A SHAPE-MEMORY NETWORK

Mechanism To illustrate the mechanisms of the Shape-Memory Network (SMN), we first define the mathematical notations and expressions. Let $\mathbb{E}$ and $\mathbb{D}$ be the encoder and decoder of the SMN, respectively. Then, the encoded feature-map (f_k) would be represented as $\mathbb{E}(I_i) = [f_k]_{k=1,2,...,\mathcal{C}}$, where $I_i \in I \subset \mathbb{R}^{H\times W\times 3}$ is the input image with its height (H) and width (W), and $\mathcal{C}$ is the number of feature-map, and the final output, the segmentation map ($Y_i \in Y \subset \mathbb{R}^{H\times W\times C}$), is represented as $Y = \mathbb{D}([f_k]) = (\mathbb{D}\circ\mathbb{E})(I)$, where C is the number of categories in the datasets. In the pipeline for the class activation map (CAM), the encoded feature-maps are average-pooled, such that $\sum_{x,y} f_k(x,y)$ represents an individual feature. Here, suppose W_k^c for the weight for the density-regression for a category (c), and then density prediction($d^c : \mathbb{R}^{H\times W\times 3} \to \mathbb{R}$) is calculated as $d^c(I_i) = \sum_k W_k^c \sum_{x,y} f_k(x,y) = \sum_k W_k^c \sum_{x,y} \mathbb{E}(I_i)$, such that $[d^c(I_i)]_{c=1,2,...,C} \in \mathbb{R}^C$. Furthermore, the predicted densities for each category are mapped to the control signal ($s^{\text{ct}} \in \mathbb{R}^{\mathbb{N}}$) with the number of individual pixels of feature-maps in $\mathbb{D}$ ($\mathbb{N}$) via dense layers with the trainable matrix ($\mathcal{M} \in \mathbb{R}^{C\times\mathbb{N}}$) such that $s^{\text{ct}} = [d^c(I_i)]\cdot\mathcal{M}$, where $\cdot$ is the matrix multiplication. Subsequently, the s^{ct} is imported into the $\mathbb{D}$, and thus the final prediction of the SMN is implemented in detail as below:

$$\begin{aligned} Y_i &= \mathbb{D}(\mathbb{E}(I_i); s^{\text{ct}}) \\ &= \mathbb{D}\Big(\mathbb{E}(I_i); [\sum_k^{\mathcal{C}} W_k^c \sum_{x,y} \mathbb{E}(I_i)]\cdot\mathcal{M}\Big) \end{aligned} \tag{3}$$

Note that all elements of the segmentation map (Y_i) are the softmax output, such that $0 \le Y_i\|_{h,w,c} \le 1$. Therefore, we can define the trainable parameters of the SMN as (1) parameters of encoder and decoder, such that $\theta_{\mathbb{E}}$ and $\theta_{\mathbb{D}}$; (2) parameters (W_k^c) for the CAM pipeline as a dense layer; (3) matrix ($\mathcal{M}$) to map the predicted density to a control signal. Note that only the matrix $\mathcal{M}$ is optimized in the inference phase to change the structure of the SMN. To summarize, the key outputs by the SMN are listed as below:

$$\begin{aligned} \text{Segmentation Map: } Y_i &= \mathbb{D}\Big(\mathbb{E}(I_i); [\sum_k^{\mathcal{C}} W_k^c \sum_{x,y} \mathbb{E}(I_i)]\cdot\mathcal{M}\Big) \\ \text{Density-regression: } d^c(I^i) &= \sum_k W_k^c \sum_{x,y} \mathbb{E}(I_i) \\ \text{Class Activation Map: } \mathbf{C}^c(I^i) &= \sum_k W_k^c f_k = \sum_k W_k^c \mathbb{E}(I^i) \\ \text{Entropy-map: } \mathcal{E}(I^i) &= -Y_i \log Y_i \end{aligned} \tag{4}$$

In addition, as illustrated in Sec.3.2 in the manuscript, remember that the entropy-map function (EM(I^i)) reconstructs the entropy-map from the CAMs ($\mathbf{C}^c$ for $c = 1, 2, ..., C$) as below:

$$\text{EM}(I^i)\|_{h,w} = -\sum_c^C \bar{d}^c(I^i)\|_{h,w} \log \bar{d}^c(I^i)\|_{h,w} \text{ where } \bar{d}^c(I^i)\|_{h,w} = \frac{e^{[\sum_k W_k^c \mathbb{E}(I^i)]\|_{h,w}}}{\sum_c^C e^{[\sum_k W_k^c \mathbb{E}(I^i)]\|_{h,w}}} \tag{5}$$

The training process of the SMN is illustrated in Sec 3.1 of the manuscript. The predictive procedure of the SMN in the inference phase is listed as the following:

1. The SMN generates the pseudo-labels for the segmentation map and density-regressions.
2. The entropy-maps are generated via the pipeline of the segmentation ($\mathcal{E}(I^i)$) and the reconstruction algorithm (EM(I^i)).
3. To fine-tune $\mathcal{M}$, minimize the similarity loss ($\mathcal{L}_{\text{ssim}}$) between $\mathcal{E}(I^i)$ and EM(I^i), such that $\mathcal{M}' = \underset{\mathcal{M}}{\text{argmin}} \mathcal{L}_{\text{ssim}}(\mathcal{E}(I^i), \text{EM}(I^i))$.

4. The SMN with $\mathcal{M}'$ predicts the final output as: $\mathbb{D}\Big(\mathbb{E}(I_i); [\sum_k^{\mathcal{C}} W_k^c \sum_{x,y} \mathbb{E}(I_i)] \cdot \mathcal{M}'\Big)$

It's important to note that during the training phase, $\mathcal{M}$ is trained, with each individual $\mathcal{M}$ being mapped to unique domains, representing different characteristics of contextual semantic information. Hence, $\mathcal{M}_i$ represents sub-domain $\mathcal{X}_i$, where the intersection of all $\mathcal{X}_i$ equates to the dataset $\mathcal{X}$, but no intersection exists among individual $\mathcal{X}_i$. During the inference phase, the similar $\mathcal{M}_i$ is derived by fine-tuning the Shape-Memory Network (SMN) for sample $x_i \in \mathcal{X}_i$. This process showcases how the SMN redeploys its saved structure by modifying its architecture, which led to the network being dubbed the *Shape-Memory Network*, and Fig. 5 in the manuscript verifies the effective utilization of similar M_i. Additionally, an illustration of the optimization of $\mathcal{M}$, as well as the provision of the final predicted segmentation map by the SMN, is presented in Algorithm 1.

Algorithm 1: Fine-tuning and Inference of the Shape-Memory Network

Input : sample x_i in test-set ($\mathcal{X} \subset \mathbb{R}^{H \times W \times 3}$), such that $x_i \in \mathcal{X}$, where H and W are height and width, respectively, and the pre-trained SMN (M).

Output : Predicted segmentation map ($y_i \in \mathcal{Y} \subset \mathbb{R}^{H \times W \times C}$) corresponding to input (x_i), where C is the number of category.

Assumption : $\mathcal{X} = \bigcup_i^N \mathcal{X}_i$ where N is the number of subset. $\bigcap_i^N \mathcal{X}_i = \emptyset$, indicating that the $\mathcal{X}_i$ represents distinct domain.

