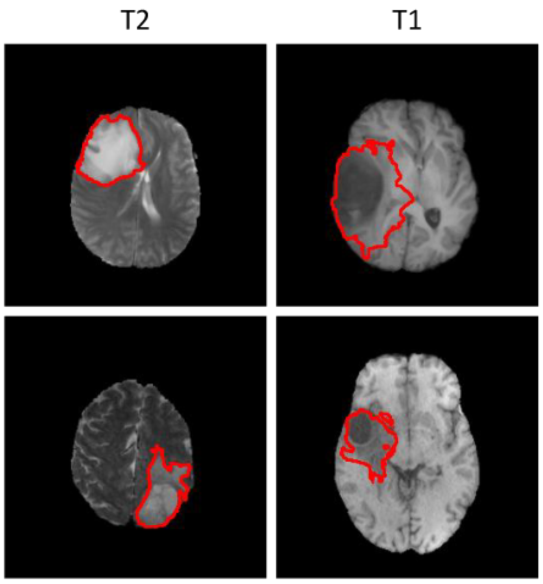
Generalizable Cross-modality Medical Image Segmentation via Style Augmentation and Dual Normalization

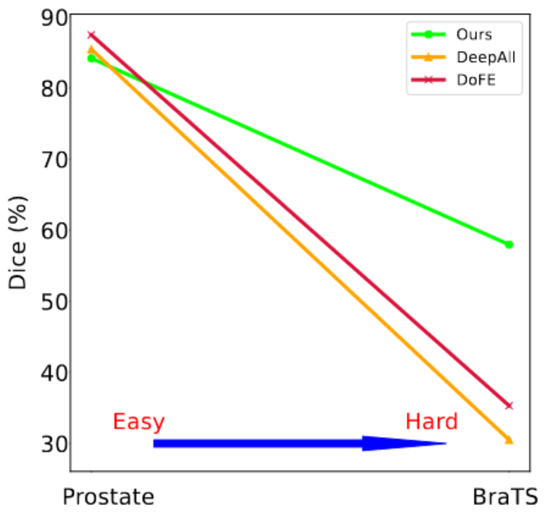
Abstract

For medical image segmentation, imagine if a model was only trained using MR images in source domain, how about its performance to directly segment CT images in target do- main? This setting, namely generalizable cross-modality segmentation, owning its clinical potential, is much more challenging than other related settings, e.g., domain adap- tation. To achieve this goal, we in this paper propose a novel dual-normalization model by leveraging the aug- mented source-similar and source-dissimilar images dur- ing our generalizable segmentation. To be specific, given a single source domain, aiming to simulate the possible appearance change in unseen target domains, we first uti- lize a nonlinear transformation to augment source-similar and source-dissimilar images. Then, to sufficiently ex- ploit these two types of augmentations, our proposed dual- normalization based model employs a shared backbone yet independent batch normalization layer for separate nor- malization. Afterward, we put forward a style-based selec- tion scheme to automatically choose the appropriate path in the test stage. Extensive experiments on three publicly available datasets, i.e., BraTS, Cross-Modality Cardiac, and Abdominal Multi-Organ datasets, have demonstrated that our method outperforms other state-of-the-art domain generalization methods. Code is available at https:// github.com/zzzqzhou/Dual-Normalization.

Keywords: Medical image segmentation, Style augmentation, Cross-modality.



(a) Example slices of T2 & T1.



(b) Performance.

Figure 1. (a) Example slices from BraTS dataset; (b) Comparison of our method with “DeepAll” and “DoFE” methods on the Crosscenter prostate segmentation task and the Cross-modality brain tumor segmentation task.

1 Introduction

In recent years, profound progress in medical image segmentation has been achieved by deep convolutional neural networks [20, 28, 33]. Benefiting from these recent efforts, the accuracy of segmentation on medical images has now been substantially improved. Despite their success, the distribution shift between training (or labeled) and test (or unlabeled) data usually results in a severe performance degeneration during the deployment of trained segmentation models. The reason for distribution shift typically come from different aspects, e.g., different acquisition parameters, various imaging methods or diverse modalities.

To fight against domain shift, several practical settings have been investigated, among which unsupervised domain adaptation (UDA) based segmentation [6, 14, 42]is the most popular one. Specifically, in UDA setting, by assuming that test or unlabeled data could be observed, the model is firstly trained on labeled source domain (i.e., training set) along with unlabeled target domain (i.e., test set), by reducing their domain gap. Then, the trained model is employed to segment the images from target domain. Nevertheless, these UDA based models require that target domain could be observed and even allowed to be trained. This prerequisite sometimes is difficult or infeasible to satisfy in the real application. For example, to protect personal privacy information, target domain (or test set) in some institutes cannot be directly accessed.

To alleviate the requirement of target domain in UDA, we consider a more feasible yet challenging setting, domain generalization (DG), to achieve generalizable medical image segmentation against domain shift. We notice that most existing DG models merely perform well in cross-central setting with small variations between domains, whereas the large domain shift (e.g., cross-modality) is seldom investigated that could largely deteriorate their performance [23, 24, 38].

