Sky Segmentation with Deep Learning for Embedded Systems

Vinicius Wolosky Muchulski
Department of Computer Science
Federal University of Santa Catarina (UFSC) - Brazil
Email: vinicius.muchulski@grad.ufsc.br

Roberto Rodrigues-Filho
Department of Computer Science
University of Brasília (UnB) - Brazil
Email: roberto.filho@unb.br

Antônio C. Sobieranski
Department of Computer Science
Federal University of Santa Catarina (UFSC) - Brazil
Email: a.sobieranski@ufsc.br

Alison R. Panisson
Department of Computer Science
Federal University of Santa Catarina (UFSC) - Brazil
Email: alison.panisson@ufsc.br

***Abstract*—This paper presents a semantic sky segmentation model designed to run on low-power devices with limited computational resources. Our approach employs a convolutional neural network based on the U^2-Net Lite architecture to produce a binary mask corresponding to the sky region in a given input image. The model was trained using a dataset of sky segmentation masks generated by SAM3. Experimental results show that the proposed model achieves approximately 95,8% IoU on test images from the SAM3-generated dataset. The model implementation supports variable input image resolutions, allowing deployment across a wide range of embedded platforms. Edge AI experiments demonstrated successful deployment of the quantized model on an ESP32-S3 microcontroller using 96×96 pixel inputs and efficient inference on a Raspberry Pi 5 at multiple input resolutions.**

I. Introduction

Visual foundation models, such as SAM3 [1], represent the state of the art in addressing fundamental problems in computer vision, including semantic segmentation and object detection. However, most of these models operate at very large scales, comprising billions of parameters, and consequently demanding substantial computational resources for inference.

In contrast, many edge AI systems, particularly autonomous robots and Unmanned Aerial Vehicles (UAVs), must operate under strict constraints on power consumption, memory, and onboard processing capabilities, while still requiring reliable perception in real time to support safe navigation and decision-making. This mismatch between the computational demands of state-of-the-art vision models and the hardware limitations of embedded platforms poses a significant challenge for deploying advanced perception algorithms in practical scenarios.

In this context, applications that require real-time perception on low-power devices, such as edge AI systems, robotics, and UAVs, face significant challenges in leveraging recent advances in deep learning-based computer vision due to strict constraints on computation, memory, and energy. This limitation is particularly critical in tasks such as sky segmentation, which must remain reliable under highly variable environmental conditions, illumination, and weather. In such scenarios, traditional image processing techniques often fail to provide the accuracy and robustness required for real-world deployment [2].

To address these challenges, we propose a lightweight binary segmentation model specifically designed to balance segmentation accuracy, computational efficiency, and deployment flexibility across different classes of embedded hardware. Additionally, we introduce a data generation pipeline in which SAM3 is used to automatically produce segmentation masks for a customized sky dataset. This dataset is then used to train a compact model that achieves performance close to SAM3, while being suitable for deployment on resource-constrained devices. The main contributions of this work are as follows: (i) the design of a lightweight sky segmentation model based on a modified U^2-Net Lite architecture, optimized for low-latency inference and robustness to quantization on resource-constrained devices; (ii) a scalable data generation pipeline that leverages SAM3 prompt-based segmentation to automatically produce a large and diverse sky segmentation dataset from unlabeled images; (iii) a comprehensive experimental evaluation, including dataset quality assessment against ADE20K, demonstrating that the generated pseudo-labels achieve consistency with human-annotated ground truth; (iv) an analysis of deployment efficiency, showing that the proposed model maintains high accuracy after INT8 quantization and can be deployed across different classes of embedded platforms, including the ESP32-S3 microcontroller and the Raspberry Pi 5; (v) a comparative study using DeepLabV3+ with MobileNetV2 and ResNet34 backbones, highlighting the trade-offs between model complexity and segmentation performance on the proposed dataset.

II. Background

A. Sky Segmentation

Sky segmentation is a binary semantic segmentation task that assigns each pixel in an image to either sky or non-sky [2]. It plays a key role in perception pipelines for autonomous systems, particularly UAVs, where reliable sky

detection supports horizon estimation, obstacle avoidance, and robot navigation [2].

