Proposed Model for Camouflage Detection

1 Proposed Model

Camouflaged object detection (COD) is a challenging problem in computer vision due to the difficulty of distinguishing objects that blend seamlessly into their surroundings. Conventional segmentation models struggle to capture the subtle visual cues necessary for detecting such objects. To address this, we propose a novel segmentation model that integrates a SegFormer backbone with a custom segmentation head featuring an attention-based enhancement mechanism. The goal of our approach is to balance computational efficiency with accurate segmentation, making it suitable for real-time and resource-constrained applications.

2 Model Architecture

The proposed model for camouflage detection integrates the lightweight SegFormer-B0 architecture along with its bigger variants as its backbone with a custom segmentation head enhanced by an attention mechanism. This design aims to efficiently detect camouflaged objects, which are characterized by subtle boundaries and complex background blending. The model balances computational efficiency with high performance, making it suitable for resource-constrained applications. The proposed architecture consists of two primary components:

- **SegFormer Backbone:** A transformer-based encoder that efficiently extracts multi-scale features.
- **Custom Segmentation Head:** A decoder utilizing a novel SplitConvAttention (STCA) module to refine and upsample feature maps for accurate camouflage segmentation.

2.1 SegFormer Backbone

The backbone of our model is based on the SegFormer architecture. We introduce three model variants:

- **Small Model:** SegFormer-B0 (lightweight, optimized for efficiency)
- **Medium Model:** SegFormer-B2 (balances performance and computational cost)
- **Large Model:** SegFormer-B5 (deeper architecture, optimized for accuracy)

These backbones are pre-trained on datasets such as Cityscapes, ADE20K, and COCO-Stuff, allowing the model to leverage rich semantic features.

Given an input image of shape $(3, 512, 512)$, the encoder extracts multi-scale features and outputs a spatially downsampled feature map:

$$F = f_{SegFormer}(I), \quad F \in \mathbb{R}^{16\times16\times C} \tag{1}$$

where:

- I represents the input image.
- $f_{SegFormer}$ is the SegFormer encoder function.
- F is the extracted feature map with spatial dimensions 16×16 and C channels.

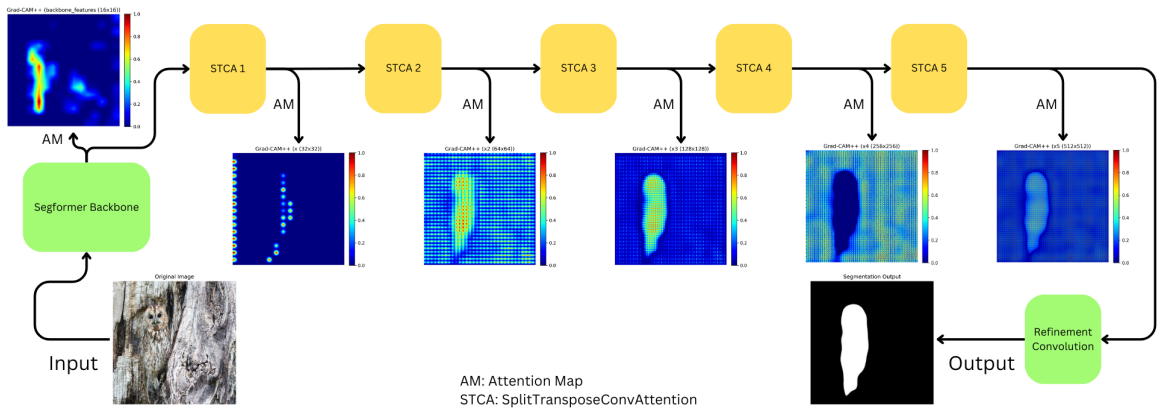

Figure 1: Proposed Model Architecture

2.2 STCA Block: Attention-Based Feature Refinement

To reconstruct the segmentation mask at the original resolution (512, 512) and enhance feature refinement for camouflaged regions, we introduce a custom segmentation head. This head employs a series of SplitConvAttention (STCA) block, which combine transposed convolutions with an attention mechanism inspired by squeeze-and-excitation (SE) to focus on salient features. The STCA block operates as follows:

1. **Input Splitting**: The input feature map (e.g., (16, 16, 256) initially) is divided along the channel dimension into four equal splits (e.g., 64 channels each), allowing parallel processing to diversify feature representations.