We now illustrate two types of generalizable segmentation scenarios (i.e., cross-center and cross-modality) to clarify our motivation. Specifically, in Figure 1b, we show results of our method (denoted as "Ours"), "DeepAll" baseline and a state-of-the-art cross-center DG method ("DoFE") [38] on two different DG tasks. The first task is the cross-center prostate segmentation task [25]. As illustrated in Figure 1b, all of these methods achieve rela- tively high Dice scores ($> 80\%$) on this task, and gaps of different methods are quite small. However, when we apply these three methods to BraTS dataset [27], a cross-modality brain tumor segmentation dataset, "DeepAll" and "DoFE" methods show a drastic degradation on Dice scores ($< 40\%$), while our method still achieves a competitive Dice score ($> 50\%$). The reason of this large performance degradation lies in large domain shift. For example, in Figure 1a, we illustrate T2- and T1-weighted images in BraTS dataset [27]. The brain tumors that need to be segmented are delineated with red curves. It is obvious that T2 and T1 modalities show large distinct appearances. Accordingly, we notice that cross-modality DG task is more challenging than cross-center DG task due to the former should tackle larger domain shift. In this paper, we aim to deal with DG task with large domain shift (e.g., cross-modality task), and most previous DG methods are not designed for this.

Our setting owns its clinical meaning. For example, large distribution shift, caused by some unpredictable factors (e.g., interference from light source) during imaging, poses challenge to current generalization methods. Also, in some cases, data scarcity occurs in target domain makes UDA becomes hard to realize. In a nut shell, we intend to develop a robust method for realizing domain distribution shift insensitive modeling.

Based on above motivations, we propose a generalizable cross-modality medical image segmentation method trained on a single source domain (e.g., CT) and directly applied to unseen target domain (e.g., MRI) without any retraining. We notice that in medical images, modality discrepancy usually manifests in gray-scale distribution discrepancy. Being aware of this fact, we wish to simulate possible appearance changes in unseen target domains. In order to tackle this challenging cross-modality DG task, we introduce a module that can randomly augment source do- main into different styles. To be specific, we utilize B ́ezier Curves [29] as transformation functions to generate two groups of images: one group of images are similar to source domain images (i.e., source-similar domain), and another group of images have a large distribution gap with source domain images (i.e., source-dissimilar domain). Then, we introduce a segmentation model with a dual-normalization module to preserve style information of source-similar domain and source-dissimilar domain. Finally, a style-based path selection module is developed to help target domain images select the best normalization path to achieve optimal segmentation results. The main contributions of this paper are summarized as:

• We propose a deep dual-normalization model to tackle a more challenging DG task, i.e., generalizable cross- modality segmentation, that could directly segment the images from unseen target domains without re-training.

• We enhance the diversity of source domain via generating source-similar and source-dissimilar images based on B ́ezier Curves and develop a dual-normalization network for effective exploitation. Besides, we propose the style- based path selection scheme in the test stage.

• Extensive experiments demonstrate our effectiveness. On BraTS dataset, our method achieves the Dice of 54.44% and 57.98% on T2 and T1CE source domains, respectively, which is quite close to UDA [5] (59.30% on T2 source domain) as our upper bound . On both Cross-Modality Cardiac and Abdominal Multi-Organ datasets, our method outperforms the state-of-the-art DG methods.

2 Related works

2.1 Unsupervised Domain Adaptation

The purpose of unsupervised domain adaptation (UDA) is to learn a model with labeled source domain and unlabeled target domain that retains promising performance on target domain [2,3,6,14,26,37,42]. And UDA has attracted considerable attention recently. Some UDA methods utilize distribution alignment in pixel-level and adopt a generative network to narrow the domain gap between source and target domains [6,14,42]. Chang et al. [2] use an adversarial training policy to align feature distribution between source and target domains to maintain semantic feature-level consistency across different domains. Also, Chang et al. [3] separate batch normalization layers for each domain which allows the model to distinguish domain-specific and domain-invariant information. However, in some scenarios, e.g., the data could not be accessed during training due to privacy protection. Target domains are not available in the training process, causing UDA methods could not be directly uesd.

2.2 Domain Generalization

Unlike UDA, domain generalization (DG), by training models purely on source domains, aims to directly generalize to target domains that could not be observed during the training process [4, 8, 11, 13, 32, 39, 40]. Recently, a large number of works on DG tasks have been proposed. Among previous efforts, some methods are designed to learn domain-invariant representations by minimizing the domain discrepancy across multiple source domains [10, 12, 15, 19, 21, 30, 40]. Additionally, some methods use meta-learning, which employs episodic training policy by splitting source domain into meta-train and meta-test domains at each training iteration to simulate domain shift [1,7,17,18,24]. Besides, some methods tackle DG task by modifying normalization layers, e.g. batch normalization (BN) and instance normalization (IN) [9,31,34,35]. For example, Pan et al. [31] propose IBN-Net, which merges IN and BN layers in an unified framework, where IN could maintain invariant representation and BN is able to preserve discriminative features simultaneously. Also, Seo et al. [35] introduce domain specific optimized normalization (DSON) to learn a joint embedding space across all source domains by optimizing domain-specific normalization layers. Segu et al. [34] utilize domain-specific batch normalization layers to collect distribution statistics to model relations between features of source and target domains. There also exists some data augmentation approaches to increase the diversity of source domains for the sake of improving generalization ability on unseen target domains [36,41,43,44].