Despite its apparent simplicity, the task is challenging due to high variability in environmental conditions (e.g., fog, twilight, and direct sunlight) and subtle visual similarities between sky and other regions such as water. Classical methods based on color thresholding can perform reasonably well but lack robustness across diverse scenarios [3]. This has motivated the adoption of deep learning approaches, which achieve higher accuracy but often require substantial computational resources, limiting their applicability on embedded platforms.

In this work, we present deep learning-based techniques for sky segmentation tailored to resource-constrained environments.

B. Deep Learning and Convolutional Neural Networks for Segmentation

Deep learning has provided significant advances to computer vision, particularly in image classification, object detection, and semantic segmentation, by enabling neural networks to learn hierarchical representations directly from raw data. Among these approaches, Convolutional Neural Networks (CNNs) have become the dominant architecture for image analysis, as they effectively capture spatial features such as edges, textures, and shapes, making them well suited for pixel-wise prediction tasks.

Several CNN-based architectures have been proposed for segmentation, with U-Net [4] standing out due to its encoder-decoder structure and skip connections, which preserve spatial details while incorporating contextual information. These properties make U-Net effective for segmentation tasks, even with limited annotated data. For this reason, this work adopts architectures derived from U-Net. The following subsections describe the U-Net model and its evolution to U^2-Net.

C. U-Net and U^2-Net Architectures

U-Net [4] is a CNN originally proposed for biomedical image segmentation. It follows an encoder-decoder structure, where the encoder extracts contextual features and the decoder progressively restores spatial resolution. A key component of U-Net is the use of skip connections, which combine high-resolution features from the encoder with the upsampled outputs of the decoder, enabling precise localization and accurate segmentation.

As an evolution of U-Net, U^2-Net [5] introduces a nested U-shaped architecture designed for Salient Object Detection. Its core component is the Residual U-Block (RSU), which captures multi-scale contextual features within a single stage while preserving spatial resolution. This design mitigates the loss of fine details typically caused by repeated pooling operations. Each RSU block combines local feature extraction, a U-shaped internal structure for multi-scale representation, and a residual connection to fuse features effectively.

The original U^2-Net includes a full model with approximately 44 million parameters and a lightweight variant, U^{2}Net-lite, with a size of about 4.7 MB. In this work, we adopt the U^{2}Net-lite architecture and implement it using TensorFlow and TensorFlow Lite to enable efficient sky segmentation on resource-constrained devices.

III. Sky Segmentation in Embedded Systems

In this section, we present our approach to sky segmentation for edge devices with limited computational resources. First, Section III-A describes the dataset constructed to train the proposed model. Then, Section III-B introduces a lightweight sky segmentation model designed for deployment on resource-constrained embedded platforms. Fig. 1 shows the pipeline for our approach.

A. Dataset Construction and Evaluation

Obtaining a representative dataset for sky segmentation is challenging due to the need to cover diverse environments and atmospheric conditions [2]. To address this, we curated a large collection of approximately 27k unlabeled images from public sources such as Kaggle, Roboflow Universe, and HuggingFace, prioritizing diversity in scene types and environmental conditions. The dataset includes images from multiple domains, such as road and maritime scenarios, as well as landscape, weather, and cloud classification datasets. Maritime scenes were specifically included to reduce confusion between visually similar regions such as sky and ocean.

The collected images were automatically annotated using SAM3, leveraging its prompt-based capability. For each image, the prompt *sky* was used to generate binary masks (sky vs. non-sky), transforming the dataset into a fully annotated segmentation corpus. This prompt-guided zero-shot pipeline enables efficient knowledge transfer, allowing compact models to be trained without manual labeling.

To evaluate the quality of the proposed approach, we compared the generated masks against those from the ADE20K dataset [6], using a publicly available version[1] adapted to binary sky segmentation. In this setup, all non-sky classes were merged into a single category. The SAM3-generated masks achieved an IoU of 90.84% relative to the ADE20K ground truth. However, manual inspection suggests that much of the discrepancy is due to limitations in the ADE20K annotations, which often exhibit coarse boundaries or omit visible sky regions. In several cases, SAM3 produces more detailed segmentation, indicating that the reported IoU likely underestimates the true mask quality, as illustrated in Fig. 2.

