Deep Learning for Medical Image Segmentation: Tricks, Challenges and Future Directions

Dong Zhang†, Yi Lin†, Hao Chen*, Zhuotao Tian, Xin Yang, Jinhui Tang, Kwang-Ting Cheng, *Fellow, IEEE*

Abstract—Over the past few years, the rapid development of deep learning technologies for computer vision has greatly promoted the performance of medical image segmentation (MedISeg). However, the recent MedISeg publications usually focus on presentations of the major contributions (*e.g.*, network architectures, training strategies, and loss functions) while unwittingly ignoring some marginal implementation details (also known as "tricks"), leading to a potential problem of the unfair experimental result comparisons. In this paper, we collect a series of MedISeg tricks for different model implementation phases (*i.e.*, pre-training model, data pre-processing, data augmentation, model implementation, model inference, and result post-processing), and experimentally explore the effectiveness of these tricks on the consistent baseline models. Compared to paper-driven surveys that only blandly focus on the advantages and limitation analyses of segmentation models, our work provides a large number of solid experiments and is more technically operable. With the extensive experimental results on both the representative 2D and 3D medical image datasets, we explicitly clarify the effect of these tricks. Moreover, based on the surveyed tricks, we also open-sourced a strong MedISeg repository, where each of its components has the advantage of plug-and-play. We believe that this milestone work not only completes a comprehensive and complementary survey of the state-of-the-art MedISeg approaches, but also offers a practical guide for addressing the future medical image processing challenges including but not limited to small dataset learning, class imbalance learning, multi-modality learning, and domain adaptation. The code has been released at: MedISeg.

Index Terms—Medical Image Analysis, Convolutional Neural Networks, Medical Image Segmentation, Computer Applications.

1 Introduction

MEDICAL image segmentation (MedISeg) is one of the most representative and comprehensive research topics in both communities of computer vision and medical image analysis [1]–[3]. It can not only recognize the object category but also locate the pixel-level positions [4]–[9]. In clinical practice, MedISeg has been successfully used in a wide range of potential applications with qualitative and quantitative analyses, *e.g.*, cancer diagnosis [10], tumor change detection [11], treatment planning [12], and computer-integrated surgery [13]. To achieve satisfactory segmentation performance, one of the key challenges is to enable the segmentation model to learn a set of rich yet discriminative feature representations [14]–[17].

In recent years, MedISeg performance has greatly improved, thanks to remarkable progress in deep learning technologies of image processing [17]–[21], MedISeg performance has greatly improved [5], [22]–[29]. Advanced backbones (*e.g.*, AlexNet [30], VGG [31], ResNet [18], DenseNet [21], MobilNet [32], ShuffleNet [33], ResNeXt [34], HRNet [35], RegNet [36], ViT [37], SwinTransformer [38], CMT [39], ConFormer [40], CvT [41]) can inherently learn rich semantic feature representations, directly facilitating MedISeg recognition ability. Certain elaborate feature regulations (*e.g.*, lateral connection [42], residual mapping [18], [21], encoder-decoder scheme [43], [44], dense connection [8], feature pyramid [45], and the global context aggregation [46], [47]) also enhance performance. The integration of these sophisticated elements into a uniform MedISeg system is the main reason that the MedlSystem performs so well. Besides, some training strategies (*e.g.*, co-training [48], [49], co-teaching [50], [51], co-learning [52], [53], and test-

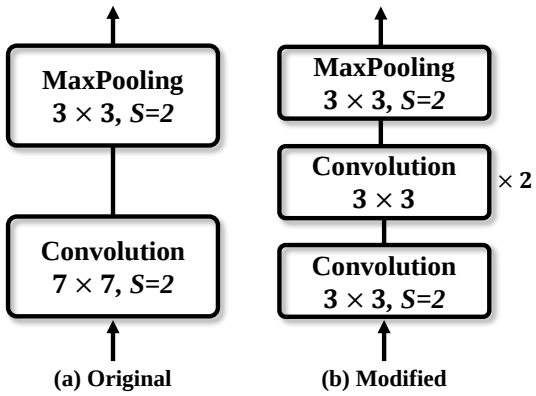


Fig. 1. Two implementation schemes of the input stem in ResNet [18], where (a) is the original implementation as claimed in its paper, and (b) is the modified implementation to reduce the computational costs. "S" denotes the stride size. "×2" denotes that this block is repeated twice.

- *D. Zhang, Y. Lin, H. Chen, and K. Cheng are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. E-mail: {dongz, jhc, timcheng}@ust.hk; yi.lin@connect.ust.hk.*
- *Z. Tian is with SmartMore Inc, China. E-mail: zttian@cse.cuhk.edu.hk.*
- *X. Yang is with the Department of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: xinyang2014@hust.edu.cn.*
- *J. Tang is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China. E-mail: jinhuitang@njust.edu.cn.*

† *These two authors contributed equally to this work.*
∗ *Corresponding author: Hao Chen.*

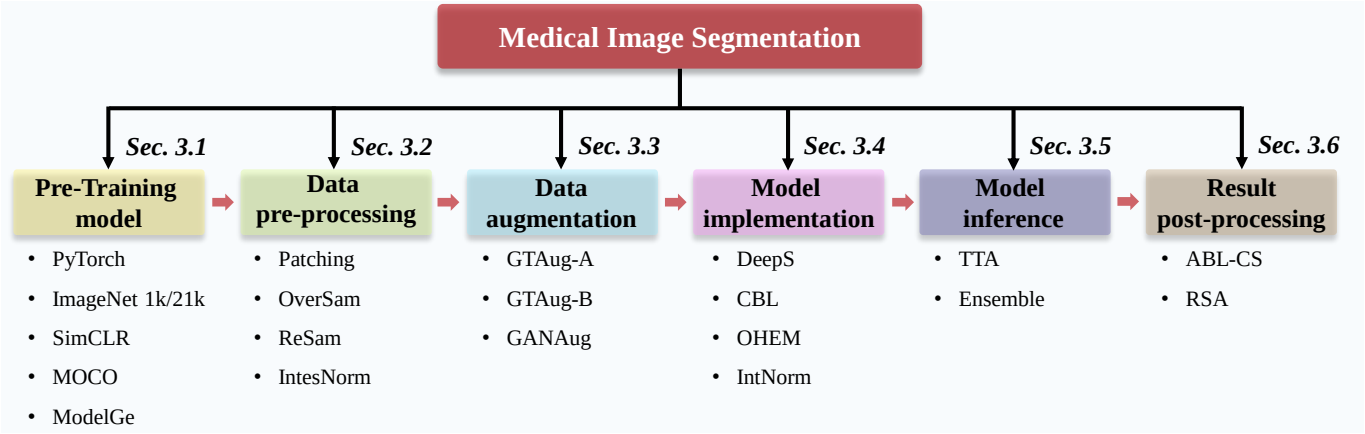


Fig. 2. An illustration of the surveyed MedISeg tricks and their latent relations. We separate the MedISeg model into six implementation phases, which are pre-training model, data pre-processing, data augmentation, model implementation, model inference, and result post-processing. For each trick, we experimentally explore its effectiveness on two typical segmentation baseline models, namely the 2D-UNet [43] and 3D-UNet [71] with the help of the representative CNNs backbones on four medical image segmentation datasets, *i.e.*, 2D ISIC 2018 [72], 2D CoNIC [73], 3D KiTS19 [74], and 3D LiTS [63].

time-training [54], [55]) and some mature loss functions (*e.g.*, cross-entropy loss, dice loss, and Lovasz-softmax loss [56], [57]) are also indispensable components that can affect model performance [58], [59].

However, signs of progress are not proposed solely, they are usually mixed with the existing implementations [60], [61]. In particular, at the present moment, an unabridged MedISeg system is usually composed of a large number of implementation details (including some non-learning model-agnostic pre-processing procedures) to achieve the ideal state-of-the-art recognition performance [5], [62]–[65]. Unfortunately, marginal implementation descriptions (also named "tricks") rarely exist in official publications or are only introduced in the released codes (there are some present in the supplementary material). For example, as illustrated in Figure 1, in the modified input stem of the prevalent ResNet [18] architecture (which is commonly treated as a prevailing backbone network for an MedISeg model), three cumulated 3×3 convolutional layers (in Figure 1 (b)) are used to replace the original 7×7 convolutional layer (in Figure 1 (a)) in the input stem to reduce the computational costs [66]–[68]. Although this subtle change can significantly improve accuracy [21], [61], [69], [70], few publications explicitly mention this. Therefore, the comparison of performance based on such a modified implementation and performance based on the original implementation is inherently unfair.

The devil is in the details. In this work, to reveal effects of tricks on an MedISeg model, as illustrated in Figure 2, according to a complete set of implementation phases including pre-training model (*ref.* Section 3.1), data pre-processing (*ref.* Section 3.2), data augmentation (*ref.* Section 3.3), model implementation (*ref.* Section 3.4), model inference (*ref.* Section 3.5), and result post-processing (*ref.* Section 3.6), we first collect a series of practical and representative tricks that are overlooked in the current MedISeg models. Then the effectiveness of these tricks is experimentally explored on the consistent segmentation baseline models including the typical 2D-UNet [43] and 3D-UNet [71] with the help of the representative convolution neural networks (CNNs) backbone networks, such that the influence of model variants (*i.e.*, performance changes due to model changes) can be avoided. Compared to the existing paper-driven technical surveys that only blandly focus on the advantage and limitation analyses of the image segmentation model, our work provides a large number of solid experimental results and is more technically operable for future work. Based on extensive experimental results on four medical image datasets (*i.e.*, the challenging 2D ISIC 2018 lesion boundary segmentation dataset [72], 2D colon nuclei identification and counting challenge dataset [73], [75], 3D kidney tumor segmentation 2019 dataset [74], and 3D liver tumor segmentation challenge dataset [63]), we explicitly clarify the effect of these tricks. Moreover, based on the surveyed tricks and the used baseline models, we also open-sourced a strong MedISeg repository, where each of its components has the advantage of plug-and-play. It is believed that this milestone work not only completes a comprehensive technological survey of the state-of-the-art MedISeg approaches, but also offers a practical guide for addressing future medical image processing (especially the dense image predicted tasks) challenges including small dataset learning, class imbalance learning, multi-modality learning, and domain adaptation.

The main contributions of this work can be summarized as follows:

- We collect a series of MedISeg tricks for different implementation phases, and experimentally explore the effectiveness of these tricks on consistent CNNs baseline models.
- We explicitly clarify the effectiveness of these tricks, and a large number of solid experimental results on both 2D and 3D medical image datasets compensate for the implementation neglect in MedISeg.
- We open-sourced a strong MedISeg repository, which includes rich segmentation tricks and each of them has the plug-and-play advantage.
- This milestone work will facilitate subsequent efforts to compare the experimental results of the MedISeg model under a fair environment.

- This work will provide practical guidance for a wide range of medical image processing especially segmentation challenges in the future.

The remainder of this paper is structured as follows: In Section 2, we first introduce the preliminary experimental setups including baseline models, experimental settings, the datasets used, and evaluation metrics. In Section 3, we introduce the collected tricks according to the ordinal training phases and provide extensive experimental results and the experimental analyses in detail. In Section 4, the whole paper including the challenges of this task is comprehensively discussed. Finally, in Section 5, we make a conclusion and show the potential directions.

2 PRELIMINARIES

2.1 Baselines

In this work, to ensure the model comprehensiveness, we choose the commonly used and representative 2D-UNet [43] and 3D-UNet [71] as our baseline models. Details of these two models are as follows:

2D-UNet [43]. 2D-UNet consists of an encoder network and a decoder network, which has been extensively used in the medical image segmentation domain [76]–[80]. The encoder network follows the classic fully convolutional architecture in that it has four spatial-tapering stages. Each stage consists of two 3×3 convolutional layers followed by a rectified linear unit (ReLU) activation function [81], [82] and a global max pooling layer (with the stride size $S = 2$). The decoder network, which takes the output of the encoder network as the input, also has four stages that correspond to the same spatial encoder stages. Within each decoder stage, a 2D transposed convolutional operator first upsamples feature maps $2\times$ via bilinear interpolation operation [83], and then two 3×3 convolutional layers and a ReLU activation function [81], [82] are deployed in sequence. In particular, in the last decoder network layer, the channel size of the output feature maps is assigned to the class size of the used dataset via a 2×2 convolutional layer.

3D-UNet [71]. The 3D-UNet has almost the same network architecture as the 2D-UNet [43] in that it consists of a 3D encoder network and a 3D decoder network [84]–[89]. 3D-UNet is usually used to handle the 3D image dataset for segmentation [90]–[92]. The implementation differences between the 3D-UNet and the 2D-UNet are i) the 2D convolution layer is replaced by the 3D convolution layer [93]; ii) a lateral connection is added between the same level of the encoder stage and a decoder stage, which has the same spatial and channel size; and iii) an intensity normalization layer [94] is implemented on the input images [81], [82].

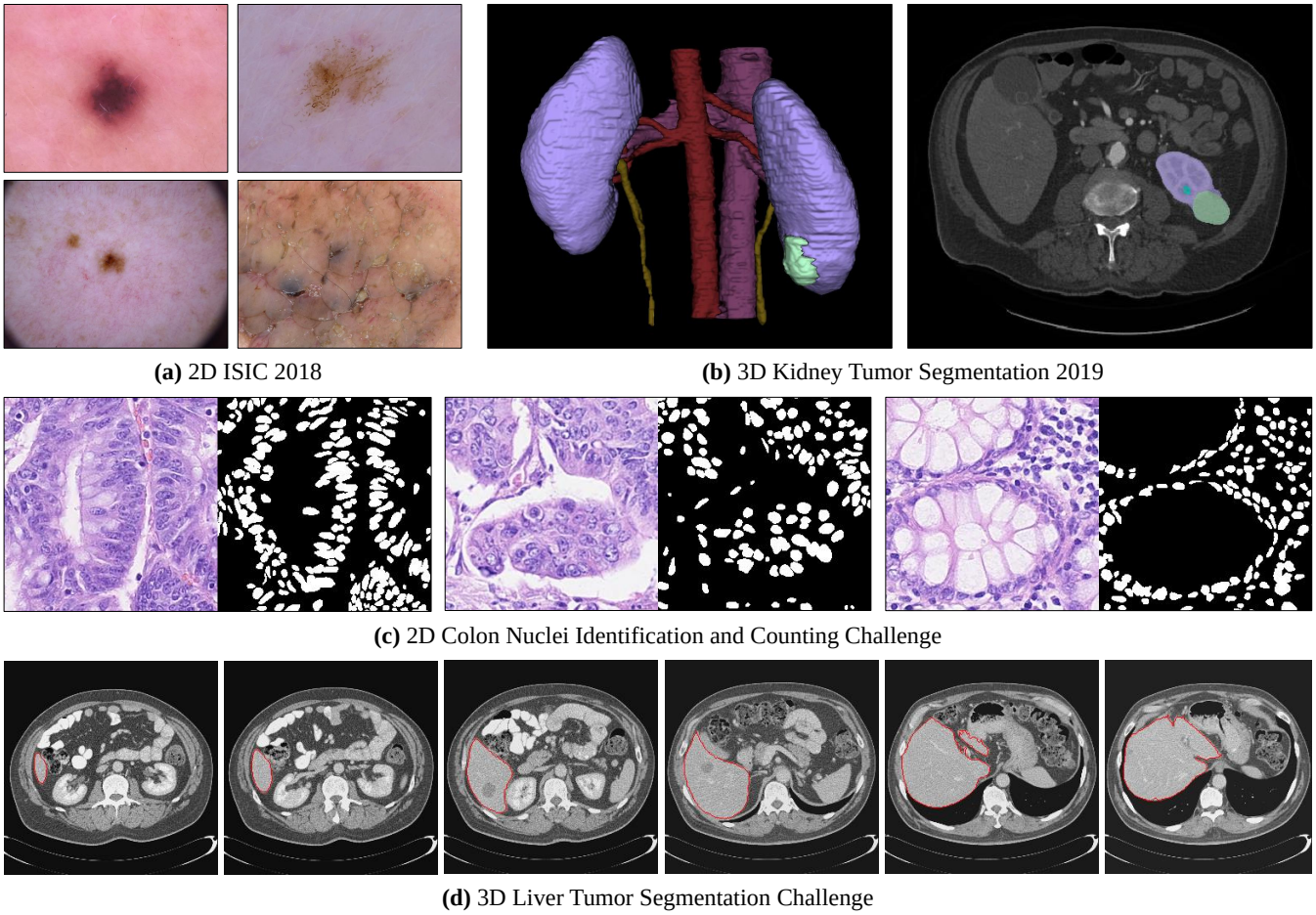
2.2 Datasets

In our experiments, to achieve comprehensive experimental implementation, and avoid the performance specificity of one specific dataset, four representative medical image datasets are chosen, which are 2D datasets with a common object size (*i.e.*, the ISIC 2018 lesion boundary segmentation dataset [72]), a 2D dataset with small object size (*i.e.*, the Colon Nuclei Identification and Counting Challenge (CoNIC) dataset [73], [75], [95]), the 3D kidney tumor segmentation 2019 (KiTS19) dataset [74], and the 3D Liver Tumor Segmentation Challenge (LiTS) dataset [63]. Figure 3 shows some experimental samples of these four datasets, and details of each dataset are introduced as follows:

2D ISIC 2018 [72]. ISIC 2018 is one of the most representative yet challenging 2D skin lesion boundary segmentation datasets in the computer-aided diagnosis domain, which consists of $2,594$ JPEG dermoscopic images and $2,594$ PNG ground truth (GT) images. In each image, as visualized in Figure 3 (a), one or more lesion regions are included with different sizes. Each skin lesion image has a uniform spatial size of 600×450. For this dataset, we only need to segment two categories of regions, *i.e.*, the foreground "lesion" region and the "background". In our experiments, we randomly split this dataset into 80% images for the training set and 20% images for the test set as in [96], [97]. In each cross-validation, we further randomly split 10% from the training set as the validation set.

