A Impact Statement

In an era dominated by over-parameterized models, designing resource-aware AI models is becoming increasingly important, especially for time-consuming tasks like medical segmentation. Our insights into model efficiency training have the potential to broaden the application of deep neural networks in this area. Overall, this work advances our fundamental understanding of dynamic sparse training and offers future perspectives for scalable and efficient AI models. We do not anticipate any negative societal impacts resulting from this research.

B Appendix.

B.1 Related Work

B.1.1 3D Medical Image Segmentation

Convolutional neural networks (CNNs) have become the dominant architecture for 3D medical image segmentation in recent years (e.g. 3D UNet [Çiçek et al., 2016], UNet++ [Zhou et al., 2018], UNet3+ [Huang et al., 2020], PaNN [Zhou et al., 2019] and nnUNet [Isensee et al., 2021]), due to their ability to capture local and weight-sharing dependencies [d'Ascoli et al., 2021, Dai et al., 2021]. However, some recent methods have attempted to incorporate transformer modules into CNNs (e.g. CoTr [Xie et al., 2021], TransBTS [Wang et al., 2021]), or use pure transformer architectures (e.g. ConvIt [Karimi et al., 2021], nnFormer [Zhou et al., 2021], Swin UNet [Cao et al., 2021]), in order to capture long-range dependencies. These transformer-based approaches often require large amounts of training data, longer training times, or specialized training techniques, and can also be computationally expensive. Most recently, a novel architecture called Mamba Gu and Dao [2023] has shown potential for computational efficiency as a State Space model in handling long sequences and has been applied to medical image segmentation tasks Ruan and Xiang [2024], Xing et al. [2024], Wang et al. [2024]. However, it has led to underwhelming performance compared to state-of-the-art convolutional models. In this paper, we propose an alternative method for efficiently incorporating 3D contextual information using a restricted depth-shift strategy in 3D convolutions, and further improving performance through adaptive multi-scale feature fusion.

B.1.2 Feature Fusion in Medical Image Segmentation

Multi-scale feature fusion is a crucial technique in medical image segmentation that allows a model to detect objects across a range of scales, while also recovering spatial information that is lost during pooling [Wang et al., 2022, Xie et al., 2021]. However, effectively representing and processing multi-scale hierarchy features can be challenging, and simply summing them up without distinction can lead to semantic gaps and degraded performance [Wang et al., 2022, Tan et al., 2020]. To address this issue, various approaches have been proposed, including adding learnable operations to reduce the gap with residuals [Ibtehaz and Rahman, 2020], attention blocks [Oktay et al., 2018]. More recently, UNet++ [Zhou et al., 2018] and its variants [Li et al., 2020, Huang et al., 2020, Jha et al., 2019] have adapted the gating signal to dense nesting levels, taking into account as many feature levels as possible. NAS-UNet [Weng et al., 2019] tries to automatically search for better feature fusion topology. While these methods have achieved better performance, they can also incur significant computational and information redundancy. Dynamic convolution [Su et al., 2020, Chen et al., 2020] utilizes coefficient prediction or attention modules to dynamically aggregate convolution kernels, thereby reducing computation costs. In our paper, we propose an intuitive approach to optimizing multi-scale feature fusion, which enables selective leveraging of **sparse** feature representations from fine-grained to semantic levels through the proposed dynamic sparse feature fusion mechanism.

B.1.3 Sparse Training

Recently, sparse training techniques have shown the possibility of training an efficient network with sparse connections that match (or even outperform) the performance of dense counterparts with lower computational cost [Mocanu et al., 2018, Liu et al., 2021b]. Beginning with [Mocanu et al., 2016], it has been demonstrated that initializing a static sparse network without optimizing its topology during training can also yield comparable performance in certain situations [Lee et al., 2018, Tanaka et al., 2020, Wang et al., 2019]. However, Dynamic Sparse Training (DST), also known as sparse training

20

Algorithm 1 The Training Process of Dynamic Sparse Feature Fusion (DSFF)

Require: Dataset $\mathcal{X}$ with label $\mathcal{Y}$; feature sparsity S; backbone $f_\Theta(.)$; Output Module: f_{out}; Total training epochs: T;

   evolution period: ΔT; connection updating number: $f_{decay}\left(\Delta T; \alpha, T\right) = \frac{\alpha}{2}\left(1 + \cos\left(\frac{\Delta T \pi}{T}\right)\right)$, α represents the number of updated connections during the initial topology update, which is set to $1/2$; Loss function: $\mathcal{L}(.)$; fusion operation: $\mathcal{F}^{j,i}(\cdot)$ with convolution kernels $\theta^{j,i}$, where the numbers of input and output channel are $C_{in}^{j,i}$, $C_{out}^{j,i}$.