The final dataset, comprising 27,374 images, was split into 70% for training, 15% for validation, and 15% for testing. Images were normalized to `float32` in the range $[0, 1]$, and data augmentation techniques, including brightness and contrast adjustments, rotations, and horizontal flipping were applied. Additionally, dynamic multi-scale resizing was used during training, where input resolutions are randomly selected, improving generalization and enabling flexible deployment.

[1] https://www.kaggle.com/datasets/awsaf49/ade20k-dataset

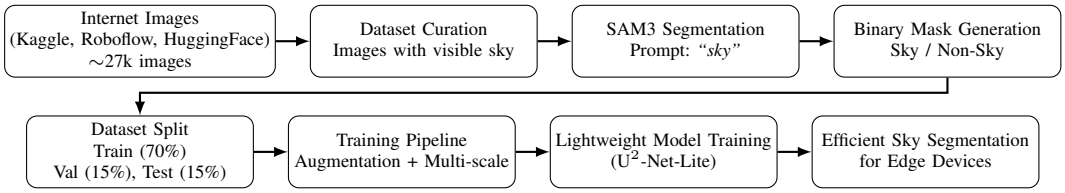


Fig. 1. Pipeline of the proposed approach.



Fig. 2. Example comparison between SAM3-generated masks and ADE20K ground truth.

B. Proposed Embedded Model

The architecture proposed for the sky segmentation task is a customized and compact U^2-Net-Lite model, which we call W-Net[2], employing modified RSU residual blocks designed to reduce both training and inference computational costs. The model was restructured to maximize efficiency on edge devices. The final architecture was selected after evaluating twenty lightweight network variants, including compact U-Net and modified U^2-Net models. Besides segmentation accuracy, the selection considered robustness in challenging scenarios, such as images containing only sky or no sky at all, where several candidate models produced false positive segmentation.

Compared to the original U^2-Net-Lite, which employs six encoder-decoder stages and heterogeneous residual blocks (RSU7, RSU6, RSU5, RSU4, and RSU4F), W-Net adopts a simplified architecture with only four encoder-decoder stages and exclusively RSU4F blocks. Since RSU4F replaces internal pooling with dilated convolutions, spatial resolution is preserved while maintaining a large receptive field. Fig. 3 illustrates the proposed architecture.

Although the resulting architecture contains slightly more parameters than the original U^2-Net-Lite due to the 256-channel bottleneck, it maintains low inference latency and proved more robust to INT8 quantization in our experiments, exhibiting negligible accuracy degradation compared to its FP32 version. As a consequence of reducing the architecture to four depth stages, the adopted deep supervision strategy generates only five probability maps (four side outputs and one final fused output $d0$). This mechanism injects gradient signals into intermediate layers, accelerating convergence while preserving both global context and fine spatial details. The final model contains ≈2.18M parameters. TensorFlow

[2]W-Net is an abbreviation for Wolosky-Net

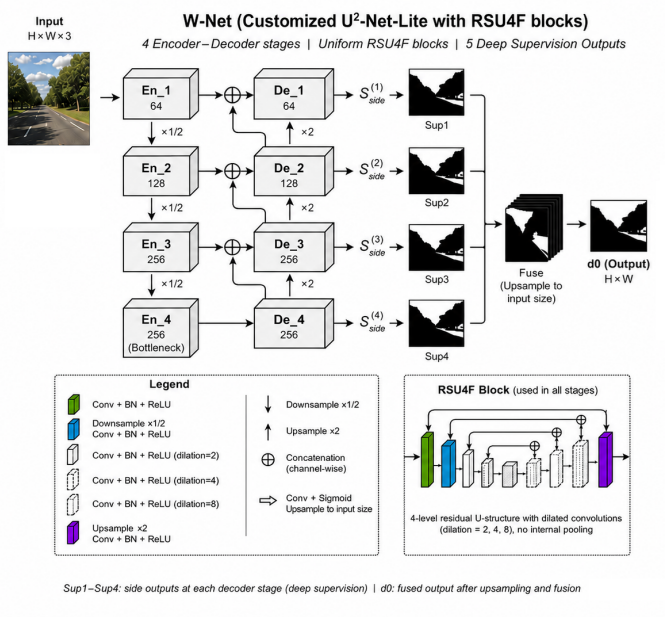


Fig. 3. Illustration of the proposed W-Net architecture.

was selected due to its native support for TensorFlow Lite, facilitating deployment on embedded platforms. After training, the model was exported in `.keras`, `.tflite`, and `.onnx` formats.