2. **Convolutional Processing**: Each split is processed by a 2D convolution with reduced filters (e.g., 32 per split for 256 input channels), using a 3x3 kernel, ReLU activation, and "same" padding. The outputs are concatenated to form a merged feature map (e.g., (16, 16, 128)).

3. **Channel-wise Attention**: A global average pooling operation compresses the merged features into a channel-wise descriptor. Two dense layers follow: the first reduces the channel count by a factor of 4 (reduction ratio) with ReLU activation, and the second restores half the channels with sigmoid activation. These attention weights are multiplied with the merged features to emphasize relevant channels.

4. **Upsampling**: A 2D transposed convolution with a 3x3 kernel and stride of 2 upsamples the enhanced features (e.g., from (16, 16, 128) to (32, 32, 128)), followed by batch normalization to stabilize training.

The channel-wise attention mechanism is defined as:

$$A_c = \sigma(W_2(\text{ReLU}(W_1(\text{GAP}(F))))) \tag{2}$$

where:

- $\text{GAP}(F)$ is the global average pooling of feature map F.
- W_1 and W_2 are dense layers.
- σ is the sigmoid activation function.
- A_c represents the channel-wise attention weight vector.

The attended feature map is then computed as:

$$F' = A_c \cdot F \tag{3}$$

This layer is applied five times in sequence, upsampling the feature maps from (16, 16, 256) to (512, 512, 8) through intermediate resolutions of (32, 32, 128), (64, 64, 64), (128, 128, 32), and (256, 256, 16). The channel dimensions are progressively halved, maintaining efficiency while preserving critical information.

