NeuralSegmentello: U-Net-Based Architectures for Mask Refinement

Andrea Gentilini 2043590 | Michele Magrini 2066963 | Leo Petrarca 2087113 | Iacopo Scandale 2085989

June 2025

1 Introduction

Interactive object segmentation is a critical task in computer vision, where the objective is to precisely segment an object of interest within an image based on minimal human input. Traditional fully automated segmentation algorithms often struggle to generalize across diverse contexts, especially when the object of interest is ambiguous or context-dependent. To mitigate this, *user-guided segmentation methods*, where the user provides rough hints such as scribbles, clicks, or brush strokes, have gained increasing attention for their ability to incorporate semantic priors from human intuition into the learning process.
This project, **NeuralSegmentello**, addresses the problem of *refining coarse user-provided masks into high-precision object segmentations*. Formally, given an input image $I \in R^{H \times W \times 3}$ and a coarse binary mask $M_{\text{rough}} \in \{0,1\}^{H \times W}$ representing a user-drawn brush stroke over an object of interest, the goal is to produce a refined mask M_{refined} such that the object boundary is accurately delineated.

1.1 Motivation

In many real-world applications, such as digital content creation, biomedical imaging, and human-in-the-loop labeling systems, a complete manual annotation is time-consuming and requires domain expertise. However, coarse annotations, such as a single brush stroke over the target object, are quick and intuitive to produce. Leveraging these weak supervisory signals with deep neural networks can dramatically reduce the annotation burden while still achieving high-quality segmentation. This motivates the development of a system capable of *learning to segment from user-imprecise cues*.

1.2 Objectives

The objectives of this project are as follows:

- Design and train lightweight deep neural architectures (based on U-Net) capable of learning a mapping from coarse binary masks and input images to precise segmentation outputs.
- Explore and compare different architectural variants, including residual connections and attention-based decoding modules.
- Evaluate performance using loss functions tailored to segmentation tasks, including *Binary Cross-Entropy (BCE)*, *Dice Loss*, and *Boundary Loss*.
- Develop a minimal yet reproducible experimental framework, providing reduced datasets and full training pipelines to ensure portability and result replication.

1.3 Role of User Input

In our system, the *user input is represented as a coarse mask* simulating a brush stroke over the object of interest. This input acts as a localization prior for the segmentation model, enabling the network

to attend only to relevant regions. The model thus functions as a *mask refinement engine* conditioned both on the image content and the structural information encoded in the user's rough input. Unlike conventional interactive segmentation relying on iterative feedback (e.g., extreme points or bounding boxes), we focus on *single-shot inference* from a one-time input, which reduces interaction latency.

Interactive segmentation has been the subject of extensive research. A detailed overview of related work is provided in Section 3.6.

1.4 Reproducibility

This project ensures reproducibility through:

- Fixed random seed and deterministic data loading
- Explicit configuration files for architecture and training parameters
- A reduced dataset for demonstration and debugging purposes (`reduce_dataset.ipynb`)
- Open-source publication under the MIT license

The full codebase and instructions are available at: github.com/andrea-gentilini/NeuralSegmentello

2 Methodology

2.1 Dataset

The dataset utilized in this project is the Microsoft Common Objects in Context (MS COCO) dataset [1]. MS COCO is a large-scale object detection, segmentation, and captioning dataset comprising over 330k images, with more than 200k images annotated for object detection and segmentation tasks. The dataset includes 80 object categories and provides per-instance segmentation masks, making it suitable for training and evaluating segmentation models.

The annotations in the MS COCO dataset are provided in JSON format and include segmentation masks represented as polygons, not directly as binary masks. Each annotation includes a polygon outlining the object, from which a binary mask can be generated.

For this project, a subset of the MS COCO dataset was used for computational complexity purpose. The images have a median resolution of 640×480 pixels, providing sufficient detail for segmentation while maintaining computational efficiency.