To enable deployment on resource-constrained hardware, post-training quantization (PTQ) was applied during TensorFlow Lite conversion using the standard TensorFlow optimization pipeline and a representative calibration dataset. Full-integer INT8 quantization was adopted instead of hybrid quantization because one of the target platform, ESP32-S3, supports only fully quantized integer models. Consequently,

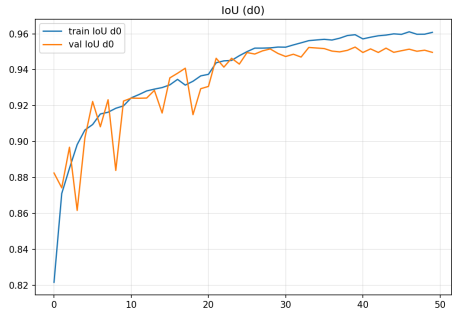


Fig. 4. Training and validation IoU of the proposed model.

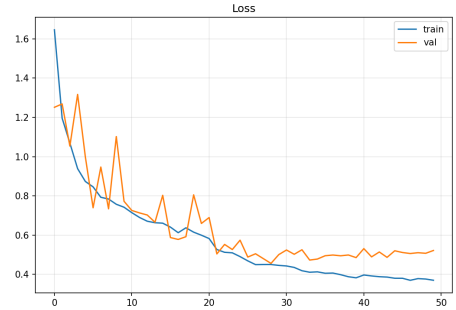


Fig. 5. Training and validation Loss of the proposed model.

two TensorFlow Lite models were generated: a full-precision FP32 version and a full-integer INT8 version. Likewise, the ONNX model was exported in both FP32 and INT8 formats, allowing deployment across different inference environments[3].

IV. Evaluation

A quantitative evaluation was conducted on the held-out test set, corresponding to 15% of the total dataset. Unless otherwise stated, all quantitative evaluation metrics reported in this section were obtained on the desktop platform using the 320×320 test set. The base model (Keras) achieved an IoU of 95.84%. Figures 4 and 5 show the IoU and loss evolution during training, indicating stable convergence of the model. Both training and validation loss decrease consistently, while the IoU increases throughout training, suggesting effective learning. A small gap between training and validation curves can be observed after approximately 25 epochs, indicating mild overfitting. However, this gap remains limited, and the validation metrics remain stable, suggesting good generalization performance without significant degradation. Techniques such as restoring the best weights with early stopping were adopted to prevent overfitting and retain the best-performing checkpoint.

To validate its feasibility for edge deployment, the performance of the exported TensorFlow Lite models was compared. The floating-point version (FP32) achieved an IoU of 95.82%, while the fully quantized version (INT8) obtained 95.75%. The marginal degradation of only 0.09% relative to the base model demonstrates the robustness of the proposed architecture to quantization, enabling efficient inference without significant loss of accuracy. The comparative results are summarized in Table I.

TABLE I
PERFORMANCE COMPARISON (IOU) WITH 320X320 IMAGE RESOLUTION BETWEEN THE ORIGINAL MODEL AND THE EXPORTED TFLITE VERSIONS.

Model	Format	Precision	IoU (%)
W-Net (Base)	Keras	FP32	95.84
W-Net	TFLite	FP32	95.82
W-Net	TFLite	INT8	95.75

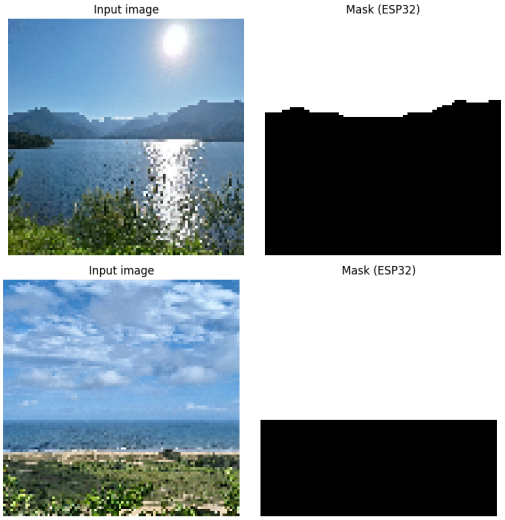


Fig. 6. Segmented Image resized to 96 X 96 and tested in the ESP32-S3.