$\bar{y}^i \leftarrow \mathbb{D}\Big(\mathbb{E}(I_i); [\sum_k^{\mathcal{C}} W_k^c \sum_{x,y} \mathbb{E}(I_i)] \cdot \mathcal{M}\Big)$; `/* Predict pseudo-label */`

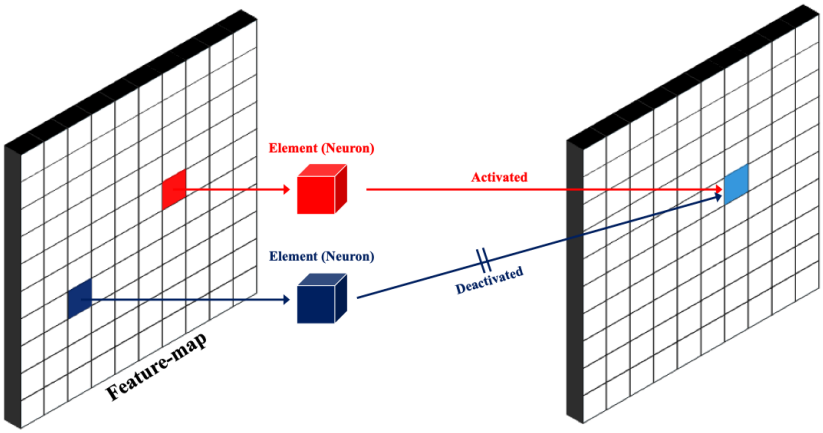
$\bar{\mathcal{C}}_1 \leftarrow -\bar{y}^i \log \bar{y}^i$; `/* Entropy-map by segmentation pipeline */`

$\bar{\mathcal{C}}_2 \leftarrow -\sum_c^C \bar{d}^c(I^i)\|_{h,w} \log \bar{d}^c(I^i)\|_{h,w}$; `/* Entropy-map by EM algorithm */`

$\mathcal{M}' \leftarrow \underset{\mathcal{M}}{\text{argmin}} \mathcal{L}_{\text{ssim}}(\mathcal{E}(I^i), \text{EM}(I^i))$; `/* Fine-tune` $\mathcal{M}$ `*/`

$y^i \leftarrow \mathbb{D}\Big(\mathbb{E}(I_i); [\sum_k^{\mathcal{C}} W_k^c \sum_{x,y} \mathbb{E}(I_i)] \cdot \mathcal{M}'\Big)$;

To summarize, the mechanism of the SMN is (1) to store the appropriate architecture for a certain domain; (2) to restore its structure corresponding to the input domain by fine-tuning a parameter; and (3) to provide precise prediction to the input image. Here, the structural mutation is achieved by fine-tuning the control signal that supervises the connections of neurons. The neural connections in the SMN are activated or deactivated based on the control signal, and thus fine-tuning the parameter that supervises the control signal enables the SMN to modify its network structure corresponding to the input image.



Appendix Figure 1: Schematic illustration of control neurons in the feature-maps.

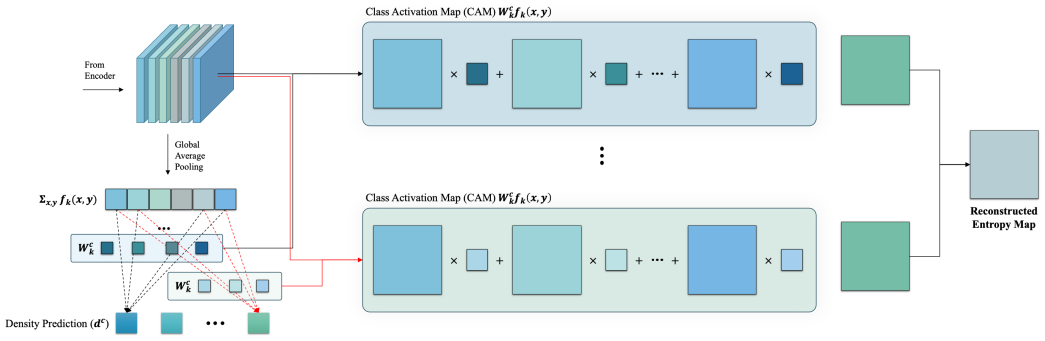
Appendix Fig. 1 illustrates how the control signal achieves the structural mutation. By activating and deactivating the output of each neuron, which is the individual element in a feature-map, the condition signal changes the structure of the SMN, and thus the control signal supervises the adaptive domain adaptation with respect to the contextual semantic information of inputs. Thereby, the structural adjustment by the control signal fine-tuned with the contextual semantic information can bring out a superior segmentation performance of the SMN.

Contributions To summarize, our contributions, in this paper, are listed below:

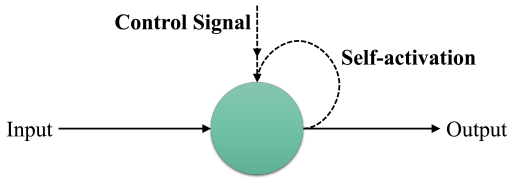
- **Construction of Shape-Memory Network**. We designed the shape-memory network that can explicitly interpret the contextual semantic information by employing the run-time adaptation method via structural modification.
- **Design of Control Neuron**. We newly devised a control neuron that can adaptively change the connections to other control neurons, leading to the implementation of structural modification of the SMN. This mechanism represents the close emulation of the human brain connectome and synapse mechanism.
- **Implementation of Entropy-Map Reconstruction Algorithm**. For the explicit interpretation of the contextual semantic information in the SMN, we newly devised the entropy-map reconstruction algorithm to train the SMN using the class activation maps regarding the contextual semantic information. The devised algorithm incorporates the contextual semantic information in the training of the SMN.

B Entropy-Map Reconstruction from Class Activation Map

In the previous research (Zhou et al., 2016), it was revealed that the Class Activation Map (CAM) identifies the regions of significant relevance to the primary task. As a result, when tasked with density regression, the CAM is influenced to concentrate on areas specific to the target object (c), leading to the derivation of **Proposition V**. In this context, density refers to the proportion of pixels in the input image representing the target object compared to the total pixel count. In response, we developed the Shape-Memory Network, which integrates multi-label density regression tasks to yield multiple CAMs for each category (c). Parameters specific to the density regression process for each category are then multiplied with the encoded feature-map, and then, a CAM for each category is generated. Subsequently, we incorporate the CAMs that exhibit attention areas for each category to produce a pseudo-entropy-map 2.



Appendix Figure 2: Schematic illustrations of reconstructing entropy-map from the class activation map (CAM). Each CMAs for each category are leveraged to generate entropy-map via a probability-based normalization method.



Appendix Figure 3: Schematic illustration of control neurons.

C CONTROL NEURON

Note that the control neuron represents an element in a feature-map (See Appendix Fig.1). Therefore, the control neuron refers to pixel-wise activation rather than convolutional weights. Suppose there are $\mathbb{N}$ numbers of control neurons in the SMN, and each control neuron has individual intrinsic threshold value (λ_n) for n^{th} control neuron. The output of the control neuron is activated when (1) the value of the control signal (s^{ct}) is above the intrinsic threshold value, such that $s^{\text{ct}} \geq \lambda_n$ or (2) self-activation is true. Therefore, the s_n^{out} is activated when the following condition is satisfied:

$$(\text{self-activation}) | (s^{\text{ct}} \geq \lambda_n) \tag{6}$$

Here, the self-activation indicates that the current control neuron (pixel or element) is more informative than other elements in the same feature-map. To avoid the loss of the informative features from the feature extraction process, the self-activation is designed. Therefore, the logical or ($|$) operator is placed in Eq. 6. Suppose f_k for the encoded feature-map by $\mathbb{E}$. In f_k with its height ($\mathcal{H}_k$) and width ($\mathcal{W}_k$), the number of control neurons are $\mathcal{H}_k\mathcal{W}_k$, and the n^{th} control neuron is more informative when the condition below is satisfied:

$$s_n^{\text{out}} \text{ is in top} - k\% \text{ among all elements in } f_k. \tag{7}$$

In the previous study Lee et al. (2022), the sampling elements met the Eq. 7 is formulated as below:

$$s_n^{\text{out}} \text{ is informative when } s_n^{\text{out}} \geq m_{f_k} + z_k * \mathrm{v}_{f_k} \tag{8}$$

where m_{f_k} and v_{f_k} are the mean and the standard deviation values of all elements in f_k, and the z_k refers to a statistical z-value for the Z-table corresponding to $k\%$.

