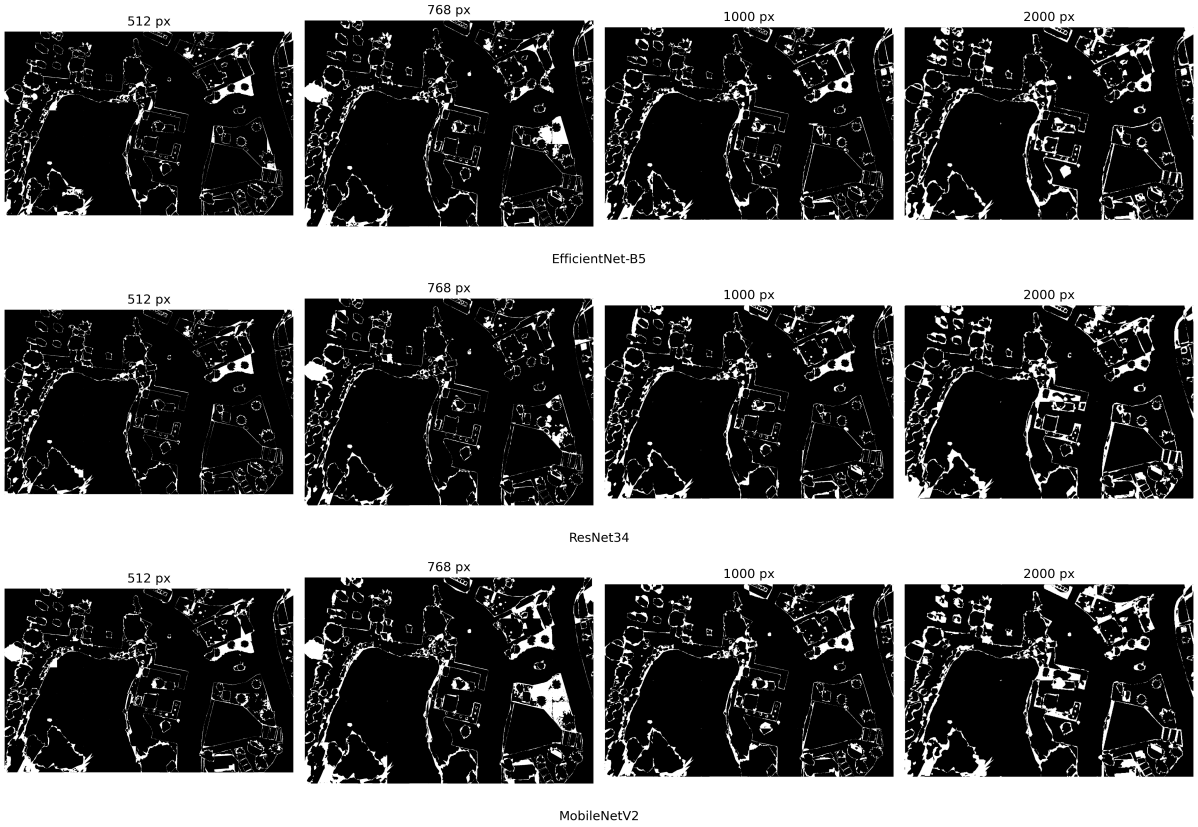


Figure 3: Segmentation performance difference over increasing tile dimensions for three models: EfficientNet-B5 (top), ResNet34 (middle), and MobileNetV2 (bottom).

quantization, model pruning is a valuable technique. This method reduces the number of connections within the network by keeping only the most essential ones, thus simplifying the model.

The deep compression technique [?], which combines pruning, quantization, and Huffman coding, can substantially reduce the storage and computational needs of neural networks. Pruning alone can decrease the number of connections by 9× to 13×, and quantization can further compress the model by reducing the bit-width of weights from 32 bits to as few as 5 bits, significantly decreasing model size.

For example, the deep compression technique has demonstrated a reduction in model storage by 35× for AlexNet, from 240MB to 6.9MB, and by 49× for VGG-16, from 552MB to 11.3MB, without sacrificing accuracy. This compression allows models to fit within on-chip SRAM caches rather than relying on off-chip DRAM memory. Additionally, compressed models exhibit 3× to 4× layer-wise speedup and 3× to 7× improvements in energy efficiency, which translates to enhanced battery life and computational performance on various hardware platforms, including mobile GPUs [?].

Incorporating these compression techniques can make models like MobileNetV2 more viable for drone deployment, optimizing both computational efficiency and practical application.

5.4 Specifically designed workstations

A final solution could be to deploy the models on workstations specifically designed for edge devices. These tailored workstations, such as the Jetson AGX Xavier, offer a balance between performance and power efficiency, making them ideal for real-time processing in resource-constrained environments. The Jetson AGX Xavier, for instance, is an AI hardware platform designed for autonomous machines, offering the performance of a GPU workstation in a compact form factor with power consumption under 30W. This platform is particularly well-suited for drones, robots, and other autonomous systems, providing the necessary computational power to run deep learning models like MobileNetV2 while maintaining operational efficiency in the field [?].

In support of this, Safavi et al. [?] demonstrated the potential of such workstations by training a UNet model with MobileNetV2 as the encoder and converting it to TensorRT engines. Their study involved iterating through 1,000 static frames over 20 rounds on a Jetson workstation, yielding an inference rate of 15 FPS with a latency of 0.07 seconds and an average power consump-

tion of just 5.9 watts. These results indicate that deploying deep learning models on specifically designed edge workstations is feasible and effective.

