Appendix

A. Details of the Network Architecture

We have introduced the network architecture of RepMode in Sec. 3.2. To guarantee reproducibility, we provide more details in this section. As shown in Fig. 9, the encoder-decoder architecture of RepMode is mainly constructed of the symmetrical downsampling and upsampling blocks. Moreover, between the downsampling and upsampling blocks, two successive MoDE blocks are employed to further refine the feature maps. Finally, a MoDE block without BN and ReLU is used to produce predictions. It is worth noting that, using the proposed MoDE block and GatRep, any plain network designed for dense prediction tasks can obtain the powerful capability to handle multiple tasks and meanwhile maintain the original architecture, since only the convolutional layers need to be modified.

B. Pixel-Level Form of GatRep

In Sec. 3.4, we have described the matrix form of GatRep for an intuitive understanding. In this section, we provide the pixel-level form as an extension. Note that here we follow the notations described in Sec. 3.4.

Step1: serial merging. In this step, we aim to merge $\mathbf{W}$ and $\mathbf{W}^{\mathrm{a}}$ of an Avgp - Conv expert into an integrated kernel $\mathbf{W}^{\mathrm{e}}$. This merging is accomplished by using $\mathbf{W}$ to perform a convolution operation on $\mathbf{W}^{\mathrm{a}}$, formulated as

$$\mathbf{W}^{\mathrm{e}} = \mathbf{W} \circledast \mathbf{W}^{\mathrm{a}}, \tag{7}$$

which is equivalent to

$$\mathbf{W}^{\mathrm{e}}_{c_{\mathrm{O}},c_{\mathrm{I}},d,h,w} = \sum_{i=1}^{C_{\mathrm{I}}} \mathbf{W}_{c_{\mathrm{O}},i,1,1,1} * \mathbf{W}^{\mathrm{a}}_{i,c_{\mathrm{I}},d,h,w}, \tag{8}$$

where the subscripts denote the indexes of tensors in the corresponding dimensions and $*$ is the multiplication.

Step 2: parallel merging. In this step, we aim to merge the kernels of all experts $\mathbf{W}^{\mathrm{e}}_t$ where $t = 1, 2, ..., T$. This merging is accomplished by a linear weighted summation with the gating weights $\hat{\mathbf{G}}_t = \{\hat{\mathbf{g}}_t\}_{t=1}^{T}$, formulated as

$$\hat{\mathbf{W}}^{\mathrm{e}} = \sum_{t=1}^{T} \hat{\mathbf{g}}_t \odot \mathrm{Pad}(\mathbf{W}^{\mathrm{e}}_t, K'), \tag{9}$$

which is equivalent to

$$\hat{\mathbf{W}}^{\mathrm{e}}_{c_{\mathrm{O}},c_{\mathrm{I}},d,h,w} = \sum_{t=1}^{T} \hat{\mathbf{g}}_{t,c_{\mathrm{O}}} * \mathbf{W}^{\mathrm{p}}_{t,c_{\mathrm{O}},c_{\mathrm{I}},d,h,w}, \tag{10}$$

where $\mathbf{W}^{\mathrm{p}}$ denotes the kernel processed by $\mathrm{Pad}(\cdot, K')$.

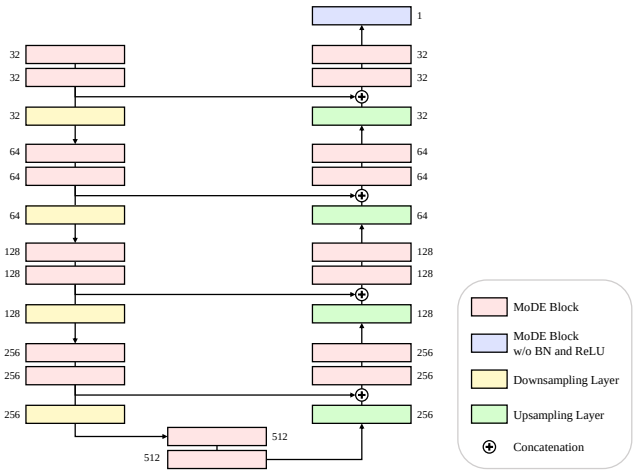

Figure 9. Detailed architecture of the proposed RepMode. The channel number of the output feature maps is shown next to each block. Note that we omit some components (*e.g.* skip connections and the final MoDE block) in Fig. 2 for the sake of brevity.