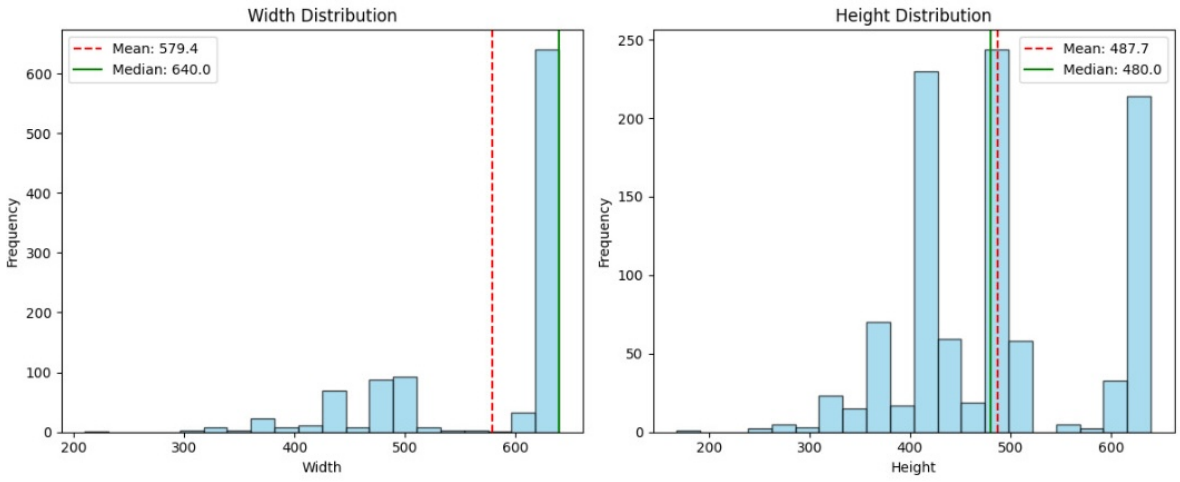

Figure 1: Distributions of image dimensions in the dataset. The left histogram shows the distribution of image widths, while the right histogram shows the distribution of heights. In both plots, the red dashed line represents the mean value, and the solid green line denotes the median. Widths are more right-skewed with a median of 640 pixels and a mean of approximately 579, while the height distribution is more symmetric, with a median of 480 and a mean around 488.

Prior to training, all images were normalized to the [0, 1] range to facilitate model convergence. We chose not to apply data augmentation, as the dataset was already large and highly diverse. Due to computational constraints, we trained the models using only a subset of 1,000 images from the dataset (900 for training and 100 for validation).

2.2 Coarse Mask Generation

To simulate user-provided coarse annotations, we generated synthetic brush stroke masks by perturbing the ground truth segmentation masks through a combination of morphological operations and randomized noise. Specifically, we applied erosion to extract object boundaries and then randomly masked parts of these boundaries using circular regions, mimicking the irregularity of hand-drawn strokes. Additional morphological opening and Gaussian blurring were employed to smooth the resulting contours. The final binary masks approximate the appearance of user-drawn strokes and are used as coarse localization cues for the segmentation model.

2.3 Model Architecture

The architectures employed in this project are variations of the U-Net model. The models follow an encoder-decoder structure with skip connections that facilitate the flow of spatial information.