To generalize, suppose a feature-map ($\mathcal{F}$) in the SMN, and the n^{th} control neuron in $\mathcal{F}$. Therefore, we can define the function ($g(s_n^{\text{out}}; \mathcal{F})$) that determines whether s_n^{out} is informative or not as below:

$$g(s_n^{\text{out}}; \mathcal{F}) = \begin{cases} 1(\text{if } s_n^{\text{out}} \geq m_{\mathcal{F}} + z_k * \mathrm{v}_{\mathcal{F}}) \\ 0(\text{else}) \end{cases} \tag{9}$$

In this case, the z_k is not trainable since the z_k is not arithmetically placed, but in the conditional statement. To make the z_k be trainable, the Heaviside step function and its approximation are utilized. Additionally, the Heaviside step function ($H(x)$) is approximated to the sigmoid function ($\sigma(-2\alpha x)$) with a large value of α. Therefore, we formulate the Eq. 9 as below:

$$\begin{aligned} g(s_n^{\text{out}}; \mathcal{F}) &= H(g(s_n^{\text{out}}; \mathcal{F}) - m_{\mathcal{F}} + z_k * \mathrm{v}_{\mathcal{F}}) \\ &= \sigma\big(-2\alpha(g(s_n^{\text{out}}; \mathcal{F}) - m_{\mathcal{F}} + z_k * \mathrm{v}_{\mathcal{F}})\big) \end{aligned} \tag{10}$$

Here, α and z_k are trainable. Therefore, the self-activation that determines the current condition neuron is informative or not is trained during the training phase, and the pre-trained self-activation determines the activation of the condition neuron in the inference phase.

Furthermore, another condition in Eq. 6 related to the control signal is formulated using the approximation of the Heaviside step function as below:

$$\sigma\big(-2\alpha(s^{\text{ct}}-\lambda_n)\big) \tag{11}$$

In addition, the logical or operator is replaced by the addition operator in arithmetic and analysis, and thus the Eq. 6 is substituted as below:

$$\sigma\big(-2\alpha(g(s_n^{\text{out}};\mathcal{F})-m_{\mathcal{F}}+z_k*\mathrm{v}_{\mathcal{F}})\big)+\sigma\big(-2\alpha(s^{\text{ct}}-\lambda_n)\big) \tag{12}$$

Therefore, let the input signals be s^{in}, and thus the final output ($\mathbf{s}_n^{\text{out}}$) value of the n^{th} control signal is provided as below:

$$\mathbf{s}_n^{\text{out}}=s^{\text{in}}*\Big(\sigma\big(-2\alpha_1(g(s_n^{\text{out}};\mathcal{F})-m_{\mathcal{F}}+z_k*\mathrm{v}_{\mathcal{F}})\big)+\sigma\big(-2\alpha_2(s^{\text{ct}}-\lambda_n)\big)\Big) \tag{13}$$

Here, in addition to the trainable parameters in **Appendix A**, the z_k, α_1, and α_2 are trainable, and z_k represents the adaptive threshold to discriminate the informative features, and α_1 and α_2 are the conditional values for the approximation. In the empirical analysis and experiments, the value of α is trained at nearly 10.0.

D Experimental Environment Description

Implementations The experiments were implemented in the Apple Macbook Pro with M1 Max and 64GB unified memories. Besides, we developed our neural network and the state-of-the-art deep learning models using Tensorflow (for ARM processor) version 2.9.0 (Abadi et al., 2016) for precise implementation. For the training, the batch size (Bottou, 2010) of the training was set to 32, and the Adam optimizer was utilized with the default values of all parameters (Kingma & Ba, 2014). Every parameter of the neural networks and the optimizer was initialized with the Gaussian distribution, of which the mean and the standard deviation values are 0.0 and 1.0.

Comparative Models To demonstrate the segmentation performance of the Shape-Memory Network (SMN), four groups of deep learning models were utilized as shown in the followings; (1) Baseline models including the early vanilla models of U-NetRonneberger et al. (2015), and SegFormer (Xie et al., 2021); (2) Multi-Path models for the segmentation task including LADDERNET (Zhuang, 2018) and MPDNet (Bai & Zhou, 2020); (3) SotA models for the segmentation task, including InterImage (Wang et al., 2022b) and BeiT-3 (Wang et al., 2022c); (4) SotA models for the video object segmentation (VOS), including Xmem (Cheng & Schwing, 2022) and AOST (Yang et al., 2022). The baseline networks were compared to demonstrate the standard feasibility of the SMN for the segmentation task. While the SoTA models, used here, were utilized to exhibit superior segmentation performance of the SMN for the benchmark datasets of scene parsing and autonomous driving, including VOS. Here, the best-performing SotA models were selected by referring to *Kaggle* benchmark lists. Additionally, the multi-path models were employed to compare the SMN in terms of the ensemble models for Eq. (1) in the manuscript.

Dataset In the experiments, five categories of distinct datasets were employed to evaluate the segmentation performance of the SMN compared to other baseline and SotA models; (1) Scene parsing benchmark using ADE20K (Zhou et al., 2017) and Youtube-VOS (Xu et al., 2018); (2) Autonomous driving using BDD100K (Yu et al., 2020); (3) Aerial image datasets of Inria (Maggiori et al., 2017b;a) and LoveDA (Wang et al., 2021); (4) Medical Imaging datasets using MRI for a brain tumor (Buda et al., 2019) and ultrasound dataset for breast cancer (Al-Dhabyani et al., 2019); (5) Synthetic images of GTA5 (Richter et al., 2016). To demonstrate the general feasibility of SMN for semantic segmentation, the datasets for scene understanding and autonomous driving datasets were utilized. In addition, since the density, which is a crucial feature for SMN, of objects is most important in the segmentation of aerial images, the benchmarks using aerial images were utilized. Furthermore, to evaluate the scalability of the SMN, the medical imaging datasets and benchmark for the synthetic images were employed. Note that since the density of the target object, especially the disease area,

is a significant key feature in medical imaging, the SMN could be expected to provide its superior segmentation performance in the medical imaging field. Furthermore, the precise segmentation performance of the SMN could provide the potential for transfer learning and extensibility to large-scale models. For training models, the images in each dataset are divided into ten-fold for the k-fold cross-validation.

Comparison to Domain Adaptation Models A great deal of DA methods, such as adversarial training (Ganin et al., 2016), maximum mean discrepancy minimization (Tzeng et al., 2014), unsupervised DA (Ganin & Lempitsky, 2015), and self-ensembling (French et al., 2017), have demonstrated success in reducing discrepancies between distinct source and target domains. However, methods typically assume the availability of labeled source domain data and unlabeled target domain data during the training phase, a condition that may not hold in real-world scenarios. Test-time DA (TTDA) methods, in contrast, aim to refine models at the inference stage, by leveraging the test data distribution without explicit access to the labels. Techniques such as transductive parameter transfer (Shu et al., 2018), and test-time self-supervised learning (Azimi et al., 2022; Lee et al., 2021; Wang et al., 2022a) have been proposed to bridge the gap between the training and test data distributions.

Despite the promising results achieved by the aforementioned methods, they are primarily designed for addressing domain discrepancies between two or more distinct domains, rather than within a single domain. To apply the domain adaptation method, two significantly distinct domains should be identified. However, in our study, the key factor for the domain discrepancy is contextual semantic information, and the contextual semantic information could be identified by the deep learning models, not by the human, and thus the labels for the different domains regarding the contextual semantic information could not be provided. Therefore, despite the promising performance of the domain adaptation decreasing the domain gap, the domain adaptation method could not be applied and implemented to resolve the issues addressed in the problem statement (Sec. 2).

E EXPERIMENTS

Verification of $\mathcal{M}$ Appendix Fig. 4 illustrates the similarity between $\mathcal{M}_i$ and $\mathcal{M}_j$ for samples of x_i and x_j in the same domain $\mathcal{X}$ alongside three error rates. This experiment was conducted to measure the justification that fine-tuning $\mathcal{M}$ could represent similar or different architecture for the SMN within different domains.