C. Details of the Experimental Setup

In this section, we provide more details of the experimental setup to highlight the comprehensiveness and reproducibility of our experiments. First, we would provide more descriptions of datasets and implementation details in Appendix C.1 and Appendix C.2 respectively. Then, we would provide mathematical definitions of the evaluation metrics in Appendix C.3. Finally, we would further describe the comparing state-of-the-art methods in Appendix C.4.

C.1. Datasets

In the experiments, we adopt a dataset collection [47] to evaluate the performance of the comparing methods and the proposed RepMode in SSP. The reason why we call it "dataset collection" is because it totally contains twelve partially labeled cell image datasets for SSP. In this dataset collection, each dataset contains 54 to 80 high-resolution 3D z-stack image pairs, where each bright-field input is associated with a fluorescent label (as we defined in Sec. 3.1). We consolidate these datasets into one single partially labeled dataset to conduct our experiments. Totally, there are 628 (resp. 70, 233) image pairs for training (resp. validation, test). With a patch-based training scheme, a dataset of this size is sufficient for such a 3D dense prediction task, which is also recognized by [62, 70].

C.2. Implementation details

All experiments are accomplished with PyTorch 1.12.1 and CUDA 11.6, and run on a single NVIDIA V100 GPU with 32GB memory. For a fair comparison, all random seeds are fixed at 0 in each experiment. Moreover, automatic mixed precision (AMP) is used to accelerate train-

ing. Due to variable image sizes and memory limitations, we adopt a patch-based training scheme in the experiments. Accordingly, in the validation and test phase, we utilize the Gaussian sliding window strategy [30] to aggregate patch-based predictions output by the network to obtain the final predictions of full images. Specifically, we implement the Gaussian sliding window strategy exactly following [70] and the window size is set to the same size of training patches (*i.e.* $32 \times 128 \times 128$).

C.3. Evaluation metrics

The evaluation metrics that we adopted in the experiments include MSE, MAE, and R^2. Following the notations described in Sec. 3.1, let $\mathbf{y}_n$ and $\mathbf{f}_n$ denote the ground-truth label and the output prediction of n-th image pairs respectively. Furthermore, let y_{ni} and f_{ni} indicate the i-th pixel intensity of $\mathbf{y}_n$ and $\mathbf{f}_n$ respectively. These evaluation metrics can be formulated as

$$\text{MSE}(\mathbf{y}_n, \mathbf{f}_n) = \frac{1}{P_n} \sum_{i=1}^{P_n} (y_{ni} - f_{ni})^2, \tag{11}$$

$$\text{MAE}(\mathbf{y}_n, \mathbf{f}_n) = \frac{1}{P_n} \sum_{i=1}^{P_n} |y_{ni} - f_{ni}|, \tag{12}$$

$$R^2(\mathbf{y}_n, \mathbf{f}_n) = 1 - \frac{\sum_{i=1}^{P_n} (y_{ni} - f_{ni})^2}{\sum_{i=1}^{P_n} (y_{ni} - \bar{y}_n)^2}, \tag{13}$$

where P_n is the total pixel number of n-th image pairs and $\bar{y}_n$ is the average of y_{ni}. We adopt MSE and MAE since they are two commonly used evaluation metrics for regression. In addition to these two metrics, R^2 is also be used in our experiments for two following reasons: 1) Compared to MSE and MAE, R^2 further takes into account the variance of the pixel intensity of a ground-truth label (see Eq. (13)); 2) MSE and MAE have arbitrary ranges, while R^2 normally ranges from 0 to 1 and thus is a more intuitive measure.