1: $\mathbf{M}^{j,i} \leftarrow$ random initialize masks for all levels and stages, satisfying that $\|\mathbf{M}^{j,i}\|_0$ equals $(1 - S) \times C_{in}^{j,i} \times C_{out}^{j,i}$

2: **for** $t = 1$ **to** T **do**

3:     Sample a batch $I_t, Y_t \sim \mathcal{X}, \mathcal{Y}$

4:     Generate multi-scaled features: $\left(\mathbf{x}^{0,1}, \mathbf{x}^{0,2}, \ldots, \mathbf{x}^{0,L}\right) = f_\Theta(I_t)$

5:     **for** each stage $j = 1$ **to** $L - 1$ **do**

6:       **for** each level $i = 1$ **to** $L - j$ **do**

7:         **if** $i = 1$ **then**

8:           $\mathbf{x}^{j,i} = \mathcal{F}^{j,i}([\mathbf{x}^{j-1,1}, \mathcal{U}(\mathbf{x}^{j-1,2})])$

9:         **else**

10:          $\mathbf{x}^{j,i} = \mathcal{F}^{j,i}([\mathcal{D}(\mathbf{x}^{j-1,i-1}), \mathbf{x}^{j-1,i}, \mathcal{U}(\mathbf{x}^{j-1,i+1})])$

11:         **end if**

12:       **end for**

13:     **end for**

14:     $l_t = 4/7\mathcal{L}(f_{out}\left(\mathbf{x}^{L-1,1}\right), Y_i) + 2/7\mathcal{L}(f_{out}\left(\mathbf{x}^{L-2,2}\right), \mathcal{D}(Y_i)) + 1/7\mathcal{L}(f_{out}\left(\mathbf{x}^{L-3,3}\right), \mathcal{D}(\mathcal{D}(Y_i)))$

15:     **if** $(t \bmod \Delta T) == 0$ **then**

16:       **for** each stage $j = 1$ **to** $L - 1$ **do**

17:         **for** each level $i = 1$ **to** $L - j$ **do**

18:           $u = (C_{in}^{j,i} \times C_{out}^{j,i}) f_{decay}\left(t; \alpha, T\right)\left(1 - S\right)$

19:           $IS \leftarrow$ importance score ($L1$-norm of corresponding kernel) for activated each feature connection

20:           $\mathbb{I}_{activate} = RandomK(\mathbb{I}_{inactivate}, u)$

21:           $\mathbb{I}_{inactivate} = ArgTopK\left(-IS, u\right)$

22:           $\mathbf{M}^{j,i} \leftarrow$ Update $\mathbf{M}^{j,i}$ using $\mathbb{I}_{inactivate}$ and $\mathbb{I}_{activate}$

23:         **end for**

24:       **end for**

25:     **else**

26:       Training the E2ENet using SGD optimizer

27:     **end if**

28: **end for**

---

with dynamic sparsity [Mocanu et al., 2018], offers a different approach by jointly optimizing the sparse topology and weights during the training process starting from a sparse network [Liu et al., 2021a, 2022, Evci et al., 2020, Jayakumar et al., 2020, Mostafa and Wang, 2019, Yuan et al., 2021]. This allows the model's sparse connections to gradually evolve in a prune-and-grow scheme, leading to improved performance compared to naively training a static sparse network [Liu et al., 2021c, Xiao et al., 2022]. In contrast to prior methods that aim to find sparse networks that can match the performance of corresponding dense networks, we aim to leverage DST to adaptively fuse multi-scale features in a computationally efficient manner for 3D medical image segmentation.

B.2   Algorithm

B.3   Datasets and Experiment Setup

AMOS-CT: The Abdominal Multi-Organ Segmentation Challenge (AMOS) [Ji et al., 2022] task 1 consists of 500 computerized tomography (CT) cases, including 200 scans for training, 100 for validation, and 200 for testing. These cases have been collected from a diverse patient population and

21

include annotations of 15 organs. The scans are from multiple centers, vendors, modalities, phases, and diseases.

BTCV: The Beyond the Cranial Vault (BTCV) abdomen challenge dataset [4] consists of 30 CT scan images for training and 20 for testing. These images have been annotated by interpreters under the supervision of radiologists, and include labels for 13 organs.

BraTS: The Brain Tumor Segmentation Challenge in the Medical Segmentation Decathlon (MSD) [Antonelli et al., 2022, Simpson et al., 2019] consists of 484 MRI images from 19 different institutions. These images contain three different tumor regions of interest (ROIs): edema (ED), non-enhancing tumor (NET) and enhancing tumor (ET). The goal of the challenge is to segment these ROIs in the images accurately.

B.4 Implementation Details

In our work, we utilized the PyTorch toolkit [Paszke et al., 2019] on an NVIDIA A100 GPU for all our experimental evaluations. We also used the nnUNet codebase [Isensee et al., 2021] to pre-process data before training our proposed E2ENet model. For the AMOS dataset, we used the nnUNet codebase as the benchmark implementation.

