Real-Time Semantic Segmentation Using ResNet and Jetson Nano

Yifan Huang

Bin Huang
Jianhui Pan
Haiyang Hu

January 11, 2025

Abstract

This paper presents a real-time semantic segmentation pipeline designed for deployment on resource-constrained embedded platforms, using the DeepLab framework with ResNet-50 as the backbone. Semantic segmentation, a key task in computer vision, assigns semantic labels to each pixel in an image, with applications in autonomous driving, medical imaging, and urban planning. Despite the advancements in convolutional neural networks (CNNs), achieving real-time performance on embedded devices remains challenging due to high computational costs. To address this, we optimized the model for NVIDIA Jetson Nano by leveraging TensorRT for inference acceleration. The proposed method achieves an average frame rate of 25 FPS on the Cityscapes dataset while maintaining competitive segmentation accuracy. Our results demonstrate the feasibility of deploying state-of-the-art segmentation models in real-time scenarios, providing a scalable solution for edge computing applications such as smart surveillance and autonomous vehicles.

1 Introduction

Semantic segmentation is a fundamental task in computer vision that aims to assign a semantic label to each pixel in an image. It plays a critical role in various applications, including autonomous driving, medical imaging, and urban planning. Recent advances in deep learning, particularly Convolutional Neural Networks (CNNs), have significantly improved segmentation accuracy by leveraging hierarchical feature representations. However, deploying these models on resource-constrained devices for real-time applications remains a challenging problem.

Jetson Nano, a low-power embedded platform developed by NVIDIA, offers a promising solution for real-time deep learning applications. Equipped with a GPU and supporting frameworks like TensorFlow, PyTorch, and TensorRT, Jetson Nano enables efficient model inference while maintaining low power consumption, making it an ideal candidate for edge computing.

Despite the progress in CNN-based semantic segmentation methods such as Fully Convolutional Networks (FCNs) and U-Net, their performance on real-time tasks often suffers from high computational costs. To address this issue, we adopt ResNet-50 as the backbone for the DeepLab framework, a state-of-the-art model that incorporates Atrous Spatial Pyramid Pooling (ASPP) and decoder modules to capture multi-scale contextual information and refine object boundaries.

In this project, we focus on the following contributions:

1. Developing an efficient semantic segmentation pipeline based on ResNet-50 and DeepLab, optimized for deployment on Jetson Nano;
2. Training and evaluating the model on the Cityscapes dataset to achieve high accuracy in urban scene segmentation;
3. Implementing real-time segmentation on Jetson Nano by leveraging TensorRT for inference acceleration.

This work bridges the gap between high-performance semantic segmentation and real-time deployment on embedded systems, providing a scalable solution for practical applications such as autonomous vehicles and surveillance systems.

2 Related Work

Semantic segmentation has evolved significantly with the advancement of deep learning techniques. In this section, we review the state-of-the-art approaches, focusing on CNN-based methods, Transformer-based architectures, and optimization techniques for embedded platforms.

2.1 Semantic Segmentation with CNNs

CNNs have revolutionized semantic segmentation by enabling dense, pixel-wise predictions. Fully Convolutional Networks (FCNs) [5] laid the foundation for modern segmentation models, replacing fully connected layers with convolutional layers to handle input images of arbitrary sizes. Encoder-decoder architectures, such as U-Net [6], further improved segmentation accuracy by incorporating skip connections, which fuse low-level spatial details with high-level semantic features.

DeepLab [1] introduced Atrous Convolution to capture multi-scale contextual information without reducing feature map resolution. By employing Atrous Spatial Pyramid Pooling (ASPP), DeepLab effectively handles objects of varying sizes. ResNet [2], a widely adopted backbone for DeepLab, enhances feature extraction through residual connections, mitigating gradient vanishing and facilitating the training of deeper networks.