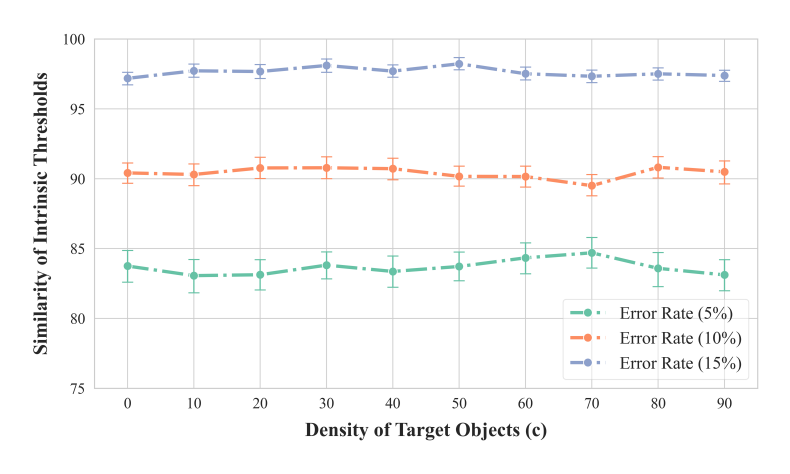
To measure the similarity, the following function is devised:

$$\mathcal{S}(x, y; r) := \begin{cases} 1 \text{ (if } \frac{|x-y|}{x} \leq r) \\ 0 \text{ (else)} \end{cases} \tag{14}$$

where $0 \leq r \leq 1$ represents the error rate, and thus $\mathcal{S}$ represent 1 if two elements is within the error rate. Here, the Similarity of the Intrinsic Threshold is calculated below:

Appendix Table 1: Detailed description of the datasets. To validate, 10-fold cross-validation was used.

Dataset	Samples	Train	Test	Category
ADE20K	27,574	24,817	2,757	150
Youtube-VOS	7,945	7,150	795	65
BDD100K	8,000	7,200	800	20
Inria	144,000	129,600	14,400	2
LoveDA	4191	3,772	419	8
BrainMRI	7,858	7,073	785	2
BUSI	789	709	80	2
GTA5	24,966	22,470	2,496	27



Appendix Figure 4: Similarity of the intrinsic threshold of control neurons containing the similar density of the target objects.

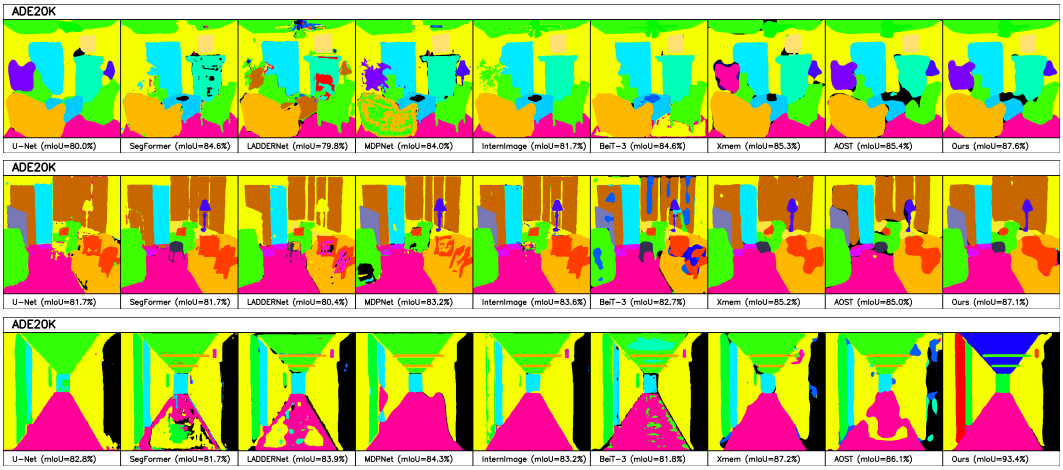
$$\frac{1}{C\mathbb{N}} \sum_{c,n}^{C,\mathbb{N}} \mathcal{S}(\mathcal{M}_i\|_{c,n}, \mathcal{M}_j\|_{c,n}; r) \tag{15}$$

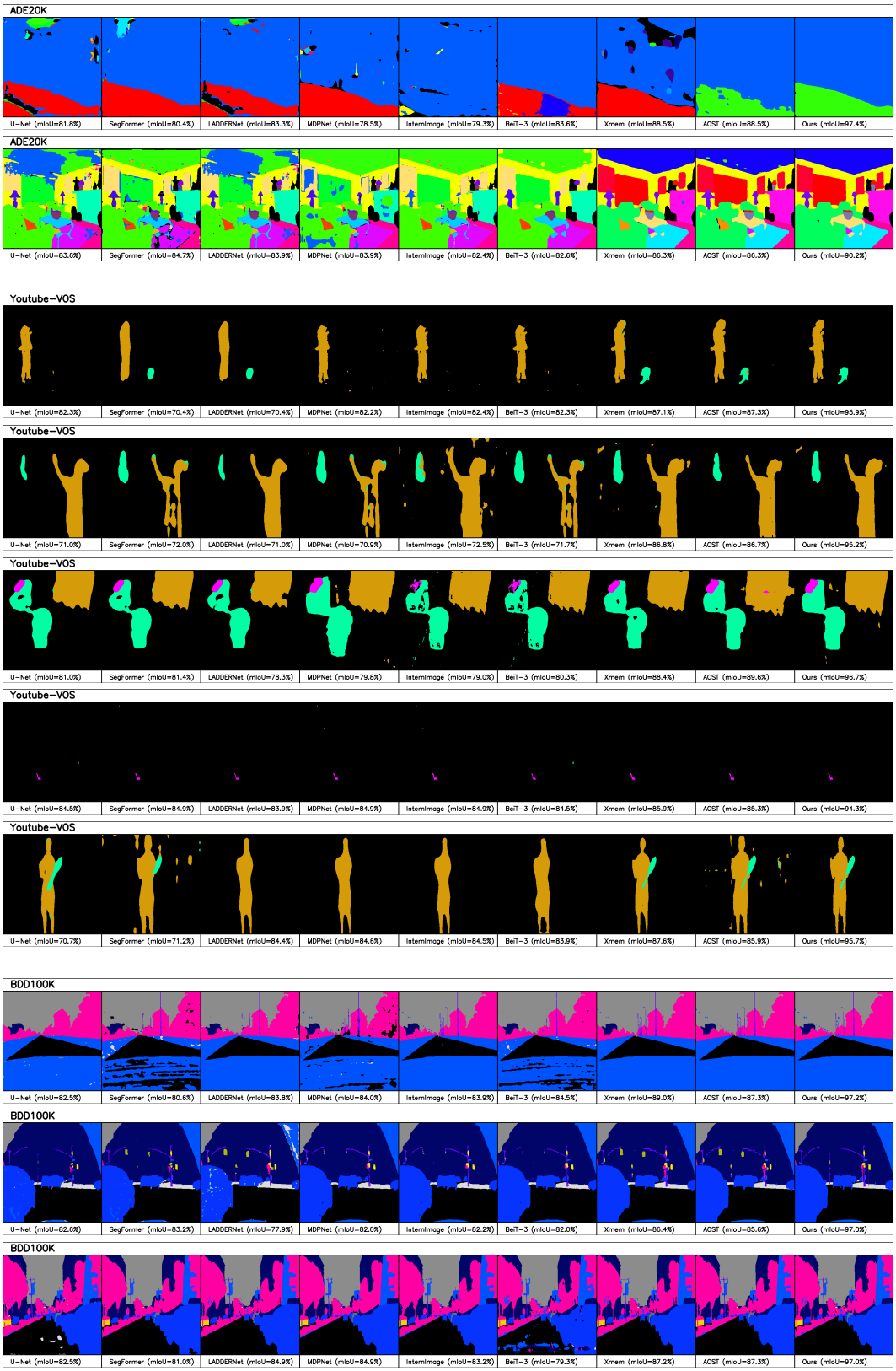
If the error rate decreases, the Similarity of the Intrinsic Threshold guarantees a higher similarity, whereas a large value of the error rate is a rough condition. Therefore, Appendix Fig. 4 verifies that the fine-tuned $\mathcal{M}$ exhibits similar values regarding the same domain.