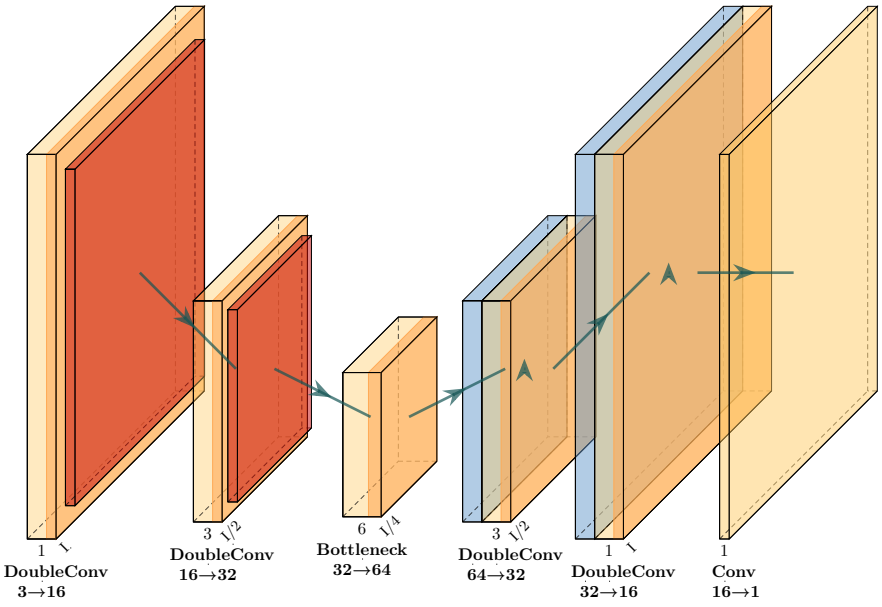

Figure 2: Overview of the U-Net base architecture.

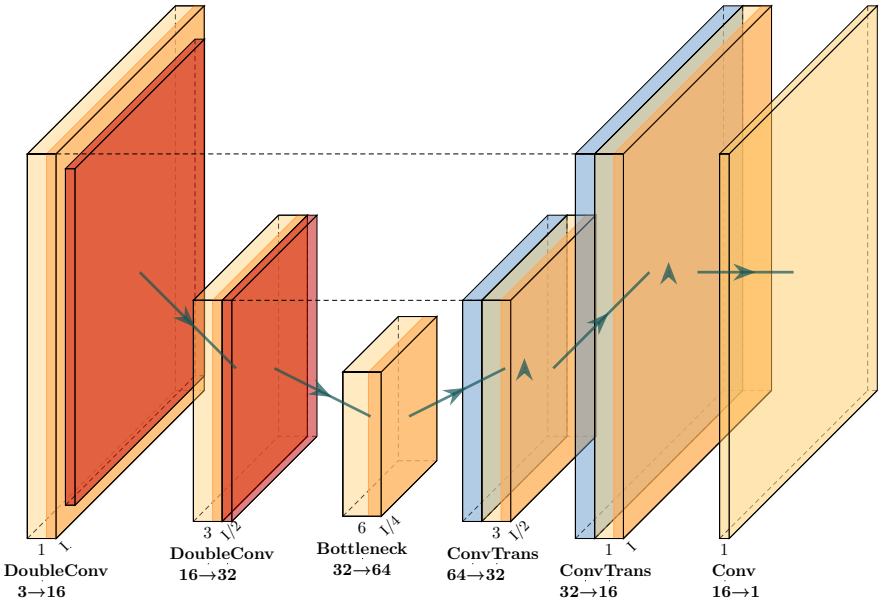

Figure 3: Overview of the U-Net architecture with residual connections.

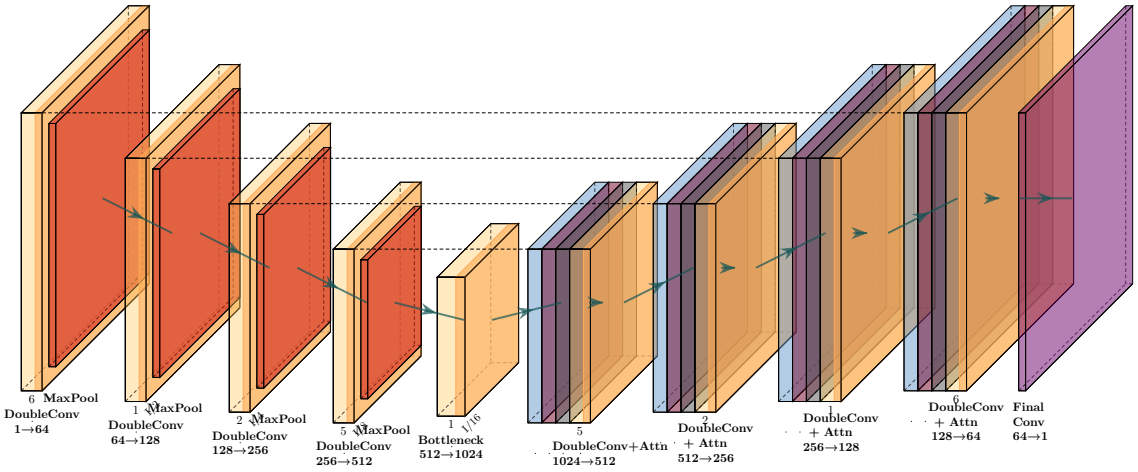

Figure 4: Overview of the U-Net architecture with attention mechanism.