In medical image analysis, several previous works have studied generalizable medical image segmentation tasks. For example, Zhang et al. [41] propose a deep-stacked transformation approach that employs a series of transformations to simulate domain shift for a specific medical imaging modality. Wang et al. [38] build a domain knowledge pool to store domain-specific prior knowledge and then use domain attributes to aggregate features from different domains. Liu et al. [23] further improve the performance of cross-domain medical image segmentation by combining continuous frequency space interpolation with episodic training strategy. However, most existing DG methods for medical image segmentation work under small domain distribution shift. When large domain shift occurs, they might suffer performance degradation.

3 Methodology

3.1 Definition and Overview

We denote our single source domain as $D^s = \{x_i^s, y_i^s\}_{i=1}^{N^s}$,where s represents the domain ID,x_i^s is the i-th image in the source domain s,y_i^s is the segmentation mask of x_i^s,and N^s is the total number of samples.Our purpose is to train a segmentation model $S_\theta : x- > y$ on source domain D^s,where x and y represent the image set and label set in source domain D^s,S_θ represents the segmentation model and θ is model parameters. We hope the model S_θ can generalize well to unseen target domain D^t.

Specifically, we first propose a style augmentation module with several transformation functions to augment the source domain D^s into source-similar domain D^{ss} and source-dissimilar domain D^{sd}.Then, based on the generated domain D^{ss} and D^{sd},a network equipped with a dual-normalization (DN) module is introduced in our method.We train the DN-based model on D^{ss} and D^{sd} domains.DN can preserve domain style information after model training. Finally, according to domain style information in DN and instance style information of the target domain, we can select the closest normalization statistics in DN to normalize features of target domain and get optimal segmentation results. The diagram of our method is shown in Figure 2. We now discuss the technical details of our method.

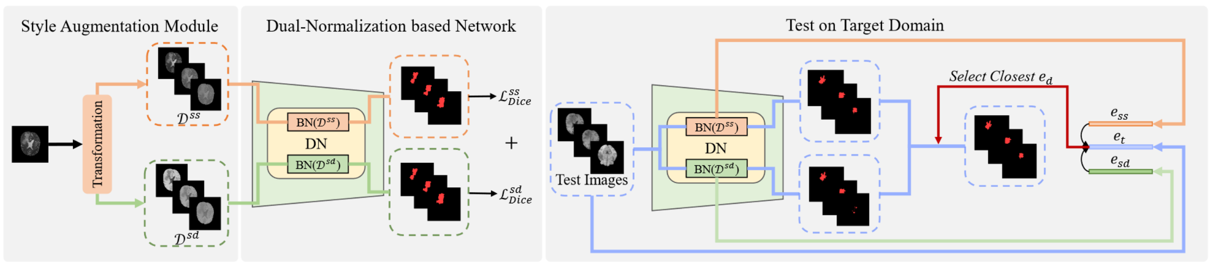


Figure 2. The overall framework of our method. We first employ a style augmentation module to generate source domain into different styles and split them into source-similar domain (D^{ss}) and source-dissimilar domain (D^{sd}). Then, we train a dual-normalization (DN) segmentation network on (D^{ss}) and (D^{sd}). Finally, we test the trained network on target domains by style-based selection module.

3.2 Style Augmentation Module

For generalizable medical image segmentation task, using a single source domain to train the model is very tough. The style bias between different modalities will dramatically degrade the performance. From this perspective, we propose a simple yet effective style augmentation module to generate different stylized images from source domain.

Popular medical image modalities (e.g., X-ray, CT, and magnetic resonance images (MRI)) are usually gray-scale images. As is shown in Figure 1, in T2-weighted MR brain images, whole tumor regions are much brighter than surroundings. In contrast, in T1-weighted MR brain images, foregrounds of whole tumors are darker than background regions. A simple idea to generate different styles is adjusting the gray value distribution of images. Inspired by the previous work Model Genesis [45], we adopt several monotonic non-linear transformation functions to map pixel values of original images to new values. Thus, the operation of