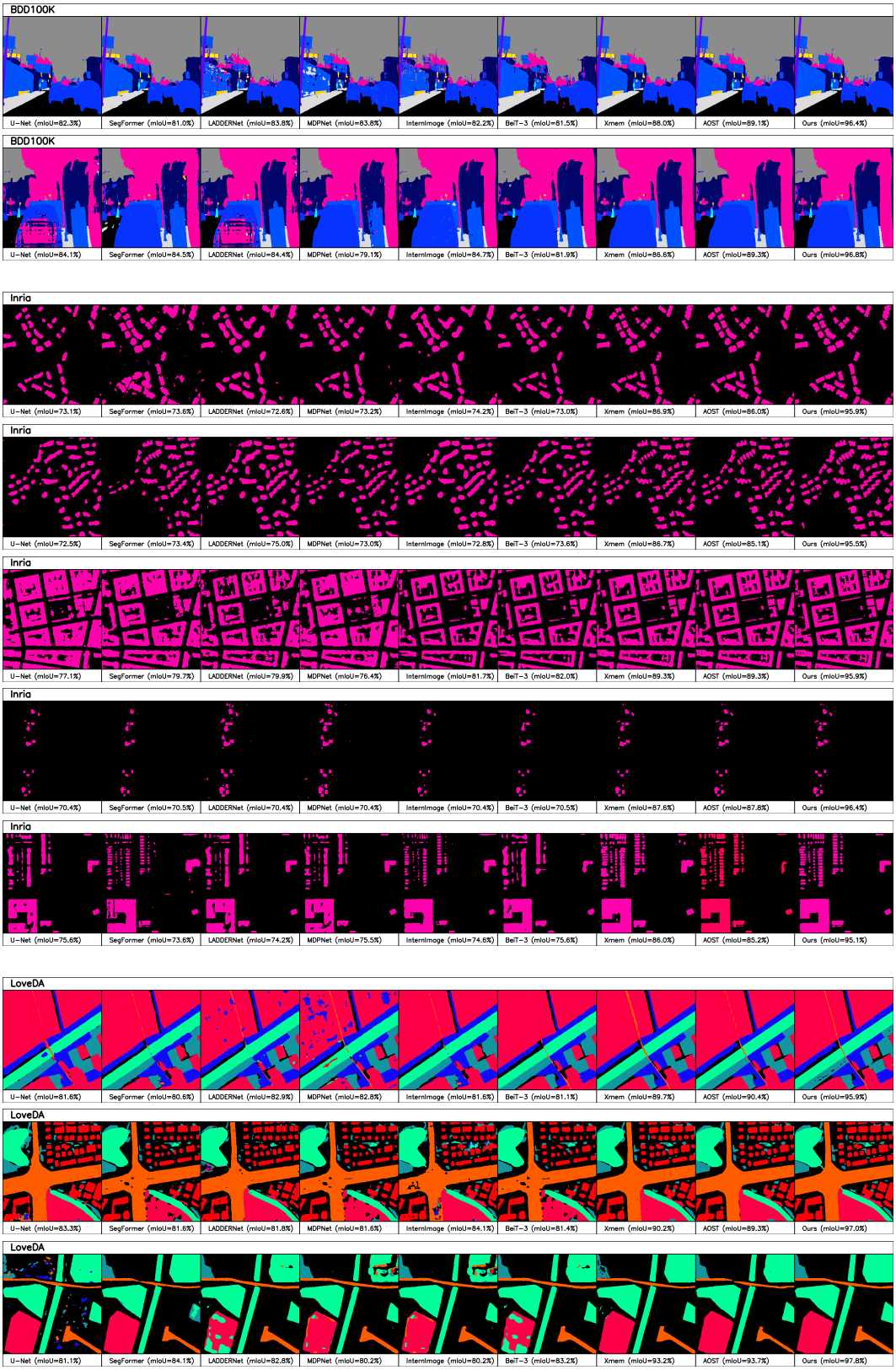
The SMN contains a small number of parameters compared to other state-of-the-art models, but the SMN significantly provides precise prediction in the segmentation task due to its effective fine-tuning mechanism. Additionally, despite the fine-tuning mechanism of the SMN, the SMN exhibits an efficient FPS due to only a small number of parameters ($\mathcal{M}$) being optimized in an inference phase.

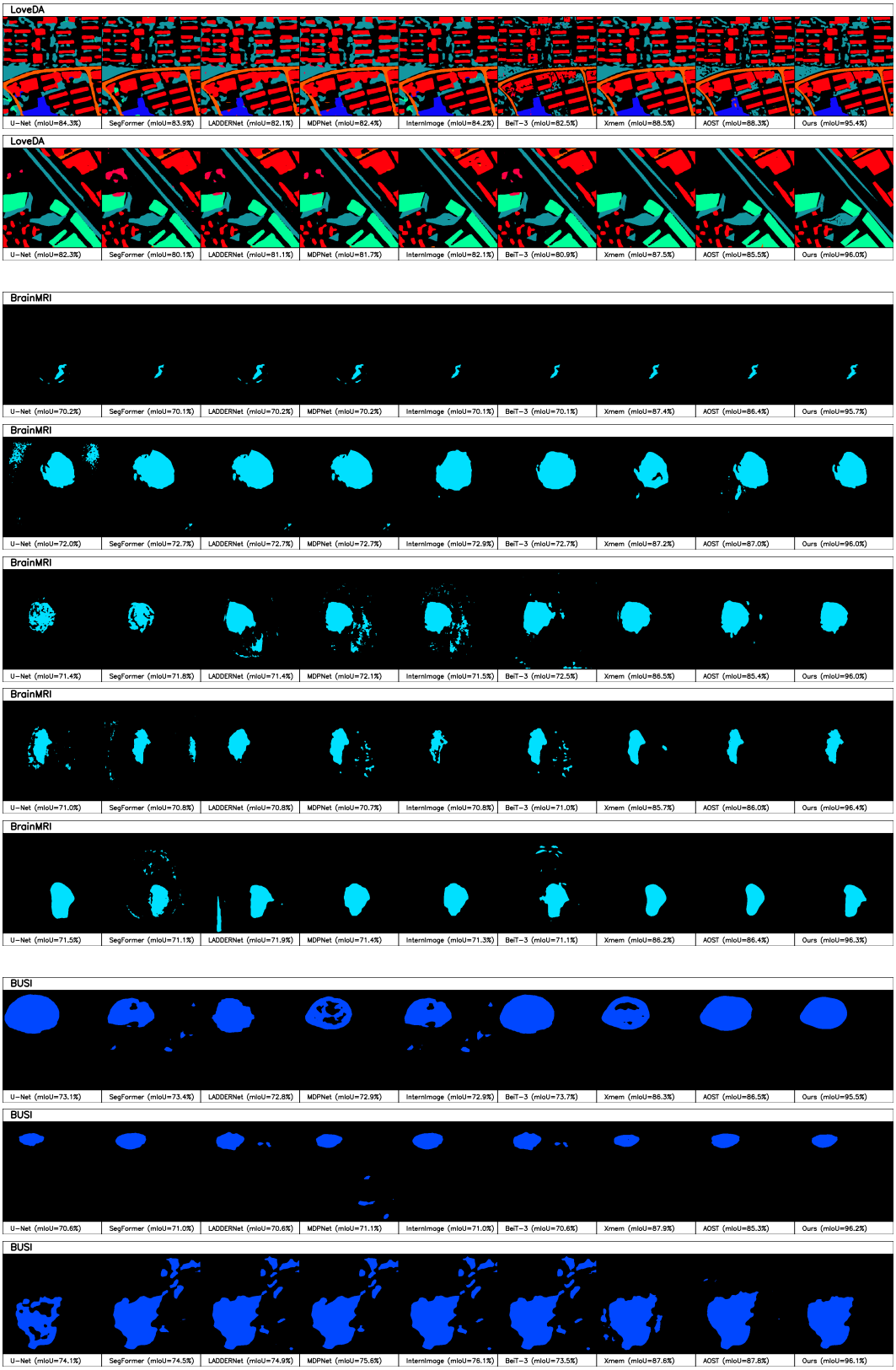
Segmentation Performance This section illustrates the evaluation results of our model compared to other deep learning models, including baseline models, multi-path models, SotA models for segmentation, and the SotA models for VOS.