For training, we use the stochastic gradient descent (SGD) optimizer with an initial learning rate of 0.01, which is gradually decreased through a "poly" decay schedule. The optimizer is configured with a momentum of 0.99 and a weight decay of 3×10^{-5}. The maximum number of training epochs is 1000, with 250 iterations per epoch. For the loss function, we combine both cross-entropy loss and Dice loss as in [Isensee et al., 2021]. To improve performance, various data augmentation techniques such as random rotation, scaling, flipping, adding Gaussian noise, blurring, adjusting brightness and contrast, simulating low resolution, and Gamma transformation are used before training.

We employ a 5-fold cross-validation strategy on the training set for all experiments, selecting the final model from each fold and simply averaging their outputs for the final segmentation predictions. In the testing stage, we employ the sliding window strategy, where the window sizes are equal to the size of the training patches. Additionally, post-processing methods outlined in [Isensee et al., 2022] are applied for the AMOS-CT dataset during the testing phase.

B.5 The Architecture

The backbone generates a total of $L = 6$ multi-scale feature levels, each with a specified number of channels: [c1, c2, c3, c4, c5, c6] = [48, 96, 192, 320, 320, 320]. At each level of feature generation, there are two convolution layers with a kernel size of (1, 3, 3), followed by instance normalization and the application of leaky ReLU activation. The down-sampling ratios for each level are as follows: ((1, 2, 2), (2, 2, 2), (2, 2, 2), (2, 2, 2), (2, 2, 2), (2, 2, 2), (2, 2, 2)).

B.6 Evaluation Metrics

B.6.1 Mean Dice Similarity Coefficient

To assess the quality of the segmentation results, we use the mean Dice similarity coefficient (mDice), which is a widely used metric in medical image segmentation. The mDice is calculated as follows:

$$mDice = \frac{1}{N} \sum_{j=1}^{N} \frac{2|\mathbf{y}_j \cdot \hat{\mathbf{y}}_j|}{(|\mathbf{y}_j| + |\hat{\mathbf{y}}_j|)}, \tag{3}$$

where N is the number of classes, $\cdot$ is the pointwise multiplication, $\mathbf{y}_j$ and $\hat{\mathbf{y}}_j$ represent the ground truth and predicted masks of the j-th class, respectively, which are encoded in one-hot format. $\frac{2|\mathbf{y}_j \cdot \hat{\mathbf{y}}_j|}{(|\mathbf{y}_j| + |\hat{\mathbf{y}}_j|)}$ is the Dice of j-th class, which measures the overlap between the predicted and ground truth segmentation masks for that class.

---

[4]https://www.synapse.org/#!Synapse:syn3193805/wiki/89480

22

B.6.2 Number of Parameters

The size of the network can be estimated by summing the number of non-zero parameters (Params), which includes the parameters of activated sparse feature connections (kernels) and parameters of the backbone. The calculation is given by the following equation:

$$Params = \|\Theta\|_0 + \sum_{j=1}^{L-1} \sum_{i=1}^{L-j} \sum_{c_{in}=1}^{C_{in}^{j,i}} \sum_{c_{out}=1}^{C_{out}^{j,i}} \mathbf{M}_{c_{in},c_{out}}^{j,i} \|\theta_{c_{in},c_{out}}^{j,i}\|_0. \tag{4}$$

Here, Θ is the parameter from backbone, L is the total number of feature levels, $\mathbf{M}^{j,i}$ is a matrix of size $C_{in}^{j,i} \times C_{out}^{j,i}$, and $\mathbf{M}_{c_{in},c_{out}}^{j,i}$ indicates whether the kernel $\theta_{c_{in},c_{out}}^{j,i}$ connecting the c_{in}-th input and c_{out}-th output feature map exist or not. The L_0 norm $\|\theta_{c_{in},c_{out}}^{j,i}\|_0$ provides the number of non-zero entries of $\theta_{c_{in},c_{out}}^{j,i}$.

B.6.3 Float Point Operations

Floating point operations (FLOPs) is a commonly used metric to compare the computational cost of a sparse model to that of a dense counterpart [Hoefler et al., 2021] [5]. In our comparison, it is calculated by counting the number of multiplications and additions performed in only one forward pass of the inference process without considering postprocessing. The inference FLOPs are estimated layer by layer and depend on the sparsity level of the network. For each convolution or transposed convolution layer, the inference FLOPs is calculated as follows:

$$FLOPs_{conv} = (2K_d K_h K_w C_{in}(1 - S) + 1) \times C_{out} HWD, \tag{5}$$

where K_d, K_h and K_w are the kernel sizes in depth, height and width; S is the feature sparsity level, for layers that are not part of the DSFF mechanism, $S = 0$ is used; C_{in} and C_{out} are the numbers of input feature and output feature; H, W and D are the height, width and depth of output features. For each fully connected layer, the inference FLOPs is calculated as follows:

$$FLOPs_{fc} = (2C_{in}(1 - S) + 1) \times C_{out}. \tag{6}$$

B.6.4 Performance Trade-Off Score

The accuracy-efficiency trade-offs could be further analyzed, from comparing resource requirements to describing holistic behaviours (including mDice, Params and inference FLOPs) for the 3D image segmentation methods. To quantify these trade-offs, we introduce the Performance Trade-Off (PT) score, which is defined as follows:

$$PT = \alpha_1 \frac{mDice}{mDice_{max}} + \alpha_2 (\frac{Params_{min}}{Params} + \frac{FLOPs_{min}}{FLOPs}), \tag{7}$$

where α_1 and α_2 are weighting factors, which control the trade-off between accuracy performance and resource requirements, and $mDice_{max}$, $Params_{min}$, and $FLOPs_{min}$ denote the highest mDice score, the smallest number of parameters, and the lowest inference FLOPs among the compared methods for a specific dataset, respectively. The term $\frac{mDice}{mDice_{max}}$ measures the segmentation accuracy, while $\frac{Params_{min}}{Params} + \frac{FLOPs_{min}}{FLOPs}$ measures the resource cost.

In most cases, we consider both segmentation accuracy and resource cost to be equally important, thus we set $\alpha_1 = 1$ and $\alpha_2 = 1/2$ in the following experiments. However, we also explore the impact of different choices of α_1 and α_2, as detailed in Section **??**. The PT score serves as a valuable metric for evaluating the trade-offs between segmentation accuracy and efficiency.

23

Table 9: Quantitative comparisons (class-wise Dice (%) ↑, mDice(%)↑, Params(M)↓, inference FLOPs(G)↓, PT score↑ and mNSD(%)↑) of segmentation performance on the validation set of AMOS-CT dataset. **Bold** indicates the best and underline indicates the second best. Note: Spl: spleen, RKid: right kidney, LKid: left kidney, Gall: gallbladder, Eso: esophagus, Liv: liver, Sto: stomach, Aor: aorta IVC: inferior vena cava, Pan: pancreas, RAG: right adrenal gland, LAG: left adrenal gland, Duo: duodenum, Bla: bladder, Pro/Uth: prostate/uterus. The class-wise Dice, mDice and mNSD results of baselines, except for nnUNet, are collected from the [Ji et al., 2022]. [†] indicates the results without postprocessing that are collected from the AMOS website. [‡] denotes the results with postprocessing that are reproduced by us. [*] indicates the results with postprocessing.

Methods	Spl	RKid	LKid	Gall	Eso	Liv	Sto	Aor	IVC	Pan	RAG	LAG	Duo	Bla	Pro/Uth	mDice	Params	FLOPs [3]	PT score	mNSD
CoTr	91.1	87.2	86.4	60.5	80.9	91.6	80.1	93.7	87.7	76.3	73.7	71.7	68.0	67.4	40.8	77.1	41.87	1510.53	1.07	64.2
nnFormer	95.9	93.5	94.8	78.5	81.1	95.9	89.4	94.2	88.2	85.0	75.0	75.9	78.5	83.9	74.6	85.6	150.14	1343.65	1.12	74.2
UNETR	92.7	88.5	90.6	66.5	73.3	94.1	78.7	91.4	84.0	74.5	68.2	65.3	62.4	77.4	67.5	78.3	93.02	391.03	1.41	61.5
Swin UNETR	95.5	93.8	94.5	77.3	83.0	96.0	88.9	94.7	89.6	84.9	77.2	78.3	78.6	85.8	77.4	86.4	62.83	1562.99	1.14	75.3
VNet	94.2	91.9	92.7	70.2	79.0	94.7	84.8	93.0	87.4	80.5	72.6	73.2	71.7	77.0	66.6	82.0	45.65	1737.57	1.10	67.9
nnUNet[†]	97.1	96.4	96.2	83.2	87.5	97.6	92.2	96.0	92.5	88.6	81.2	81.7	85.0	90.5	85.0	90.0	30.76	1067.89	1.30	82.1
nnUNet[‡]	97.1	97.0	97.1	86.6	87.7	97.9	92.4	96.0	92.7	88.8	81.6	82.1	85.0	90.6	85.2	90.5	30.76	1067.89	1.31	83.0
E2ENet[*] (s=0.7)	97.1	96.9	97.1	86.0	87.6	97.9	92.3	95.7	92.3	89.0	81.5	82.4	84.9	90.3	83.8	90.3	11.23	969.32	1.54	82.7
E2ENet[*] (s=0.8)	97.1	96.9	97.0	85.2	87.5	97.9	92.3	95.7	92.3	89.0	81.3	82.1	84.6	90.1	84.8	90.3	9.44	778.74	1.65	82.5
E2ENet[*] (s=0.9)	96.7	96.9	97.0	84.2	87.0	97.7	92.2	95.6	92.0	88.6	81.0	81.8	84.0	89.9	83.8	89.9	7.64	492.29	1.89	81.8
E2ENet (s=0.7)	97.1	96.6	96.5	83.4	87.6	97.5	92.3	95.8	92.3	89.0	81.4	82.3	84.9	90.3	83.8	90.1	11.23	969.32	1.54	82.3
E2ENet (s=0.8)	97.1	96.6	96.5	83.4	87.5	97.5	92.3	95.8	92.3	89.0	81.3	82.0	84.5	90.1	84.8	90.0	9.44	778.74	1.65	82.3
E2ENet (s=0.9)	96.7	95.4	96.4	82.6	86.9	97.4	92.2	95.6	92.0	88.6	80.9	81.7	84.0	89.9	83.8	89.6	7.64	492.29	1.88	81.4
E2ENet(static, s=0.9)	96.6	95.5	96.3	82.6	86.9	97.4	92.2	95.6	92.0	88.6	80.9	81.7	84.0	89.9	83.8	89.6	7.64	492.29	1.88	81.4