2D CoNIC [73]. In 2D CoNIC challenge dataset, images are from the Lizard dataset [75], which consists of six nuclear categories (*i.e.*, "epithelial cells", "connective tissue cells", "lymphocytes", "plasma cells", "neutrophils", and "eosinophils") and one "background". For the Lizard dataset [73], [75], [95], as visualized in Figure 3 (c), each original image has one ground truth label, which contains the information of instance map, nuclear categories, bounding boxes segmentation mask, and nuclei counts. In our experiments, we use the segmentation mask for image segmentation, and we only distinguish between the foreground object "nuclei" and the "background", *i.e.*, it is a binary segmentation task as in [72]. In this dataset, $4,981$ image patches with the size of 256×256 are provided in the RGB format. In our experiments, following [98]–[100], we randomly split all images into 80% for training and 20% for testing. In each cross-validation, we further randomly split 10% from the training set as the validation set.

3D KiTS19 [74]. For KiTS19, a total of 210 high-quality annotated patients' images of the 3D abdominal computed tomography images are publicly provided. As visualized in Figure 3 (b), KiTS19 consists of three categories including "kidney", "tumor", and one "background". We can observe that the positions of the kidney and tumor are relatively fixed in the given images. In our experiments, following [74], [101], [102], we validate the binary segmentation performance of two common settings, *i.e.*, settings-i) the foreground "kidney" and the "background"; settings-ii) the foreground "tumor" and the "background". Although some other human structures (*e.g.*, ureters, arteries, and veins) are provided in KiTS19, they are not in the range that needs to be segmented, *i.e.*, they are uniformly considered as the background. Each image and the corresponding GT mask in KiTS19 are provided in the NIFTI format, which includes the number of slices (216 slices on average), height (512), and width (512). Following [74], [103], due to the concern over GT quality, case 15, 23, 37, 68, 125 and 133 are removed from the original dataset, and the remainder 204 cases are randomly divided into 80% for training, and 20% for testing. Following [74], [101], in each cross-validation, we further randomly split split off 10% from the training set as



(a) 2D ISIC 2018

(b) 3D Kidney Tumor Segmentation 2019

(c) 2D Colon Nuclei Identification and Counting Challenge

(d) 3D Liver Tumor Segmentation Challenge

Fig. 3. Some experimental samples, where images in (a) are from the 2D ISIC 2018 [72] training set, images in (b) are a rendering scenario and a segmented axial slice from the 3D Kidney tumor segmentation 2019 dataset [74], images in (c) are from the 2D colon nuclei identification and counting challenge dataset [75], and images in (d) are from the 3D liver tumor segmentation challenge dataset [63]. (Images are from the released dataset or its official website).

the validation set. Besides, although there are extra 90 test cases provided on its official website, the GT masks are not public access, so we neglect these cases.

3D LiTS [63]. Liver cancer is one of the most common tumors, and it usually occurs far more often in men (*i.e.*, the fifth most commonly occurring cancer) than in women (*i.e.*, the ninth most commonly occurring cancer) [63], [104], [105]. To improve the progress of the automatic segmentation of lesions, LiTS proposes a benchmark based on contrast-enhanced abdominal computed tomography scans. LiTS contain 130 scans for training and 70 scans for testing. A set of cross-section images is visualized in Figure 3 (d), where the red area represents the region of the liver. The images from left to right are a series of cross-sections of the same liver region. Images of this dataset are primitively provided in the PNG format with the spatial size of 256×256, and each image contains two categories: "background" and "liver". Following [106], [107], we treat the tumor area and the liver area the same. In the training phase, all the images are clipped into the spatial size of 200×200. In each cross-validation, we further randomly split off 10% from the training set as the validation set as in [105].

2.3 Settings

Platform. All experiments in this paper are implemented on PyTorch [108] deep learning platform with NVIDIA GeForce RTX 2080 GPUs. In our implementation, to avoid coding flaws as much as possible, we use the PyTorch official functions to complete the corresponding operations, including basic operations, loss calculations, and metric calculations.

Backbone. Backbone network is one of the most important factors affecting model performance. Considering the dataset size and the maturity of the network application, the representative ResNet-50 [18] is used as the default backbone network, which follows the classic implementation of the fully convolutional network architecture [17], [109], [110]. It is also noticed that the input stem of the backbone network is its original implementation via the cumulated convolutions, *i.e.*, the one in Figure 1 (a).

Implementations for 2D-UNet [43]. Following experiments in [96], [97], 2D training images in the ISIC 2018 dataset [72] used for the lesion boundary segmentation and in the CoNIC dataset [73], [75], [95] used for the colon nuclei segmentation are uniformly resized into the fixed input size of 200×200, and are also normalized using the mean value and the standard deviation of the ImageNet dataset [111] as

in [57], [79], [112]. The initial learning rate is set to 0.0003. The adaptive moment estimation [113] is used as the optimizer, where the weight decay is set to 0.0005. The whole model is trained with 200 training epochs with a batch size of 32, and the results of the best model on the validation set are used for testing as in [56], [102]. Following [8], [63], we employ the commonly used pixel-level cross-entropy [114] loss as the default loss function.

Implementations for 3D-UNet [71]. Following the commonly used experimental settings of the 3D-UNet for KiTS19 [74] and LiTS [63] in [63], [74], [103]–[105], [115], [116], Hounsfield units are clipped to $[-79, ..., 304]$ and the voxel spacing is resampled by the coefficient of $3.22 \times 1.62 \times 1.62$ mm^3 as in [56], [101], [102]. We down-sample the computed tomography scans on the cross-section and resample it to adjust the z-axis spacing of all data to 1mm for LiTS [63]. The training voxel size (*i.e.*, the image patching size) is resampled into $96 \times 96 \times 96$, which is just right to be crammed into an NVIDIA GeForce RTX 2080 GPU. The stochastic gradient descent [117] is used as the optimizer, where the momentum is set to 0.9 and the weight decay is set to 0.0001, respectively. The initial learning rate is set to 0.01, and the batch size is set to 2. The model is trained in an end-to-end fashion with 100 epochs, and the results of the best model on the validation set are used for testing as in [56], [102]. The cross-entropy loss is used as the unique loss function as in [115], [118]. In the inference phase, we perform patch-wise overlap with 50% region overlap (*i.e.*, $48 \times 48 \times 48$).

2.4 Evaluation Metrics

To evaluate the model performance, we employ the commonly used Recall, Precision, Dice [119], and Intersection over Union (IoU) as our primary metrics [74], [102], [103], [116]. In particular, we perform five-fold cross-validation to evaluate the model performance and report the average result, thereby avoiding random experimental results and the overfitting problem.

3 Methods and Experiments

We separate an unabridged MedISeg system into six implementation phases; these phases are: pre-training model described in subsection 3.1; data pre-processing described in subsection 3.2; data augmentation described in subsection 3.3; model implementation described in subsection 3.4; model inference described in subsection 3.5;, and result post-processing described in subsection 3.6. Based on each phase, a set of representative MedISeg tricks are empirically explored. It is noticed that all the experiments are implemented on the preliminary settings as described in Section 2.3 without a specific declaration.

3.1 Pre-Training Model

The pre-training model (*i.e.*, using the pre-trained weights as the initialized parameters for model fine-tuning) provides favorable parameters, such that the training convergence can be easily accelerated and the potential model can obtain a strong generalization ability [38], [120]. Different pre-training models have distinctly different influences, but this trick is usually overlooked. For example, for the pre-training model of the popular ImageNet [111], authors generally state: "Our network is pre-trained on the ImageNet [111].", while pre-training on ImageNet consists of at least two basic forms (*i.e.*, 1k and 21k versions). Only the experimental result comparisons that clarify a detailed pre-training implementation are fair. In this subsection, we explore six commonly used and publicly released pre-trained weights on MedISeg, and these pre-trained weights can be divided into the following two main camps: the fully-supervised one (*i.e.*, PyTorch official weights [108] and Model-oriented ImageNet 1k/21k weights [111]), and the self-supervised one (*i.e.*, SimCLR weights [121], MOCO weights [122] and Model genesis weights [123]). Implementation details of each pre-trained weight and the experimental results on MedISeg are as follows:

PyTorch official weights [108]. In the PyTorch repository, there are some backbone pre-trained weights provided by `torchvision.models` [108]. These pre-trained weights are obtained by training the corresponding backbone network for the single-label image classification task on ImageNet 1k dataset [111], where "1K" denotes that this dataset consists of $1,000$ classes of the common scenes.

Model-oriented ImageNet 1k weights [111]. Except for the PyTorch official pre-trained weights, the model creator usually releases the pre-trained weights on ImageNet [111], [124] as well. For example, as the backbone network that we use, the model-oriented ImageNet 1k weights can be obtained by training ResNet-50 [18] for image classification on the ImageNet 1k dataset. We then use the obtained trained weights for downstream vision tasks.

Model-oriented ImageNet 21k weights [111]. Compared to ImageNet 1k [111], ImageNet 21k is a more general and comprehensive dataset version, which has about $21,000$ object classes in total [125] for (weakly-supervised or semi-supervised or fully-supervised) image classification. Therefore, the trained weights on ImageNet 21k are more conducive to improving recognition performance of the downstream computer vision models [37], [38], [126].

SimCLR weights [121]. SimCLR demonstrates that introducing a learnable nonlinear transformation between feature representations and the contrastive learning loss can greatly improve model representation quality [127], [128]. Based on this assumption, SimCLR mainly consists of three implementation steps: 1) the input image is first grouped into some image patches; 2) different data augmentation strategies are then implemented on image patches for different batches; 3) the model is finally trained to obtain the similar results for the same image patches with different augmentations, and mutually exclude other results. In our work, we use the SimCLR weights that are trained on ImageNet 1k [111] for classification on ResNet-50 [18].

MOCO weights [122]. MoCo is one of the classical self-supervised contrastive learning methods. It aims to address the problem of sampled feature inconsistency in the memory bank [129], [130]. To this end, MoCo uses a queue to store and sample the negative samples; that is, it stores feature vectors of multiple recent batches used for training. In its implementation, the fixed network remains unchanged and a linear layer with the softmax layer is added to the

TABLE 1
Experimental results on different pre-trained weights. "PyTorch", "Imag-1k", "Imag-21k", "SimCLR", "MOCO" and "ModelGe" denotes the PyTorch [108] official weights, model-oriented ImageNet 1k [125] weights, model-oriented ImageNet 21k [125] weights, SimCLR [121] weights, MOCO [122] weights and the model genesis [123] weights, respectively. "+" means fine-tuning the baseline model on the corresponding weights.

Methods	Recall (%)	Percision (%)	Dice (%)	IoU (%)	Recall (%)	Percision (%)	Dice (%)	IoU (%)
2D-UNet [43]	ISIC 2018 [72]				CoNIC [75]			
	88.18	89.88	86.89	85.80	78.12	77.25	77.23	77.58
+ PyTorch [108]	89.28$_{+1.10}$	90.08$_{+0.20}$	88.09$_{+1.20}$	87.07$_{+1.27}$	78.08$_{-0.04}$	79.21$_{+1.96}$	78.08$_{+0.85}$	78.38$_{+0.80}$
+ Imag-1k [125]	89.19$_{+1.01}$	90.07$_{+0.19}$	87.99$_{+1.10}$	86.93$_{+1.13}$	79.48$_{+1.36}$	78.69$_{+1.44}$	78.70$_{+1.47}$	78.77$_{+1.19}$
+ Imag-21k [125]	90.21$_{+2.03}$	91.48$_{+1.60}$	89.38$_{+2.49}$	88.00$_{+2.20}$	78.79$_{+0.67}$	79.66$_{+2.41}$	78.75$_{+1.52}$	78.91$_{+1.33}$
+ SimCLR [121]	88.09$_{-0.09}$	89.93$_{+0.05}$	86.95$_{+0.06}$	85.90$_{+0.10}$	77.87$_{-0.25}$	77.44$_{+0.19}$	77.17$_{-0.06}$	77.53$_{-0.05}$
+ MOCO [122]	87.99$_{-0.19}$	90.11$_{+0.23}$	86.88$_{-0.01}$	85.84$_{+0.04}$	77.98$_{-0.14}$	77.65$_{+0.40}$	77.35$_{+0.12}$	77.67$_{+0.09}$
3D-UNet [71]	KiTS19 [74]: settings-i				KiTS19 [74]: settings-ii			
	91.01	95.20	92.50	87.35	27.35	46.71	29.63	21.51
+ ModelGe [123]	90.70$_{-0.31}$	95.48$_{+0.28}$	92.29$_{-0.21}$	87.18$_{-0.17}$	28.17$_{+0.82}$	47.62$_{+0.91}$	29.95$_{+0.32}$	21.75$_{+0.24}$

TABLE 2
Experimental results on 3D LiTS dataset [63]. "ModelGe" denotes the model genesis [123] weight. "+" means fine-tuning the baseline model on the corresponding pre-trained weights.

Methods	Recall (%)	Percision (%)	Dice (%)	IoU (%)
3D-UNet [71]	89.33	84.03	86.11	76.44
+ ModelGe [123]	90.54$_{+1.21}$	84.66$_{+0.63}$	86.99$_{+0.88}$	77.67$_{+1.23}$

end of the backbone for classification in an unsupervised training manner. In our work, we use the MOCO weights that are trained on ImageNet 1k [111] for classification on ResNet-50 [18].

Model genesis (ModelGe) weights [123]. ModelGe is an advanced self-supervised model pre-training technology, which usually consists of four transformation operations (*i.e.*, non-linear, local pixel shuffling, out-painting, and in-painting) for single image restoration on computed tomography and magnetic resonance imaging images [131], [132]. In its implementation, the network is trained to learn a general visual representation by restoring the original image patch from the transformed one, where the weight for these transformation operations is set to 0.9 for non-linear, 0.5 for local pixel shuffling, 0.8 for out-painting, and 0.2 for in-painting following [133]–[135]. Based on the trained model, the obtained trained weights can be used as the pre-trained weights for downstream models.