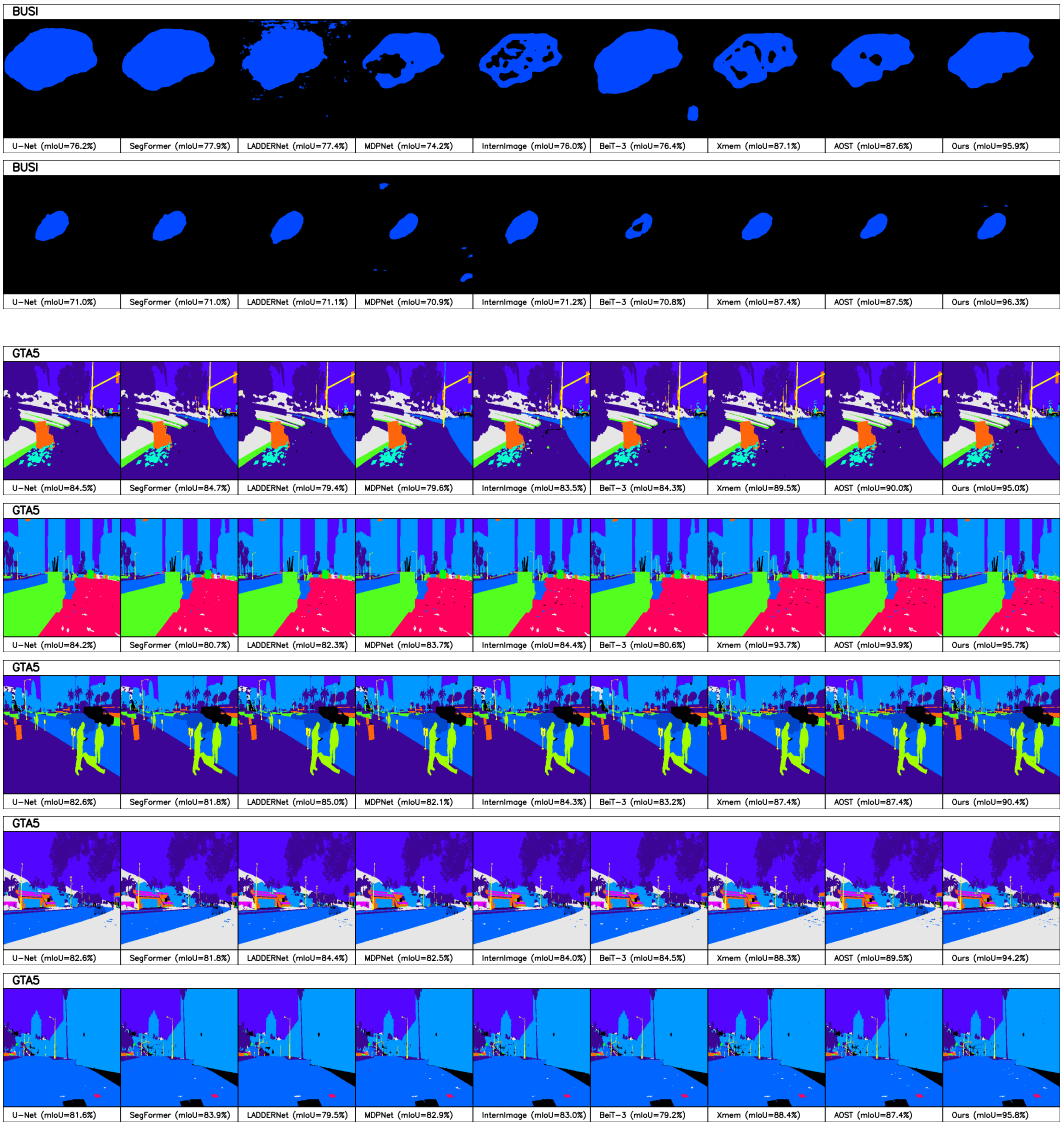
In addition, the figure below illustrates the samples of the predicted segmentation by the SMN and other comparative models in eight datasets.




ADE20K
U-Net (mIoU=81.8%)
SegFormer (mIoU=80.4%)
LADDERNet (mIoU=83.3%)
MDPNet (mIoU=78.5%)
InternImage (mIoU=79.3%)
BeiT-3 (mIoU=83.6%)
Xmem (mIoU=88.5%)
AOST (mIoU=88.5%)
Ours (mIoU=97.4%)
ADE20K
U-Net (mIoU=83.6%)
SegFormer (mIoU=84.7%)
LADDERNet (mIoU=83.9%)
MDPNet (mIoU=83.9%)
InternImage (mIoU=82.4%)
BeiT-3 (mIoU=82.6%)
Xmem (mIoU=86.3%)
AOST (mIoU=86.3%)
Ours (mIoU=90.2%)
Youtube-VOS
U-Net (mIoU=82.3%)
SegFormer (mIoU=70.4%)
LADDERNet (mIoU=70.4%)
MDPNet (mIoU=82.2%)
InternImage (mIoU=82.4%)
BeiT-3 (mIoU=82.3%)
Xmem (mIoU=87.1%)
AOST (mIoU=87.3%)
Ours (mIoU=95.9%)
Youtube-VOS
U-Net (mIoU=71.0%)
SegFormer (mIoU=72.0%)
LADDERNet (mIoU=71.0%)
MDPNet (mIoU=70.9%)
InternImage (mIoU=72.5%)
BeiT-3 (mIoU=71.7%)
Xmem (mIoU=86.8%)
AOST (mIoU=86.7%)
Ours (mIoU=95.2%)
Youtube-VOS
U-Net (mIoU=81.0%)
SegFormer (mIoU=81.4%)
LADDERNet (mIoU=78.3%)
MDPNet (mIoU=79.8%)
InternImage (mIoU=79.0%)
BeiT-3 (mIoU=80.3%)
Xmem (mIoU=88.4%)
AOST (mIoU=89.6%)
Ours (mIoU=96.7%)
Youtube-VOS
U-Net (mIoU=84.5%)
SegFormer (mIoU=84.9%)
LADDERNet (mIoU=83.9%)
MDPNet (mIoU=84.9%)
InternImage (mIoU=84.9%)
BeiT-3 (mIoU=84.5%)
Xmem (mIoU=85.9%)
AOST (mIoU=85.3%)
Ours (mIoU=94.3%)
Youtube-VOS
U-Net (mIoU=70.7%)
SegFormer (mIoU=71.2%)
LADDERNet (mIoU=84.4%)
MDPNet (mIoU=84.6%)
InternImage (mIoU=84.5%)
BeiT-3 (mIoU=83.9%)
Xmem (mIoU=87.6%)
AOST (mIoU=85.9%)
Ours (mIoU=95.7%)
BDD100K
U-Net (mIoU=82.5%)
SegFormer (mIoU=80.6%)
LADDERNet (mIoU=83.8%)
MDPNet (mIoU=84.0%)
InternImage (mIoU=83.9%)
BeiT-3 (mIoU=84.5%)
Xmem (mIoU=89.0%)
AOST (mIoU=87.3%)
Ours (mIoU=97.2%)
BDD100K
U-Net (mIoU=82.6%)
SegFormer (mIoU=83.2%)
LADDERNet (mIoU=77.9%)
MDPNet (mIoU=82.0%)
InternImage (mIoU=82.2%)
BeiT-3 (mIoU=82.0%)
Xmem (mIoU=86.4%)
AOST (mIoU=85.6%)
Ours (mIoU=97.0%)
BDD100K
U-Net (mIoU=82.5%)
SegFormer (mIoU=81.0%)
LADDERNet (mIoU=84.9%)
MDPNet (mIoU=84.9%)
InternImage (mIoU=83.2%)
BeiT-3 (mIoU=79.3%)
Xmem (mIoU=87.2%)
AOST (mIoU=87.3%)
Ours (mIoU=97.0%)


