Real-Time Human Detection and Segmentation using Mask2Former with a Swin Transformer Backbone

Anonymous Author(s) [1]

Abstract

I present a system for real-time human detection and segmentation that combines the state-of-the-art Mask2Former architecture with a Swin Transformer backbone. Our approach processes images, prerecorded videos, and live webcam feeds to generate accurate segmentation masks for each human instance while achieving an average processing rate of approximately 15 FPS. Experimental evaluations on the COCO panoptic dataset report competitive performance (person AP $\approx$ 0.50 and mask IoU $\approx$ 0.60). I detail both the conceptual model aspects and the practical code implementation, discussing modularity, real-time optimizations, and the the integration of open-source libraries. Applications in construction safety, surveillance, and security are highlighted. Code available at: https://anonymous.4open.science/r/RealTimeHumanDetection-8000/README.md

1. Introduction

Real-time human segmentation is essential in many real-world applications, such as enforcing safety protocols on construction sites, improving public surveillance systems, and monitoring protected areas. In scenarios where split-second decisions are crucial, not only must a system quickly detect the presence of individuals, but it must also delineate their exact boundaries to enable downstream processing.

The deep learning era has seen tremendous progress with transformer-based architectures in computer vision. Among these, Mask2Former (4) has emerged as one of the top methods for segmentation tasks due to its powerful masked attention mechanism. When combined with a Swin Transformer backbone (8), which provides an efficient hierarchical representation of visual features, the resulting system exhibits strong performance in both accuracy and speed. Our work focuses on tailoring this model for human segmentation and, more importantly, on developing robust code that can process images, videos, and live webcam feeds in real time.

[1]Paper under double-blind review.. Correspondence to: Anon <>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

In this paper, I detail the design of our system, discuss key challenges, and present both quantitative and qualitative performance metrics. I also elaborate on the code structure and engineering decisions behind integrating libraries such as HuggingFace Transformers, OpenCV, and Matplotlib.

2. Related Work

The field of object detection and segmentation has evolved significantly in the past decade. Traditional detectors such as Faster R-CNN (10) and YOLO (9) laid the groundwork by demonstrating that deep CNNs could reliably localize objects and provide bounding boxes in real time. However, these approaches typically generate coarse representations of objects without pixel-level precision.

Subsequent advances introduced segmentation-specific methods. Fully Convolutional Networks (FCNs) (11) enabled end-to-end dense prediction, and Mask R-CNN (6) extended detection frameworks by adding a dedicated mask branch for instance segmentation. Although effective, these methods result in high computational costs, making real-time performance challenging without significant optimization.

The emergence of transformer-based architectures has reinvigorated the field. Vision Transformers (ViT) (5) and DETR (1) demonstrated that global attention mechanisms could dramatically improve object detection and segmentation. MaskFormer (3) recast segmentation as a mask classification problem, simplifying the segmentation pipeline, while Mask2Former (4) further refined this approach with masked attention to improve instance distinction in overlapping regions.

Recent trends have emphasized open source code bases and reproducible research. Many of these methods have accompanying code releases that standardize training and inference procedures. Our work follows this tradition by integrating open source libraries (PyTorch, HuggingFace Transformers, OpenCV) into a unified pipeline, focusing on a modular code design that supports various input modalities (image, video, webcam). This modularity aids rapid prototyping and real-world testing, ensuring that segmentation models can transition from academic research to practical applications with minimal modification.

3. Problem Definition

Given an input frame $I \in R^{H \times W \times 3}$, our task is to produce a set of binary masks $\{M_i\}_{i=1}^{N}$, where each mask $M_i \in \{0,1\}^{H \times W}$ corresponds to a detected human instance. The mapping is represented as:

$$f : I \rightarrow \{M_1, M_2, \ldots, M_N\}.$$

This formulation must handle several challenges:

- **Crowded Scenes:** Distinguishing overlapping individuals in dense environments.
- **Scale Variance:** Detecting and segmenting small or distant persons while preserving details for larger instances.
- **Mask Precision:** Ensuring high Intersection-over-Union (IoU) with ground truth, ideally ≈ 0.60.
- **Real-Time Performance:** Achieving an average processing rate of approximately 15 frames per second (FPS) on modern hardware.

4. Methodology

Our system is based on a pre-trained Mask2Former model (*facebook/mask2former-swin-base-coco-panoptic*) and is implemented in Python with several open-source libraries. Below, I detail both the theoretical underpinnings and the specific code implementations that enable real-time segmentation.