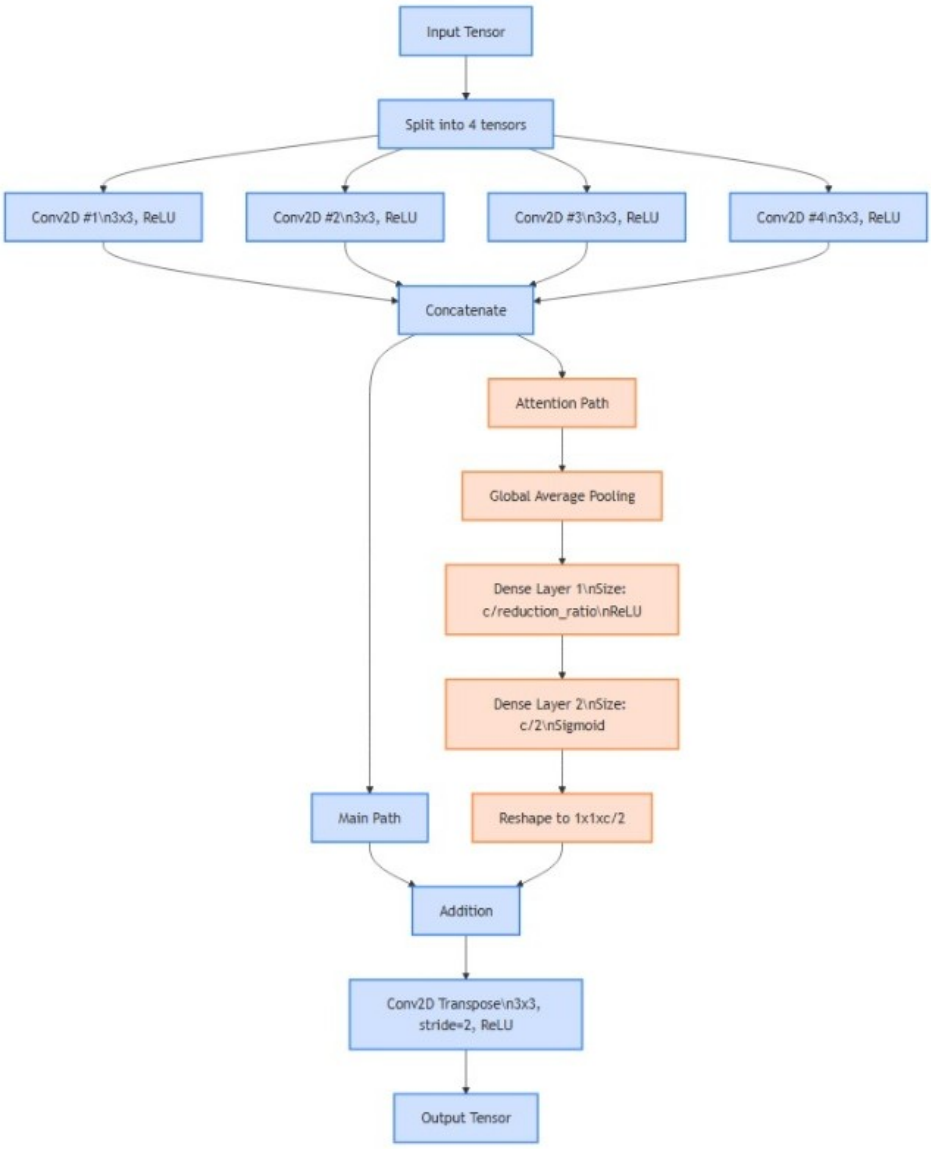

Figure 2: STCA Block Architecture

2.3 Output Layer

The final feature map of shape (512, 512, 8) is processed by a 1x1 convolution with a single output channel and sigmoid activation, yielding a probability map of shape (512, 512, 1). This map indicates the likelihood of each pixel belonging to a camouflaged object. The resulting model takes an input tensor of shape (3, 512, 512) and outputs a segmentation mask of shape (512, 512, 1). It can be represented mathematically as,

$$P = \sigma(W_o F') \tag{4}$$

where W_o is a 1x1 convolution kernel.

3 Loss Function

In this study, we introduce a custom loss function, termed IDB, specifically designed to optimize the performance of our camouflage detection model. The IDB loss function integrates three complementary components to ensure both accuracy and robustness in detecting camouflaged objects:

- **Binary Cross-Entropy (BCE)**: This component evaluates the pixel-wise discrepancy between the predicted segmentation mask and the ground truth, ensuring accurate classification at the pixel

level. It is weighted by a factor of 0.5 to balance its contribution relative to the other terms.

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}\left[G_i \log P_i + (1 - G_i)\log(1 - P_i)\right] \tag{5}$$

- **Dice Loss**: Derived from the Dice Coefficient, which quantifies the overlap between the predicted and actual regions, the Dice Loss is calculated as 1 minus the Dice Coefficient. The Dice Coefficient is given by

$$\text{Dice Loss} = 1 - \frac{2\sum(P \cdot G) + \epsilon}{\sum P + \sum G + \epsilon} \tag{6}$$

  where $\epsilon = 1 \times 10^{-6}$ is a smoothing factor to prevent division by zero and P and G represent the predicted and ground truth segmentation masks, while N is the total number of pixels.. With a weight of 2.0, this term emphasizes the importance of region overlap, which is crucial for capturing small or irregularly shaped camouflaged objects.

- **Intersection over Union (IoU) Loss**: Computed as 1 minus the IoU score, where IoU is defined as

$$\text{IoU Loss} = 1 - \frac{\sum(P \cdot G) + \epsilon}{\sum P + \sum G - \sum(P \cdot G) + \epsilon} \tag{7}$$

  where P and G represent the predicted and ground truth segmentation masks, and N is the total number of pixels. This loss focuses on the alignment of the predicted boundaries with the ground truth. It is weighted by 1.0, providing an additional penalty for misalignment.