Experimental results. The experimental results on ISIC 2018 [72], CoNIC [73], and KiTS19 [136] are shown in Table 1, and the results on LiTS [63] are shown in Table 2. In general, we can observe that compared to the baseline model that is trained from scratch (*i.e.*, without using the pre-trained weights), the fine-tuned model can achieve an overall better performance. This observation not only validates the effectiveness of the pre-trained weights but also verifies that the effect of different pre-trained weights is variable. For example, compared to the baseline results of 2D-UNet [43] on ISIC 2018, using the pre-trained weights improves performance under almost every evaluation metric. Specifically, fine-tuning can bring the max performance gain of Recall, Precision, Dice, and IoU by 2.03%, 1.60%, 2.49%, and 2.20%, respectively. These maximum performance gains all benefits from the ImagNet-21k [111] pre-trained weights. This experimental observation validates the powerful representation ability of ImagNet-21k [38], [137]. Besides, the performance gain of the pre-training model on Precision is relatively small, *i.e.*, only 0.45% on average. We can obtain similar experimental observations and conclusions from the CoNIC [73] dataset. In particular, the performance gains on Recall are relatively small (*i.e.*, only 0.41% on average), and the performance gains on Precision are relatively large (*i.e.*, 1.08% on average) on CoNIC [73]. The above experimental results demonstrate that the same tricks may have different effects on different datasets. Moreover, using the same dataset, different evaluation metrics will also perform differently. Although the model performance has dropped on some evaluation metrics, we believe that this is not due to the pre-trained weights. Since the pre-trained weights are obtained from natural scenes, our task is about the medical images. There may be a domain gap between natural scense and medical images. This problem can be solved in the future by using the pre-trained weights of medical images. Compared to the baseline results on KiTS19 [136] and LiTS [63], the fine-tuned 3D-UNet using the released ModelGe weights can bring the max performance gain on setting-ii (*i.e.*, segment the foreground "tumor" and the "background") by 1.21%, 0.91%, 0.88%, and 1.23% of Recall, Precision, Dice, and IoU, respectively. However, on setting-ion KiTS19 [136], three-quarters of the performance has a slight drop, *i.e.*, −0.31% Recall, −0.21% Dice, and −0.17% IoU. We speculate that this may be because the kidney region in KiTS19 [136] is more sensitive to the initialized parameters. This observation inspires us, in the future, to consider not only the differences between the network architectures of the fine-tuned models but also the state of the used dataset as important factors. These 3D experimental results demonstrate that under the same implementation trick with the same datasets and evaluation metrics, different experimental settings will lead to differences in performance [61], [138].

3.2 Data Pre-Processing

Due to data specificity (*e.g.*, image modalities, and versatile image resolutions) of 3D medical images for deep CNNs, data pre-processing is necessary for obtaining satisfactory recognition performance [139], [140]. We mainly explore the effectiveness of four commonly used image pre-processing tricks in 3D-UNet [71], namely, patching [62], oversampling (OverSam) [141], resampling (ReSam) [142], and intensity normalization (IntesNorm) [102]. Implementation details of each data pre-processing strategy and the experimental results on MedISeg are as follows:

Patching [62]. Some specific categories of medical images (*e.g.*, MRI [143], and pathology images [62]) tend to be very large in spatial size and lack sufficient training samples in quantitative terms. Hence it is impractical to train a MedISeg model directly using these images [144], [145]. Instead, people usually resample the whole image into different image patches at smaller spatial scales with/without overlaps, such that the model can be implemented with less GPU memory costs and can be better trained. Intuitively, Patching size is one of the most important factors affecting model performance. In our work, following the experimental settings on 3D-UNet [56], [102], the training patching size is set to $96 \times 96 \times 96$ without the overlap, and the patching size is set to $96 \times 96 \times 96$ with 50% region overlap in the inference phase.

OverSam [141]. OverSam strategy is proposed to address problem of class imbalance between the positive and negative samples. OverSam is mainly used in the minority class samples [146]. At present, a group of OverSam schemes have been proposed, *e.g.*, random oversampling [141], synthetic minority oversampling (SMOTE) [147], borderline SMOTE [148], and the adaptive synthetic sampling [149]. A wide range of experimental results validates that OverSam schemes do not affect the model slope, but can amplify the model intercept [150], [151]. In our work, we follow the prevailing oversampling strategy as in [102], *i.e.*, 70% of the selected training samples are from the random image locations, while 30% of the image patches are guaranteed to hold at least one training foreground class. In this way, each training sample can simultaneously contain one training foreground image patch and one random image patch.

ReSam [142]. Image ReSam strategy is proposed to improve the representational ability of the used dataset via the machine learning model. Because the available sample ability is sometimes limited and heterogeneous, a better sub-sample dataset can be obtained via a random/nonrandom ReSam strategy [152], [153]. In its implementation, ReSam mainly consists of four steps: 1) spacing interpolation; 2) window transform; 3) acquisition of mask effective range, and 4) generation of sub-images. Based on the reorganized sub-sample dataset, a better-performing recognition model can be trained [142], [154]. In our baseline implementation, the commonly used random ReSam strategy has been used. To demonstrate its importance in MedISeg, in this section, we explore the effect of the ReSam strategy by removing it (*i.e.*, /o) in our experiments, *i.e.*, the image pixels are directly interpolated and scaled, such that the actual distances represented by the pixels are the same.

IntesNorm [155]. IntesNorm is a specific normalization strategy that is mainly used for medical images [156], [157]. There are usually two commonly used IntesNorm methods: z-scoring for all modalities and another one for computed tomography images [102]. In our work, we mainly explore the effectiveness of IntesNorm by removing it (*i.e.*, /o) on KiTS19 [74] in our experiments. Following the common implementation [102], [155], a global normalization scheme is adopted in this paper, where 0.5% of the foreground voxels is used for clipping and computing the foreground mean, and 99.5% of the foreground voxels is used for computing the standard deviation.

Experimental results. Experimental results on KiTS19 [136] and on LiTS [63] are given in Table 3 and Table 4, respectively. We can observe that compared to the baseline model on KiTS19 [136], the effect of these pre-processing operations on setting-ii is more sensitive than that on setting-i. More specifically, 1) the larger the patching size, the better the model's overall performance, where the baseline model is based on Patching_{96}. When the patching size is relatively small (*e.g.*, the patching size is set to 32), the segmentation model is even corrupted on both KiTS19 [136] and LiTS [63]. When the patching size is set to 128, the model on KiTS19 [136] can achieve the best performance on setting-ii by 37.81% Recall, 52.28% Precision, 38.06% Dice, and 28.81% IoU, which surpasses the baseline by 10.46% Recall, 5.57% Precision, 8.43% Dice, and 7.30% IoU, respectively. Experimental results on LiTS [63] show that the model performance gain increases gradually with the increase of patching size. Although there is a slight performance drop on setting-i under Patching_{128} when compared to the baseline performance, this is due to the experimental setting on the image scale rather than the patching scheme [102]. This experimental phenomenon is consistent with the conclusions of previous papers in [56], [102], which is recommended that we should use as large an image patch size as the GPU memory can accommodate. 2) With the help of the OverSam strategy [146], [149], the model performance is overall improved. In particular, + OverSam on KiTS19 [136] can bring the remarkable performance gain of 0.84% Recall, 0.43% Dice and 0.73% IoU on setting-i, and 9.11% Recall, 6.98% Precision, 6.06% Dice and 5.05% IoU on setting-ii, respectively. 3) Under the absence of ReSam [158], the model performance is significantly reduced on both KiTS19 [136] and LiTS [63]. For example, on KiTS19, the model can result in a performance decrease of 90.83% Recall, 69.71% Precision, 92.15% Dice, and 87.17% IoU on setting-i. Surprisingly, the model is completely corrupted without using ReSam under setting-ii. 4) IntesNorm [155] has relatively weak effect on experimental results on both setting-i and setting-iion KiTS19. For example, without using the IntesNorm on the baseline 3D-UNet [71] can only reduce the max model performance of 0.78% Recall, 0.30% Dice, and 0.70% IoU on setting-i, and 0.20% Recall and 0.00% Dice on setting-ii, respectively. However, IntesNorm has a significant effect on the experimental results on LiTS [63]. This phenomenon shows that the same method has different effects on different datasets in terms of the data pre-processing strategies. The experimental results from 2) to 4) validate the importance of OverSam, ReSam, and IntesNorm on MedISeg.

TABLE 3
Experimental results on data pre-processing tricks, where the two columns of results are experiments of setting-i and setting-ii (as introduced in section 2.2), respectively. "OverSam", "ReSam", "IntesNorm" denotes Oversampling [141], Resampling [142], and Intensity normalization [102], respectively. "/o" denotes that this trick is not implemented under this setting, and "NaN" denotes that the model under this setting is corrupted.

Methods	Recall (%)	Percision (%)	Dice (%)	IoU (%)	Recall (%)	Percision (%)	Dice (%)	IoU (%)
3D-UNet [71]	KiTS19 [74]: settings-i				KiTS19 [74]: settings-ii			
	91.01	95.20	92.50	87.35	27.35	46.71	29.63	21.51
Patching$_{32}$ [102]	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Patching$_{64}$ [102]	37.38$_{-53.63}$	59.51$_{-35.69}$	43.47$_{-49.03}$	29.87$_{-57.48}$	5.91$_{-21.44}$	9.40$_{-37.31}$	5.07$_{-24.56}$	2.89$_{-18.62}$
Patching$_{96}$ [102]	91.01$_{+0.00}$	95.20$_{+0.00}$	92.50$_{+0.00}$	87.35$_{+0.00}$	27.35$_{+0.00}$	46.71$_{+0.00}$	29.63$_{+0.00}$	21.51$_{+0.00}$
Patching$_{128}$ [102]	89.37$_{-1.64}$	95.24$_{+0.04}$	91.56$_{-0.94}$	85.88$_{-1.47}$	37.81$_{+10.46}$	52.28$_{+5.57}$	38.06$_{+8.43}$	28.81$_{+7.30}$
+ OverSam [141]	91.85$_{+0.84}$	95.05$_{-0.15}$	92.93$_{+0.43}$	88.08$_{+0.73}$	35.46$_{+8.11}$	53.69$_{+6.98}$	35.69$_{+6.06}$	26.56$_{+5.05}$
(/o) ReSam [142]	0.18$_{-90.83}$	25.49$_{-69.71}$	0.35$_{-92.15}$	0.18$_{-87.17}$	NaN	NaN	NaN	NaN
(/o) IntesNorm [94]	90.23$_{-0.78}$	95.33$_{+0.13}$	92.20$_{-0.30}$	86.65$_{-0.70}$	27.15$_{-0.20}$	49.29$_{+2.58}$	29.63$_{+0.00}$	21.73$_{+0.22}$

TABLE 4
Experimental results on 3D LiTS dataset [63] on data pre-processing tricks. "OverSam", "ReSam", "IntesNorm" denotes Oversampling [141], Resampling [142], and Intensity normalization [102], respectively. "/o" denotes that this trick is not implemented under this setting, and "NaN" denotes that the model under this setting is corrupted.

Methods	Recall (%)	Percision (%)	Dice (%)	IoU (%)
3D-UNet [71]	89.33	84.03	86.11	76.44
Patching$_{32}$ [102]	NaN	NaN	NaN	NaN
Patching$_{64}$ [102]	73.27$_{-16.06}$	78.57$_{-5.46}$	75.14$_{-10.97}$	60.71$_{-15.73}$
Patching$_{96}$ [102]	89.33$_{+0.00}$	84.03$_{+0.00}$	86.11$_{+0.00}$	76.44$_{+0.00}$
Patching$_{128}$ [102]	92.05$_{+2.72}$	91.55$_{+7.52}$	91.44$_{+5.33}$	84.60$_{+8.16}$
Patching$_{160}$ [102]	93.29$_{+3.96}$	94.94$_{+10.91}$	93.88$_{+7.77}$	88.76$_{+12.32}$
Patching$_{192}$ [102]	93.31$_{+3.98}$	95.35$_{+11.32}$	94.08$_{+7.97}$	89.18$_{+12.74}$
+ OverSam [141]	89.78$_{+0.45}$	87.55$_{+3.52}$	88.14$_{+2.03}$	79.85$_{+3.41}$
(/o) ReSam [142]	73.19$_{-16.14}$	78.86$_{-5.17}$	73.87$_{-12.24}$	60.84$_{-15.60}$
(/o) IntesNorm [94]	87.64$_{-1.69}$	81.14$_{-2.89}$	83.68$_{-2.43}$	72.37$_{-4.07}$

3.3 Data Augmentation

Data augmentation is one of the most fundamental technologies in the computer vision domain. It is usually used to deal with the problem of insufficient training samples in quantitative terms and it can be used to alleviate the overfitting problem, to give the strong model generalization ability, and to confer robustness [159]–[163]. Especially, for medical images, data augmentation is usually used to address data shortage problems [164], [165]. The data augmentation schemes used for MedISeg can be mainly divided into the following two categories: geometric transformation-based data augmentation (GTAug) [160], and generative adversarial network (GAN)-based data augmentation (GANAug) [162]. Implementation details of each data augmentation strategy and the experimental results on MedISeg are as follows:

GTAug [160]. GTAug is proposed to eliminate the effect of some geometric object variations in training images, *e.g.*, positions, scales, and viewing angles. The prevalently used GTAug includes flipping, cropping, rotations, translating, color jittering, contrast, simulation of low resolution, Gaussian noise injection, mixing images, random erasing, Gaussian blur, mixup, and cutmix [56], [159], [161], [163]. In our work, we choose the commonly used random brightness contrast (with the setting of brightness limit = 0.2, contrast limit = 0.2, and $p = 0.5$), random gamma (with the setting of gamma limit = $(80, 120)$, and $p = 0.5$), CLAHE, random noise with $p = 0.5$, gamma adjust with $p = 0.5$, shift scale rotate (with the setting of shift limit=0.1, scale limit=0.1, rotate limit=45, and $p = 0.5$), horizontal flip with $p = 0.5$, vertical flip with $p = 0.5$, random scale([0.85, 1.25]) with $p = 0.5$, random rotation([90, 180, 279]), random mirroring along the X, Y, Z axial directions in an MedISeg model. By dividing the above data augmentation schemes into two groups, we implement two types of data augmentation, named GTAug-A (*i.e.*, the pixel-level transform) and GTAug-B (*i.e.*, the spatial-level transform). To be specific, for experiments on 2D-UNet [43], random brightness contrast, random gamma, and CLAHE are used in GTAug-A, and shift scale rotate horizontal flip, and vertical flip are used in GTAug-B. All schemes are deployed with their default parameters. For experiments on 3D-UNet [71], random noise and gamma adjust are used in GTAug-A, and random scale, random rotation, and random mirroring are used in GTAug-B with their default parameters.

GANAug [166]. The intrinsic prerequisite for data augmentation is to introduce the domain knowledge or other incremental information into the training dataset [167]–[169]. From this aspect, GANAug can be viewed as a loss function that focuses on guiding the network to generate some real data that is close to the source dataset domain [112], [159]. Especially, given the small dataset in medical images, the data distribution fitted by the generative model of GAN is better than that of the discriminative model. Therefore, GANAug is suitable for MedISeg task [170], [171]. In our work, the classical pixel-level image-to-image translation with conditional adversarial networks [166] is used for data augmentation with its default setting.

Experimental results. In Table 5 and Table 6, we show the experimental results of different data augmentation schemes. We can observe that compared to the baseline 2D-UNet [43] on ISIC 2018 [72] in Table 5, GTAug-B can

TABLE 5
Experimental results on different data augmentation schemes. "GTAug" and "GANAug" denote the geometric transformation-based and the generative adversarial network-based data augmentation [166], respectively. "All" denotes that both GTAug and GANAug are used in the model training phase.