With these metrics, we report the performance on twelve datasets and present the overall performance by averaging the metrics over all image pairs in Tab. 1. For a clear comparison, we also report the relative overall performance improvement over Multi-Net which is the most naive baseline. Let m_i and m_i' denote the overall results of a random method and Multi-Net on the i-th metric. The relative overall performance improvement of this method over Multi-Net on the i-th metric can be calculated as

$$\Delta_{\text{Imp}}(m_i, m_i') = (-1)^{v_i} \frac{m_i - m_i'}{m_i'}, \tag{14}$$

where $v_i = 1$ if a lower value means better performance for the i-th metric, and 0 otherwise. With such an informative measure, the performance differences in the experiments can be clearly presented (see Tab. 1).

C.4. Comparing methods

In Sec. 4.2, we have briefly introduced the comparing state-of-the-art methods of the experiments. Here we provide detailed descriptions of these methods: 1) Multi-Net [47]: multiple individual networks, each of which aims to handle one single-label prediction task; 2) Multi-Head: a partially-shared network composed of a shared feature extractor and multiple task-specific heads, including two variants, *i.e.* multiple task-specific decoders (denoted by Dec.) or last layers (denoted by Las.); 3) Conditional Network (CondNet) [15]: a task-conditional network where the task-aware prior is encoded as feature maps by a predefined hash function; 4) Task Switching Networks (TSNs) [55]: a task-conditional network that uses a fully connected module to learn the task embedding for adaptive instance normalization; 5) Pyramid Input Pyramid Output Feature Abstraction Network (PIPO-FAN) [16]: a network that consists of a U-shape pyramid architecture with multi-resolution images as input, and a deep supervision mechanism to refine the output in different scales; 6) Dynamic On-Demand Network (DoDNet) [70]: a task-conditional network composed of a shared encoder-decoder architecture, a controller for filter generation, and a dynamic convolutional head (*i.e.* three convolutional layers); 7) Task-Guided Network (TGNet) [62]: an improved version of DoDNet, where task-guided residual blocks and attention modules are further introduced to emphasize the features related to the specified task. Notably, we have equipped these networks with the same backbone of RepMode to ensure fairness.

D. Additional Analysis and Discussion

D.1. Task-incremental learning

We have conducted the corresponding experiments in Sec. 4.4 to verify that the proposed RepMode can serve as a better task-incremental learner. Here we detail the experimental setup and provide additional analysis.

Experimental setup. We select the mainstream solutions of SSP, *i.e.* Multi-Net and Multi-Head, for a comparison. For Multi-Head, we select its "Dec." variant since it contains more task-specific parameters. First, all these networks are pretrained on eleven datasets. Then, the pretrained networks are extended to a new task by being trained on the remaining dataset. Note that the training of these two phases also follows the implementation details that we describe in Sec. 4.1 and Appendix C.2. Specifically, the strategies of these networks for task-incremental learning are: 1) Multi-Net: employ a new network to be trained on the new dataset from scratch; 2) Multi-Head (Dec.): add a new decoder to handle the new dataset and fine-tunes the whole network; 3) RepMode: introduce an extra expert (here we choose a Conv $3 \times 3 \times 3$ expert) and a new gating module in each MoDE block, and only fine-tune the

Blocks	MSE	MAE	R^2
ACNet Block [12]	.5075	.4197	.4611
RepVGG Block [14]	.5034	.4122	.4654
DBB [13]	.5023	.4102	.4667
MoDE Block	**.4956**	**.4078**	**.4735**

Table 4. Comparison with other SOTA re-param blocks in SSP. Note that we modify these blocks to adapt to our GatRep.

newly-introduced components with the other ones frozen. We adopt the datasets of two basic subcellular structures, *i.e*. nucleolus and cell membrane, for the experiments of task-incremental learning.

Results and analysis. As shown in Tab. 3, the proposed RepMode can achieve superior performance in task-incremental learning. The main reason is that the experts of RepMode are trained in a task-agnostic manner and thus capable of learning the generalized domain knowledge of SSP. When trained on a new dataset, RepMode can utilize the pretrained experts to "transfer" such knowledge to the new task. With this strategy, RepMode can easily adapt to a new task of an unseen subcellular structure, rather than learning it from scratch. Moreover, as long as the previous gating weights have been stored, the fine-tuned RepMode can maintain the original performance on the previous tasks since the parameters of the frozen experts are fixed and preserved. Whereas, Multi-Net requires training a new network and thus achieve poor performance in task-incremental learning. Besides, Multi-Head needs to fine-tune the whole network, which would result in an inevitable performance drop on the previous tasks.