To evaluate the proposed model in representative edge AI scenarios, experiments were conducted on two embedded platforms with different computational capabilities. The ESP32-S3 microcontroller was used to assess deployment feasibility under severe memory constraints, while the Raspberry Pi 5 was used to evaluate inference performance at different input resolutions. The ESP32-S3 deployment used the fully quantized INT8 TFLite model with an input resolution of 96×96 pixels[4] and the TensorFlow Lite Micro framework, enabling inference directly on the microcontroller. The quantized INT8 model occupied approximately 2.19 MB of flash memory. During inference, TensorFlow Lite Micro reserved a 2 MB tensor arena in PSRAM, of which approximately 1.75 MB was effectively used. The average inference time over 21 RGB images was 120.4 s per image. Fig. 6 presents representative segmentation results obtained directly on the ESP32-S3.

Fig. 7 presents representative segmentation results obtained with the ONNX W-Net model on aerial and drone images.[5]

[3]The FP32 ONNX model can also be converted to FP16 when targeting GPU-based inference engines such as TensorRT, providing additional flexibility without requiring a dedicated FP16 export during training.

[4]The ESP32-S3 deployment uses an input resolution of 96×96 pixels due to the limited memory available for TensorFlow Lite Micro inference. Preliminary experiments with higher resolutions (e.g., 320×320 pixels) exceeded the available memory budget, making deployment infeasible on the target hardware.

[5]Training code and additional examples are available at https://github.com/vini-muchulski/Sky-Segmentation-with-Deep-Learning-for-Embedded-Systems-WoloSky-Net.

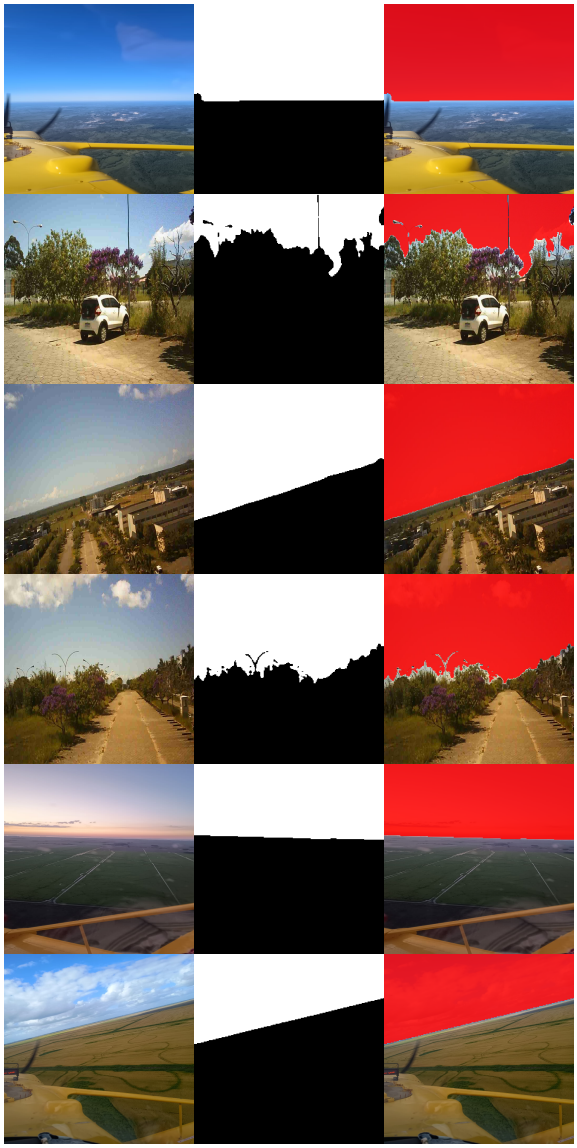


Fig. 7. ONNX W-Net model results in representative images.