Methods	Recall (%)	Percision (%)	Dice (%)	IoU (%)	Recall (%)	Percision (%)	Dice (%)	IoU (%)
2D-UNet [43]	ISIC 2018 [72]				CoNIC [75]			
	88.18	89.88	86.89	85.80	78.12	77.25	77.23	77.58
+ GTAug-A	87.67$_{-0.51}$	90.19$_{+0.31}$	86.87$_{-0.02}$	85.68$_{-0.12}$	78.94$_{+0.82}$	77.54$_{+0.29}$	77.87$_{+0.64}$	78.05$_{+0.47}$
+ GTAug-B	88.32$_{+0.14}$	91.11$_{+1.23}$	88.07$_{+1.18}$	86.98$_{+1.18}$	79.28$_{+1.16}$	82.53$_{+5.28}$	80.33$_{+3.10}$	80.35$_{+2.77}$
+ GANAug [166]	87.78$_{-0.40}$	89.59$_{-0.29}$	86.47$_{-0.42}$	85.64$_{-0.16}$	78.43$_{+0.31}$	77.05$_{-0.20}$	77.37$_{+0.14}$	77.62$_{+0.04}$
+ All	87.77$_{-0.41}$	90.25$_{+0.37}$	87.30$_{+0.41}$	86.63$_{+0.83}$	81.17$_{+3.05}$	80.27$_{+3.02}$	80.39$_{+3.16}$	80.13$_{+2.55}$
3D-UNet [71]	KiTS19 [74]: settings-i				KiTS19 [74]: settings-ii			
	91.01	95.20	92.50	87.35	27.35	46.71	29.63	21.51
+ GTAug-A	89.60$_{-1.41}$	95.44$_{+0.24}$	91.65$_{-0.85}$	86.13$_{-1.22}$	21.81$_{-5.54}$	45.89$_{-0.82}$	25.68$_{-3.95}$	18.19$_{-3.32}$
+ GTAug-B	84.40$_{-6.61}$	94.58$_{-0.62}$	88.00$_{-4.50}$	80.94$_{-6.41}$	14.02$_{-13.33}$	37.70$_{-9.01}$	15.64$_{-13.99}$	10.71$_{-10.80}$
+ GANAug [166]	91.89$_{+0.88}$	94.88$_{-0.32}$	92.87$_{+0.37}$	87.98$_{+0.63}$	29.19$_{+1.84}$	47.69$_{+0.98}$	30.99$_{+1.36}$	22.89$_{+1.38}$
+ All	85.97$_{-5.04}$	90.55$_{-4.65}$	86.93$_{-5.57}$	78.91$_{-8.44}$	6.86$_{-20.49}$	30.39$_{-16.32}$	9.93$_{-19.70}$	6.29$_{-15.22}$

TABLE 6
Experimental results on 3D LiTS dataset [63] on different data augmentation schemes. "GTAug" and "GANAug" denote the geometric transformation-based and the generative adversarial network-based data augmentation [166], respectively. "All" denotes that both GTAug and GANAug are used in the model training phase.

Methods	Recall (%)	Percision (%)	Dice (%)	IoU (%)
3D-UNet [71]	89.33	84.03	86.11	76.44
+ GTAug-A	90.28$_{+0.95}$	84.24$_{+0.21}$	86.62$_{+0.51}$	76.89$_{+0.45}$
+ GTAug-B	84.85$_{-4.48}$	81.60$_{-2.43}$	82.45$_{-3.66}$	70.97$_{-5.47}$
+ GANAug [166]	75.83$_{-13.50}$	76.42$_{-7.61}$	75.00$_{-11.11}$	61.49$_{-14.95}$
+ All	70.32$_{-19.01}$	77.04$_{-6.99}$	71.69$_{-14.42}$	57.89$_{-18.55}$

markedly improve the recognition performance according to all the evaluation metrics, while GTAug-A brings a slight performance reduction under three metrics, *i.e.*, -0.51% Recall, -0.02% Dice, and -0.12% IoU. The above experimental observations validate that the pixel-level transform outperforms the spatial-level transform on 2D medical images. In contrast to GTAug, GANAug achieves 87.78% Recall, 89.59% Precision, 86.47% Dice, and 85.64% IoU, Which represents reductions in performance of -0.40% Recall, -0.29% Precision, -0.42% Dice, and -0.16% IoU, respectively. The overall performance of GANAug is worse than the recognition performance of GTAug. When both GTAug and GANAug are implemented on 2D-UNet [43] on ISIC 2018 [72] (*i.e.*, + All in Table 5), the model improve the performance according to three evaluation metrics, *i.e.*, 0.37% Precision, 0.41% Dice, and 0.83% IoU, which finally achieves the performance of 87.77% Recall, 90.25% Precision, 87.30% Dice, and 86.63% IoU. Compared to the experimental results on the baseline 2D-UNet [43] on CoNIC [73], we can observe that almost all the values under these evaluation metrics are improved. Particularly, there is only a 0.20% Precision reduction. By comparing the experimental results on ISIC 2018 [72] and CoNIC [73], we can observe that GTAug-B can achieve better experimental results on both datasets than GTAug-A and GANAug, demonstrating the importance of shift scale rotate, horizontal flip, and vertical flip in medical image segmentation. Besides, the above two sets of experimental results show that when faced with different datasets, we should choose the appropriate data augmentation tricks. The same observation and conclusion are obtained in Section 3.1.

For the experimental results on 3D KiTS19 [136] and LiTS [63], in general, using GANAug can boost the max performance gain, and using GTAug is barely beneficial on KiTS19 [136]. While the experimental results on LiTS [63] show that only GTAug-A is beneficial, both GTAug-B and GANAug are detrimental to the experimental performance. For example, using GTAug on 3D-UNet on KiTS19 [136] reduces the max performance of -13.33%, 9.01%, 13.99%, and 10.80% by Recall, Precision, Dice, and IoU, respectively. Comparatively, implementing GANAug achieves performance gains of 0.88% Recall, 0.37% Dice and 0.63% IoU on setting-i, and 1.84% Recall, 0.98% Precision, 1.36% Dice and 1.38% IoU on setting-ii. No surprise, when both GTAug and GANAug are implemented on KiTS19 [136], the model achieves the lower performance of 85.97% Recall, 90.55% Precision, 86.93% Dice and 78.91% IoU on setting-i, and 6.86% Recall, 30.39% Precision, 9.93% Dice and 6.29% IoU on setting-ii. In particular, results on setting-ii have the max performance reduction, which is -20.49%, -16.32%, -19.70% and -15.22% on Recall, Precision, Dice and IoU, respectively. Experimental results on LiTS [63] show that GTAug-A can bring performance improvements to every evaluation metric, *i.e.*, 0.95% Recall, 0.21% Precision, 0.51% Dice, and 0.45% IoU, respectively. In contrast, deploying GANAug and GTAug at the same time (*i.e.*, +All) will bring the most performance degradation. These experimental results on KiTS19 [136] and LiTS [63] demonstrate that we need to consider the specific dataset state when choosing the data augmentation method. A more detailed analysis can be found in Section 4.

TABLE 7
Experimental results on some model implementation tricks. "DeepS", "CBL", "OHEM", and IntNorm denotes deep supervision [173], class balance loss [174], online hard example mining [178], and instance normalization [94], respectively.

Methods	Recall (%)	Percision (%)	Dice (%)	IoU (%)	Recall (%)	Percision (%)	Dice (%)	IoU (%)
2D-UNet [43]	ISIC 2018 [72]				CoNIC [75]			
	88.18	89.88	86.89	85.80	78.12	77.25	77.23	77.58
+ DeepS [173]	88.91$_{+0.73}$	89.85$_{-0.03}$	87.42$_{+0.53}$	86.18$_{+0.38}$	78.16$_{+0.04}$	77.02$_{-0.23}$	77.13$_{-0.10}$	77.46$_{-0.12}$
+ CBL$_{\text{Dice}}$ [175]	89.59$_{+1.41}$	89.89$_{+0.01}$	87.71$_{+0.82}$	86.36$_{+0.56}$	79.38$_{+1.26}$	78.37$_{+1.12}$	78.36$_{+1.13}$	78.51$_{+0.93}$
+ CBL$_{\text{Focal}}$ [176]	88.06$_{-0.12}$	87.32$_{-2.56}$	85.41$_{-1.48}$	84.65$_{-1.15}$	81.78$_{+3.66}$	73.75$_{-3.50}$	77.12$_{-0.11}$	77.24$_{-0.34}$
+ CBL$_{\text{Tvers}}$ [57]	89.40$_{+1.22}$	90.19$_{+0.31}$	87.87$_{+0.98}$	86.42$_{+0.62}$	78.65$_{+0.53}$	78.79$_{+1.54}$	78.23$_{+1.00}$	78.40$_{+0.82}$
+ CBL$_{\text{WCE}}$ [177]	89.72$_{+1.54}$	88.15$_{-1.73}$	86.72$_{-0.17}$	85.63$_{-0.17}$	82.24$_{+4.12}$	74.08$_{-3.17}$	77.54$_{+0.31}$	77.58$_{+0.00}$
+ OHEM [178]	88.06$_{-0.12}$	89.81$_{-0.07}$	86.85$_{-0.04}$	85.80$_{+0.00}$	77.35$_{-0.77}$	77.72$_{+0.47}$	77.06$_{-0.17}$	77.47$_{-0.11}$
3D-UNet [71]	KiTS19 [74]: settings-i				KiTS19 [74]: settings-ii			
	91.01	95.20	92.50	87.35	27.35	46.71	29.63	21.51
+ DeepS [173]	90.13$_{-0.88}$	95.07$_{-0.13}$	91.88$_{-0.62}$	86.42$_{-0.93}$	26.69$_{-0.66}$	46.03$_{-0.68}$	29.27$_{-0.36}$	21.73$_{+0.22}$
+ CBL$_{\text{Dice}}$ [175]	90.50$_{-0.51}$	93.24$_{-1.96}$	91.17$_{-1.33}$	85.07$_{-2.28}$	41.91$_{+14.56}$	44.70$_{-2.01}$	37.15$_{+7.52}$	26.74$_{+5.23}$
+ CBL$_{\text{Focal}}$ [176]	74.88$_{-16.13}$	95.64$_{+0.44}$	81.29$_{-11.21}$	72.27$_{-15.08}$	9.72$_{-17.63}$	37.80$_{-8.91}$	12.98$_{-16.65}$	8.29$_{-13.22}$
+ CBL$_{\text{Tvers}}$ [57]	87.72$_{-3.29}$	93.94$_{-1.26}$	89.99$_{-2.51}$	83.13$_{-4.22}$	35.89$_{+8.54}$	44.15$_{-2.56}$	33.49$_{+3.86}$	23.68$_{+2.17}$
+ CBL$_{\text{WCE}}$ [177]	92.57$_{+1.56}$	88.01$_{-7.19}$	89.49$_{-3.01}$	82.49$_{-4.86}$	36.07$_{+8.72}$	33.53$_{-13.18}$	29.85$_{+0.22}$	21.39$_{-0.12}$
+ OHEM [178]	91.01$_{+0.00}$	94.81$_{-0.39}$	92.17$_{-0.33}$	86.94$_{-0.41}$	28.45$_{+1.10}$	46.68$_{-0.03}$	30.58$_{+0.95}$	22.21$_{+0.70}$
+ IntNorm [94]	73.53$_{-17.48}$	85.64$_{-9.56}$	77.45$_{-15.05}$	65.93$_{-21.42}$	16.92$_{-10.43}$	32.84$_{-13.87}$	18.45$_{-11.18}$	11.67$_{-9.84}$

3.4 Model Implementation

A MedISeg model usually consists of many implementation details, and this is especially true nowadays. In practice, each unconcerned transform may have a potential noticeable effect on performance [172]. Therefore, model implementation tricks are crucial for a MedISeg model. Especially for the fair result comparison of experimental results, it is necessary to ensure an adequate explanation of the implementation details. In our work, we choose three categories of the commonly used implementation tricks and explore their segmentation effectiveness. These three categories are: deep supervision (DeepS) [173]; class balance loss (CBL) [174], which includes four loss functions (*i.e.*, CBL$_{\text{Dice}}$ [175], CBL$_{\text{Focal}}$ [176], CBL$_{\text{Tvers}}$ [57] and CBL$_{\text{WCE}}$ [177]); online hard example mining (OHEM) [178]; and instance normalization (IntNorm) [94]. Details of each implementation trick and the experimental results on MedISeg are as follows:

DeepS [173]. DeepS is an auxiliary learning trick, which is proposed in DSN [173] and used for common scene image classification in [179]. This trick is implemented to supervise the backbone network by adding an auxiliary classifier or segmenter on some intermediate hidden layers in a direct or indirect manner [45], [180]–[182]. It can be used to solve problems of the training gradient disappearance or slow convergence speed. For image segmentation, this trick is usually implemented by adding image-level classification loss. In our work, following [180], [181], [183], we first extract feature maps from the last three decoder layers and use a 1×1 convolutional layer to project the lesion mask into the same channel size. Then, output feature maps from different layers are upsampled into the same spatial size as the input image for the segmentation head network [109], [182], [184] by bilinear interpolations.

TABLE 8
Experimental results on 3D LiTS dataset [63] on some model implementation tricks. "DeepS", "CBL", "OHEM", and IntNorm denotes deep supervision [173], class balance loss [174], online hard example mining [178], and instance normalization [94], respectively.

Methods	Recall (%)	Percision (%)	Dice (%)	IoU (%)
3D-UNet [71]	89.33	84.03	86.11	76.44
+ DeepS [173]	90.42$_{+1.09}$	84.02$_{-0.01}$	86.60$_{+0.49}$	77.17$_{+0.73}$
+ CBL$_{\text{Dice}}$ [175]	83.09$_{-6.24}$	71.34$_{-12.69}$	75.51$_{-10.60}$	62.45$_{-13.99}$
+ CBL$_{\text{Focal}}$ [176]	88.37$_{-0.96}$	81.61$_{-2.42}$	84.16$_{-1.95}$	73.16$_{-3.28}$
+ CBL$_{\text{Tvers}}$ [57]	86.60$_{-2.73}$	72.93$_{-11.10}$	78.70$_{-7.41}$	65.52$_{-10.92}$
+ CBL$_{\text{WCE}}$ [177]	91.18$_{+1.85}$	80.27$_{-3.76}$	84.92$_{-1.19}$	74.57$_{-1.87}$
+ OHEM [178]	90.14$_{+0.81}$	85.64$_{+1.61}$	87.35$_{+1.24}$	78.24$_{+1.80}$
+ IntNorm [94]	77.23$_{-12.10}$	87.27$_{+3.24}$	80.94$_{-5.17}$	68.62$_{-7.82}$

CBL [174]. CBL is usually used to learn a general class weight, *i.e.*, the weight for each class is only related to the object category. Compared to some traditional segmentation loss functions (*e.g.*, cross-entropy loss) on the class imbalanced dataset [173], [174], CBL can improve model representational ability. In the used dataset, CBL introduces the effective number of samples to represent the expected volume representations of the selected dataset, and weights different classes by the number of effective samples rather than the number of original samples. In this paper, we mainly explore the effect of four commonly used CBL loss functions for the class imbalanced problem in medical image domain, including Dice loss (CBL$_{\text{Dice}}$) [175], Focal loss (CBL$_{\text{Focal}}$) [176], Tversky loss (CBL$_{\text{Tvers}}$ [57]), and the weighted cross-entropy loss (CBL$_{\text{WCE}}$ [177]).

OHEM [178]. The core idea of OHEM is first to filter out some hard learning samples (*i.e.*, images, objects, and pixels) via the loss function, and these selected hard examples all have a high impact on the recognition tasks [178]. Then, these samples are applied to gradient descent in the model training process. Extensive experimental results on different vision tasks show that OHEM is not only efficient but also performs well on various datasets [66], [67], [185], [186]. In our work, we validate the effectiveness of OHEM on both the 2D and 3D medical datasets with its default setting.

IntNorm [94]. IntNorm is a popular normalization algorithm that is suitable for recognition tasks with higher requirements on a single pixel. In its implementation, every single sample and all elements for a single channel of the sample are taken into consideration when computing the statistic normalization [94], [187], [188]. In the medical image domain, an important reason why IntNorm is used is that the batch size is usually set to a small value (especially for 3D images) during the training process, which makes the use of batch normalization invalid. In this paper, we demonstrate the effect of the IntNorm by replacing it with the intensity normalization [102] in 3D experiments.