In each model, the encoder consists of a series of convolutional layers with increasing feature dimensions, capturing hierarchical representations of the input image. The decoder mirrors the encoder structure, employing transposed convolutions to upsample the feature maps and reconstruct the segmentation mask.

To incorporate the coarse mask into the architecture, the binary mask is concatenated with the input image (normalized) along the channel dimension. In addition, the gradient is added, resulting in a 3-channel input. This enables the model to leverage localization cues provided by the coarse mask during the segmentation process.

In total, eight different models were trained, all representing slight variants of the following three architectural families:

- **U-Net Tiny** [2]: A compact version of the U-Net model with reduced feature dimensions, designed to balance performance and computational efficiency.

- **U-Net Tiny with Residual Connections** [3]: Incorporates residual connections within both encoder and decoder blocks to improve gradient flow and facilitate faster convergence.
- **U-Net with Attention Mechanisms** [4] [5]: Integrates attention modules within the decoder to enhance focus on semantically relevant regions guided by the coarse input.

Specifically, the following configurations were explored:

- U-Net Tiny: `tiny16-32`, `tiny16-128`
- U-Net Tiny with Residual Connections: `res16-32`, `res16-128`, `res32-256`
- U-Net Tiny / Residual with Attention: `attn32-256`

These configurations differ in their depth and feature map widths, but all follow one of the three core architectural designs described above.

2.4 Loss Functions

The training objective combines multiple loss functions to address different aspects of the segmentation task:

- **Binary Cross-Entropy (BCE) Loss**: Measures the pixel-wise classification error between the predicted and ground truth masks.
- **Dice Loss**: Focuses on the overlap between the predicted and ground truth masks, promoting accurate segmentation of object regions.
- **Boundary Loss**: Emphasizes the alignment of object boundaries, encouraging precise delineation of object contours.

Not all models use the same combination of loss terms. Instead, each model variant incorporates one or more of these losses depending on its configuration. Specifically, the loss components used in training are indicated in the model name (e.g., `bce-dice`, `bce-dice-bound`).
When multiple loss functions are used, the total loss is computed as a weighted combination:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{BCE}} \cdot \mathcal{L}_{\text{BCE}} + \lambda_{\text{Dice}} \cdot \mathcal{L}_{\text{Dice}} + \lambda_{\text{Boundary}} \cdot \mathcal{L}_{\text{Boundary}}$$

where λ_{BCE}, λ_{Dice}, and $\lambda_{\text{Boundary}}$ are weighting coefficients that balance the contribution of each component. When $\mathcal{L}_{\text{Boundary}}$ is not included, all λ values are equal. In contrast, when $\mathcal{L}_{\text{Boundary}}$ is part of the final loss, the weights are set as follows: $\lambda_{\text{BCE}} = 0.4$, $\lambda_{\text{Dice}} = 0.4$, and $\lambda_{\text{Boundary}} = 0.2$.

2.5 Training

The models were implemented using the PyTorch framework and trained on a workstation equipped with an NVIDIA GeForce GTX 1650 GPU (4GB VRAM). Each training session was conducted for 50 epochs with a batch size of 1, due to varying input image dimensions. Optimization was performed using the Adam optimizer with a learning rate of 1×10^{-3}. To mitigate overfitting, early stopping was applied based on the validation loss, with a patience threshold of 5 epochs.

The training losses of the best and worst models, as determined by pixel accuracy (see Section 3.2), are shown in Figure 5.