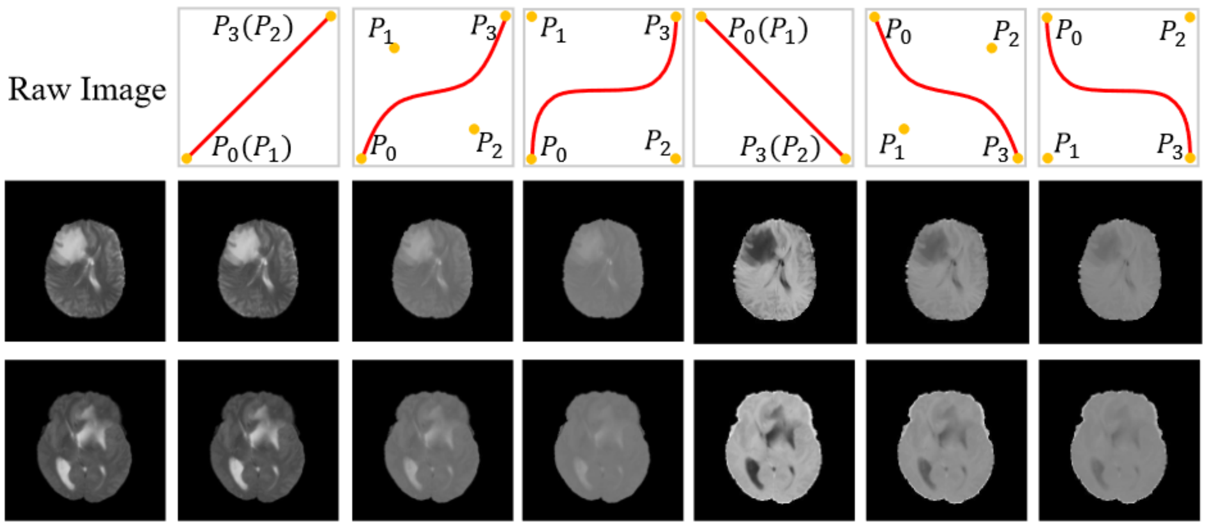


Figure 3. T2 weight augmented MR Brain images and augmented images.

changing gray distribution of images can be realized. Similar to [45], we use smooth and monotonic B´ezier Curve [29]as our transformation function.

B´ezier Curve can be generated from two end points (P0 and P3) and two control points(P1 and P2). The function is defined as follows:

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} P_i (1-t)^{n-i} t^i, n = 3, t \in [0,1],$$

where t is a fractional value along the length of the line. The domain and range of all B´ezier Curves are $[-1, 1]$. In Figure 3, we illustrate the original T2-weighted BraTS image and its augmented images. We set $P_0 = (-1, -1)$ and $P_3 = (1, 1)$ to get an increasing function and opposite to get a decreasing function. When $P_0 = P_1$ and $P_2 = P_3$, the B´ezier Curve is a linear function (shown in Columns 2, 5). Then, we randomly generate another two pairs of control points. Specifically, we set $P_1 = (-v, v)$ and $P_2 = (v, -v)$ $(v \in (0, 1))$.We randomly generate two different vs for each image, so we get two increasing curves (shown in Columns 3 and 4) and two decreasing curves by inversion (shown in Columns 6, 7). Finally, we get 6 non-linear transformation functions (three increasing and three decreasing) for augmentation. In our three tasks, we normalize each sample to $[-1, 1]$. It should be noted that we only perform transformation operations on foreground regions.

Obviously, in gray-scale medical images, monotonically increasing transformation functions have less impact on image style. So we classify these transformed images obtained by increasing transformation functions as images similar to the source domain images, which we call source-similar domain (D^{ss}). On the contrary, these images generated by decreasing transformation functions will be treated as source-dissimilar domain $\left(D^{sd}\right)$.We assume that those images with gray distribution close to source domain images can have good generalization performance on the model trained on D^{ss}, while other images having large distribution gaps with source domain could generalize well on models trained on D^{sd}. We use both domains to train the DN-based model introduced in the next section.

3.3 Dual-normalization based Network

It has been proven that batch normalization [16] can make neural networks easy to capture data bias in their internal latent space [22]. However, data bias captured by neural networks with BN depends on domain distribution, which may degrade generalization ability when tested on novel domains. For our style-augmented images, simply adopting BN will make the model lose domain-specific distribution information of D^{ss} and D^{sd}. As a result, our model may not be able to generalize well on target domains.

To capture different domain distribution information in D^{ss} and D^{ss}, inspired by previous work [2], we adopt two different BN layers in a model to normalize activated features of D^{ss} and D^{ss}, respectively. We call this DualNormalization (DN). The DN module can be written as

$$DN\left(z;d\right) = \gamma_d \frac{z-\mu_d}{\sqrt{\sigma_d^2+\varepsilon}} + \beta_d, \tag{1}$$

where z represents activated features of the model from domain d, d represents the domain label, γ_d and β_d are affine parameters of domain d, (μ_d, σ_d^2) represent the mean and variance of the input feature z from domain d and $\varepsilon > 0$ is a small constant value to avoid numerical instability.

During training, BN layers estimate means and variances of activated features by exponential moving average with update factor α. For DN, they can be written as

$$\bar{\mu}_d^{t+1} = \left(1-\alpha\right)\bar{\mu}_d^t + \alpha\mu_d^t, \tag{2}$$

$$\left(\bar{\sigma}_d^{t+1}\right)^2 = \left(1-\alpha\right)\left(\bar{\sigma}_d^t\right)^2 + \alpha\left(\sigma_d^t\right)^2, \tag{3}$$

where t represents current training iteration, $\bar{\mu}_d^t$ and $(\bar{\sigma}_d^t)$ are the estimated means and variances of domain d at t-th iteration.The estimated means and variances of DN for each domain can be considered as domain distribution information.When testing on target domain, these domain distribution information (μ_d, σ_d^2) can be used to compare with distribution information (μ_t, σ_t^2) of target domains.So that the model can select suitable domain distribution statistics to normalize activated features from target domains.

3.4 Style-Based Path Selection

The DN module allows the model to learn multiple source distribution of D^{ss} and D^{sd}. So that the estimated statistics in DN can be considered as domain style information of D^{ss} and D^{sd}. Hence, we get a light-weight ensemble model, where each domain shares same model parameters except for normalization statistics.