The total loss function is a weighted sum of these components, formulated as:

$$\mathcal{L}_{\text{total}} = 0.5 \times \mathcal{L}_{\text{BCE}} + 2.0 \times \mathcal{L}_{\text{Dice}} + 1.0 \times \mathcal{L}_{\text{IoU}}.$$

The selection of weights reflects a deliberate emphasis on different aspects of segmentation quality. Specifically, the higher weight assigned to the Dice Loss underscores the priority of achieving strong region overlap, which is essential for effective camouflage detection. Meanwhile, the BCE and IoU losses contribute to maintaining pixel-wise accuracy and boundary precision, respectively. This composite loss function enables the model to address the multifaceted challenges inherent in camouflage detection, leading to improved performance in identifying objects that blend seamlessly with their surroundings.

4 Results

4.1 Datasets

To assess the capabilities of our proposed camouflage segmentation model, we conducted experiments using three well-established benchmark datasets: CHAMELEON, CAMO, NC4K, and COD10K. These datasets collectively offer a broad spectrum of camouflage challenges, featuring diverse object types such as animals, plants, and artificial items. They also present varying complexities, including background blending, color disruption, and indistinct edges, making them ideal for testing the robustness and adaptability of our approach. Consistent with the methodology outlined in [40], we utilized the training portions of CAMO and COD10K and 3000 samples from NC4K to train our model, reserving the remaining images for testing purposes.

4.2 Evaluation Metrics

The performance of our model was evaluated using four widely adopted metrics designed to capture different aspects of segmentation quality. These include:

- **Structural Similarity Measure** S_α, which quantifies the structural consistency between predicted masks and ground truth;
- **Adaptive E-measure** E_ϕ^{ad}, which evaluates edge alignment accuracy;
- **Weighted F-measure** F_β^w, providing a balanced assessment of precision and recall; and
- **Mean Absolute Error** M, which measures pixel-wise differences between predictions and ground truth.

Together, these metrics offer a comprehensive evaluation of our model's effectiveness.

4.3 Experimental Process

Our model was developed and implemented using the TensorFlow framework and executed on Google Colab system powered by NVIDIA A100 GPU. For training, input images were standardized to a resolution of 512×512 pixels. The Segformer backbone encoder was initialized with weights from a pre-trained Segformer model, and a learning rate of 0.00006 was set. During testing, inference was performed on the input images with no additional post-processing applied to refine the outputs.

4.4 Comparison to the State-of-the-Art

To demonstrate the superiority of our approach, we benchmarked it against 27 contemporary state-of-the-art methods spanning multiple domains, including object detection (e.g., FPN), semantic segmentation (e.g., PSPNet), instance segmentation (e.g., Mask RCNN), shadow detection (e.g., DSC), medical image segmentation (e.g., UNet++), salient object detection (e.g., PiCANet), and camouflage object segmentation (e.g., SINet). The full list of compared methods includes: FPN, PSPNet, Mask RCNN, HTC, MSRCNN, DSC, BDRAR, UNet++, PraNet, PiCANet, BASNet, CPD, PFANet, EGNet, F3Net, GCPANet, MINet-R, SINet, PFNet, UGTR, RankNet, BgNet, ERRNet, FEDER, MRRNet, and BCNet. The results of this comparison are summarized in Table 1, which presents quantitative performance across the four evaluation metrics on all three datasets. Our method consistently outperformed the competing approaches, achieving top results (highlighted in bold) across all metrics and datasets, underscoring its effectiveness and versatility.

Table 1: Performance metrics of various methods on Chameleon, Camo-Test, and COD10K-Test datasets.