Moreover, combining these workstations with model quantization could further enhance the model's inference capability, with only a minor loss in accuracy. Quantization would reduce the computational load and power requirements even further, making it a compelling approach for optimizing real-time performance on drones and other autonomous platforms.

6 Discussion on Tile Size

Small Tile Size Wins: All models performed best with tiles of 512x512 pixels, indicating that smaller tile sizes are more effective for capturing detailed features and improving segmentation accuracy. Many classes in the dataset consist of small objects at the drone altitude (e.g., person, window, door, dog). Given that all tiles are resized to 256x256 pixels, the 512x512 pixel tiles capture the most detailed features. However, a larger tile size (for example, 768x768 pixels) could provide a better balance in terms of computational requirements and accuracy, as it reduces the number of tiles per image by half, resulting in only about a 1% decrease in performance (35 tiles at 768x768 pixels versus 77 tiles at 512x512 pixels). Increasing the tile size further to 1,000x1,000 pixels could lower the computational requirements of the network by a factor of 3, decreasing the number of tiles from 77 per image to 24. This reduction would lead to higher frames per second (FPS) for detection. The choice of optimal tile dimensions ultimately depends on the application's priorities: whether to prioritize higher accuracy at the expense of FPS or to accept slightly lower accuracy for improved FPS.

7 Conclusion

This study conducted a comprehensive evaluation of the DeepLabv3+ architecture using EfficientNet-B5, ResNet34, and MobileNetV2 encoders for semantic segmentation of urban drone imagery, focusing on the Semantic Drone Dataset. Our findings reveal several key insights:

- **Small Tile Size Wins:** All models performed best with tiles of 512x512 pixels. However, these tile dimensions impose significant computational requirements due to the high number of tiles per image (77 tiles). Larger tile sizes, such as 768x768 pixels or 1,000x1,000 pixels, offer nearly equivalent performance while reducing the number of tiles per image by a factor of 2 or 3, respectively, thereby maintaining high performance with lower computational demands.
- **EfficientNet-B5** consistently delivered the highest scores across all metrics, achieving an accuracy of 93.32%, an mIoU of 88.87%, and a mean Dice coefficient of 93.24% at 512x512 pixels. **ResNet34** followed closely, with an accuracy of 93.27%, an mIoU of 88.64%, and a mean Dice coefficient of 93.05%. However, these larger models are not fit for running on drones due to the limited computing power and battery on-board.
- **MobileNetV2** demonstrated a remarkable balance between performance and efficiency, achieving an accuracy of 92.89%, an mIoU of 88.04%, and a mean Dice coefficient of 92.65% at 512x512 pixels. The differences in performance metrics between EfficientNet-B5 and MobileNetV2 are shown in Table [**??**], with the delta of performance consistently smaller than 1%. Despite its lighter architecture, MobileNetV2's performance was impressively close to that of EfficientNet-B5, making it particularly well-suited for applications in drone autonomous driving that require rapid, on-the-fly image analysis.

Metric	EffNet-B5	MobNetV2	Delta
Accuracy	93.32	92.89	0.43
Mean IoU	88.87	88.04	0.83
Mean Dice	93.24	92.65	0.59

Table 6: Performance comparison across different metrics (expressed as percentages %) of EfficientNet-B5 and MobileNetV2 with tiles of 512x512 pixels. Notice that the delta is smaller than 1%.

However, deploying MobileNetV2 on drones presents challenges due to the limited on-board resources. Safavi et al. [**?**] have demonstrated that deploying MobileNetV2 on drones is feasible using specifically designed workstations like the Jetson AGX Xavier. Their study showed that MobileNetV2 could achieve real-time performance with acceptable power consumption, making it a strong candidate for drone applications.

To further enhance MobileNetV2's deployment on drones, several strategies should be explored:

- **Remote Image Processing:** Offloading computational tasks to ground stations or cloud servers can reduce the on-board computational burden.
- **Frame Rate Reduction:** Lowering the frame rate can decrease the computational load while still providing effective segmentation in slowly changing environments.
- **Model Quantization and Pruning:** These techniques can optimize model size and computational efficiency, potentially improving

inference times and reducing power consumption with minimal impact on accuracy.

These strategies can help bridge the gap between the high accuracy of larger models like EfficientNet-B5 or ResNet34 and the computational limitations of drone-based applications.

In conclusion, while EfficientNet-B5 offers marginally superior segmentation accuracy, MobileNetV2 emerges as a highly promising solution for drone-based urban imagery analysis. Its balance of efficiency and performance, coupled with appropriate deployment strategies, positions it well for real-world applications, such as autonomous drone navigation, drone deliveries, and infrastructure inspection. Future research should focus on further optimizing these models for edge devices and for specifically designed workstations.

References

[1] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 833–851, 2018.

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, 06 2016.

[3] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Conference: 2019 International Conference on Machine Learning ICML*, 2019.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018.

[6] S. Han, H. Mao, and W. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Conference: 2016 International Conference on Learning Representations ICLR*, 10 2016.

[7] T. Tommy, "Semantic segmentation using deep neural networks for mavs.," 2022.

[8] F. Safavi, I. Ali, V. Dasari, G. Song, T. Zhu, and M. Rahnemoonfar, "Efficient semantic segmentation on edge devices," 2023.