The model with DN module is trained on D^{ss} and D^{sd}. After training, DN module will preserve statistics μ_d and σ_d^2 and affine parameters γ_d and β_d of training data from domain d. Therefore, we will get two series of μ_d and σ_d^2, which can be denoted as

$$e_d = [e_d^1, e_d^2, ..., e_d^L] = [\left(\mu_d^1, (\sigma_d^1)^2\right), \left(\mu_d^2, (\sigma_d^2)^2\right), ..., \left(\mu_d^L, (\sigma_d^L)^2\right)],$$

where d represents the domain label of D^{ss} and D^{sd}, the superscripts $l \in [1, 2, ..., L]$ denote the l-th BN layer in the model. This can be defined as a batch normalization embedding [36] [34] for a certain domain d. In our work, we denote ed as style embedding of domain d.

For a target sample xt, we can capture instance statistics (μ_t, σ_t^2) by forward propagation. The style embedding et of target domain sample can be described as

$$e_t = [e_t^1, e_t^2, ..., e_t^L] = [\left(\mu_t^1, (\sigma_t^1)^2\right), \left(\mu_t^2, (\sigma_t^2)^2\right), ..., \left(\mu_t^L, (\sigma_t^L)^2\right)],$$

Each e_t^1 represents instance style statistics of target domain sample at certain layer l during forward propagation. Once the instance style embedding of target domain sample is available, it is possible to measure similarities of a target domain sample x_t to D^{ss} and D^{sd} by calculating distances between e_t and e_d.

To measure the distance between style embeddings of source domain and target domain, we adopt a symmetric distance function that satisfies the triangle inequality. In our method, we choose the Euclidean Distance. The distance of the l-th layer embeddings can be written as

$$W\left(e_t^l, e_d^l\right) = \|\mu_t^l - \mu_d^l\|_2^2 + \|(\sigma_t^l)^2 - (\sigma_d^l)^2\|_2^2 \tag{4}$$

We measure the distance between target sample x_t and source domain d by summing over the distance between the style embeddings e_t^l and e_d^l of all layers:

$$Dist\left(e_t, e_d\right) = \sum_{l \in 1,2,...,L} W\left(e_t^l, e_d^l\right) \tag{5}$$

Once the distance to each source domain is computed, we can choose the nearest source domain style embedding and affine parameters γ_d and β_d to normalize the input feature z_t of target domain:

$$c = \arg\min_d Dist(e_t, e_d) \tag{6}$$

The normalization on target domain feature z_t is written as

$$Norm(z_t; c) = \gamma_c \frac{z_t - \mu_c}{\sqrt{\sigma_c^2 + \varepsilon}} + \beta_c \tag{7}$$

Since our model shares all parameters except batch normalization layers on D^{ss} and D^{sd}, the model we trained can predict results on target domains by normalized features in Equation (7). We denote $S_\theta\left(\right)$ as our segmentation model, where θ represents parameters in the model except for batch normalization layers. So we can form predicting result on target domain t as $S_\theta\left(z_t^c\right)$, where z_t^c represents normalized target domain features from Equation (7).

3.5 Training Details

As aforementioned, DN module contains two separate batch normalization layers in our model—one is for D^{ss}, and another is for D^{sd}. At the beginning, we augment the source domain into D^{ss} and D^{sd} by style augmentation module. Then, we feed them in the DN-based model to get soft predictions. Afterwards, we optimize the model S_θ by segmentation loss. To overcome the class imbalance issue between relative small-sized foreground and large-sized background, we employ a sum of soft Dice loss of D^{ss} and D^{sd} to train the segmentation network:

$$L_{seg} = L_{Dice}(S_\theta(x_{ss}), y_{ss}) + L_{Dice}(S_\theta(x_{sd}), y_{sd}), \tag{8}$$

where (x_{ss}, y_{ss}) and (x_{sd}, y_{sd}) represent pairs of image and related one-hot ground truth from D^{ss} and D^{sd}, $S_\theta\,()$ yields soft prediction. We show the overall framework of our method in Figure 2.