[3] The inference FLOPs are calculated based on the patch sizes of $1 \times 128 \times 128 \times 128$ without considering postprocessing cost.

B.7 More Experimental Results

B.7.1 Class-wise Dice of AMOS-CT

B.7.2 BTCV Challenge

We compare the performance of our E2ENet model to several baselines (CoTr [Xie et al., 2021], RandomPatch [Tang et al., 2021], PaNN [Zhou et al., 2019], UNETR [Hatamizadeh et al., 2022], and nnUNet [Isensee et al., 2021]) on the test set of BTCV challenge, and report class-wise Dice, mDice, Params and inference FLOPs on the test set in Table 10. It is worth noting that nnUNet is a strong performer that uses an automatic model configuration strategy to select and ensemble two best of multiple U-Net models (2D, 3D and 3D cascade) based on cross-validation results. In contrast, E2ENet is designed to be computationally and memory efficient, using a consistent 3D network configuration. Swin UNETR [Tang et al., 2022] is among the best on the leaderboard for this challenge. However, we do not include it in our comparison because it employs self-supervised learning with extra data. This falls outside of our goal of trading off training efficiency and accuracy without using extra data.

Our proposed E2ENet, a single 3D architecture without cascade, has achieved comparable performance to nnUNet, with mDice of 88.3%. Additionally, it has a significantly smaller number of parameters, 11.25 M, compared to other methods such as nnUNet (30.76 M), CoTr (41.87 M), and UNETR (92.78 M).

B.7.3 Statistical significance of designed modules

To demonstrate the advantages of individual modules, we plot a critical distance diagram using the Nemenyi post-hoc test with a p-value of 0.05 to establish the statistical significance of our modules. In Figure 7, the top line represents the axis along which the methods' average ranks, and a lower value indicates better performance. Methods joined by thick horizontal

Figure 7: The critical distance diagram on the AMOS-CT validation dataset, with the evaluation metric being mDice.

black lines are considered not statistically different. From the diagram, we can clearly observe that E2ENet with depth shift significantly outperforms E2ENet without depth shift. Additionally, the incorporation of dynamic sparse feature fusion into E2ENet results in a substantial reduction in both

---

[5]This is because current sparse training methods often use masks on dense weights to stimulate sparsity. This is done because most deep learning hardware is optimized for dense matrix operations. As a result, using these prototypes doesn't accurately reflect the true memory and speed benefits of a truly sparse network [Hoefler et al., 2021].

24

Table 10: Quantitative comparisons of segmentation performance on BTCV test set. Note: Spl: spleen, RKid: right kidney, LKid: left kidney, Gall: gallbladder, Eso: esophagus, Liv: liver, Sto: stomach, Aor: aorta IVC: inferior vena cava, Veins: portal and splenic veins, Pan: pancreas, AG: adrenal gland. The results (class-wise Dice and mDice) for these baselines are from [Hatamizadeh et al., 2022]. $^{+}$ denotes that the training of UNETR^{+} is without using any extra data outside the challenge. The results of nnUNet‡, E2ENet and Hausdorff Distance (HD)↓ of UNETR are from the standard leaderboard of BTCV challenge, while the results of nnUNet are from the free leaderboard.