Experimental results. Experimental results of using the detailed model implementation tricks on 2D-UNet [43] on 2D ISIC 2018 [72] and 2D CoNIC [73], and 3D-UNet [71] on KiTS19 [136] and 3D LiTS [63] datasets are given in Table 7 and Table 8. In summary, implementing these tricks brings more significant improvements in recognition accuracy on 2D datasets than on 3D datasets. For example, we can observe that compared to the baseline results on 2D-UNet [43] on ISIC 2018 [72], implementing DeepS [173], CBL_{Dice} [175], and $\text{CBL}_{\text{Tvers}}$ [57] can significantly boost model performance. Specifically, $+\text{CBL}_{\text{Dice}}$ has the performance gains of 1.41% Recall, 0.01% Precision, 0.82% Dice, and 0.56% IoU. $+\text{CBL}_{\text{Tvers}}$ has performance of 89.40% Recall, 90.19% Precision, 87.87% Dice, and 86.42% IoU. $+\text{CBL}_{\text{Tvers}}$ has the performance gains of 1.22% Recall, 0.31% Precision, 0.98% Dice, and 0.62% IoU. In contrast, other tricks do not bring comprehensive performance improvements on 2D ISIC 2018 [72] and 2D CoNIC [73]. For example, +DeepS on ISIC 2018 [72] has performance gains of 0.73% Recall, 0.53% Dice, and 0.38% IoU. Frustratingly, $+\text{CBL}_{\text{Focal}}$, $+\text{CBL}_{\text{WCE}}$, and + OHEM [178] reduces model performance on most of these evaluation metrics on both 2D datasets. For example, $+\text{CBL}_{\text{Focal}}$ has the performance reduction of -0.12% Recall, -2.56% Precision, -1.48% Dice, and -1.15% IoU. $+\text{CBL}_{\text{Focal}}$ finally achieves 88.06% Recall, 87.32% Precision, 85.41% Dice, and 84.65% IoU. The same experimental result observations and conclusions can be congruously obtained on 2D-UNet [43] on both ISIC 2018 [72] and CoNIC [73]. In particular, we can see that both DeepS [173] and CBL [174] do not perform well on Recall. Besides, compared to the experimental results on ISIC, the effect on CoNIC is more significant. The reason may be that since these tricks were initially designed for 2D datasets and validated on 2D datasets, they do not work on 3D datasets. Experimental results on 3D-UNet [71] on KiTS19 [136] and 3D LiTS [63] show that these model implementation tricks cannot improve performance remarkably performance improvement. Particularly, on KiTS19 [136], $+\text{CBL}_{\text{Dice}}$ can bring performance gains of 14.56% Recall, 7.52% Dice, and 5.23% IoU, and $+\text{CBL}_{\text{Tvers}}$ can bring performance gains of 8.54% Recall, 3.86% Dice, and 2.17% IoU on setting-ii. OHEM [178] can also useful boost model performance. For example, + OHEM on KiTS19 [136] achieves 28.45% Recall, 30.58% Dice, and 22.21% IoU on setting-ii. The last set of experimental results in Table 7 shows that the effect of IntNorm [94] is not as good as that of the intensity normalization [102] on the 3D KiTS19 dataset. Specifically, replacing IntNorm with the intensity normalization [102] can reduce by the max performance of -17.48% Recall, -13.87% Precision, -15.05% Dice, and -21.42% IoU. The phenomenon that the 2D tricks do not fit on the 3D datasets reminds us that the dataset format needs to be considered in future model designs. In addition, to make the above tricks work in the 3D dataset, we may have to revise them before deploying them during the application process.

3.5 Model Inference

Using implementation tricks is an important strategy for improving the recognition performance [137], [189], [190]. In our work, we mainly explore two kinds of commonly used inference tricks, namely, test time augmentation (TTA) and the model ensemble. The implementation details of these two tricks are as follows:

TTA. TTA is currently a popular data augmentation mechanism in the model inference phase for improving the model accuracy. TTA can be used to improve recognition performance without training, so it has the potential to be a plug-and-play. At the same time, it can improve the ability of model calibration, which is beneficial for visual tasks [191], [192]. In this work, we follow the same image augmentation strategy as in subsection 3.3 by three aspects: 1) implementing the TTA strategy on the baseline model (*i.e.*, $\text{TTA}_{\text{baseline}}$). 2) implementing the test time augmentation GTAug-A on the baseline model under the same data augmentation strategy GTAug-A (*i.e.*, $\text{TTA}_{\text{GTAug-A}}$). 3) implementing the test time augmentation GTAug-B on the baseline model under the same data augmentation strategy GTAug-B (*i.e.*, $\text{TTA}_{\text{GTAug-B}}$).

Ensemble. The model ensemble strategy aims to unite multiple trained models and achieve a multi-model fusion result on the test set based on a certain ensemble mechanism, such that the final result can learn from each of the trained models and improve the overall generalization ability [110], [193]. The commonly used model ensemble methods are voting, averaging, stacking, and non-cross-stacking (blending). In our work, we first independently conduct 5 training sets with different random seeds (*i.e.*, $2022 - 2026$) obtaining 5 different models in fold-validation. Then, we choose the commonly used voting and averaging as the model ensemble strategies [193], termed as EnsVot and EnsAvg, respectively.

Experimental results. Experiments for model inference are carried out on both 2D-UNet [43] and 3D-UNet [71]. The experimental results are given in Table 9 and Table 10. We can observe that these tricks can lead to significant overall performance gains for 2D-UNet [43] on ISIC 2018 [72] and CoNIC [73], but not all of them are beneficial to 3D-UNet [71] on KiTS19 [74] and LiTS [63]. For example,

TABLE 9
Experimental results on model inference tricks. "All" denotes that both $\text{TTA}_{\text{GTAug}}$ and the corresponding model ensemble strategy are used.

Methods	Recall (%)	Percision (%)	Dice (%)	IoU (%)	Recall (%)	Percision (%)	Dice (%)	IoU (%)
2D-UNet [43]	ISIC 2018 [72]				CoNIC [75]			
	88.18	89.88	86.89	85.80	78.12	77.25	77.23	77.58
+ $\text{TTA}_{\text{baseline}}$	88.55$_{+0.37}$	90.29$_{+0.41}$	87.42$_{+0.53}$	86.26$_{+0.46}$	79.13$_{+1.01}$	80.62$_{+3.37}$	79.37$_{+2.14}$	79.41$_{+1.83}$
+ $\text{TTA}_{\text{GTAug-A}}$	88.28$_{+0.10}$	90.46$_{+0.58}$	87.37$_{+0.48}$	86.13$_{+0.33}$	80.19$_{+2.07}$	80.57$_{+3.32}$	80.00$_{+2.77}$	79.86$_{+2.28}$
+ $\text{TTA}_{\text{GTAug-B}}$	90.21$_{+2.03}$	90.94$_{+1.06}$	88.94$_{+2.05}$	87.59$_{+1.79}$	80.04$_{+1.92}$	83.81$_{+6.56}$	81.32$_{+4.09}$	81.22$_{+3.64}$
+ EnsAvg [193]	91.08$_{+2.90}$	89.50$_{-0.38}$	88.52$_{+1.63}$	87.21$_{+1.41}$	79.97$_{+1.85}$	79.90$_{+2.65}$	79.45$_{+2.22}$	79.85$_{+2.27}$
+ EnsVot [193]	91.04$_{+2.86}$	89.41$_{-0.47}$	88.46$_{+1.57}$	87.15$_{+1.35}$	80.22$_{+2.10}$	79.42$_{+2.17}$	79.34$_{+2.11}$	79.73$_{+2.15}$
+ All (TTA-EnsAvg)	88.64$_{+0.46}$	90.96$_{+1.08}$	87.90$_{+1.01}$	86.72$_{+0.92}$	79.98$_{+1.86}$	80.60$_{+3.35}$	79.83$_{+2.60}$	80.15$_{+2.57}$
+ All (TTA-EnsVot)	88.61$_{+0.43}$	90.86$_{+0.98}$	87.87$_{+0.98}$	86.69$_{+0.89}$	79.89$_{+1.77}$	80.56$_{+3.31}$	79.76$_{+2.53}$	80.09$_{+2.51}$
3D-UNet [71]	KiTS19 [74]: settings-i				KiTS19 [74]: settings-ii			
	91.01	95.20	92.50	87.35	27.35	46.71	29.63	21.51
+ $\text{TTA}_{\text{baseline}}$	72.94$_{-18.07}$	97.82$_{+2.62}$	81.38$_{-11.12}$	72.08$_{-15.27}$	19.70$_{-7.65}$	39.60$_{-7.11}$	21.81$_{-7.82}$	15.39$_{-6.12}$
+ $\text{TTA}_{\text{GTAug-A}}$	71.32$_{-19.69}$	97.81$_{+2.61}$	80.22$_{-12.28}$	70.60$_{-16.75}$	17.20$_{-10.15}$	38.28$_{-8.43}$	19.16$_{-10.47}$	13.27$_{-8.24}$
+ $\text{TTA}_{\text{GTAug-B}}$	82.43$_{-8.58}$	94.91$_{-0.29}$	86.65$_{-5.85}$	79.24$_{-8.11}$	10.59$_{-16.76}$	34.69$_{-12.02}$	14.17$_{-15.46}$	9.52$_{-11.99}$
+ EnsAvg [193]	93.00$_{+1.99}$	96.69$_{+1.49}$	94.39$_{+1.89}$	90.02$_{+2.67}$	27.09$_{-0.26}$	54.71$_{+8.00}$	29.15$_{-0.48}$	21.26$_{-0.25}$
+ EnsVot [193]	92.90$_{+1.89}$	96.62$_{+1.42}$	94.31$_{+1.81}$	89.87$_{+2.52}$	27.65$_{+0.30}$	56.65$_{+9.94}$	29.44$_{-0.19}$	21.40$_{-0.11}$
+ All (TTA-EnsAvg)	79.75$_{-11.26}$	98.54$_{+3.34}$	86.86$_{-5.64}$	79.01$_{-8.34}$	24.49$_{-2.86}$	48.17$_{+1.46}$	27.02$_{-2.61}$	19.56$_{-1.95}$
+ All (TTA-EnsVot)	79.03$_{-11.98}$	98.73$_{+3.53}$	86.31$_{-6.19}$	78.28$_{-9.07}$	23.38$_{-3.97}$	50.05$_{+3.34}$	26.42$_{-3.21}$	19.11$_{-2.40}$

TABLE 10
Experimental results on 3D LiTS dataset [63] on model inference tricks. "All" denotes that both $\text{TTA}_{\text{GTAug}}$ and the corresponding model ensemble strategy are used.

Methods	Recall (%)	Percision (%)	Dice (%)	IoU (%)
3D-UNet [71]	89.33	84.03	86.11	76.44
+ $\text{TTA}_{\text{baseline}}$	81.03$_{-8.30}$	85.35$_{+1.32}$	82.32$_{-3.79}$	70.81$_{-5.63}$
+ $\text{TTA}_{\text{GTAug-A}}$	82.32$_{-7.01}$	85.28$_{+1.25}$	83.06$_{-3.05}$	71.45$_{-4.99}$
+ $\text{TTA}_{\text{GTAug-B}}$	84.82$_{-4.51}$	83.48$_{-0.55}$	83.38$_{-2.73}$	72.42$_{-4.02}$
+ EnsAvg [193]	90.21$_{+0.88}$	88.39$_{+4.36}$	88.77$_{+2.66}$	80.73$_{+4.29}$
+ EnsVot [193]	90.12$_{+0.79}$	87.18$_{+3.15}$	88.09$_{+1.98}$	79.61$_{+3.17}$
+ All (TTA-EnsAvg)	84.16$_{-5.17}$	88.23$_{+4.20}$	85.46$_{-0.65}$	75.52$_{-0.92}$
+ All (TTA-EnsVot)	84.16$_{-5.17}$	87.70$_{+3.67}$	85.19$_{-0.92}$	75.05$_{-1.39}$

compared to the experimental results of baseline 2D-UNet on the ISIC 2018 dataset, the max performance gains are 2.90% Recall, 1.08% Precision, 2.05% Dice, and 1.79% IoU. Among these tricks, $\text{TTA}_{\text{GTAug-B}}$ has the max performance improvements, and $\text{TTA}_{\text{GTAug-A}}$ has the min performance improvements on average. In particular, + EnsVot and + EnsAvg have small performance reductions on Precision by -0.38% and -0.47%. Compared to the experimental results on ISIC 2018 [72] without using the data augmentation tricks in the training process, models using data augmentation tricks in both training and testing phases show about the same performance gains. In terms of the model ensemble on ISIC 2018 [72], both EnsAvg [193] and EnsVot [193] can improve the model performance by 0.46% Recall, 1.08% Precision, 1.01% Dice, and 0.92% IoU, and 0.43% Recall, 0.98% Precision, 0.98% Dice, and 0.89% IoU. For the experimental results on 2D-UNet [43] using the CoNIC [73] dataset, we can observe that all values under these evaluation metrics are improved, strongly demonstrating the effect of test time augmentation and the model ensemble tricks on the medical image segmentation task [159], [163], [194], [195]. Compared to the baseline results on KiTS19 [74] and LiTS [63], the experimental results with TTA do not show a significant performance improvement. For example, on KiTS19 [74], + EnsAvg [193] and + EnsVot [193] can boost the model performance by 1.99%, 1.49%, 1.89%, and 2.67% on Recall, Precision, Dice, and IoU on setting-i, respectively. + EnsAvg [193] and + EnsVot [193] can also boost the model performance by 1.89%, 1.42%, 1.81%, and 2.52% on Recall, Precision, Dice, and on setting-i, respectively. In addition, + All can improve the Precision on setting-i and setting-ii. The 3D setting model finally can achieve the best performance of 93.00%, 98.73%, 94.39%, and 90.02% on Recall, Precision, Dice, and IoU on setting-i on KiTS19 [74], respectively. The 3D setting model finally can also achieve the best performance of 27.65%, 56.65%, 29.44%, and 21.40% on Recall, Precision, Dice, and IoU on setting-i on KiTS19 [74], respectively. From the experimental results on 3D-UNet [71] on KiTS19 [74] and LiTS [63], we can make the same observations and reach the same conclusions as in subsection 3.3. In general, the effectiveness of the data augmentation scheme itself needs to be considered in a MedISeg model implementation, at the same time, the state (*e.g.*, modality, distribution, and lesion size) of the dataset is one of the important reasons for determining whether the trick is effective or not [163]–[165], [196].

TABLE 11
Experimental results on post-processing tricks. "ABL-CS" and "RSA" denotes all-but-largest-component-suppression strategy [199] and remove small area strategy [198], respectively.

Methods	Recall (%)	Percision (%)	Dice (%)	IoU (%)	Recall (%)	Percision (%)	Dice (%)	IoU (%)
2D-UNet [43]	ISIC 2018 [72]				CoNIC [75]			
	88.18	89.88	86.89	85.80	78.12	77.25	77.23	77.58
+ ABL-CS [200]	$87.64_{-0.54}$	$90.57_{+0.69}$	$86.99_{+0.10}$	$86.00_{+0.20}$	-	-	-	-
+ RSA [198]	$88.15_{-0.03}$	$89.87_{-0.01}$	$86.91_{+0.02}$	$85.83_{+0.03}$	$78.00_{-0.12}$	$77.33_{+0.08}$	$77.23_{+0.00}$	$77.58_{+0.00}$
3D-UNet [71]	KiTS19 [74]: settings-i				KiTS19 [74]: settings-ii			
	91.01	95.20	92.50	87.35	27.35	46.71	29.63	21.51
+ ABL-CS [200]	$90.04_{-0.97}$	$95.63_{+0.43}$	$92.14_{-0.36}$	$86.81_{-0.54}$	$23.49_{-3.86}$	$49.00_{+2.29}$	$28.23_{-1.40}$	$21.26_{-0.25}$
+ RSA [198]	$90.76_{-0.25}$	$95.33_{+0.13}$	$92.41_{-0.09}$	$87.22_{-0.13}$	$26.82_{-0.53}$	$44.30_{-2.41}$	$29.20_{-0.43}$	$21.36_{-0.14}$

TABLE 12
Experimental results on 3D LiTS dataset [63] on post-processing tricks. "ABL-CS" and "RSA" denotes all-but-largest-component-suppression strategy [199] and remove small area strategy [198], respectively.