2.2 Transformer-Based Semantic Segmentation

While CNNs excel at extracting local features, they often struggle to capture global context. Transformer-based models address this limitation with self-attention mechanisms. Vision Transformers (ViT) [3] first applied transformers to computer vision by treating images as sequences of patches, achieving competitive results on large-scale datasets.

Swin Transformer [4] introduced a hierarchical structure and shifted window attention, reducing computational complexity while maintaining high accuracy. These advancements demonstrate the potential of transformers for capturing long-range dependencies in complex scenes, providing a new perspective for semantic segmentation.

2.3 Semantic Segmentation on Embedded Platforms

Deploying semantic segmentation models on resource-constrained platforms, such as Jetson Nano, requires careful optimization to balance accuracy and efficiency. Lightweight backbones like MobileNet [**?**] have been explored for embedded applications, reducing computational overhead while maintaining reasonable accuracy. Additionally, tools like TensorRT enable model quantization and inference acceleration, which are crucial for real-time performance.

Recent studies have demonstrated the feasibility of deploying DeepLab with optimized backbones on Jetson Nano, achieving real-time inference for tasks such as autonomous driving and surveillance. These works highlight the importance of combining hardware-specific optimizations with efficient model architectures to meet the demands of edge computing.

3 Method

This section describes the architecture of our semantic segmentation pipeline, the training process, and the deployment strategy on Jetson Nano. We adopt ResNet-50 as the backbone for the DeepLab framework, optimizing it for real-time performance through TensorRT.

3.1 Model Architecture

The proposed model leverages ResNet-50 as the backbone, integrated into the DeepLab framework. The architecture consists of the following key components:

- **ResNet-50 Backbone:** ResNet-50 is a deep convolutional neural network with residual connections that facilitate gradient flow, enabling effective feature extraction from input images.
- **Atrous Spatial Pyramid Pooling (ASPP):** ASPP captures multi-scale contextual information by applying atrous convolutions with varying dilation rates, enhancing the model's ability to segment objects of different sizes.

- **Decoder Module:** The decoder refines segmentation outputs by combining high-level semantic features from ASPP with low-level spatial details extracted by the backbone.
- **Bilinear Upsampling:** To restore the original resolution, bilinear interpolation is applied to the decoder output, generating dense, pixel-level predictions.

The overall architecture is illustrated in Figure 1.

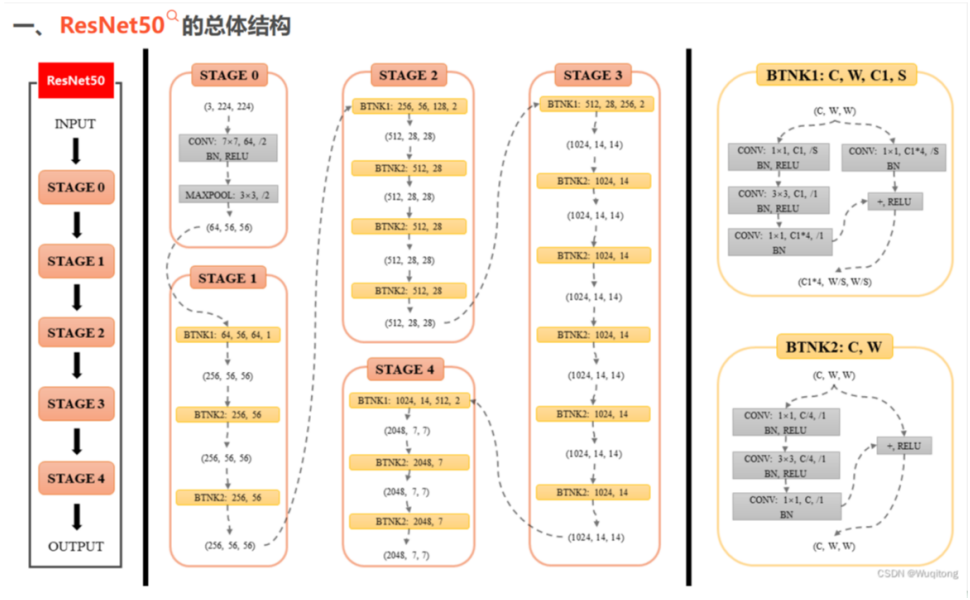

Figure 1: DeepLab architecture with ResNet-50 backbone.