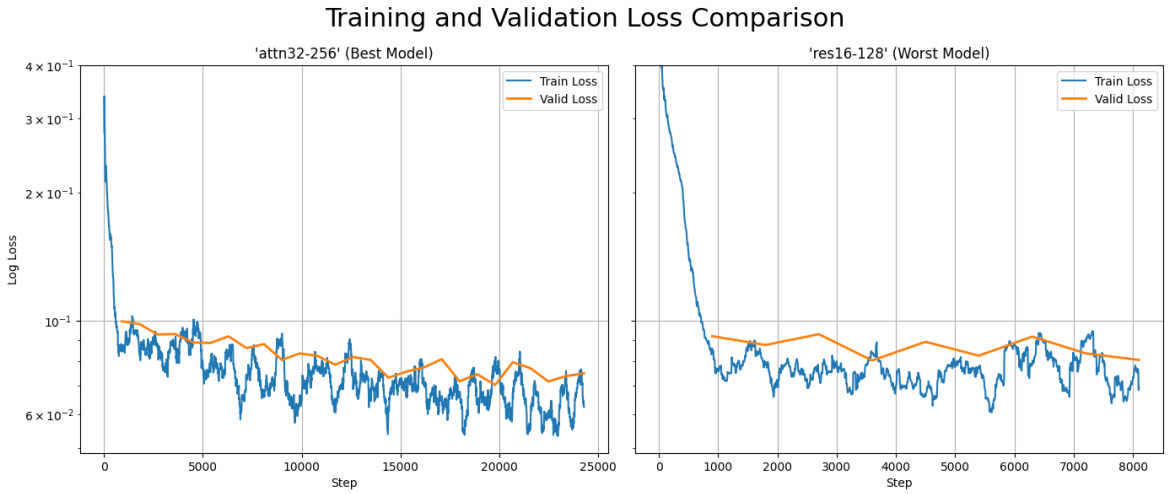

Figure 5: Training loss curves for the best and worst performing models according to pixel accuracy (respectively *attn32-256 bce-dice-bound* and *res16-128 bce-dice*). The figure highlights the difference in convergence speed and final loss value between the two models.

3 Results and Discussion

3.1 Evaluation Metrics

To quantitatively assess the performance of our segmentation models, we employed the following metrics, designed to evaluate different aspects of segmentation quality:

- **Intersection over Union (IoU)**: Computes the ratio between the intersection and the union of the predicted and ground truth masks. This is a standard metric that quantifies the overall agreement between the predicted segmentation and the ground truth.
- **Pixel Accuracy**: Calculates the proportion of correctly classified pixels (both foreground and background) over the total number of pixels. While easy to interpret, it may be misleading in cases of strong class imbalance.
- **Boundary IoU**: Focuses on the overlap between the boundaries of the predicted and ground truth masks. This metric is more sensitive to errors along the object contours and is particularly useful for evaluating segmentation refinement.
- **Hausdorff Distance**: Measures the greatest of all the distances from a point in one mask boundary to the closest point in the other. It captures the worst-case deviation between the predicted and ground truth contours and is useful for assessing boundary accuracy in a geometric sense.

These metrics collectively provide a comprehensive evaluation of segmentation performance, accounting for global overlap, per-pixel correctness, contour alignment, and worst-case boundary deviation.

3.2 Quantitative Results

We evaluated our models on the test set derived from the MS COCO dataset [1]. The Table 1 summarizes the performance of the various model configurations according to the evaluation metrics defined in Section 3.1. Additionally, the size of each trained model is also reported in Table 2.

Baseline Comparison. The smallest architectures (`tiny16-32_bce-dice` and `res16-32_bce-dice`, both ∼1.40 MB; see Table 2) already yield strong performance, with IoU scores above 0.86 and Hausdorff distances around 43 pixels. The residual variant offers marginal improvements across all metrics, confirming the benefit of skip connections even in shallow models.