Methods	**Year**	**Chameleon (76 images)**				**Camo-Test (250 images)**				**COD10K-Test (2026 images)**			
		$S_\alpha \uparrow$	$E_\phi^{ad} \uparrow$	$F_\beta^w \uparrow$	$M \downarrow$	$S_\alpha \uparrow$	$E_\phi^{ad} \uparrow$	$F_\beta^w \uparrow$	$M \downarrow$	$S_\alpha \uparrow$	$E_\phi^{ad} \uparrow$	$F_\beta^w \uparrow$	$M \downarrow$
FPN+ [45]	2017	0.794	0.835	0.590	0.075	0.684	0.791	0.483	0.131	0.697	0.711	0.411	0.075
PSPNet+ [46]	2017	0.773	0.814	0.555	0.085	0.663	0.778	0.455	0.139	0.678	0.688	0.377	0.080
Mask-RCNN+ [47]	2017	0.643	0.780	0.518	0.099	0.574	0.716	0.430	0.151	0.613	0.750	0.402	0.080
UNet++ [52]	2018	0.695	0.808	0.501	0.094	0.599	0.740	0.392	0.149	0.623	0.718	0.350	0.086
DSCA [50]	2018	0.850	0.888	0.714	0.050	0.736	0.830	0.592	0.105	0.758	0.788	0.542	0.052
PiCANet [54]	2018	0.769	0.836	0.536	0.085	0.609	0.753	0.356	0.156	0.649	0.678	0.322	0.090
BDRAR [51]	2018	0.779	0.831	0.663	0.064	0.759	0.825	0.664	0.093	0.753	0.836	0.591	0.051
HTC [48]	2019	0.517	0.490	0.204	0.129	0.476	0.442	0.174	0.172	0.548	0.521	0.221	0.088
BASNet [49]	2019	0.637	0.682	0.443	0.091	0.617	0.670	0.413	0.133	0.641	0.708	0.419	0.073
CPD [55]	2019	0.853	0.878	0.706	0.052	0.726	0.802	0.550	0.115	0.747	0.763	0.508	0.059
PFANet [57]	2019	0.679	0.732	0.378	0.144	0.659	0.735	0.391	0.172	0.636	0.619	0.286	0.128
EGNet [36]	2019	0.848	0.879	0.702	0.050	0.732	0.827	0.583	0.104	0.737	0.777	0.509	0.056
F3Net [37]	2020	0.854	0.899	0.749	0.045	0.779	0.840	0.666	0.091	0.786	0.832	0.617	0.046
GCPANet [16]	2020	0.876	0.891	0.748	0.041	0.778	0.840	0.646	0.092	0.791	0.799	0.592	0.045
PraNet [53]	2020	0.860	0.898	0.763	0.044	0.769	0.833	0.663	0.094	0.789	0.839	0.629	0.045
MINet-R [58]	2020	0.844	0.919	0.746	0.040	0.749	0.835	0.635	0.090	0.759	0.832	0.580	0.045
SINet* [40]	2020	0.869	0.899	0.740	0.044	0.751	0.834	0.606	0.100	0.771	0.797	0.551	0.051
PFNet [59]	2020	0.882	0.942	0.810	0.033	0.782	0.852	0.695	0.085	0.800	0.868	0.660	0.040
UGTR [31]	2021	0.888	0.921	0.804	0.031	0.784	0.858	0.747	0.086	0.817	0.850	0.670	0.036
RankNet* [19]	2021	0.890	0.936	0.835	0.030	0.787	0.859	0.756	0.080	0.804	0.833	0.699	0.037
BgNet* [60]	2022	0.885	0.936	0.822	0.032	0.804	0.872	0.766	0.075	0.804	0.866	0.676	0.039
FRRNet* [61]	2022	0.877	0.923	0.815	0.036	0.760	0.830	0.722	0.088	0.780	0.856	0.656	0.044
FEDER* [17]	2023	0.887	0.943	0.847	0.030	0.804	0.877	0.765	0.071	0.822	0.891	0.740	0.032
SegMaR* [62]	2023	0.882	0.924	0.812	0.033	0.781	0.870	0.727	0.076	0.822	0.859	0.695	0.036
BCNet [63]	2023	0.833	0.932	0.821	0.034	0.799	0.875	0.759	0.076	0.800	0.858	0.673	0.041
CAMEMF	2024	0.853	0.914	0.776	0.028	0.802	0.865	0.727	0.069	0.802	0.890	0.676	0.032
Our (S)	2025	0.884	0.854	0.861	0.039	0.830	0.788	0.826	0.075	0.785	0.705	0.787	0.049
Our(M)	2025	0.901	0.883	0.896	0.028	0.855	0.816	0.857	0.065	0.833	0.757	0.831	0.039
Our(L)	2025	0.915	0.895	0.912	0.026	0.890	0.870	0.894	0.047	0.857	0.794	0.851	0.033

4.5 Performance Evaluation

To demonstrate the proposed model capabilities and transparency, the proposed models are evaluated on the COD10K individual classes. The table 2 shows the model performance.