BDD100K
U-Net (mIoU=82.3%)
SegFormer (mIoU=81.0%)
LADDERNet (mIoU=83.8%)
MDPNet (mIoU=83.8%)
InternImage (mIoU=82.2%)
BeiT-3 (mIoU=81.5%)
Xmem (mIoU=88.0%)
AOST (mIoU=89.1%)
Ours (mIoU=96.4%)
BDD100K
U-Net (mIoU=84.1%)
SegFormer (mIoU=84.5%)
LADDERNet (mIoU=84.4%)
MDPNet (mIoU=79.1%)
InternImage (mIoU=84.7%)
BeiT-3 (mIoU=81.9%)
Xmem (mIoU=86.6%)
AOST (mIoU=89.3%)
Ours (mIoU=96.8%)
Inria
U-Net (mIoU=73.1%)
SegFormer (mIoU=73.6%)
LADDERNet (mIoU=72.6%)
MDPNet (mIoU=73.2%)
InternImage (mIoU=74.2%)
BeiT-3 (mIoU=73.0%)
Xmem (mIoU=86.9%)
AOST (mIoU=86.0%)
Ours (mIoU=95.9%)
Inria
U-Net (mIoU=72.5%)
SegFormer (mIoU=73.4%)
LADDERNet (mIoU=75.0%)
MDPNet (mIoU=73.0%)
InternImage (mIoU=72.8%)
BeiT-3 (mIoU=73.6%)
Xmem (mIoU=86.7%)
AOST (mIoU=85.1%)
Ours (mIoU=95.5%)
Inria
U-Net (mIoU=77.1%)
SegFormer (mIoU=79.7%)
LADDERNet (mIoU=79.9%)
MDPNet (mIoU=76.4%)
InternImage (mIoU=81.7%)
BeiT-3 (mIoU=82.0%)
Xmem (mIoU=89.3%)
AOST (mIoU=89.3%)
Ours (mIoU=95.9%)
Inria
U-Net (mIoU=70.4%)
SegFormer (mIoU=70.5%)
LADDERNet (mIoU=70.4%)
MDPNet (mIoU=70.4%)
InternImage (mIoU=70.4%)
BeiT-3 (mIoU=70.5%)
Xmem (mIoU=87.6%)
AOST (mIoU=87.8%)
Ours (mIoU=96.4%)
Inria
U-Net (mIoU=75.6%)
SegFormer (mIoU=73.6%)
LADDERNet (mIoU=74.2%)
MDPNet (mIoU=75.5%)
InternImage (mIoU=74.6%)
BeiT-3 (mIoU=75.6%)
Xmem (mIoU=86.0%)
AOST (mIoU=85.2%)
Ours (mIoU=95.1%)
LoveDA
U-Net (mIoU=81.6%)
SegFormer (mIoU=80.6%)
LADDERNet (mIoU=82.9%)
MDPNet (mIoU=82.8%)
InternImage (mIoU=81.6%)
BeiT-3 (mIoU=81.1%)
Xmem (mIoU=89.7%)
AOST (mIoU=90.4%)
Ours (mIoU=95.9%)
LoveDA
U-Net (mIoU=83.3%)
SegFormer (mIoU=81.6%)
LADDERNet (mIoU=81.8%)
MDPNet (mIoU=81.6%)
InternImage (mIoU=84.1%)
BeiT-3 (mIoU=81.4%)
Xmem (mIoU=90.2%)
AOST (mIoU=89.3%)
Ours (mIoU=97.0%)
LoveDA
U-Net (mIoU=81.1%)
SegFormer (mIoU=84.1%)
LADDERNet (mIoU=82.8%)
MDPNet (mIoU=80.2%)
InternImage (mIoU=80.2%)
BeiT-3 (mIoU=83.2%)
Xmem (mIoU=93.2%)
AOST (mIoU=93.7%)
Ours (mIoU=97.8%)


LoveDA
U-Net (mIoU=84.3%) SegFormer (mIoU=83.9%) LADDERNet (mIoU=82.1%) MDPNet (mIoU=82.4%) InternImage (mIoU=84.2%) BeiT-3 (mIoU=82.5%) Xmem (mIoU=88.5%) AOST (mIoU=88.3%) Ours (mIoU=95.4%)
LoveDA
U-Net (mIoU=82.3%) SegFormer (mIoU=80.1%) LADDERNet (mIoU=81.1%) MDPNet (mIoU=81.7%) InternImage (mIoU=82.1%) BeiT-3 (mIoU=80.9%) Xmem (mIoU=87.5%) AOST (mIoU=85.5%) Ours (mIoU=96.0%)
BrainMRI
U-Net (mIoU=70.2%) SegFormer (mIoU=70.1%) LADDERNet (mIoU=70.2%) MDPNet (mIoU=70.2%) InternImage (mIoU=70.1%) BeiT-3 (mIoU=70.1%) Xmem (mIoU=87.4%) AOST (mIoU=86.4%) Ours (mIoU=95.7%)
BrainMRI
U-Net (mIoU=72.0%) SegFormer (mIoU=72.7%) LADDERNet (mIoU=72.7%) MDPNet (mIoU=72.7%) InternImage (mIoU=72.9%) BeiT-3 (mIoU=72.7%) Xmem (mIoU=87.2%) AOST (mIoU=87.0%) Ours (mIoU=96.0%)
BrainMRI
U-Net (mIoU=71.4%) SegFormer (mIoU=71.8%) LADDERNet (mIoU=71.4%) MDPNet (mIoU=72.1%) InternImage (mIoU=71.5%) BeiT-3 (mIoU=72.5%) Xmem (mIoU=86.5%) AOST (mIoU=85.4%) Ours (mIoU=96.0%)
BrainMRI
U-Net (mIoU=71.0%) SegFormer (mIoU=70.8%) LADDERNet (mIoU=70.8%) MDPNet (mIoU=70.7%) InternImage (mIoU=70.8%) BeiT-3 (mIoU=71.0%) Xmem (mIoU=85.7%) AOST (mIoU=86.0%) Ours (mIoU=96.4%)
BrainMRI
U-Net (mIoU=71.5%) SegFormer (mIoU=71.1%) LADDERNet (mIoU=71.9%) MDPNet (mIoU=71.4%) InternImage (mIoU=71.3%) BeiT-3 (mIoU=71.1%) Xmem (mIoU=86.2%) AOST (mIoU=86.4%) Ours (mIoU=96.3%)
BUSI
U-Net (mIoU=73.1%) SegFormer (mIoU=73.4%) LADDERNet (mIoU=72.8%) MDPNet (mIoU=72.9%) InternImage (mIoU=72.9%) BeiT-3 (mIoU=73.7%) Xmem (mIoU=86.3%) AOST (mIoU=86.5%) Ours (mIoU=95.5%)
BUSI
U-Net (mIoU=70.6%) SegFormer (mIoU=71.0%) LADDERNet (mIoU=70.6%) MDPNet (mIoU=71.1%) InternImage (mIoU=71.0%) BeiT-3 (mIoU=70.6%) Xmem (mIoU=87.9%) AOST (mIoU=85.3%) Ours (mIoU=96.2%)
BUSI
U-Net (mIoU=74.1%) SegFormer (mIoU=74.5%) LADDERNet (mIoU=74.9%) MDPNet (mIoU=75.6%) InternImage (mIoU=76.1%) BeiT-3 (mIoU=73.5%) Xmem (mIoU=87.6%) AOST (mIoU=87.8%) Ours (mIoU=96.1%)



Appendix Table 3: Segmentation Results of the SMN and other comparative models.

Network Complexity The SMN contains the trainable parameters of (1) parameters of encoder and decoder, such that $\theta_{\mathbb{E}}$ and $\theta_{\mathbb{D}}$; (2) parameters (W_k^c) for the CAM pipeline as a dense layer; (3) matrix ($\mathcal{M}$) to map the predicted density to a control signal. To verify the feasibility and scalability for a real-world application, we compared the SMN to other deep learning models in terms of the number of parameters (# of Param), the FLoating point Operations Per Second (FLOPs), and Frame Per Second for generating predictions (FPS). The predictions were performed using the Apple Macbook Pro with M1 Max and 64GB unified memories.