4 Implementation details

4.1 Comparing with the released source codes

```
import nibabel as nib
import numpy as np
import os
import SimpleITK as sitk
from bezier_curve import bezier_curve
from tqdm import tqdm

modality_name_list = {'t1': '_t1.nii.gz',
          't1ce': '_t1ce.nii.gz',
          't2': '_t2.nii.gz',
          'flair': '_flair.nii.gz'}

def resize_image_itk(itkimage, newSize, resamplemethod=sitk.
    sitkNearestNeighbor):
resampler = sitk.ResampleImageFilter()
originSize = itkimage.GetSize()
originSpacing = itkimage.GetSpacing()
newSize = np.array(newSize, float)
factor = originSize / newSize
newSpacing = originSpacing * factor
newSize = newSize.astype(np.int)
resampler.SetReferenceImage(itkimage)
resampler.SetSize(newSize.tolist())
resampler.SetOutputSpacing(newSpacing.tolist())
resampler.SetTransform(sitk.Transform(3, sitk.sitkIdentity))
resampler.SetInterpolator(resamplemethod)
itkimgResampled = resampler.Execute(itkimage)
return itkimgResampled
```

```

def save_img(slice, label, dir):
np.savez_compressed(dir, image=slice, label=label)

def norm(slices):
max = np.max(slices)
min = np.min(slices)
slices = 2 * (slices - min) / (max - min) - 1
return slices

def nonlinear_transformation(slices):

points_1 = [[-1, -1], [-1, -1], [1, 1], [1, 1]]
xvals_1, yvals_1 = bezier_curve(points_1, nTimes=100000)
xvals_1 = np.sort(xvals_1)

points_2 = [[-1, -1], [-0.5, 0.5], [0.5, -0.5], [1, 1]]
xvals_2, yvals_2 = bezier_curve(points_2, nTimes=100000)
xvals_2 = np.sort(xvals_2)
yvals_2 = np.sort(yvals_2)

points_3 = [[-1, -1], [-0.5, 0.5], [0.5, -0.5], [1, 1]]
xvals_3, yvals_3 = bezier_curve(points_3, nTimes=100000)
xvals_3 = np.sort(xvals_3)

points_4 = [[-1, -1], [-0.75, 0.75], [0.75, -0.75], [1, 1]]
xvals_4, yvals_4 = bezier_curve(points_4, nTimes=100000)
xvals_4 = np.sort(xvals_4)
yvals_4 = np.sort(yvals_4)

points_5 = [[-1, -1], [-0.75, 0.75], [0.75, -0.75], [1, 1]]
xvals_5, yvals_5 = bezier_curve(points_5, nTimes=100000)
xvals_5 = np.sort(xvals_5)

"""
slices, nonlinear_slices_2, nonlinear_slices_4 are source-similar images
nonlinear_slices_1, nonlinear_slices_3, nonlinear_slices_5 are source-dissimilar images
"""
```

```
nonlinear_slices_1 = np.interp(slices, xvals_1, yvals_1)
nonlinear_slices_1[nonlinear_slices_1 == 1] = -1

nonlinear_slices_2 = np.interp(slices, xvals_2, yvals_2)

nonlinear_slices_3 = np.interp(slices, xvals_3, yvals_3)
nonlinear_slices_3[nonlinear_slices_3 == 1] = -1

nonlinear_slices_4 = np.interp(slices, xvals_4, yvals_4)

nonlinear_slices_5 = np.interp(slices, xvals_5, yvals_5)
nonlinear_slices_5[nonlinear_slices_5 == 1] = -1

return slices, nonlinear_slices_1, nonlinear_slices_2, \
nonlinear_slices_3, nonlinear_slices_4, nonlinear_slices_5


def main(data_root, modality, target_root):
list_dir = os.listdir(data_root)
tbar = tqdm(list_dir, ncols=70)
count = 0
for name in tbar:
nib_img = nib.load(os.path.join(data_root, name, name +
    modality_name_list[modality]))
nib_mask = nib.load(os.path.join(data_root, name, name + '_seg.
    nii.gz'))

affine = nib_img.affine.copy()

slices = nib_img.get_fdata()
masks = nib_mask.get_fdata()
masks[masks != 0] = 1

slices = norm(slices)
slices, nonlinear_slices_1, nonlinear_slices_2, \
nonlinear_slices_3, nonlinear_slices_4, nonlinear_slices_5 =
    nonlinear_transformation(slices)

if not os.path.exists(os.path.join(target_root, modality + '_ss')
    ):
```

```python
        os.makedirs(os.path.join(target_root, modality + '_ss'))
        if not os.path.exists(os.path.join(target_root, modality + '_sd')):
        os.makedirs(os.path.join(target_root, modality + '_sd'))

        for i in range(slices.shape[2]):
        """
        Source-Similar
        """
        save_img(slices[:, :, i], masks[:, :, i], os.path.join(target_root, modality + '_ss', 'sample{}_0.npz'.format(count)))
        save_img(nonlinear_slices_2[:, :, i], masks[:, :, i], os.path.join(target_root, modality + '_ss', 'sample{}_1.npz'.format(count)))
        save_img(nonlinear_slices_4[:, :, i], masks[:, :, i], os.path.join(target_root, modality + '_ss', 'sample{}_2.npz'.format(count)))
        """
        Source-Dissimilar
        """
        save_img(nonlinear_slices_1[:, :, i], masks[:, :, i], os.path.join(target_root, modality + '_sd', 'sample{}_0.npz'.format(count)))
        save_img(nonlinear_slices_3[:, :, i], masks[:, :, i], os.path.join(target_root, modality + '_sd', 'sample{}_1.npz'.format(count)))
        save_img(nonlinear_slices_5[:, :, i], masks[:, :, i], os.path.join(target_root, modality + '_sd', 'sample{}_2.npz'.format(count)))
        count += 1

        if __name__ == '__main__':
        data_root = 'D:\MICCAI_BraTS_2018_Data_Training\HGG'
        target_root = 'D:\BraTs2018\\npz_data\\test\\flair'
        modality = 'flair'
        main(data_root, modality, target_root)
```

The data set downloaded from BraTs is a 3D nii medical image. In order to perform better image segmentation, the 3D nii image needs to be converted into a 2D npz image, and the above code realizes the conversion.The data set is divided into D^{ss} and D^{sd} based on Bessel curve

```
import numpy as np
import os
from matplotlib import pyplot as plt
sampled_batch = np.load("D:\ Target\\tice\\t1ce_ss\sample10448_0.npz")
image = sampled_batch["image"].T
plt.imshow(image, cmap='gray')
plt.show()
```