Methods	Recall (%)	Percision (%)	Dice (%)	IoU (%)
3D-UNet [71]	89.33	84.03	86.11	76.44
+ ABL-CS [200]	$89.31_{-0.02}$	$87.38_{+3.35}$	$87.79_{+1.68}$	$79.13_{+2.69}$
+ RSA [198]	$89.32_{-0.01}$	$84.19_{+0.16}$	$86.19_{+0.08}$	$76.58_{+0.14}$

3.6 Result Post-Processing

The purpose of the post-processing operations is mainly to improve model performance via non-learnable approaches. For example, the segmentation result can be refined by aggregating the global image information. In this paper, we try to investigate two commonly used result post-processing schemes in the domain of medical image analysis. These schemes are all-but-largest-component-suppression (ABL-CS) [197], and removal of small area (RSA) [198].

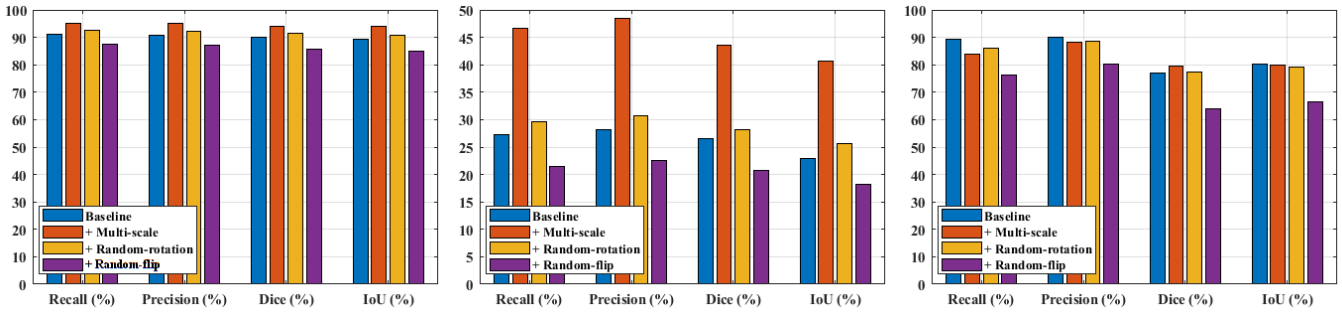
ABL-CS [197]. ABL-CS aims to remove some wrong areas in the segmentation results based on knowledge of the organism's physical properties [199]. For example, for the heart segmentation task, we all know that every person has only one heart, so if there are small segmentation areas in the obtained mask, we need to remove this small areas [200].

RSA [198]. For MedISeg, the imaging protocol is generally unchanged, such that the area of each instance segmentation mask remains unvaried as well. Based on this physical property, we can set a pixel-level threshold to remove some instance masks that are too small (*i.e.*, under the given threshold) in the obtained segmentation masks [198]. In our work, following [201], [202], the threshold is set to 5000 for setting-i, 80 for for setting-ii, 120 for ISIC 2018 [72], and 10 for CoNIC [73] and LiTS [63]. This means that all masks in the segmented area with less than this threshold are eliminated from the final results.

Experimental results. Experiments for these post-processing strategies are carried out on 2D-UNet [43] for ISIC 2018 [72] and CoNIC [73] datasets, and 3D-UNet [71] for KiTS19 [136] and LiTS [63] datasets, respectively. Results are shown in Table 11 and Table 12. We can observe that compared to the baseline 2D-UNet on ISIC 2018 [72], implementing ABL-CS on the baseline model can boost three-quarters of the evaluation metrics. For example, 0.69% Precision, 0.10% Dice, and 0.20% IoU. Implementing RSA on the baseline model can boost 0.02% Dice, and 0.03% IoU. We can also observe that +RSA has a weak effect on performance improvements on ISIC 2018 [72], *e.g.*, the increased model performance is almost always less than 0.1%. Since ABL-CS does not apply to the CoNIC [73] dataset (we cannot assume the number of colon nuclei in each input image), we only implement RSA on CoNIC [73]. We can observe that values under these evaluation metrics have very little changes. The experimental results on 3D-UNet [71] on KiTS19 [136] show that + ABL-CS can improve model performance on Precision by 0.43% and 2.29% on setting-i and setting-ii, respectively. Besides, + RSA only works on setting-i by improving 0.13% Precision. The above experimental results of + ABL-CS and + RSA on the 3D setting on KiTS19 [136] show that these two post-processing schemes are ineffective in improving performance for Recall, Dice, and IoU. The experimental results on LiTS [63] show that these two schemes can improve all evaluation metrics except Recall. These experimental results show that the performance of the same post-processing operation on different datasets and different evaluation metrics is also different. It also shows us that we need to pay attention to the state of the datasets when choosing the post-processing tricks, and to what evaluation metrics we are focusing on.

4 Discussions

In this section, we discuss the potential challenges and problems corresponding to the above experimental tricks. In addition, we also point out the limitations of these tricks. As illustrated in Figure 2, an unabridged MedISeg system can be separated into six main implementation phases, *i.e.*, pre-training model, data pre-processing, data augmentation, model implementation, model inference, and result post-processing. We collected and explored a series of practical yet representative tricks in each phase on the consistent CNNs-based baseline models. The potential challenges we addressed included but not limited to **small dataset learning**, **class imbalance learning**, **multi-modality learning**, and **domain adaptation**.



(a) KiTS19: settings-i **(b)** KiTS19: settings-ii **(c)** LiTS

Fig. 4. The segmentation result illustrations by adding each single data augmentation scheme on 3D KiTS19 [74] and LiTS [63] datasets, respectively. The baseline model is 3D-UNet [71]. (a) the foreground "kidney" and the "background" in KiTS19 [74], (b) the foreground "tumor" and the "background" in KiTS19 [74], (c) the foreground "liver" and the "background" in LiTS [63].

4.1 Challenges

By implementing these collected tricks on the different segmentation baselines with consistent CNNs-based backbones, we found significant differences in performance. Furthermore, most of these tricks empirically do not cause excessive computational overhead, which further emphasizes their potential practical value in the image segmentation task. In our work, it is worth noting that although the differences in performance can be attributed to these tricks on baseline implementation, the performance improvements are latent since some fundamental yet challenging medical image processing problems are intrinsically handled. Despite the different formulations in deployment and appearance, the problems solved by tricks are potentially alike in terms of purposes with advanced approaches. For example, we observe that fine-tuning the existing pre-trained weights can improve model performance by a large margin (*ref.* Table 1 and Table 2) according to most of the evaluation metrics. First, this observation demonstrates that the pre-trained weights on the natural scenes can be transferred into the medical scene via a plain fine-tuning strategy, thereby essentially solves **multi-modality learning** and **domain adaptation** (*i.e.*, cross-domain representations between natural scenarios and medical scenarios) problems. Although these problems can also be solved by some advanced transfer learning [203]–[205] or meta-learning approaches [194], [206], fine-tuning is a cheap solution. Second, it shows that training on more data samples (the pre-training dataset + the fine-tuning dataset) helps to improve the model performance, which corresponds to problems of **small dataset learning** and overfitting. These two problems are accentuated in models with very strong learning capabilities (*e.g.*, the transformer-based models), as such models generally require very large datasets to train.

In addition to the pre-training model, we also can make similar observations and conclusions on the data pre-processing (*ref.* Table 3 and Table 4), the data augmentation (*ref.* Table 5 and Table 6, and Table 11 and Table 12), and the test time augmentation (*ref.* Table 9 and Table 10) tricks. Specifically, the experimental dataset representation ability can be implicitly enhanced via data pre-processing and data augmentation tricks, such that challenges of the insufficient training samples can be alleviated via Patching, OverSam, and ReSam. The problem of **small dataset learning** can be addressed via Patching, and OverSam. The problem of **class imbalance learning** can be addressed via Patching, OverSam, ReSam, and IntesNorm. Fortunately, these categories of tricks do not incur a noticeable computational cost, which is evidence that these tricks are hidden in their implementations. Although in some scenarios, data augmentation and test time augmentation strategies do not improve the model performance, we do believe that this is not because the strategies themselves have difficulty dealing with medical images, but is determined by the dataset specificity. For example, implementing the practical $\text{TTA}_{\text{baseline}}$, and $\text{TTA}_{\text{GTAug}}$ on 3D-UNet baseline on CoNIC [73] does not improve all performance improvement. The reason for this phenomenon may be that the position of human tissues is relatively fixed in the given images (*ref.* Figure 3 (b)). But some of the geometric transformation-based data augmentation schemes will destroy such inductive biases in model learning, resulting in poor segmentation accuracy.

Besides, there is also a significant difference between different individual geometric transformation-based data augmentation schemes. As shown in Figure 4, we visualize the segmentation illustrations of each single data augmentation scheme on 3D on KiTS19 [136] and LiTS [63] datasets. We can observe that multiple scales and rotation improve model performance improvement on some evaluation metrics on KiTS19 [136], while the use of flip markedly reduces the model performance. Results on LiTS [63] show that multi-scale improves only a portion of the evaluation metrics (*e.g.*, Dice), and the remaining other data augmentation schemes did not work. These separate experimental results can validate our hypothesis above, *i.e.*, that data augmentation methods with non-destructive inductive biases (*i.e.*, multi-scale) are more beneficial to images with relatively fixed object positions. Comparatively, the generative adversarial network-based data augmentation can improve dataset representation without destroying such inductive biases, to achieve better performance on KiTS19 [136] (*ref.* Table 5. These experiments remind us that in future data augmentation design, we also need to consider the specific state (*e.g.*, modality, distribution, and lesion size) of the used dataset. In addition, the **class imbalance learning** is also an obvious topic that can be handled by these collected tricks (*ref.* Table 7 and Table 8). In summary, based on the above analysis, our work therefore not only completed

an MedISeg survey, but also is a practical guide for addressing future medical image processing challenges (especially the segmentation-related tasks) including but not limited to **small dataset learning**, **class imbalance learning**, **multi-modality learning**, and **domain adaptation**.

The actual situation of human tissues also needs to be considered when deploying the post-processing tricks on medical images. Natural scene images differ significantly from medical images [203], [207], [208]. For example, implementing the ABL-CS and RSA on KiTS19 [74] both reduce the model Recall (*ref.* Table 5 and Table 11). This is because, for some ISIC images, not one lesion area but also multiple lesion areas are included. The simple implementation of RSA results in the removal of some small but existing lesion areas. For example, most people have two kidneys, but some people (due to surgery, disease, or congenital) have only one; in this case, the use of ABL-CS would reduce the phenomenon of Recall [2], [3], [144].

4.2 Limitations

There are some limitations of this work. Firstly, the collected tricks are limited, *i.e.*, tricks collected and experimented with in our paper are only a finite subset of all the medical image segmentation tricks. This point can be elaborated into the following two aspects:

(1) We surveyed a finite subset of tricks. We only collected a representative number of tricks for the same category of phase. For example, there are many data augmentation schemes, *e.g.*, translating, color jittering, contrast, simulation of low resolution, Gaussian noise injection, mixing images, random erasing, Gaussian blur, mixup, and cutmix [56], [159], [161], [163], [182]. However, considering the practical requirements of medical image processing and the characteristics of the experimental datasets, only a small number of these were used.

(2) The application of the surveyed tricks is limited. In our work, we choose the commonly used 2D-UNet [43] and 3D-UNet [71] as our baseline models, and validated the effectiveness of these tricks on four representative 2D and 3D medical image datasets. However, there are a large number of progressive MedISeg models with different backbones at different levels that were not involved.

(3) The effect of the surveyed tricks is limited. The medical image processing is closely related to clinical practice, and involves many complex challenging problems (*e.g.*, sheltered objects [209], blurred images [210], the long tail problem [211], the incremental learning problem in medical images [212], and the object boundary detection and refinement [2]). We only handle some of the fundamental ones.

5 Conclusions and Future Directions

We collected a bag of MedISeg tricks for different implementation phases, namely, the pre-training model, data pre-processing, data augmentation, model implementation, model inference, and result post-processing. These tricks cover nearly all the common and fundamental schemes used for the medical image segmentation task; other elaborate tricks can be viewed as a more complex combinations of these. In our work, to avoid the performance ambiguity from implementation variations, we experimentally explored the effectiveness of the collected tricks on the consistent 2D-UNet [43] and 3D-UNet [71] baseline models. Witnessed by the experimental results on 2D ISIC 2018 [72], 2D CoNIC [73], [75], [95], 3D KiTS19 [74] and 3D LiTS [63], we explicitly clarified the effect of these tricks. Moreover, based on the surveyed tricks and baseline models, we open-sourced a strong MedISeg repository for both 2D and 3D medical images, where each of its components has the advantage of plug-and-play. Compared to the existing paper-driven segmentation surveys [10], [77], [144], [145], [159], [163], [170], [195] that only blandly focus on advantage and limitation analyses, our work can provide extensive experiments and is more technically operable.

One of the important contributions of our work is to explicitly explore the effect of these collected tricks. Our work can not only promote the subsequent methods to pay attention to tricks but also achieve a fair result comparison. This is potentially necessary, especially at the moment, when the network architecture is becoming more and more sophisticated in the face of some complex tasks, *e.g.*, image segmentation [180], [184], object detection [45], [176], and image generation [77], [112]. Besides, there may be conflicts in the implementation or offsets among tricks in a MedISeg model when we integrate all the commonly used tricks into a unified framework [61], which can provide the empirical and coordinating guidance for the forthcoming segmentation pipelines, including network architectures, training strategies, and loss functions.

In the future, we will work in the following directions:

(1) Survey and develop more tricks on MedISeg. In clinical practice, we are often faced with very complicated situations and MedISeg is a fundamental research topic that is closely integrated with practice. Therefore, it is of great practical value and significance to continue to explore and develop some more advanced MedISeg tricks on the existing basis to meet the requirements of different problems.

(2) Continue to explore the effectiveness of tricks on more methods and datasets. Experimental results on a small number of limited data sets are inevitably biased. Especially when faced with the MedISeg problem, different image types, distributions, and divergence of inner classes can affect the effectiveness of a specific trick. To enable a more comprehensive and fair comparison of experimental results, a thorough trick survey is necessary.

(3) Explore trick-inspired model designs. Although tricks are easily overlooked in the existing publications, the principles and ideas they contain can be used to inspire subsequent work to achieve a cheaper and computationally friendly model design.

(4) Explore attention-based tricks. Recently, the visual transformer framework, with a strong feature representation ability via the multi-head attention mechanism, has received more and more attention in communities of computer vision and medical image analysis. However, due to its complex internal architectures and the immature practice applications (especially in the face of small datasets), the further application of vision transformers is still being explored. Therefore, carrying out trick research on the vision transformer framework would also be valuable

REFERENCES

[1] C. J. Lynch and C. Liston, "New machine-learning technologies for computer-aided diagnosis," *Nature Medicine*, vol. 24, no. 9, pp. 1304–1305, 2018.

[2] R. Wang, S. Chen, C. Ji, J. Fan, and Y. Li, "Boundary-aware context neural network for medical image segmentation," *Medical Image Analysis*, vol. 78, p. 102395, 2022.

[3] R. Azad, N. Khosravi, M. Dehghanmanshadi, J. Cohen-Adad, and D. Merhof, "Medical image segmentation on mri images with missing modalities: A review," *arXiv*, 2022.

[4] G. Wang, M. A. Zuluaga, W. Li, R. Pratt, P. A. Patel, M. Aertsen, T. Doel, A. L. David, J. Deprest, S. Ourselin *et al.*, "Deepigeos: a deep interactive geodesic framework for medical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1559–1572, 2018.

[5] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annual Review of Biomedical Engineering*, vol. 19, p. 221, 2017.