Appendix Table 2: Comparison analysis in terms of mean IoU.

		ADE20K	Youtube-VOS	BDD100K	GTA5
2*Baseline Model	U-Net	42.61% (±3.75)	77.12% (±3.38)	36.69% (±1.41)	65.84% (±1.81)
	SegFormer	46.72% (±3.93)	81.07% (±2.37)	42.59% (±3.83)	65.99% (±2.17)
2*Multi-Path	LADDERNet	52.66% (±3.44)	86.04% (±3.53)	41.13% (±3.44)	68.64% (±3.61)
	MPDNet	44.3% (±3.11)	83.57% (±2.54)	40.03% (±3.6)	70.69% (±3.42)
2*Seg SotA	InternImage	51.04% (±2.67)	85.13% (±2.2)	47.25% (±3.5)	70.94% (±3.27)
	BEiT-3	51.67% (±2.08)	86.67% (±3.02)	39.85% (±1.22)	70.9% (±3.23)
2*VOS	Xmem	44.88% (±1.78)	83.36% (±3.31)	42.69% (±3.35)	68% (±2.31)
	AOST	52.06% (±3.12)	87.09% (±2.04)	43.09% (±1.41)	71.06% (±2.32)
2*Ours	Ours - SA	48.84% (±2.42)	85.66% (±1.04)	43.81% (±3.59)	69.07% (±2.97)
	Ours	**55.76% (±2.9)**	**88.66% (±2.44)**	**48.83% (±1.14)**	**76.58% (±1.14)**
		Inria	LoveDA	BrainMRI	BUSI
2*Baseline Model	U-Net	62.96% (±3.34)	47.71% (±3.12)	75.11% (±1.64)	63.69% (±2.55)
	SegFormer	67.97% (±3.19)	51.33% (±3.34)	74.28% (±3.36)	71.41% (±3.19)
2*Multi-Path	LADDERNet	64.77% (±3.09)	49.66% (±3.59)	69.75% (±3.34)	60.36% (±2.88)
	MPDNet	64.51% (±1.09)	48.25% (±3.43)	67.43% (±3.89)	67.51% (±1.28)
2*Seg SotA	InternImage	68.6% (±3.27)	49.81% (±1.17)	**76.08% (±3.82)**	70.47% (±2.27)
	BEiT-3	66.69% (±1.82)	49.63% (±1.75)	66.08% (±3.78)	67.46% (±1.63)
2*VOS	Xmem	64.85% (±2.78)	51.29% (±3.12)	61.57% (±3.04)	67.24% (±3.41)
	AOST	69.3% (±3.75)	50.57% (±3.43)	75.22% (±2.83)	60.16% (±3.2)
2*Ours	Ours - SA	68.6% (±2.57)	50.4% (±2.69)	68.86% (±2.95)	72.99% (±3.42)
	Ours	**72.72% (±1.06)**	**54.28% (±1.31)**	74.82% (±1.77)	**75.22% (±1.24)**

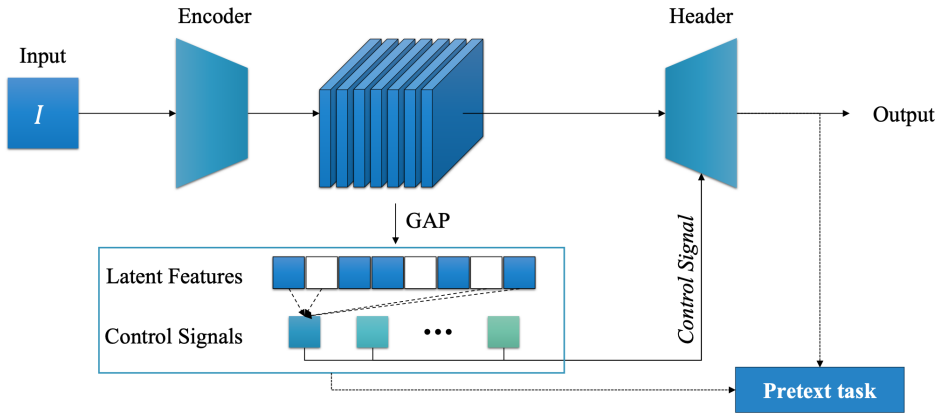
Appendix Table 4: Comparison analysis in terms of networks' complexities.

	U-Net	SegFormer	InternImage	BeiT-3	Xmem	AOST	SMN (Ours)
Resolution	512×512	512×512	384×384	384×384	512×512	512×512	512×512
# of Param	31.0M	64.1M	335M	1843M	-	65.6M	47.5M
FLOPs	224.6G	95.7G	163.2G	2859.9G	-	-	549.8G
FPS	42.5	30.7	42.6	10.2	41.7	35.2	32.8

F DISCUSSION

Extension to Other Tasks Our framework demonstrates significant potential for extension beyond semantic segmentation tasks. As illustrated in Appendix Fig. 5, the SMN architecture can be generalized via a modular design approach: maintaining the encoder with control neurons while allowing customization of the header and pretext task for specific applications. The adaptability of the SMN is achieved by two key components: (1) the latent features extracted from the encoder and (2) the control signals derived from the features. The latent features, representing high-level semantic information, are processed through task-specific headers to generate appropriate outputs (e.g., class probabilities for classification, bounding box coordinates for detection), while the control signals guide the structural adaptation of the network based on a pretext task appropriate for the target application. While our segmentation implementation uses density-based pretext tasks to identify spatial information, other applications might employ different self-supervised learning objectives - for instance, classification tasks could utilize feature correlation learning based on variational auto-encoder, while detection tasks might benefit from pretext tasks using object localization patterns.

The adaptability of our SMN architecture extends beyond semantic segmentation tasks through its modular design approach, as illustrated in Fig. 5. The architecture maintains its main feature extraction mechanism with control neurons while enabling task-specific customization through two key components: the decoder (header) and the pretext task. This design principle allows the network to be adapted for various computer vision tasks while preserving the benefits of our control neuron mechanism.



Appendix Figure 5: Generalized pipeline of SMN for various computer vision tasks.

In our preliminary study, we demonstrate this adaptability in object detection tasks, where we modified only the decoder while maintaining the control neuron mechanism and semantic information-based optimization. Our initial experiments on the COCO dataset show competitive results compared to recent state-of-the-art detection models like DiffusionDet, achieving 47.43 AP, 65.64 AP_{50}, and 52.21 AP_{75}. Notably, our approach achieves this performance with minimal architectural modifications, demonstrating particularly strong results in AP_l (63.24) for large object detection. For detection tasks, we leverage the same semantic information optimization strategy as used in segmentation, demonstrating the transferability of our core mechanism across different vision tasks.

COCO	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
DiffusionDet (1@ 500)	47.18	65.74	51.42	31.18	50.19	62.24
DiffusionDet (4@ 500)	47.36	65.62	52.13	30.72	50.37	63.18
Ours (SMN)	47.43	65.64	52.21	30.8	50.39	63.24

The flexibility of the SMN extends further through our generalized pipeline (Fig. 5), where the pretext task can be customized for different applications. While segmentation and detection tasks benefit

from semantic information optimization, other computer vision applications may require different self-supervised learning approaches. For classification tasks, we are exploring various pretext task approaches leveraging auto-encoders and VAE architectures, including feature correlation learning between augmented views, rotation prediction, and solving jigsaw puzzles of image patches. This modular architecture design, separating the core feature extraction mechanism from task-specific components, ensures that the primary strengths of our approach remain effective across different applications. The empirical experiments in both segmentation and detection tasks and additional ongoing exploration in classification demonstrate the broader applicability of our brain-inspired approach across various computer vision tasks, with the selection of appropriate pretext tasks being the key consideration for each specific application.

Reproducibility More detailed experimental results, including class-wise IoU values, and the code for the SMN will be available at https://github.com/Anonymous/Repo.