Methods	Spl	RKid	LKid	Gall	Eso	Liv	Sto	Aor	IVC	Veins	Pan	AG	mDice	Params	FLOPs [1]	PT score	HD
CoTr	95.8	92.1	93.6	70.0	76.4	96.3	85.4	92.0	83.8	78.7	77.5	69.4	84.4	41.87	636.94	1.22	/
RandomPatch	96.3	91.2	92.1	74.9	76.0	96.2	87.0	88.9	84.6	78.6	76.2	71.2	84.4	/	/	/	/
PaNN	96.6	**92.7**	95.2	73.2	79.1	97.3	89.1	91.4	85.0	80.5	80.2	65.2	85.4	/	/	/	/
UNETR^{+}	96.8	92.4	94.1	75.0	76.6	97.1	91.3	89.0	84.7	78.8	76.7	74.1	85.6	92.79	**164.91**	1.53	23.4
nnUNet	**97.2**	91.8	**95.8**	75.3	84.1	**97.7**	92.2	**92.9**	**88.1**	83.2	**85.2**	77.8	**88.4**	31.18	416.73	1.38	15.6
nnUNet‡	96.5	91.7	**95.8**	78.5	84.2	97.4	91.5	92.3	86.9	83.1	84.9	77.5	88.0	31.18	416.73	1.38	16.9
E2ENet ($s = 0.7$)	96.5	91.3	95.7	78.1	**84.5**	97.5	91.5	92.2	86.7	**83.4**	84.8	77.9	88.3	**11.25**	449.00	**1.68**	16.1

[1] The inference FLOPs are calculated based on the patch sizes of $1 \times 96 \times 96 \times 96$. The codes for RandomPatch and PaNN are not publicly available, so it is not possible for us to determine their model size and inference FLOPs.

the number of FLOPs (from 23.90M to 11.23M) and parameters (from 3069.55G to 969.32G) while maintaining comparable performance, without any significant performance degradation.

B.7.4 Qualitative Results

BTCV Challenge In Figure 8 (b), we present a qualitative comparison of our proposed E2ENet method with nnUNet as a baseline model on the BTCV challenge. Our results demonstrate the effectiveness of our proposed method in addressing some of the challenges of medical image segmentation. For example, as shown in the first and third columns, our E2ENet method accurately distinguishes the stomach from the background without over- or under-segmentation, which can be difficult due to the low contrast in the image. In the second column, E2ENet performs well in differentiating the stomach from the spleen. These examples suggest that our DSFF module can effectively encode feature information for improved performance in medical image segmentation.
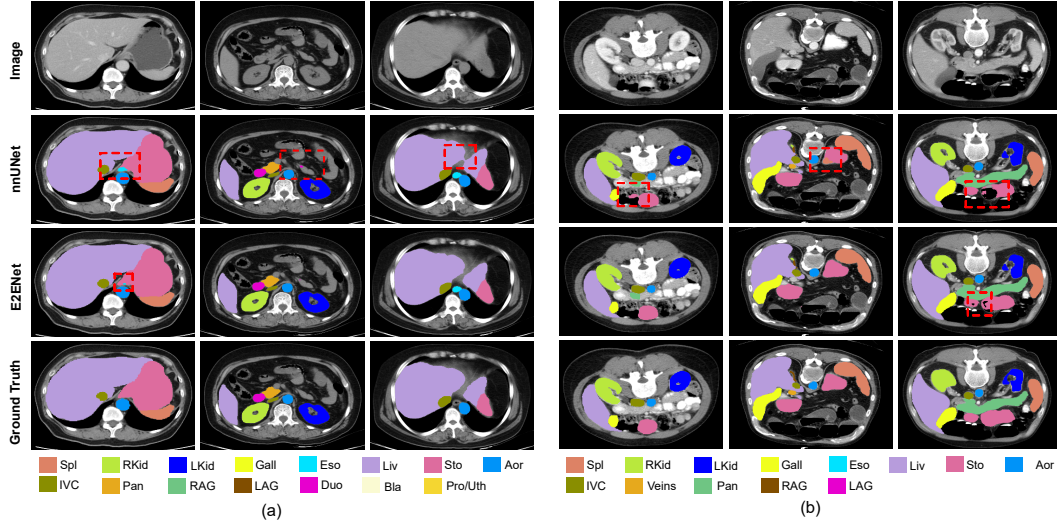
Figure 8: Qualitative comparison of the proposed E2ENet and nnUNet on AMOS-CT and BTCV challenges.