On the Raspberry Pi 5 (8 GB RAM), the model was evaluated using different input resolutions with both the FP32 and INT8 TFLite models. Fig. 8 shows the average inference performance achieved under the different resolution and quantization settings. The results indicate that the INT8 model with a 224-pixel input resolution provides an efficient trade-off between segmentation quality and inference speed on hardware without GPU acceleration[6]. Furthermore, memory profiling showed that the benefits of INT8 quantization become more pronounced as the input resolution increases. At resolutions of 96×96, 224×224, and 320×320 pixels, the INT8 model required approximately 4.5, 15.6, and 19.6 MB of additional memory, respectively, compared to 5.2, 25.6, and 51.8 MB for the FP32 model. These results further demonstrate that full-integer quantization substantially reduces the memory footprint, particularly at higher resolutions, making the proposed model more suitable for resource-constrained embedded platforms.

[6]For comparison, the ONNX FP32 model converted to TensorRT was executed on a desktop equipped with an NVIDIA RTX 3060 GPU achieved 56 FPS using a 256×256 input resolution.

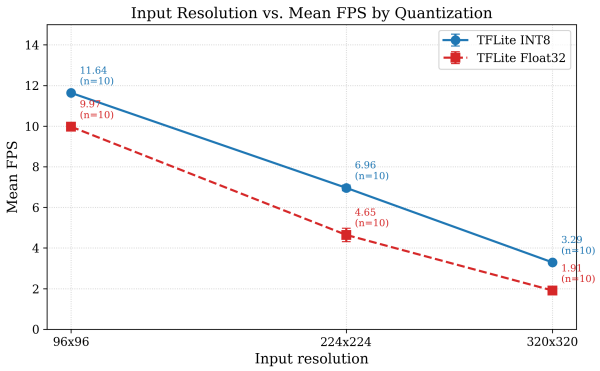


Fig. 8. Resolution vs FPS - Inference in a Raspberry Pi 5.

A. *Transfer Learning with DeepLab*

To complement the sky segmentation techniques explored in this work, we also investigated the use of the binary mask dataset generated with SAM3 to train more widely adopted semantic segmentation architectures.

For the solution of sky segmentation task that we proposed to solve, we trained two models based on the DeepLabV3+ architecture [7] denoted as W-Net-DeepLab and are also made available alongside the W-Net original lite variant. These models differ in the encoder backbone used: MobileNetV2 [8] and ResNet34 [9]. Both encoders were initialized with weights pretrained on ImageNet [10], characterizing a transfer learning approach in which knowledge from generic visual representations is transferred to the specific domain of sky segmentation [11]. The same dataset and same IoU validation metrics were used to evaluate this training process.

MobileNetV2 was selected due to its computational efficiency and suitability for inference on hardware-constrained devices, while ResNet34 was used as a higher-capacity reference architecture, enabling a direct comparison between segmentation performance and computational cost.

To ensure a fair and reproducible comparison between the proposed model and the DeepLabV3+ models, the evaluation was performed with all models exported to the *ONNX* format in full precision (FP32), eliminating potential variations introduced by quantization. Inference was conducted on the same held-out test set used to evaluate the proposed W-Net model. With these complementary models, different levels of computation power can be explored by using these larger

TABLE II
COMPARISON BETWEEN MODELS ON THE TEST SET (ONNX FP32)

Model	**Params**	**Type**	**IoU (%)**
W-Net-DeepLabV3+ ResNet34	22.44M	ONNX FP32	97.23
W-Net-DeepLabV3+ MobileNetV2	4.38M	ONNX FP32	97.05
W-Net	2.18M	ONNX FP32	95.82

models when they are needed. This experiment was not intended to outperform W-Net on embedded hardware, but rather to demonstrate that the automatically generated dataset is sufficiently general to train larger segmentation architectures.

V. Related Work

Sky segmentation has been explored in several domains, including image enhancement, environmental monitoring, and atmospheric analysis. Mihail et al. [2] studied sky segmentation under real-world conditions using a dataset of approximately 100k outdoor webcam images, showing the effects of illumination, solar position, and adverse weather on segmentation performance. Their work proposed a deep ensemble approach, but did not focus on deployment under embedded or real-time constraints.