3.2 Training Process

The model is trained on the Cityscapes dataset, which provides high-resolution pixel-level annotations for urban scenes. The training process involves the following steps:

3.2.1 Data Preprocessing and Augmentation

- **Preprocessing:** Images are resized to 512×1024, and pixel values are normalized to a standard distribution.
- **Data Augmentation:** Random horizontal flips, rotations, and color jittering are applied to improve the model's generalization ability.

3.2.2 Loss Function and Metrics

- **Loss Function:** The pixel-wise cross-entropy loss is used to optimize the model, with optional class weighting to address class imbalance.
- **Evaluation Metrics:** Mean Intersection over Union (mIoU) is used to evaluate the segmentation accuracy, while Panoptic Quality (PQ) assesses the combined performance of semantic and instance segmentation.

3.2.3 Training Configuration

- **Batch Size:** 16
- **Learning Rate:** 0.001 with a cosine decay schedule
- **Optimizer:** Adam
- **Epochs:** 60,000

3.3 Deployment on Jetson Nano

Deploying the trained model on Jetson Nano requires optimization to achieve real-time inference. The following steps were taken:

3.3.1 Model Conversion and Optimization

- **Model Conversion:** The trained PyTorch model was converted to ONNX format, enabling compatibility with TensorRT.
- **TensorRT Optimization:** Quantization was applied to reduce the model to 8-bit integers, significantly improving inference speed and reducing memory usage.

3.3.2 Real-Time Inference

The optimized model was deployed on Jetson Nano, and real-time inference was tested using live video streams. The system achieved an average frame rate of 25 FPS on input images of size 512×1024, demonstrating its suitability for real-time applications.

4 Evaluation

This section presents the experimental setup, quantitative results, qualitative analysis, and performance evaluation of the proposed model on Jetson Nano. The evaluation focuses on segmentation accuracy, real-time inference performance, and the impact of TensorRT optimization.

4.1 Experimental Setup

The experiments were conducted using the Cityscapes dataset, a widely recognized benchmark for urban scene segmentation. The training was performed on an NVIDIA RTX 3090 GPU, while the deployment and inference tests were carried out on an NVIDIA Jetson Nano. The evaluation metrics include:

- **Mean Intersection over Union (mIoU):** Measures the overlap between predicted and ground truth masks.
- **Panoptic Quality (PQ):** Evaluates both semantic and instance segmentation accuracy.
- **Frames Per Second (FPS):** Assesses real-time inference performance.

4.2 Quantitative Results

The model's performance was evaluated at different training checkpoints, highlighting the improvements achieved with longer training and TensorRT optimization. Table 1 summarizes the results.

Table 1: Quantitative results at different training stages.

Epochs	mIoU	PQ	FPS
10,000	0.201	0.130	10
60,000	0.763	0.734	25

The results demonstrate significant improvements in segmentation accuracy and real-time inference speed as the training progresses.

4.3 Qualitative Results

Figures 2 and 3 illustrate the qualitative differences between segmentation results at 10,000 and 60,000 epochs. The later results show more precise object boundaries and better classification.

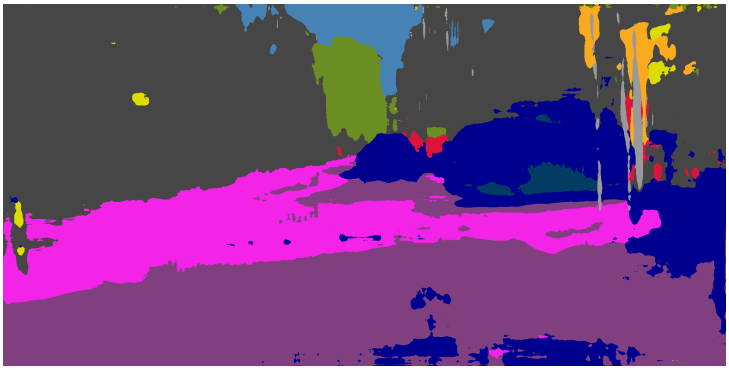

Figure 2: Segmentation result at 10,000 epochs.