Model Name	Pixel Accuracy	IoU	Boundary IoU	Hausdorff Distance
tiny16-32_bce-dice	0.9839	0.8609	0.1566	43.10
res16-32_bce-dice	0.9840	0.8611	0.1581	42.73
tiny16-128_bce-dice	0.9833	0.8528	0.1485	46.29
res16-128_bce-dice	0.9830	0.8534	0.1416	43.64
tiny16-128_bce-dice-bound	0.9864	0.8796	0.2207	40.66
res16-128_bce-dice-bound	0.9857	0.8749	0.2082	42.17
res32-256_bce-dice-bound	0.9859	0.8654	0.2040	43.03
attn32-256_bce-dice-bound	0.9866	0.8792	0.2210	40.77

Table 1: Performance metrics on the validation set for the different U-Net-based model configurations (excluding model size).

Model Name	Size (MB)
tiny16-32_bce-dice	1.40
res16-32_bce-dice	1.40
tiny16-128_bce-dice	22.34
res16-128_bce-dice	22.34
tiny16-128_bce-dice-bound	22.34
res16-128_bce-dice-bound	22.34
res32-256_bce-dice-bound	88.97
attn32-256_bce-dice-bound	90.04

Table 2: Model sizes (in megabytes) for each configuration. Models' order matches the Table 1

Effect of Depth. Increasing the model depth and feature width to 128 channels (`tiny16-128` and `res16-128`) results in a minor drop in performance for the baseline BCE+Dice loss. Despite the larger capacity (22.34 MB), these models show slightly lower IoU and boundary alignment than their shallower counterparts, suggesting potential overfitting or insufficient training data to exploit the additional parameters.

Impact of Boundary-Aware Loss. Models trained with an additional boundary-aware loss (`*-bce-dice-bound`) consistently outperform their counterparts in both IoU and especially Boundary IoU. For instance, `tiny16-128_bce-dice-bound` reaches an IoU of 0.8796 and a Boundary IoU of 0.2207, significantly higher than its non-boundary-aware counterpart (0.8528 and 0.1485, respectively). The improvement in Hausdorff Distance corroborates the enhanced precision along object borders.

Large-Scale Architectures. The models `res32-256_bce-dice-bound` and `attn32-256_bce-dice-bound` (with sizes of 88.97 MB and 90.04 MB, respectively) further validate the benefit of deeper and wider architectures. Notably, the attention-based model achieves the highest Pixel Accuracy (0.9866) and Boundary IoU (0.2210), while maintaining a competitive Hausdorff Distance (40.77). This suggests that attention mechanisms help focus on relevant structures and improve boundary delineation.

Trade-off Analysis. As shown in Table 2, model performance generally improves with size, but not linearly. The best-performing models balance architectural complexity and boundary-aware losses. In deployment scenarios where model size is a constraint, lightweight configurations such as `tiny16-32` or `res16-32` remain viable options with strong baseline performance.

3.3 Qualitative Results

To visually assess the segmentation quality across different model configurations, we present comparative examples of input images, user-provided coarse masks, ground truth masks, and model predictions. Each row corresponds to a specific U-Net variant, characterized by architectural differences and loss functions.

As illustrated in Figure 6, all model variants significantly refine the coarse user inputs into accurate segmentation masks. Architectures using deeper features and additional mechanisms such as residual