The above code can display two-dimensional npz medical images.

```
import math
import numpy as np
from model.layers import *
from model.dsbn import DomainSpecificBatchNorm2d
import torch.nn as nn
import torch.nn.functional as F
import torch

class MyUpsample2(nn.Module):
def forward(self, x):
return x[:, :, :, None, :, None].expand(-1, -1, -1, 2, -1, 2).reshape(x.size(0), x.size(1), x.size(2)*2, x.size(3)*2)


def normalization(planes, norm='gn', num_domains=None, momentum=0.1):
if norm == 'dsbn':
m = DomainSpecificBatchNorm2d(planes, num_domains=num_domains, momentum=momentum)
elif norm == 'bn':
m = nn.BatchNorm2d(planes)
elif norm == 'gn':
m = nn.GroupNorm(1, planes)
elif norm == 'in':
m = nn.InstanceNorm2d(planes)
else:
raise ValueError('Normalization type {} is not supporter'.format(norm))
return m

class ConvD(nn.Module):
    def __init__(self, inplanes, planes, norm='bn', first=False, num_domains=None, momentum=0.1):
    super(ConvD, self).__init__()

    self.first = first
    self.conv1 = nn.Conv2d(inplanes, planes, 3, 1, 1, bias=True)
    self.bn1   = normalization(planes, norm, num_domains, momentum=momentum)

    self.conv2 = nn.Conv2d(planes, planes, 3, 1, 1, bias=True)
    self.bn2   = normalization(planes, norm, num_domains, momentum=momentum)

    self.conv3 = nn.Conv2d(planes, planes, 3, 1, 1, bias=True)
    self.bn3   = normalization(planes, norm, num_domains, momentum=momentum)

    def forward(self, x, weights=None, layer_idx=None, domain_label=None):

    if weights == None:
    weight_1, bias_1 = self.conv1.weight, self.conv1.bias
    weight_2, bias_2 = self.conv2.weight, self.conv2.bias
    weight_3, bias_3 = self.conv3.weight, self.conv3.bias

    else:
    weight_1, bias_1 = weights[layer_idx+'.conv1.weight'], weights[layer_idx+'.conv1.bias']
    weight_2, bias_2 = weights[layer_idx+'.conv2.weight'], weights[layer_idx+'.conv2.bias']
    weight_3, bias_3 = weights[layer_idx+'.conv3.weight'], weights[layer_idx+'.conv3.bias']

    if not self.first:
    x = maxpool2D(x, kernel_size=2)

    #layer 1 conv, bn
    x = conv2d(x, weight_1, bias_1)
    if domain_label is not None:
    x, _ = self.bn1(x, domain_label)
```

```
else:
x = self.bn1(x)

#layer 2 conv, bn, relu
y = conv2d(x, weight_2, bias_2)
if domain_label is not None:
y, _ = self.bn2(y, domain_label)
else:
y = self.bn2(y)
y = relu(y)

#layer 3 conv, bn
z = conv2d(y, weight_3, bias_3)
if domain_label is not None:
z, _ = self.bn3(z, domain_label)
else:
z = self.bn3(z)
z = relu(z)

return z

class ConvU(nn.Module):
def __init__(self, planes, norm='bn', first=False, num_domains=
    None, momentum=0.1):
super(ConvU, self).__init__()

self.first = first
if not self.first:
self.conv1 = nn.Conv2d(2*planes, planes, 3, 1, 1, bias=True)
self.bn1   = normalization(planes, norm, num_domains, momentum=
    momentum)

self.pool = MyUpsample2()
self.conv2 = nn.Conv2d(planes, planes//2, 1, 1, 0, bias=True)
self.bn2   = normalization(planes//2, norm, num_domains, momentum
    =momentum)

self.conv3 = nn.Conv2d(planes, planes, 3, 1, 1, bias=True)
self.bn3   = normalization(planes, norm, num_domains, momentum=
    momentum)
```

```
self.relu = nn.ReLU(inplace=True)

def forward(self, x, prev, weights=None, layer_idx=None, domain_label=None):

if weights == None:
if not self.first:
weight_1, bias_1 = self.conv1.weight, self.conv1.bias
weight_2, bias_2 = self.conv2.weight, self.conv2.bias
weight_3, bias_3 = self.conv3.weight, self.conv3.bias

else:
if not self.first:
weight_1, bias_1 = weights[layer_idx+'.conv1.weight'], weights[layer_idx+'.conv1.bias']
weight_2, bias_2 = weights[layer_idx+'.conv2.weight'], weights[layer_idx+'.conv2.bias']
weight_3, bias_3 = weights[layer_idx+'.conv3.weight'], weights[layer_idx+'.conv3.bias']

#layer 1 conv, bn, relu
if not self.first:
x = conv2d(x, weight_1, bias_1, )
if domain_label is not None:
x, _ = self.bn1(x, domain_label)
else:
x = self.bn1(x)
x = relu(x)

#upsample, layer 2 conv, bn, relu
y = self.pool(x)
y = conv2d(y, weight_2, bias_2, kernel_size=1, stride=1, padding=0)
if domain_label is not None:
y, _ = self.bn2(y, domain_label)
else:
y = self.bn2(y)
y = relu(y)