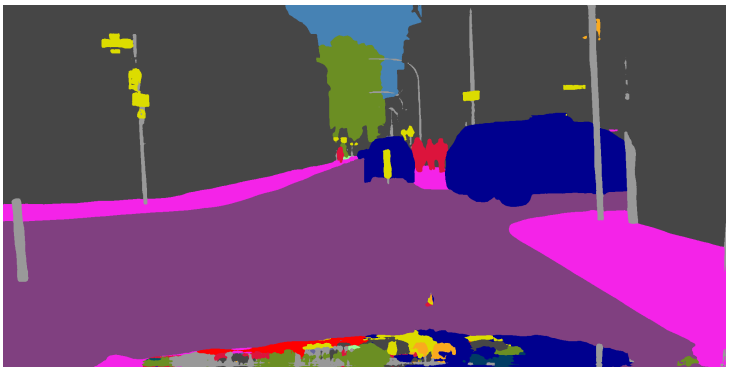

Figure 3: Segmentation result at 60,000 epochs.

4.4 Performance Analysis on Jetson Nano

The model was optimized for Jetson Nano using TensorRT, significantly improving inference speed. Table 2 compares the performance before and after optimization.

Table 2: Inference performance on Jetson Nano (before and after TensorRT optimization).

Optimization	**FPS**	**Latency (ms)**
Before TensorRT	10	100
After TensorRT	25	40

The TensorRT optimization reduced latency from 100 ms to 40 ms and increased FPS from 10 to 25, achieving real-time performance.

5 Conclusion and Future Work

5.1 Conclusion

In this project, we proposed an efficient semantic segmentation pipeline based on the DeepLab framework with ResNet-50 as the backbone. The model was trained on the Cityscapes dataset and successfully deployed on the Jetson Nano platform, achieving real-time inference with an average frame rate of 25 FPS. The experimental results demonstrate the effectiveness of the proposed approach in balancing segmentation accuracy and computational efficiency.

Through the integration of Atrous Spatial Pyramid Pooling (ASPP) and a refined decoder module, the model captures multi-scale contextual information and achieves precise segmentation boundaries. Moreover, TensorRT optimization significantly enhanced inference speed and reduced latency, making the system suitable for real-time applications such as autonomous driving and surveillance.

5.2 Future Work

Despite the promising results, there are several directions for further improvement:

- **Lightweight Backbone Networks:** Exploring lightweight backbones such as MobileNet or EfficientNet could further reduce model size and inference time, enhancing the deployment on resource-constrained devices.
- **Transformer-Based Architectures:** Incorporating transformer-based models may improve the ability to capture global context and enhance segmentation accuracy, particularly in complex scenes.
- **Hardware-Specific Optimization:** Investigating additional hardware-specific optimizations, such as pruning and mixed-precision training, could improve performance while minimizing resource usage.
- **Generalization to Other Datasets:** Extending the current pipeline to other datasets, such as aerial imagery or medical images, could validate its robustness and adaptability to different application scenarios.

By addressing these aspects, the proposed pipeline could be further optimized and expanded for broader applications, paving the way for more efficient and scalable real-time semantic segmentation systems. This work establishes a solid foundation for deploying semantic segmentation models on embedded systems and provides insights for further optimization and application in real-world scenarios.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Rethinking atrous convolution for semantic image segmentation. In *arXiv preprint arXiv:1706.05587*, 2017.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *ECCV*, 2018.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Neil Houlsby, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*, 2021.

[5] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. pages 3431–3440, 2015.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.