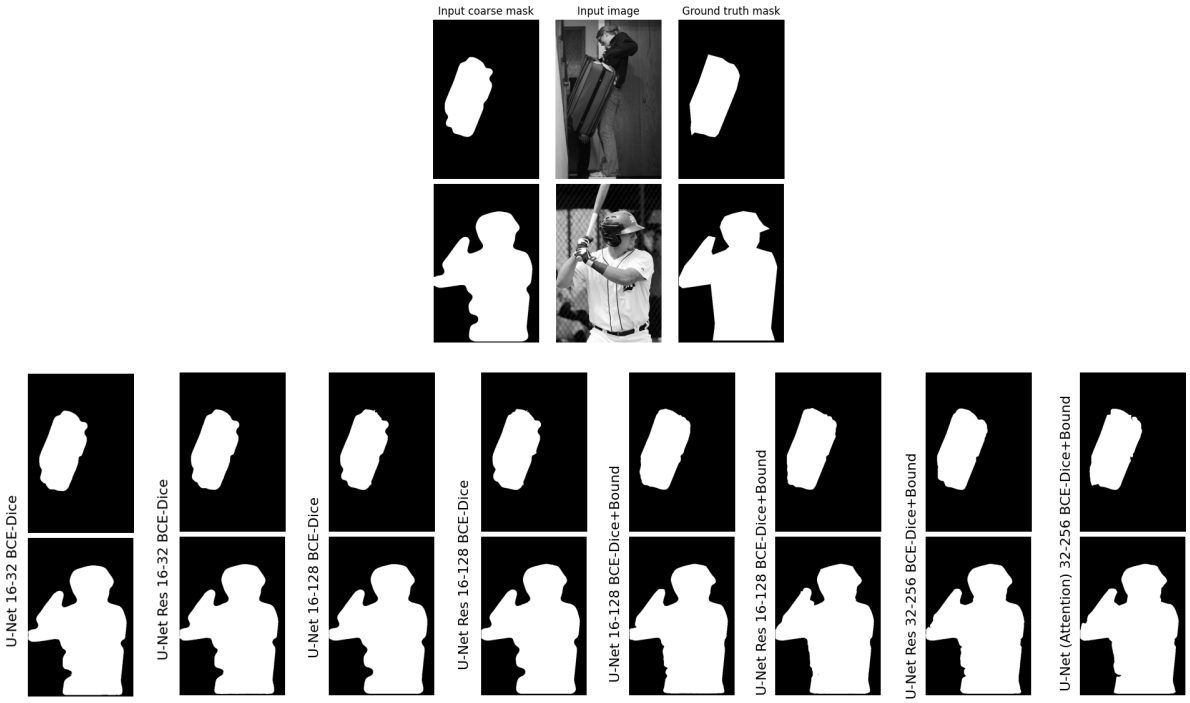

Figure 6: Qualitative comparison of segmentation results for various U-Net configurations. The first row shows the inputs to the models (image + coarse mask). The second row shows various models' predictions and ground truth (left); input and output masks for different models (right).

connections or attention modules tend to yield more precise boundaries and better alignment with the ground truth, as per quantitative results analysis.

3.4 Error Analysis and Strengths

Despite the overall strong performance, certain failure cases were observed:

- **Ambiguous Coarse Masks**: In instances where the user-provided coarse mask poorly represents the object of interest, the model struggles to accurately refine the segmentation, leading to incomplete or incorrect masks.
- **Complex Backgrounds**: Images with cluttered or complex backgrounds sometimes result in the model misclassifying background regions as part of the object, reducing precision.
- **Small or Occluded Objects**: The model occasionally fails to accurately segment small or partially occluded objects, likely due to limited contextual information.

These observations suggest that the model's performance is influenced by the quality of the user input and the complexity of the scene. The model's design prioritizes ease of use, requiring only a simple brush stroke to initiate segmenta- tion. This approach reduces the annotation burden and accelerates the segmentation process, making it practical for applications such as image editing, medical imaging, and interactive labeling tools. However, ensuring the quality of the initial user input remains crucial for optimal performance.