4.1. Model Architecture Overview

Mask2Former decouples segmentation from detection by treating mask generation as a classification problem on a set of learned queries. The model architecture includes:

1. **Swin Transformer Backbone:** The backbone splits the image into non-overlapping patches and applies shifted-window self-attention to generate multiscale feature maps, $F = \text{SwinTransformer}(I)$.
2. **Pixel Decoder:** Aggregates and refines these multiscale features to prepare for detailed segmentation.
3. **Transformer Decoder:** Utilizes fixed learned query embeddings to iteratively refine segmentation masks via masked cross-attention.

At the end of the forward pass, each query results in a mask prediction with an associated confidence score. Only masks classified as "person" (class ID 0 in COCO) are retained for further processing.

4.2. Code Implementation Details

Our implementation leverages Python and several widely used libraries. The codebase is modular, with separate modules for processing images, videos, and live webcam streams. The main modules are:

- **Image Segmentation:** Implemented in *image_segmentation_windows.py*, which contains the *HumanSegmenter* class. This class loads the Mask2Former model using HuggingFace's textitAutoImageProcessor and *Mask2FormerForUniversalSegmentation* libraries. The *segment_image* method handles image loading (using OpenCV), conversion from BGR to RGB, and then applies the model to obtain a segmentation map. Post-processing merges the segmentation results into overlays and binary masks.
- **Video Segmentation:** Encapsulated in *video_segmentation_windows.py*, the *VideoHumanSegmenter* class extends the image segmentation pipeline to process each frame of a video. It uses OpenCV to read frames, applies the segmentation model, and writes side-by-side outputs (original and segmented) to disk. The code employs a progress bar (via *tqdm*) to monitor processing progress.
- **Webcam Segmentation:** In *webcam_segmentation_windows.py*, the *WebcamHumanSegmenter* class processes live video streams. It supports interactive mode: users can switch display modes (original, mask, overlay, etc.) and quit the application via keyboard inputs. Key optimization steps include frame downsampling and adaptive FPS measurement.

A common pattern in our code is the use of pre-processing routines that convert raw frames from OpenCV's BGR format to the RGB format expected by the model. For example, the image segmentation code executes:

```
cv_image = cv2.imread(image_path)
rgb_image = cv2.cvtColor(cv_image,
cv2.COLOR_BGR2RGB)
image = Image.fromarray(rgb_image)
```

After preprocessing, the image is passed to the Mask2Former model. The postprocessing step leverages the model's built-in functions (e.g., *post_process_panoptic_segmentation*) to obtain a segmentation map, and then a loop iterates over each segment to assign colors and labels.

For video segmentation, the code structure is similar but placed inside a loop that iterates over each frame. The

Algorithm 1 Real-Time Segmentation Inference (Common Pipeline)

Require: Input frame I
Convert I from BGR to RGB.
Downsample I if high resolution.
$F \leftarrow \text{AutoImageProcessor}(I)$
$\hat{Y} \leftarrow \text{Mask2Former}(F)$
$S \leftarrow \text{PostProcess}(\hat{Y})$
$S_{person} \leftarrow \{s \in S \mid \text{class}(s) = \text{person}\}$
Generate visualization (overlay, binary mask, etc.).
return S_{person}

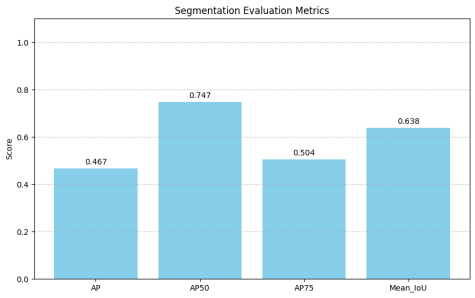

Figure 1. Evaluation Result based on 3721 COCO dataset

pseudocode in Algorithm 1 summarizes the inference pipeline common to all modules.

In addition, our code uses *argparse* for command-line options, allowing the user to specify input file paths, output locations, and display preferences. This flexibility is critical for iterative development and testing in real-time applications.

4.3. Integration of Open-Source Libraries

The implementation relies on:

- **PyTorch** for GPU-accelerated model inference.
- **HuggingFace Transformers** for loading and utilizing the pre-trained Mask2Former model.
- **OpenCV** for video capture and image processing.
- **Matplotlib** for generating and saving visualizations.
- **Tqdm** for monitoring progress during video processing.

These libraries allow the code to remain concise yet powerful, facilitating both rapid experimentation and production-level deployment.