[6] W. Zhu, L. Sun, J. Huang, L. Han, and D. Zhang, "Dual attention multi-instance deep learning for alzheimer's disease diagnosis with structural mri," *IEEE Transactions on Medical Imaging*, vol. 40, no. 9, pp. 2354–2366, 2021.

[7] M. Liu, D. Zhang, and D. Shen, "Relationship induced multi-template learning for diagnosis of alzheimer's disease and mild cognitive impairment," *IEEE Transactions on Medical Imaging*, vol. 35, no. 6, pp. 1463–1474, 2016.

[8] X. Li, H. Chen, X. Qi, Q. Dou, C.-W. Fu, and P. A. Heng, "H-denseunet: hybrid densely connected unet for liver and tumor segmentation from ct volumes," *IEEE Transactions on Medical Imaging*, vol. 37, no. 12, pp. 2663–2674, 2018.

[9] T. Liu, E. Siegel, and D. Shen, "Deep learning and medical image analysis for covid-19 diagnosis and prediction," *Annual Review of Biomedical Engineering*, vol. 24, 2022.

[10] K. Munir, H. Elahi, A. Ayub, F. Frezza, and A. Rizzi, "Cancer diagnosis using deep learning: a bibliographic review," *Cancers*, vol. 11, no. 9, pp. 1235–1246, 2019.

[11] C. Jin, H. Yu, J. Ke, P. Ding, Y. Yi, X. Jiang, X. Duan, J. Tang, D. T. Chang, X. Wu *et al.*, "Predicting treatment response from longitudinal images using multi-task deep learning," *Nature Communications*, vol. 12, no. 1, pp. 1–11, 2021.

[12] D. L. Pham, C. Xu, and J. L. Prince, "Current methods in medical image segmentation," *Annual Review of Biomedical Engineering*, vol. 2, no. 1, pp. 315–337, 2000.

[13] R. H. Taylor, "Computer-integrated interventional medicine: A 30 year perspective," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2020.

[14] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[15] D. Zhang, Y. Sun, Q. Ye, and J. Tang, "Recursive discriminative subspace learning with l1-norm distance constraint," *IEEE Transactions on Cybernetics*, vol. 50, no. 5, pp. 2138–2151, 2018.

[16] X. Lu, W. Wang, C. Ma, J. Shen, L. Shao, and F. Porikli, "See more, know more: Unsupervised video object segmentation with co-attention siamese networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[19] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv*, 2017.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] H. Chen, Q. Dou, L. Yu, J. Qin, and P. A. Heng, "Voxresnet: Deep voxelwise residual networks for brain segmentation from 3d mr images," *NeuroImage*, vol. 170, no. 1, pp. 446–455, 2018.

[23] X. Yang, C. Liu, Z. Wang, J. Yang, H. Le Min, L. Wang, and K. T. T. Cheng, "Co-trained convolutional neural networks for automated detection of prostate cancer in multi-parametric mri," *Medical Image Analysis*, vol. 42, no. 1, pp. 212–227, 2017.

[24] D. Zhang, Y. Wang, L. Zhou, H. Yuan, D. Shen, A. D. N. Initiative *et al.*, "Multimodal classification of alzheimer's disease and mild cognitive impairment," *Neuroimage*, vol. 55, no. 3, pp. 856–867, 2011.

[25] W. Zhang, R. Li, H. Deng, L. Wang, W. Lin, S. Ji, and D. Shen, "Deep convolutional neural networks for multi-modality isointense infant brain image segmentation," *NeuroImage*, vol. 108, pp. 214–224, 2015.

[26] H.-I. Suk, S.-W. Lee, D. Shen, A. D. N. Initiative *et al.*, "Hierarchical feature representation and multimodal fusion with deep learning for ad/mci diagnosis," *NeuroImage*, vol. 101, pp. 569–582, 2014.

[27] D. Zhang, D. Shen, A. D. N. Initiative *et al.*, "Multi-modal multi-task learning for joint prediction of multiple regression and classification variables in alzheimer's disease," *NeuroImage*, vol. 59, no. 2, pp. 895–907, 2012.

[28] F. Shan, Y. Gao, J. Wang, W. Shi, N. Shi, M. Han, Z. Xue, D. Shen, and Y. Shi, "Lung infection quantification of covid-19 in ct images with deep learning," *arXiv*, 2020.

[29] R. Ge, Y. He, C. Xia, C. Xu, W. Sun, G. Yang, J. Li, Z. Wang, H. Yu, D. Zhang *et al.*, "X-ctrsnet: 3d cervical vertebra ct reconstruction and segmentation directly from 2d x-ray images," *Knowledge-Based Systems*, vol. 236, p. 107680, 2022.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[32] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv*, 2017.

[33] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[34] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[35] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3349–3364, 2020.

[36] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[37] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[38] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[39] J. Guo, K. Han, H. Wu, Y. Tang, X. Chen, Y. Wang, and C. Xu, "Cmt: Convolutional neural networks meet vision transformers," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[40] Z. Peng, W. Huang, S. Gu, L. Xie, Y. Wang, J. Jiao, and Q. Ye, "Conformer: Local features coupling global representations for visual recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[41] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "Cvt: Introducing convolutions to vision transformers," in *Pro-*

ceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021.

[42] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[43] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2015.

[44] X. Guan, G. Yang, J. Ye, W. Yang, X. Xu, W. Jiang, and X. Lai, "3d agse-vnet: an automatic brain tumor mri data segmentation framework," *BMC Medical Imaging*, vol. 22, no. 1, pp. 1–18, 2022.

[45] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[46] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[47] D. Zhang, C. Zuo, Q. Wu, L. Fu, and X. Xiang, "Unabridged adjacent modulation for clothing parsing," *Pattern Recognition*, vol. 127, p. 108594, 2022.

[48] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Conference on Information and Knowledge Management (CIKM)*, 2000.

[49] J. Peng, G. Estrada, M. Pedersoli, and C. Desrosiers, "Deep co-training for semi-supervised image segmentation," *Pattern Recognition*, vol. 107, p. 107269, 2020.

[50] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[51] F. Robinet, C. Parera, C. Hundt, and R. Frank, "Weakly-supervised free space estimation through stochastic co-teaching," in *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[52] G. Song and W. Chai, "Collaborative learning for deep neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[53] H. Wei, H. Cao, Y. Cao, Y. Zhou, W. Xue, D. Ni, and S. Li, "Temporal-consistent segmentation of echocardiography with co-learning from appearance and shape," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2020.

[54] A. Bartler, A. Bühler, F. Wiewel, M. Döbler, and B. Yang, "Mt3: Meta test-time training for self-supervised test-time adaption," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

[55] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, "Test-time training with self-supervision for generalization under distribution shifts," in *International Conference on Machine Learning (ICML)*, 2020.

[56] M. Yeung, E. Sala, C.-B. Schönlieb, and L. Rundo, "Unified focal loss: Generalising dice and cross entropy-based losses to handle class imbalanced medical image segmentation," *Computerized Medical Imaging and Graphics*, vol. 95, no. 1, p. 102026, 2022.

[57] S. S. M. Salehi, D. Erdogmus, and A. Gholipour, "Tversky loss function for image segmentation using 3d fully convolutional deep networks," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2017.

[58] T. Athanasiadis, P. Mylonas, Y. Avrithis, and S. Kollias, "Semantic image segmentation and object labeling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 3, pp. 298–312, 2007.

[59] J. Ma, J. Chen, M. Ng, R. Huang, Y. Li, C. Li, X. Yang, and A. L. Martel, "Loss odyssey in medical image segmentation," *Medical Image Analysis*, vol. 71, no. 6, p. 102035, 2021.

[60] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, "How to train your vit? data, augmentation, and regularization in vision transformers," *arXiv*, 2021.

[61] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[62] Y. Lin, Z. Qu, H. Chen, Z. Gao, Y. Li, L. Xia, K. Ma, Y. Zheng, and K. T. Cheng, "Label propagation for annotation-efficient nuclei segmentation from pathology images," *arXiv*, 2022.

[63] P. Bilic, P. F. Christ, E. Vorontsov, G. Chlebus, H. Chen, Q. Dou, C.-W. Fu, X. Han, P. A. Heng, J. Hesser *et al.*, "The liver tumor segmentation benchmark (lits)," *arXiv*, 2019.

[64] W. Zhang, D. Zhang, and X. Xiang, "Cascaded and dual: Discrimination oriented network for brain tumor classification," in *Asian Conference on Machine Learning (ACML)*, 2019.

[65] Y. Li, L. Luo, H. Lin, H. Chen, and P. A. Heng, "Dual-consistency semi-supervised learning with uncertainty quantification for covid-19 lesion segmentation from ct images," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2021.

[66] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[67] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[68] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, pp. 2011–2023, 2020.

[69] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning (ICML)*, 2021.

[70] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "Repvgg: Making vgg-style convnets great again," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[71] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: learning dense volumetric segmentation from sparse annotation," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2016.

[72] N. Codella, V. Rotemberg, P. Tschandl, M. E. Celebi, S. Dusza, D. Gutman, B. Helba, A. Kalloo, K. Liopyris, M. Marchetti *et al.*, "Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic)," *arXiv*, 2019.

[73] S. Graham, M. Jahanifar, Q. D. Vu, G. Hadjigeorghiou, T. Leech, D. Snead, S. E. A. Raza, F. Minhas, and N. Rajpoot, "Conic: Colon nuclei identification and counting challenge 2022," *arXiv*, 2021.

[74] N. Heller, F. Isensee, K. H. Maier-Hein, X. Hou, C. Xie, F. Li, Y. Nan, G. Mu, Z. Lin, M. Han *et al.*, "The state of the art in kidney and kidney tumor segmentation in contrast-enhanced ct imaging: Results of the kits19 challenge," *Medical Image Analysis*, vol. 67, p. 101821, 2021.

[75] S. Graham, M. Jahanifar, A. Azam, M. Nimir, Y.-W. Tsang, K. Dodd, E. Hero, H. Sahota, A. Tank, K. Benes *et al.*, "Lizard: a large-scale dataset for colonic nuclear instance segmentation and classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[76] H. Cao, Y. Wang, J. Chen, D. Jiang, X. Zhang, Q. Tian, and M. Wang, "Swin-unet: Unet-like pure transformer for medical image segmentation," *arXiv*, 2021.

[77] T. Zhou, S. Ruan, and S. Canu, "A review: Deep learning for medical image segmentation using multi-modality fusion," *Array*, vol. 3, no. 1, pp. 4–100, 2019.

[78] A. Sinha and J. Dolz, "Multi-scale self-guided attention for medical image segmentation," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 1, pp. 121–130, 2020.

[79] Y. Zhang, H. Liu, and Q. Hu, "Transfuse: Fusing transformers and cnns for medical image segmentation," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2021.

[80] S. Feng, H. Zhao, F. Shi, X. Cheng, M. Wang, Y. Ma, D. Xiang, W. Zhu, and X. Chen, "Cpfnet: Context pyramid fusion network for medical image segmentation," *IEEE Transactions on Medical Imaging*, vol. 39, no. 10, pp. 3008–3018, 2020.

[81] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning (ICML)*, 2010.

[82] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv*, 2018.

[83] P. Smith, "Bilinear interpolation of digital images," *Ultramicroscopy*, vol. 6, no. 2, pp. 201–204, 1981.

[84] A. Hatamizadeh, Y. Tang, V. Nath, D. Yang, A. Myronenko, B. Landman, H. R. Roth, and D. Xu, "Unetr: Transformers for 3d medical image segmentation," in *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[85] Y. Xie, J. Zhang, C. Shen, and Y. Xia, "Cotr: Efficiently bridging cnn and transformer for 3d medical image segmentation," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2021.

[86] Q. Yu, D. Yang, H. Roth, Y. Bai, Y. Zhang, A. L. Yuille, and D. Xu, "C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[87] X. Wang, S. Han, Y. Chen, D. Gao, and N. Vasconcelos, "Volumetric attention for 3d medical image segmentation and detection," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2019.

[88] J. Zhang, Y. Xie, Y. Wang, and Y. Xia, "Inter-slice context residual learning for 3d medical image segmentation," *IEEE Transactions on Medical Imaging*, vol. 40, no. 2, pp. 661–672, 2020.

[89] X. Liao, W. Li, Q. Xu, X. Wang, B. Jin, X. Zhang, Y. Wang, and Y. Zhang, "Iteratively-refined interactive 3d medical image segmentation with multi-agent reinforcement learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[90] Y. Wu, K. Liao, J. Chen, D. Z. Chen, J. Wang, H. Gao, and J. Wu, "D-former: A u-shaped dilated transformer for 3d medical image segmentation," *arXiv*, 2022.

[91] X. Yan, H. Tang, S. Sun, H. Ma, D. Kong, and X. Xie, "After-unet: Axial fusion transformer unet for medical image segmentation," in *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[92] A. Hatamizadeh, Z. Xu, D. Yang, W. Li, H. Roth, and D. Xu, "Unetformer: A unified vision transformer model and pre-training framework for 3d medical image segmentation," *arXiv*, 2022.

[93] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[94] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, 2015.

[95] S. Graham, Q. D. Vu, S. E. A. Raza, A. Azam, Y. W. Tsang, J. T. Kwak, and N. Rajpoot, "Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images," *Medical Image Analysis*, vol. 58, p. 101563, 2019.

[96] N. Abraham and N. M. Khan, "A novel focal tversky loss function with improved attention u-net for lesion segmentation," in *International Symposium on Biomedical Imaging (ISBI)*, 2019.

[97] R. Azad, M. Asadi Aghbolaghi, M. Fathy, and S. Escalera, "Attention deeplabv3+: Multi-level context attention mechanism for skin lesion segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[98] M. Weigert and U. Schmidt, "Nuclei segmentation and classification in histopathology images with stardist for the conic challenge 2022," *arXiv*, 2022.

[99] M. Böhland, O. Neumann, M. P. Schilling, M. Reischl, R. Mikut, K. Löffler, and T. Scherr, "ciscnet–a single-branch cell instance segmentation and classification network," *arXiv*, 2022.

[100] C.-Y. Lee, H.-C. Chien, C.-P. Wang, H. Yen, K.-W. Zhen, and H.-K. Lin, "Using multi-scale swintransformer-htc with data augmentation in conic challenge," *arXiv*, 2022.

[101] N. Heller, N. Sathianathen, A. Kalapara, E. Walczak, K. Moore, H. Kaluzniak, J. Rosenberg, P. Blake, Z. Rengel, M. Oestreich *et al.*, "The kits19 challenge data: 300 kidney tumor cases with clinical context, ct semantic segmentations, and surgical outcomes," *arXiv*, 2019.

[102] F. Isensee, P. F. Jaeger, S. A. Kohl, J. Petersen, and K. H. Maier-Hein, "nnu-net: a self-configuring method for deep learning-based biomedical image segmentation," *Nature Methods*, vol. 18, no. 2, pp. 203–211, 2021.

[103] F. Isensee and K. H. Maier Hein, "An attempt at beating the 3d u-net," *arXiv*, 2019.

[104] C. Li, Y. Tan, W. Chen, X. Luo, Y. Gao, X. Jia, and Z. Wang, "Attention unet++: A nested attention-aware u-net for liver ct image segmentation," in *IEEE International Conference on Image Processing (ICIP)*, 2020.

[105] X. Li, L. Yu, H. Chen, C.-W. Fu, L. Xing, and P.-A. Heng, "Transformation-consistent self-ensembling model for semisupervised medical image segmentation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 523–534, 2020.

[106] S. Gul, M. S. Khan, A. Bibi, A. Khandakar, M. A. Ayari, and M. E. Chowdhury, "Deep learning techniques for liver and liver tumor segmentation: A review," *Computers in Biology and Medicine*, p. 105620, 2022.

[107] D. T. Kushnure and S. N. Talbar, "Hfru-net: High-level feature fusion and recalibration unet for automatic liver and tumor segmentation in ct images," *Computer Methods and Programs in Biomedicine*, vol. 213, p. 106501, 2022.