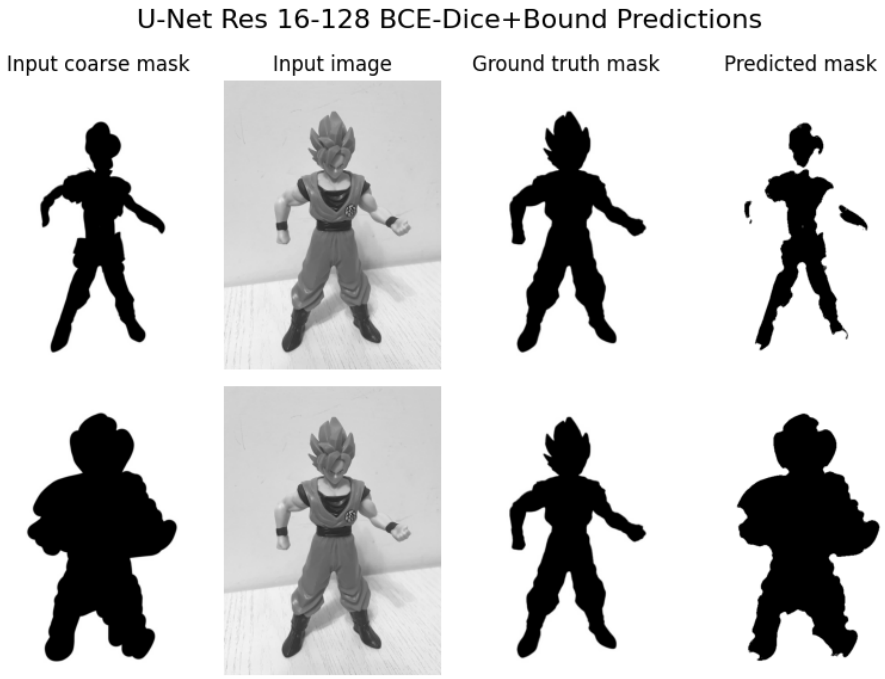

Figure 7: Example of predictions when ambiguos coarse masks is given as input (res16-128_bce-dice-bound model).

Instead, models strengths are:

- **Efficiency**: The lightweight architecture ensures fast inference times, making it suitable for real-time applications.
- **User-Friendly**: The model requires minimal user input, simplifying the segmentation process for end-users.
- **Generalization**: Demonstrates strong generalization across diverse object categories and scenes.

3.5 Future Work

To address the identified limitations and enhance the model's capabilities, the following avenues are proposed:

- **Scaling Up Training**: Train the models on significantly larger datasets using more powerful hardware. All current experiments were conducted on a single NVIDIA GeForce GTX 1650 with 4GB of memory, which limited both the dataset size and model complexity.
- **Real User Input Integration**: Develop a user interface to collect actual human inputs (e.g., brush strokes) and validate the model's performance in real-world interactive scenarios, rather than relying solely on synthetic annotations.
- **Interactive Refinement**: Incorporate iterative user feedback loops to allow dynamic refinement of segmentation outputs. This would be particularly beneficial in ambiguous or fine-grained regions.

3.6 Related Work

Interactive segmentation has been extensively studied, with notable contributions including:

- **Deep Extreme Cut (DEXTR)** [6]: Utilizes extreme points as user input to guide object segmentation.

- **F-BRS** [7]: Introduces a feature backpropagating refinement scheme for interactive segmentation.
- **RITM** [8]: Proposes a lightweight architecture for real-time interactive segmentation.
- **Segment Anything Model (SAM)** [9]: A foundation model trained on a large-scale dataset for general-purpose segmentation tasks.
- **Class-Based Styling (CBS)** [10]: Combines semantic segmentation with real-time style transfer to apply artistic styles selectively to specific object classes.

Our approach differs by focusing on refining coarse brush stroke mask inputs into precise segmentation masks, emphasizing model efficiency (low-latency deployment) and minimal user interaction.

References

[1] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[2] 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 (MICCAI)*, pages 234–241. Springer, 2015.

[3] Md Zahangir Alom, Mahmudul Hasan, Chris Yakopcic, Tarek M Taha, and Vijayan K Asari. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. *arXiv preprint arXiv:1802.06955*, 2018.

[4] truthisneverlinear. Attention u-net — pytorch. https://www.kaggle.com/code/truthisneverlinear/attention-u-net-pytorch/notebook, 2021. Accessed: 2025-05-26.

[5] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas, 2018.

[6] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. Deep extreme cut: From extreme points to object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 616–625, 2018.

[7] Konstantin Sofiiuk, Ilia Petrov, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8623–8632, 2020.

[8] Konstantin Sofiiuk, Ilia A. Petrov, and Anton Konushin. Reviving iterative training with mask guidance for interactive segmentation. *arXiv preprint arXiv:2102.06583*, 2021.

[9] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[10] Lironne Kurzman, David Vazquez, and Issam Laradji. Class-based styling: Real-time localized style transfer with semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2019.