Table 2: COD10K(2026) Data

	COD10K(2026)			
	$S_\alpha \uparrow$	$E_\phi^{ad} \uparrow$	$F_\beta^w \uparrow$	$M \downarrow$
		Amphibians		
CAT(S)	0.821	0.744	0.845	0.047
CAT(M)	0.869	0.816	0.880	0.035
CAT(L)	0.892	0.865	0.910	0.028
		Aquatic		
CAT(S)	0.783	0.731	0.783	0.068
CAT(M)	0.833	0.788	0.831	0.051
CAT(L)	0.840	0.796	0.840	0.048
		Flying		
CAT(S)	0.812	0.742	0.780	0.038
CAT(M)	0.857	0.792	0.822	0.030
CAT(L)	0.873	0.820	0.837	0.026
		Terrestrial		
CAT(S)	0.752	0.637	0.785	0.048
CAT(M)	0.802	0.688	0.828	0.041
CAT(L)	0.832	0.746	0.851	0.034
		Other		
CAT(S)	0.753	0.647	0.835	0.062
CAT(M)	0.835	0.727	0.885	0.044
CAT(L)	0.791	0.706	0.868	0.053

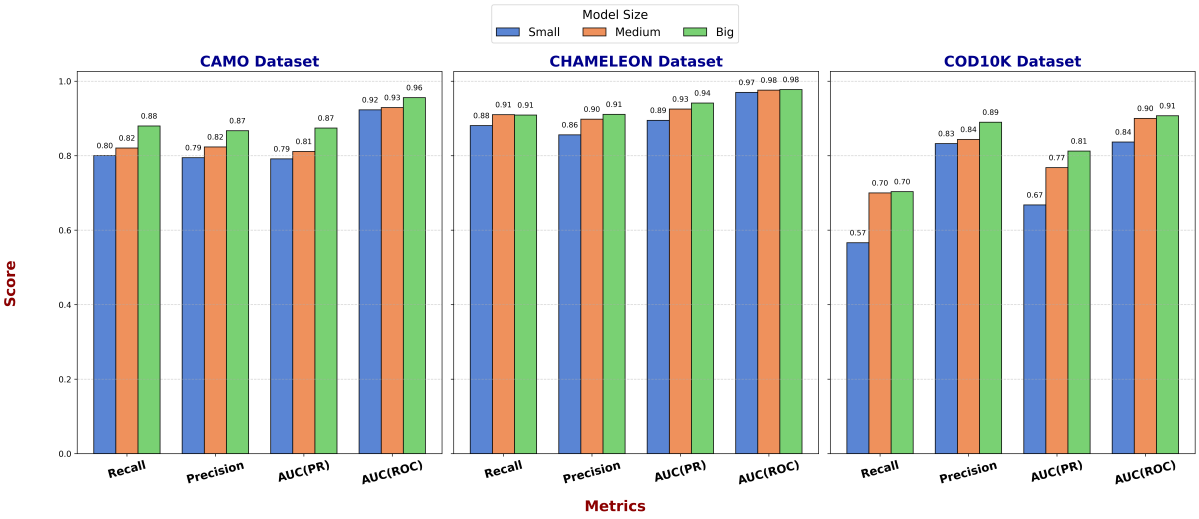

Figure 3: Performance comparison of the proposed model across different sizes (Small, Medium, Big) on CAMO, CHAMELEON, and COD10K datasets using Recall, Precision, AUC(PR), and AUC(ROC) metrics.