D.2. Comparison with other re-param blocks

The performance of the proposed MoDE block is already verified in Sec. 4.3. In this subsection, we further compare it with the existing SOTA re-param blocks [12–14] in SSP. Below we would detail the experimental setup and conduct the corresponding analysis.

Experimental setup. We select the following state-of-the-art re-param blocks and modify them to a 3D convolution version: 1) Asymmetric convolution network (ACNet) block [12]: consist of a Conv $3 \times 3 \times 3$, a Conv $3 \times 1 \times 3$, and a Conv $3 \times 3 \times 1$; 2) RepVGG block [14]: contains a Conv $3 \times 3 \times 3$, a Conv $1 \times 1 \times 1$, and a residual connection (since the channel numbers of the input and output feature maps may be different, we replace it with an additional Conv $1 \times 1 \times 1$ aiming to align the channel numbers); 3) Diverse branch block (DBB) [13]: consists of a Conv $1 \times 1 \times 1$, a Conv $1 \times 1 \times 1$ - Conv $K \times K \times K$, a Conv $1 \times 1 \times 1$ - Avgp $K \times K \times K$, and a Conv $K \times K \times K$ (here we set $K = 3, 5$ and report the best result). Moreover, in order to adapt to our GatRep for a fair comparison, all BN inside the branches are removed to ensure linearity.

Methods	Time (s)		GPU Memory (%)
	Training	Validation	
RepMode w/o GatRep	135.87	1359.41	95.04
RepMode w/ GatRep	**80.13**	**526.59**	**59.07**

Table 5. Statistics of time and memory consumption. Note that "Time" indicates the average time of a training epoch or a validation in a complete training phase, and "GPU Memory" indicates the maximum percentage of allocated GPU memory during training. These results are acquired based on an NVIDIA V100 GPU with 32GB memory.

We replace MoDE blocks with these blocks in RepMode, and follow the implementation details that we describe in Sec. 4.1 and Appendix C.2 to evaluate their performance.

Results and analysis. As we can observe in Tab. 4, the proposed RepMode can still achieve competitive performance when equipped with different re-param blocks, which reveals its applicability. Furthermore, compared to the other re-param blocks, our MoDE block can achieve better performance in SSP. This is because the MoDE block is composed of the experts with diverse configurations. Such an efficient and flexible convolution collocation works well with a task-conditioning strategy and is capable of handling more generalized situations, which is also demonstrated by the ablation studies in Sec. 4.3.

D.3. Cost reducing of GatRep

In Sec. 3.4, we have that claimed GatRep is an efficient expert utilization manner for the MoDE block. Specifically, compared to completely utilizing all experts to process the input feature maps (see Fig. 4(a)), GatRep can significantly reduce the computational and memory costs caused by the multi-branch topology of MoE. In this subsection, we provide some empirical evidence to demonstrate this benefit of GatRep. As we can observe in Tab. 5, GatRep can save 41.02% and 61.26% time in a training epoch and a validation respectively. This is because only one convolution operation is required in a MoDE block when using GatRep. Moreover, GatRep can reduce 37.85% peak GPU memory utilization, since the output feature maps of all experts are no longer separately calculated and stored. Using GatRep, our RepMode can acquire cost-economic performance improvement and the ability to handle multiple tasks in an all-shared network. As a result, RepMode can maintain a compact practical topology exactly like a plain network, and meanwhile achieves a powerful theoretical topology. Such a technique can increase the device-friendliness of RepMode in the practical scenarios of biological research.