[108] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[109] D. Zhang, H. Zhang, J. Tang, X. Hua, and Q. Sun, "Causal intervention for weakly-supervised semantic segmentation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[110] H. Li, G. Jiang, J. Zhang, R. Wang, Z. Wang, W. Zheng, and B. Menze, "Fully convolutional network ensembles for white matter hyperintensities segmentation in mr images," *NeuroImage*, vol. 183, no. 1, pp. 650–665, 2018.

[111] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Feifei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[112] H.-C. Shin, N. A. Tenenholtz, J. K. Rogers, C. G. Schwarz, M. L. Senjem, J. L. Gunter, K. P. Andriole, and M. Michalski, "Medical image synthesis for data augmentation and anonymization using generative adversarial networks," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2018.

[113] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[114] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," in *Deep Learning and Data Labeling for Medical Applications*, 2016, pp. 179–187.

[115] G. Santini, N. Moreau, and M. Rubeaux, "Kidney tumor segmentation using an ensembling multi-stage deep learning approach. a contribution to the kits19 challenge," *arXiv*, 2019.

[116] T. Estienne, M. Vakalopoulou, E. Battistella, A. Carré, T. Henry, M. Lerousseau, C. Robert, N. Paragios, and E. Deutsch, "Deep learning based registration using spatial gradients and noisy segmentation labels," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2020.

[117] N. Ketkar, "Stochastic gradient descent," in *Deep Learning with Python*, 2017, pp. 113–132.

[118] A. Qayyum, A. Lalande, and F. Meriaudeau, "Automatic segmentation of tumors and affected organs in the abdomen using a 3d hybrid model for computed tomography imaging," *Computers in Biology and Medicine*, vol. 127, no. 1, pp. 97–104, 2020.

[119] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *International Conference on 3D Vision (3DV)*, 2016.

[120] X. Wang, R. Zhang, C. Shen, T. Kong, and L. Li, "Dense contrastive learning for self-supervised visual pre-training," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[121] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International Conference on Machine Learning (ICML)*, 2020.

[122] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[123] Z. Zhou, V. Sodha, M. M. Rahman Siddiquee, R. Feng, N. Tajbakhsh, M. B. Gotway, and J. Liang, "Models genesis: Generic autodidactic models for 3d medical image analysis," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2019.

[124] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan, "Billion-scale semi-supervised learning for image classification," *arXiv*, 2019.

[125] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[126] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, "Big transfer (bit): General visual representation

learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[127] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 21 271–21 284.

[128] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 4037–4058, 2020.

[129] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *International Conference on Machine Learning (ICML)*, 2020.

[130] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[131] A. Rajwade, A. Rangarajan, and A. Banerjee, "Image denoising using the higher order singular value decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 4, pp. 849–862, 2012.

[132] A. Taleb, M. Kirchler, R. Monti, and C. Lippert, "Contig: Self-supervised multimodal contrastive learning for medical imaging with genetics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[133] H. Zhao, Y. Li, N. He, K. Ma, L. Fang, H. Li, and Y. Zheng, "Anomaly detection for medical images using self-supervised and translation-consistent features," *IEEE Transactions on Medical Imaging*, vol. 40, no. 12, pp. 3641–3651, 2021.

[134] F. Haghighi, M. R. H. Taher, Z. Zhou, M. B. Gotway, and J. Liang, "Transferable visual words: Exploiting the semantics of anatomical patterns for self-supervised learning," *IEEE Transactions on Medical Imaging*, vol. 40, no. 10, pp. 2857–2868, 2021.

[135] X. Li, X. Hu, X. Qi, L. Yu, W. Zhao, P.-A. Heng, and L. Xing, "Rotation-oriented collaborative self-supervised learning for retinal disease diagnosis," *IEEE Transactions on Medical Imaging*, vol. 40, no. 9, pp. 2284–2294, 2021.

[136] A. Kutikov and R. G. Uzzo, "The renal nephrometry score: a comprehensive standardized system for quantitating renal tumor size, location and depth," *The Journal of Urology*, vol. 182, no. 3, pp. 844–853, 2009.

[137] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr *et al.*, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[138] H. Bao, L. Dong, F. Wei, W. Wang, N. Yang, X. Liu, Y. Wang, J. Gao, S. Piao, M. Zhou *et al.*, "Unilmv2: Pseudo-masked language models for unified language model pre-training," in *International Conference on Machine Learning (ICML)*, 2020.

[139] M. N. Ahmed, S. M. Yamany, N. Mohamed, A. A. Farag, and T. Moriarty, "A modified fuzzy c-means algorithm for bias field estimation and segmentation of mri data," *IEEE Transactions on Medical Imaging*, vol. 21, no. 3, pp. 193–199, 2002.

[140] Y. Zhang, M. Brady, and S. Smith, "Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm," *IEEE Transactions on Medical Imaging*, vol. 20, no. 1, pp. 45–57, 2001.

[141] S. Kotsiantis, D. Kanellopoulos, P. Pintelas *et al.*, "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, no. 1, pp. 25–36, 2006.

[142] S. M. Pizer, P. T. Fletcher, S. Joshi, A. Thall, J. Z. Chen, Y. Fridman, D. S. Fritsch, A. G. Gash, J. M. Glotzer, M. R. Jiroutek *et al.*, "Deformable m-reps for 3d medical image segmentation," *International Journal of Computer Vision*, vol. 55, no. 2, pp. 85–106, 2003.

[143] H. Imai, S. Matzek, T. D. Le, Y. Negishi, and K. Kawachiya, "High resolution medical image segmentation using data-swapping method," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2019.

[144] M. H. Hesamian, W. Jia, X. He, and P. Kennedy, "Deep learning techniques for medical image segmentation: achievements and challenges," *Journal of Digital Imaging*, vol. 32, no. 4, pp. 582–596, 2019.

[145] B. Mainak, K. Venkatanareshbabu, S. Luca, R. E. Damodar, C.-G. Elisa, M. R. Tato, N. Andrew *et al.*, "State-of-the-art review on deep learning in medical imaging," *Frontiers in Bioscience-Landmark*, vol. 24, no. 3, pp. 380–406, 2019.

[146] B. Krawczyk, M. Koziarski, and M. Woźniak, "Radial-based oversampling for multiclass imbalanced data classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 2818–2831, 2019.

[147] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, 2002.

[148] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International Conference on Intelligent Computing (ICIC)*, 2005, pp. 878–887.

[149] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2008.

[150] W. Fithian and T. Hastie, "Local case-control sampling: Efficient subsampling in imbalanced data sets," *Annals of Statistics*, vol. 42, no. 5, p. 1693, 2014.

[151] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 559–563, 2017.

[152] M. Seppä, "High-quality two-stage resampling for 3-d volumes in medical imaging," *Medical Image Analysis*, vol. 11, no. 4, pp. 346–360, 2007.

[153] C. Lartizien, J.-B. Aubin, and I. Buvat, "Comparison of bootstrap resampling methods for 3-d pet imaging," *IEEE Transactions on Medical Imaging*, vol. 29, no. 7, pp. 1442–1454, 2010.

[154] Q. Zhu, B. Du, and P. Yan, "Boundary-weighted domain adaptive neural network for prostate mr image segmentation," *IEEE Transactions on Medical Imaging*, vol. 39, no. 3, pp. 753–763, 2019.

[155] M. Shah, Y. Xiao, N. Subbanna, S. Francis, D. L. Arnold, D. L. Collins, and T. Arbel, "Evaluating intensity normalization on mris of human brain with multiple sclerosis," *Medical Image Analysis*, vol. 15, no. 2, pp. 267–282, 2011.

[156] G. Collewet, M. Strzelecki, and F. Mariette, "Influence of mri acquisition protocols and image intensity normalization methods on texture classification," *Magnetic Resonance Imaging*, vol. 22, no. 1, pp. 81–91, 2004.

[157] R. T. Shinohara, E. M. Sweeney, J. Goldsmith, N. Shiee, F. J. Mateen, P. A. Calabresi, S. Jarso, D. L. Pham, D. S. Reich, C. M. Crainiceanu *et al.*, "Statistical normalization techniques for magnetic resonance imaging," *NeuroImage*, vol. 6, no. 1, pp. 9–19, 2014.

[158] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv*, 2016.

[159] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[160] L. Taylor and G. Nitschke, "Improving deep learning with generic data augmentation," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018.

[161] R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama, "Meta approach to data augmentation optimization," in *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[162] G. Wang, W. Kang, Q. Wu, Z. Wang, and J. Gao, "Generative adversarial network (gan) based data augmentation for palmprint recognition," in *Digital Image Computing: Techniques and Applications (DICTA)*, 2018.

[163] M. Xu, S. Yoon, A. Fuentes, and D. S. Park, "A comprehensive survey of image augmentation techniques for deep learning," *arXiv*, 2022.

[164] A. Zhao, G. Balakrishnan, F. Durand, J. V. Guttag, and A. V. Dalca, "Data augmentation using learned transformations for one-shot medical image segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[165] J. Xu, M. Li, and Z. Zhu, "Automatic data augmentation for 3d medical image segmentation," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2020.

[166] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[167] N. T. Tran, V. H. Tran, N. B. Nguyen, T. K. Nguyen, and N. M. Cheung, "On data augmentation for gan training," *IEEE Transactions on Image Processing*, vol. 30, no. 1, pp. 1882–1897, 2021.

[168] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han, "Differentiable augmentation for data-efficient gan training," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[169] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[170] X. Yi, E. Walia, and P. Babyn, "Generative adversarial network in medical imaging: A review," *Medical Image Analysis*, vol. 58, p. 101552, 2019.

[171] D. Nie, R. Trullo, J. Lian, C. Petitjean, S. Ruan, Q. Wang, and D. Shen, "Medical image synthesis with context-aware generative adversarial networks," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2017.

[172] E. Korot, Z. Guan, D. Ferraz, S. K. Wagner, G. Zhang, X. Liu, L. Faes, N. Pontikos, S. G. Finlayson, H. Khalid *et al.*, "Code-free deep learning for multi-modality medical image classification," *Nature Machine Intelligence*, vol. 3, no. 4, pp. 288–298, 2021.

[173] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial Intelligence and Statistics*, 2015, pp. 562–570.

[174] Y. Zang, C. Huang, and C. C. Loy, "Fasa: Feature augmentation and sampling adaptation for long-tailed instance segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[175] W. R. Crum, O. Camara, and D. L. Hill, "Generalized overlap measures for evaluation and validation in medical image analysis," *IEEE Transactions on Medical Imaging*, vol. 25, no. 11, pp. 1451–1461, 2006.

[176] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[177] T. H. Phan and K. Yamamoto, "Resolving class imbalance in object detection with weighted cross entropy losses," *arXiv*, 2020.

[178] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[179] L. Wang, C. Y. Lee, Z. Tu, and S. Lazebnik, "Training deeper convolutional networks with deep supervision," *arXiv*, 2015.

[180] D. Zhang, H. Zhang, J. Tang, X. Hua, and Q. Sun, "Self-regulation for semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[181] M. Phuong and C. H. Lampert, "Distillation-based training for multi-exit architectures," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[182] D. Zhang, H. Zhang, J. Tang, M. Wang, X. Hua, and Q. Sun, "Feature pyramid transformer," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[183] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Learning a discriminative feature network for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[184] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, "Context encoding for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[185] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[186] C. Liu, L. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[187] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[188] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv*, 2016.

[189] Q. Hou, L. Zhang, M. Cheng, and J. Feng, "Strip pooling: Rethinking spatial pooling for scene parsing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[190] J. Fu, J. Liu, J. Jiang, Y. Li, Y. Bao, and H. Lu, "Scene segmentation with dual relation-aware attention network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 6, pp. 2547–2560, 2020.

[191] D. Shanmugam, D. Blalock, G. Balakrishnan, and J. Guttag, "Better aggregation in test-time augmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[192] N. Moshkov, B. Mathe, A. Kertesz-Farkas, R. Hollandi, and P. Horvath, "Test-time augmentation for deep learning-based cell segmentation on microscopy images," *Scientific Reports*, vol. 10, no. 1, pp. 1–7, 2020.

[193] K. Kamnitsas, W. Bai, E. Ferrante, S. McDonagh, M. Sinclair, N. Pawlowski, M. Rajchl, M. Lee, B. Kainz, D. Rueckert *et al.*, "Ensembles of multiple models and architectures for robust brain tumour segmentation," in *Workshop on Medical Image Computing and Computer Assisted Intervention (Workshop MICCAI)*, 2017.

[194] S. Luo, Y. Li, P. Gao, Y. Wang, and S. Serikawa, "Meta-seg: A survey of meta-learning for image segmentation," *Pattern Recognition*, p. 108586, 2022.

[195] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu *et al.*, "A survey on vision transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[196] D. Shanmugam, D. Blalock, G. Balakrishnan, and J. Guttag, "When and why test-time augmentation works," *arXiv*, 2020.

[197] M. Havaei, A. Davy, D. Warde Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, and H. Larochelle, "Brain tumor segmentation with deep neural networks," *Medical Image Analysis*, vol. 35, pp. 18–31, 2017.

[198] T. Chen and D. Metaxas, "A hybrid framework for 3d medical image segmentation," *Medical Image Analysis*, vol. 9, no. 6, pp. 547–565, 2005.

[199] R. Dorent, A. Kujawa, M. Ivory, S. Bakas, N. Rieke, S. Joutard, B. Glocker, J. Cardoso, M. Modat, K. Batmanghelich *et al.*, "Crossmoda 2021 challenge: Benchmark of cross-modality domain adaptation techniques for vestibular schwnannoma and cochlea segmentation," *arXiv*, 2022.

[200] K. Saikumar, V. Rajesh, N. Ramya, S. H. Ahammad, and G. N. S. Kumar, "A deep learning process for spine and heart segmentation using pixel-based convolutional networks," *Journal of International Pharmaceutical Research*, vol. 46, no. 1, pp. 278–282, 2019.

[201] F. Mahmood, R. Chen, and N. J. Durr, "Unsupervised reverse domain adaptation for synthetic medical images via adversarial training," *IEEE Transactions on Medical Imaging*, vol. 37, no. 12, pp. 2572–2581, 2018.

[202] J. Sutherland, J. Belec, A. Sheikh, L. Chepelev, W. Althobaity, B. J. Chow, D. Mitsouras, A. Christensen, F. J. Rybicki, and D. J. La Russa, "Applying modern virtual and augmented reality technologies to medical images and models," *Journal of Digital Imaging*, vol. 32, no. 1, pp. 38–53, 2019.

[203] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[204] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[205] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International Conference on Artificial Neural Networks*, 2018.

[206] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *arXiv*, 2020.

[207] S. Niu, Y. Liu, J. Wang, and H. Song, "A decade survey of transfer learning (2010–2020)," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 2, pp. 151–166, 2020.

[208] V. Cheplygina, M. de Bruijne, and J. P. Pluim, "Not-so-supervised: a survey of semi-supervised, multi-instance, and transfer learning in medical image analysis," *Medical Image Analysis*, vol. 54, pp. 280–296, 2019.

[209] Z. Zhou, Z. Yang, S. Jiang, F. Zhang, and H. Yan, "Design and validation of a surgical navigation system for brachytherapy based on mixed reality," *Medical Physics*, vol. 46, no. 8, pp. 3709–3718, 2019.

[210] Y. Na, L. Zhao, Y. Yang, and M. Ren, "Guided filter-based images fusion algorithm for ct and mri medical images," *IET Image Processing*, vol. 12, no. 1, pp. 138–148, 2018.

[211] A. G. Roy, J. Ren, S. Azizi, A. Loh, V. Natarajan, B. Mustafa, N. Pawlowski, J. Freyberg, Y. Liu, Z. Beaver *et al.*, "Does your dermatology classifier know what it doesn't know? detecting the

long-tail of unseen conditions," *Medical Image Analysis*, vol. 75, p. 102274, 2022.
[212] P. Kumar and M. M. Srivastava, "Example mining for incremental learning in medical imaging," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018, pp. 48–51.