BraTS Challenge in MSD Figure 9 presents a qualitative comparison of our proposed E2ENet method with the nnUNet on the BraTS challenge with highly variable shapes of the segmentation targets. Based on the results of the baseline model, nnUNet, we observed that accurately distinguishing the edema (ED) from the background is difficult, as the edema tends to have less smooth boundaries. Our results suggest that E2ENet may have some potential to improve the distinguishability of the edema boundaries, as evidenced by the relatively better segmentation results in the first, second, and fourth columns. Moreover, E2ENet accurately differentiates the enhanced tumor (ET) from the

25
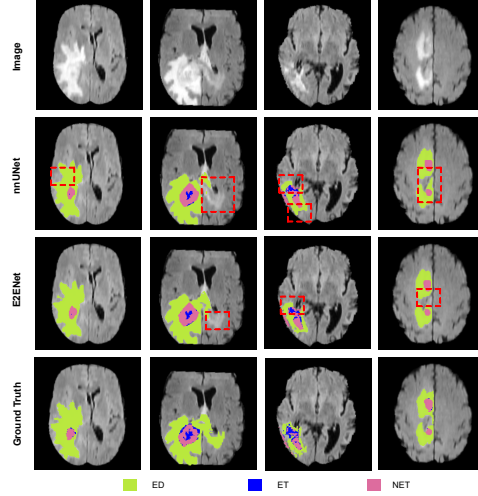
Figure 9: Qualitative comparison of the proposed E2ENet and nnUNet on BraTS Challenge in MSD.

edema, as shown in the third column, which is a challenging task due to the similarity in appearance between these two regions, and the dispersive distribution of ET. These findings suggest that E2ENet is a promising method for accurately segmenting brain tumors in challenging scenarios.

B.8 Convergence Analysis

In this section, we analyze the convergence behavior of E2ENet by examining the loss changes during topology updating (kernel activation/deactivation epochs), comparing it with the best-performing baseline nnUNet, and studying the impact of topology update frequency. From Figure 10, we observed that the activation/deactivation of weights initially led to an increase in training loss. However, over the long term, the training converged. Additionally, we compared the learning curve of E2ENet with that of nnUNet and found that E2ENet converged even faster than nnUNet, as shown in the subplot in Figure 11 (a). To account for the effect of the number of parameters, we scaled down nnUNet to have a similar number of parameters as E2ENet and observed that it converged even more slowly than the original nnUNet. We also studied the impact of topology update frequency. As shown in Figure 11 (b), when the topology updating frequency is increased, the convergence speed may decrease slightly, but the impact is not significant.
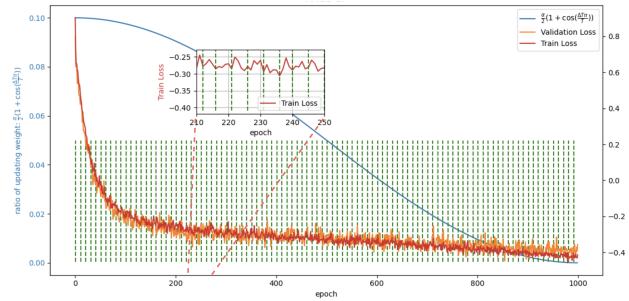
Figure 10: The learning curve of E2ENet on AMOS-CT, with green dotted vertical lines indicating the epochs of weight activation and deactivation. The blue line represents the ratio of weight deactivation/reactivation throughout the training process.

B.9 Organ Volume Statistics and Class-wise Results Visualization

In this section, we analyzed the relationship between organ volume and segmentation accuracy on the AMOS-CT, BTCV, and BraTS challenges. The results, depicted in Figures 12, 13 and 14, showed that small organs with relatively low segmentation accuracy. For the AMOS-CT challenge, RAG