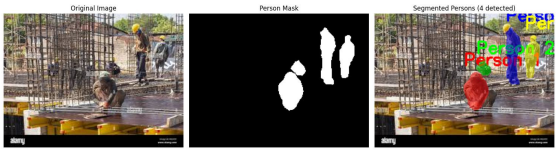

Figure 2. Segmentation overlay in a construction environment. Individuals are accurately segmented with distinct colored masks.

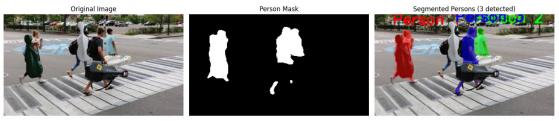

Figure 3. Segmentation results in a surveillance scenario. Foreground persons are well-delineated, though extremely small figures may be less visible.

5. Experimental Results

I evaluate our system on both quantitative and qualitative metrics.

5.1. Segmentation Quality

On static images from the COCO dataset, our model obtains:1

- **AP for Person:** Approximately 0.467.
- **Average Mask IoU:** Approximately 0.63.

These values show that the segmentation quality remains competitive with state-of-the-art systems. Figures 2 and 3 illustrate segmentation overlays in construction and surveillance scenarios, respectively. The results demonstrate clear separation of individual persons even in cluttered environments.

5.2. Real-Time Performance

Benchmarks were carried out on a system with an NVIDIA RTX 2070Ti GPU:

- **Video Processing:** 720p video streams process at original FPS.
- **Webcam Streams:** With appropriate downsampling (e.g., to 720p), live feeds also average around 15 FPS.

The synergy of careful code optimizations, GPU acceleration, and efficient pre/post-processing allows the system to maintain real-time responsiveness.

5.3. Discussion and Limitations

Despite robust performance, challenges remain:

- In extremely dense crowds, minor mask merging can occur.
- Very small or distant people may occasionally be missed.
- The system requires GPU acceleration for maintaining 15 FPS; CPU-only implementations would incur significant slowdowns.
- People that concealed by other object or people are hard to detect

Future enhancements may include multi-scale analysis and temporal smoothing across video frames to further mitigate these issues.

6. Contribution and Future Work

Our primary contribution is the adaptation of a state-of-the-art segmentation model into a real-time human segmentation system. In doing so, we:

- Integrated a sophisticated model architecture (Mask2Former with Swin Transformer) with custom code to process images, videos, and live webcam feeds.
- Demonstrated that with careful engineering and code optimizations, it is possible to achieve an average throughput of approximately 15 FPS.
- Released a modular and extensible codebase (see repository[1]) to enable reproducibility and further research in real-time segmentation.

Future work will focus on enhancing small object detection, employing model compression/quantization to further boost speed on edge devices, and integrating temporal smoothing to improve consistency across video frames.

7. Conclusion

I have presented a real-time human segmentation system that combines Mask2Former with a Swin Transformer backbone. The system achieves competitive segmentation performance (AP ≈ 0.467, IoU ≈ 0.638) while maintaining an average of 15 FPS on modern GPUs. By detailing both the theoretical foundations and the practical code implementation, I hope to bridge the gap between advanced academic research and practical real-time applications. Our work demonstrates a robust pipeline capable of processing diverse input types ranging from static images to live video thereby offering a viable solution for applications such as construction safety, surveillance, and security monitoring.

[1] https://anonymous.4open.science/r/RealTimeHumanDetection-8000/README.md

LLM Use Acknowledgment

Portions of this manuscript were generated using an LLM (OpenAI GPT-4) to assist in drafting and structuring the content, while all technical details were verified and refined by the authors.

References

[1] Carion, N., et al. (2020). End-to-End Object Detection with Transformers. ECCV.

[2] Chen, L.-C., et al. (2018). Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. ECCV.

[3] Cheng, B., et al. (2021). MaskFormer: Per-Pixel Classification is Not All You Need for Semantic Segmentation. arXiv.

[4] Cheng, B., et al. (2022). Masked-Attention Mask Transformer for Universal Image Segmentation. CVPR.

[5] Dosovitskiy, A., et al. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR.

[6] He, K., et al. (2017). Mask R-CNN. ICCV.

[7] Lin, T.-Y., et al. (2014). Microsoft COCO: Common Objects in Context. ECCV.

[8] Liu, Z., et al. (2021). Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. ICCV.

[9] Redmon, J., et al. (2016). You Only Look Once: Unified, Real-Time Object Detection. CVPR.

[10] Ren, S., et al. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NeurIPS.

[11] Long, J., et al. (2015). Fully Convolutional Networks for Semantic Segmentation. CVPR.