D.4. Experimental results of multiple runs

To further verify the effectiveness of the proposed RepMode, we perform "four-fold cross-test" and report the av-

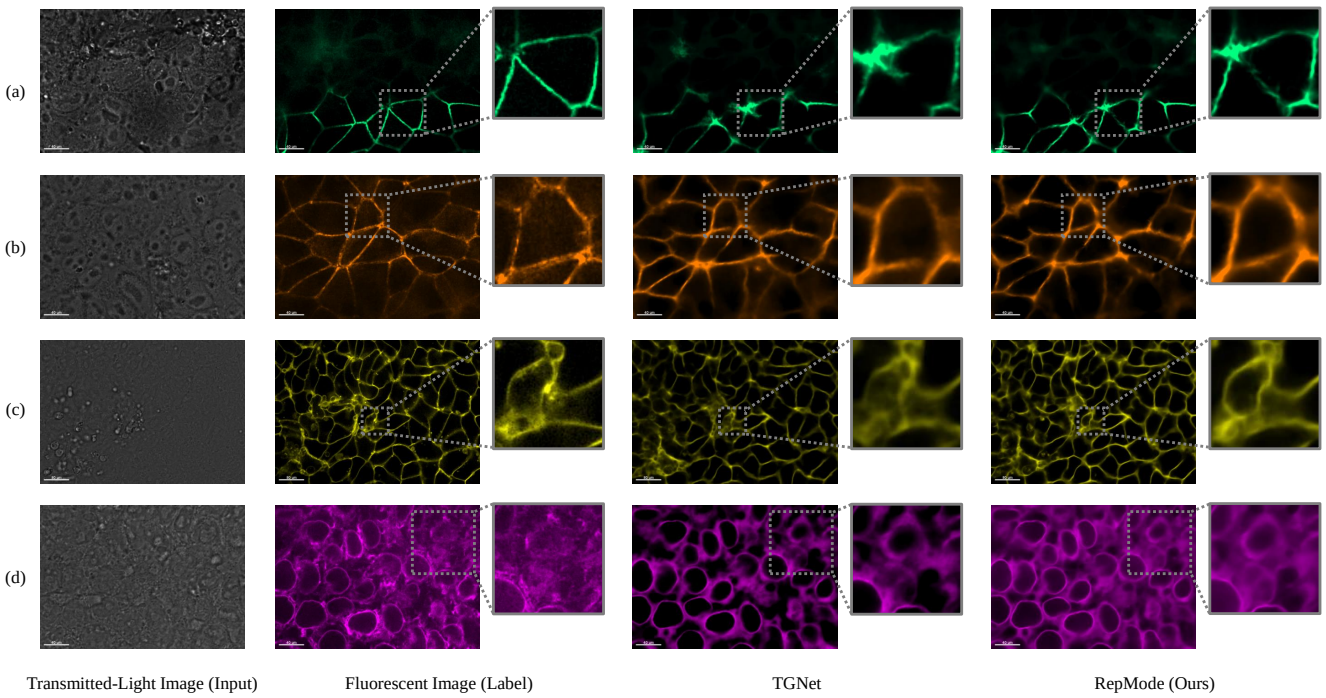

Figure 10. Examples of the prediction results on the test set, including (a) tight junction, (b) actomyosin bundle, (c) cell membrane, and (d) endoplasmic reticulum. We compare the predictions of our RepMode with the ones of TGNet [62] which is a competitive method. Note that the dotted boxes indicate the major prediction difference.

Methods	MSE	MAE	R^2
Multi-Head (Dec.)	.5204	.4247	.4466
TSNs [55]	.5134	.4202	.4538
TGNet [62]	.5123	.4186	.4549
RepMode	**.5032**	**.4124**	**.4642**

Table 6. Experimental results of "four-fold cross-test". Note that we present the average results of multiple runs.

erage results of multiple runs. Specifically, following the ratio of 25%, we divide the dataset into four parts and then select each part in turn as the test set to conduct the experiments. We compare our RepMode with a Multi-Head variant (*i.e.* Multi-Head (Dec.)) and two competitive methods (*i.e.* TSNs [55] and TGNet [62]). The experimental results show that our RepMode remains superior (see Tab. 6).

E. More Qualitative Examples

In this section, we provide more qualitative examples as an extension to Fig. 7. It is worth noting that all images, including transmitted-light images, fluorescent images, and prediction results, are visualized by Imaris 9.0.1 with identical rendering configurations respectively for a fair comparison. Moreover, all examples are randomly selected from the test set, and a random z-axis slice is presented for each example. As shown in Fig. 10, Our RepMode can produce relatively precise predictions even for those hard cases (*e.g.* Fig. 10(c)&(d)), which demonstrates the remarkable effectiveness of RepMode in SSP.