The bar chart in Figure 3 complements the evaluation by providing a visual comparison of the proposed models performance on the CAMO, CHAMELEON, and COD10K datasets, evaluated using Recall, Precision, AUC(PR), and AUC(ROC) metrics. On the CAMO dataset, the Big model achieves the highest scores in Recall (0.88), Precision (0.87), AUC(PR) (0.87), and AUC(ROC) (0.96). For the

CHAMELEON dataset, the Big model consistently excels, with top scores in Recall (0.91), Precision (0.91), AUC(PR) (0.94), and AUC(ROC) (0.98), indicating strong performance across all metrics. On the COD10K dataset, performance is notably lower, especially for the Small model (e.g., Recall of 0.57), but the Big model still leads with Recall (0.70), Precision (0.89), AUC(PR) (0.81), and AUC(ROC) (0.91). These results demonstrate that larger model sizes generally enhance performance across datasets, with the COD10K dataset appearing more challenging, as evidenced by lower overall scores compared to CAMO and CHAMELEON.

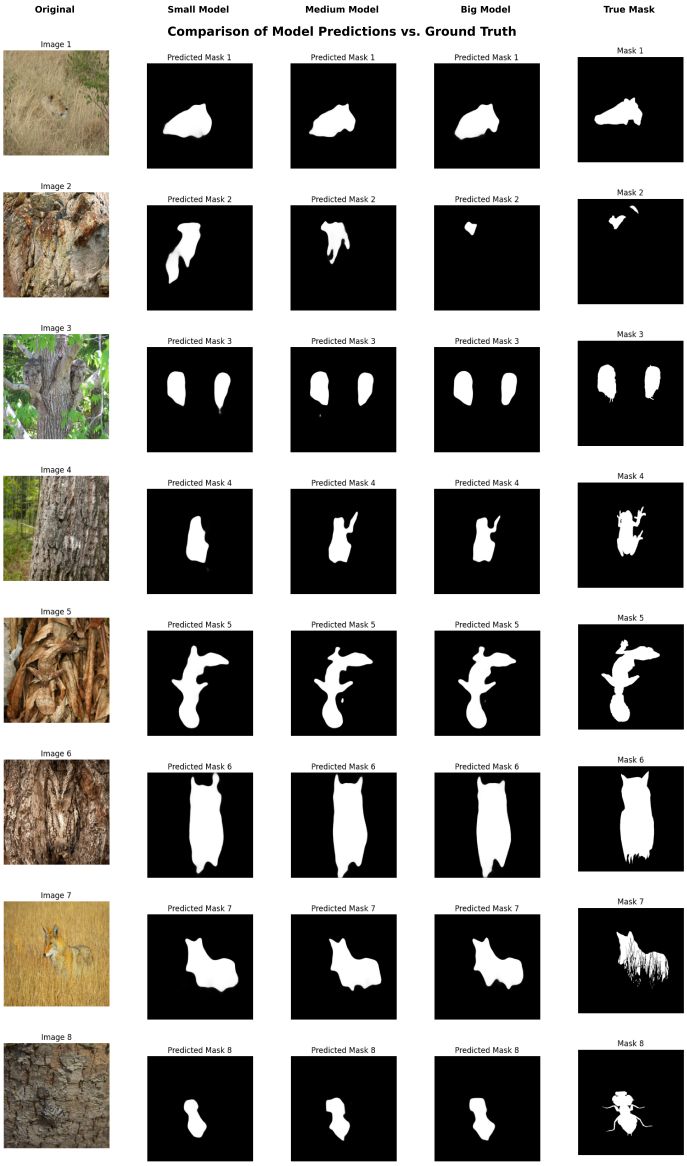

Figure 4: Prediction of proposed Small, Medium and Large Model with Ground truth Label

5 Conclusion

Our model integrates a SegFormer backbone with a novel attention-based decoder for effective camouflaged object segmentation. The introduction of three model variants allows flexibility in performance vs. computational cost. Future work includes real-world applications and further refinement of attention mechanisms.