26
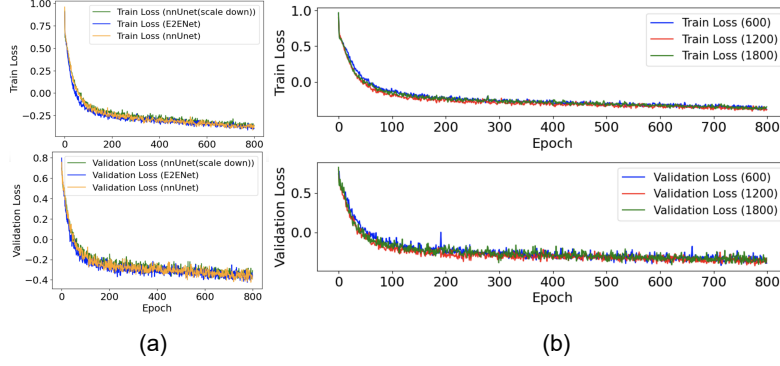
Figure 11: (a) Comparing the learning curve of E2ENet with that of nnUNet and scaled-down nnUNet (referred to as nnUNet (-)); (b) Comparing the learning curve of E2ENet with different topology update frequencies.
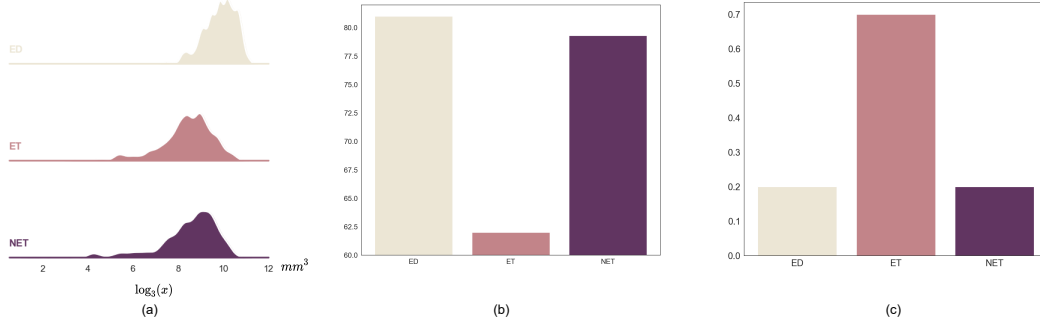
Figure 12: (a) The organ volume statistics of AMOS-CT training dataset. (b) Class-wise Dice of nnUNet without postprocessing (visualization of Table 1). (c) Class-wise Dice differences between E2ENet with feature sparsity 0.7 without postprocessing and nnUNet without postprocessing on AMOS-CT validation dataset . The positive value means that E2ENet outperforms nnUNet, vice versa.)
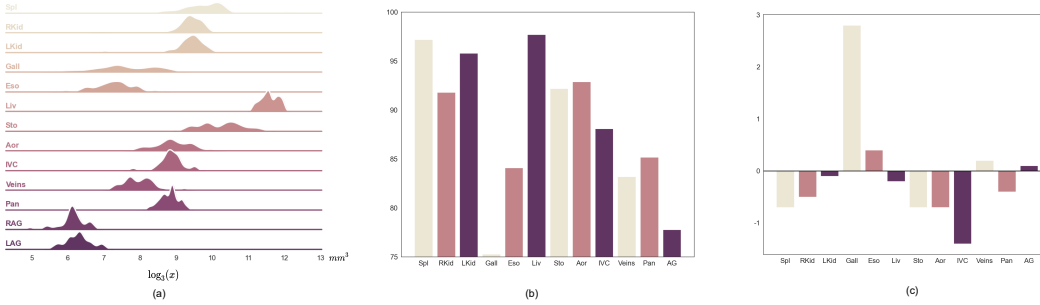
Figure 13: (a) The organ volume statistics of BTCV training dataset. (b) Class-wise Dice of nnUNet (visualization of Table 10). Note that AG denotes the average of the right and left adrenal glands (RAG and LAG). (c) Class-wise Dice differences between E2ENet with feature sparsity 0.7 and nnUNet on BTCV test dataset. The positive value means that E2ENet outperforms nnUNet, vice versa.

(right adrenal gland), LAG (left adrenal gland), Gall (gallbladder), and Eso (esophagus) are more challenging to accurately segment. This may be due to the fact that smaller organ volumes provide less visual information for the segmentation algorithm to work with. However, our proposed method, E2ENet, also demonstrated comparable (or better) performance on these small organs, particularly for the organ "LAG", in which the Dice improved from 81.7% to 82.4%. On the BTCV challenge, the Dice of "Gall", which is considered to be the most challenging organ, improves from 75.3% to

27
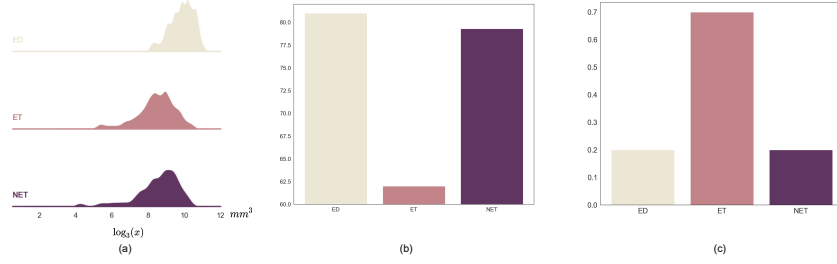
Figure 14: (a) The organ volume statistics of BraTS training dataset. (b) Class-wise Dice of nnUNet (visualization of Table 2) (c) Class-wise Dice differences between E2ENet with feature sparsity 0.7 and nnUNet on 5-fold cross-validation of the training dataset. The positive value means that E2ENet outperforms nnUNet, vice versa.

78.1% when using E2ENet compared to nnUNet. For the BraTs challenge, E2ENet demonstrates the most significant improvement in the Dice score of the "ET" region, which is considered the most challenging class, with an increase of 0.7%.

These results indicate that by applying the DSFF mechanism, E2ENet is able to effectively utilize multi-scale information, potentially leading to improved performance in segmenting small organs. It is important to note that other factors, such as the quality and resolution of the medical images, as well as the complexity of the anatomy being imaged, may also impact the performance of the segmentation algorithms. Future work could focus on further exploring the potential impact of these factors on segmentation accuracy.

28