For mobile photography, Liba et al. [12] proposed a lightweight network for sky segmentation in low-light images, followed by guided-filter edge refinement. Although designed for mobile hardware, the method targets photographic enhancement rather than real-time binary segmentation for autonomous systems.

Sky and cloud segmentation has also been investigated for solar energy forecasting. Martins [13] evaluated U-Net, HRNet, and Detectron2 for cloud segmentation in horizon-facing images, showing that relatively simple CNNs can achieve strong performance with appropriate data augmentation. However, the study focuses on multi-class cloud analysis rather than binary sky segmentation for embedded inference. More recently, Cerentini [14] investigated the use of semantic segmentation outputs to support MLLM-based visual analysis of clouds, showing that structured segmentation can improve downstream visual reasoning.

In contrast, our work combines prompt-based SAM3 segmentation for scalable sky dataset generation with a compact W-Net model optimized for edge deployment. The proposed pipeline addresses both the challenge of obtaining annotated sky segmentation data and the computational constraints of practical embedded inference.

VI. Conclusion

This work presented a customized architecture derived from U^2-Net Lite, optimized for deployment on resource-constrained embedded and edge AI platforms. The proposed model demonstrated effective performance in the binary semantic segmentation task, achieving a good balance between segmentation accuracy and computational efficiency.

The study also validated a training pipeline based on SAM3 for automatically generating high-quality segmentation masks from unlabeled images. This strategy enables scalable dataset construction while avoiding the need for manual annotation.

Post-training INT8 quantization enabled efficient deployment across different classes of embedded hardware, ranging from the ESP32-S3 microcontroller to the Raspberry Pi 5. The experimental results showed only a marginal reduction in segmentation accuracy after quantization while substantially reducing the memory footprint, demonstrating that the proposed model is suitable for resource-constrained edge AI applications. Furthermore, support for multiple input resolutions enables the same model to be deployed on devices with different computational capabilities without requiring architectural modifications.

As future work, the proposed approach could be extended to multi-class semantic segmentation tasks. In particular, the prompt-based capabilities of SAM3 could be exploited to generate labels for multiple semantic categories simultaneously, enabling the development of compact multi-class segmentation models and their evaluation in real-world edge AI scenarios.

Acknowledgments

Part of this work was performed with computational resources provided by Fundação de Amparo a Pesquisa e Inovação do Estado de Santa Catarina (FAPESC), grant nr. 2024TR000090.

References

[1] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Suris, C. Ryali, K. V. Alwala, H. Khedr, A. Huang *et al.*, "Sam 3: Segment anything with concepts," *arXiv preprint arXiv:2511.16719*, 2025.

[2] R. P. Mihail, S. Workman, Z. Bessinger, and N. Jacobs, "Sky segmentation in the wild: An empirical study," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016, pp. 1–6.

[3] N. Kulkarni, "Color thresholding method for image segmentation of natural images," *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 4, no. 1, pp. 28–34, 2012.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer, 2015, pp. 234–241.

[5] X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. R. Zaiane, and M. Jagersand, "U2-net: Going deeper with nested u-structure for salient object detection," *Pattern recognition*, vol. 106, p. 107404, 2020.

[6] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," 2018. [Online]. Available: https://arxiv.org/abs/1608.05442

[7] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," 2018. Available: https://arxiv.org/abs/1802.02611

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019. [Online]. Available: https://arxiv.org/abs/1801.04381

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.

[11] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[12] O. Liba, L. Cai, Y.-T. Tsai, E. Eban, Y. Movshovitz-Attias, Y. Pritch, H. Chen, and J. T. Barron, "Sky optimization: Semantically aware image processing of skies in low-light photography," 2020. [Online]. Available: https://arxiv.org/abs/2006.10172

[13] B. J. Martins, "Segmentation and classification of individual clouds in images captured with horizon-aimed cameras for nowcasting of solar irradiance absorption," Master's Thesis, Federal University of Santa Catarina, Florianópolis, Brasil, 2024.

[14] A. Cerentini, "Aprimorando a análise visual detalhada de nuvens por LLMs multimodais através da integração de segmentação semântica como ferramenta externa," Tese de Doutorado, Universidade Federal de Santa Catarina, Florianópolis, SC, Brasil, 2025, programa de Pós-Graduação em Ciência da Computação.