```

```python
#concatenation of two layers
y = torch.cat([prev, y], 1)

#layer 3 conv, bn
y = conv2d(y, weight_3, bias_3)
if domain_label is not None:
y, _ = self.bn3(y, domain_label)
else:
y = self.bn3(y)
y = relu(y)

return y


class Unet2D(nn.Module):
def __init__(self, c=1, n=16, norm='bn', num_classes=2,
    num_domains=4, momentum=0.1):
super(Unet2D, self).__init__()

self.convd1 = ConvD(c,      n, norm, first=True, num_domains=
    num_domains, momentum=momentum)
self.convd2 = ConvD(n,    2*n, norm, num_domains=num_domains,
    momentum=momentum)
self.convd3 = ConvD(2*n, 4*n, norm, num_domains=num_domains,
    momentum=momentum)
self.convd4 = ConvD(4*n, 8*n, norm, num_domains=num_domains,
    momentum=momentum)
self.convd5 = ConvD(8*n,16*n, norm, num_domains=num_domains,
    momentum=momentum)

self.convu4 = ConvU(16*n, norm, first=True, num_domains=
    num_domains, momentum=momentum)
self.convu3 = ConvU(8*n, norm, num_domains=num_domains, momentum=
    momentum)
self.convu2 = ConvU(4*n, norm, num_domains=num_domains, momentum=
    momentum)
self.convu1 = ConvU(2*n, norm, num_domains=num_domains, momentum=
    momentum)

self.seg1 = nn.Conv2d(2*n, num_classes, 1)
```

```

for m in self.modules():
if isinstance(m, nn.Conv2d):
nn.init.kaiming_normal_(m.weight, mode='fan_out', nonlinearity='relu')
elif isinstance(m, nn.BatchNorm2d) or isinstance(m, nn.GroupNorm):
nn.init.constant_(m.weight, 1)
nn.init.constant_(m.bias, 0)

def forward(self, x, weights=None, domain_label=None):
if weights == None:
x1 = self.convd1(x, domain_label=domain_label)
x2 = self.convd2(x1, domain_label=domain_label)
x3 = self.convd3(x2, domain_label=domain_label)
x4 = self.convd4(x3, domain_label=domain_label)
x5 = self.convd5(x4, domain_label=domain_label)

y4 = self.convu4(x5, x4, domain_label=domain_label)
y3 = self.convu3(y4, x3, domain_label=domain_label)
y2 = self.convu2(y3, x2, domain_label=domain_label)
y1 = self.convu1(y2, x1, domain_label=domain_label)

y1_pred = conv2d(y1, self.seg1.weight, self.seg1.bias, kernel_size=None, stride=1, padding=0)
else:
x1 = self.convd1(x, weights=weights, layer_idx='module.convd1', domain_label=domain_label)
x2 = self.convd2(x1, weights=weights, layer_idx='module.convd2', domain_label=domain_label)
x3 = self.convd3(x2, weights=weights, layer_idx='module.convd3', domain_label=domain_label)
x4 = self.convd4(x3, weights=weights, layer_idx='module.convd4', domain_label=domain_label)
x5 = self.convd5(x4, weights=weights, layer_idx='module.convd5', domain_label=domain_label)

y4 = self.convu4(x5, x4, weights=weights, layer_idx='module.convu4', domain_label=domain_label)
y3 = self.convu3(y4, x3, weights=weights, layer_idx='module.
    convu3', domain_label=domain_label)
y2 = self.convu2(y3, x2, weights=weights, layer_idx='module.
    convu2', domain_label=domain_label)
y1 = self.convu1(y2, x1, weights=weights, layer_idx='module.
    convu1', domain_label=domain_label)

y1_pred = conv2d(y1, weights['module.seg1.weight'], weights['
    module.seg1.bias'], kernel_size=None, stride=1, padding=0)
predictions = torch.sigmoid(input=y1_pred)

return predictions
```

This paper mainly realizes the direction of domain generalization, but the U-Net model is adopted to segment retinal blood vessel module, and the above code is U-Net model block.

4.2 Experimental environment setup

The Cross-Modality Brain Tumor Segmentation Challenge 2018 dataset (BraTS) [27] is composed of four modalities of MR images, i.e., T2, Flair, T1, and T1CE.

For image preprocessing, I normalize the image to [-1, 1] in intensity values. For the Abdominal MultiOrgan dataset, we crop the volume of each case that contains segmentation targets. Following the previous work in UDA, I randomly select 80% of patient data as training set and 20% as test set, and each slice is resized to 256×256. During training, I perform data augmentations i.e., random crop, random rotation, random scale, etc.

I employ U-Net [33] as our segmentation backbone with replacing all BN layers to our DN module. We implement our model with the PyTorch framework on a Nvidia GTX1650 with 4 GB memory.I train the model for 10 epochs with a batch size of 64. I choose the Adam optimizer with an initial learning rate lr_0 of 4×10^{-3} as our optimizer to train the model. Additionally, for stable training, the learning rate lr is decayed according to the polynomial rule.

I adopt two popularly used evaluation metrics, i.e., Dice coefficient (Dice) and Hausdorff distance (HD). The Dice coefficient measures the overlapping ratio between prediction and ground truth. The higher Dice value, the better segmentation performance. Hausdorff distance is defined between two sets in the metric space. The lower HD value, the better performance.

5 Results and analysis

I ran a total of 10 epochs and finally got a learning rate of 0.000914 and a total loss of 0.055601.

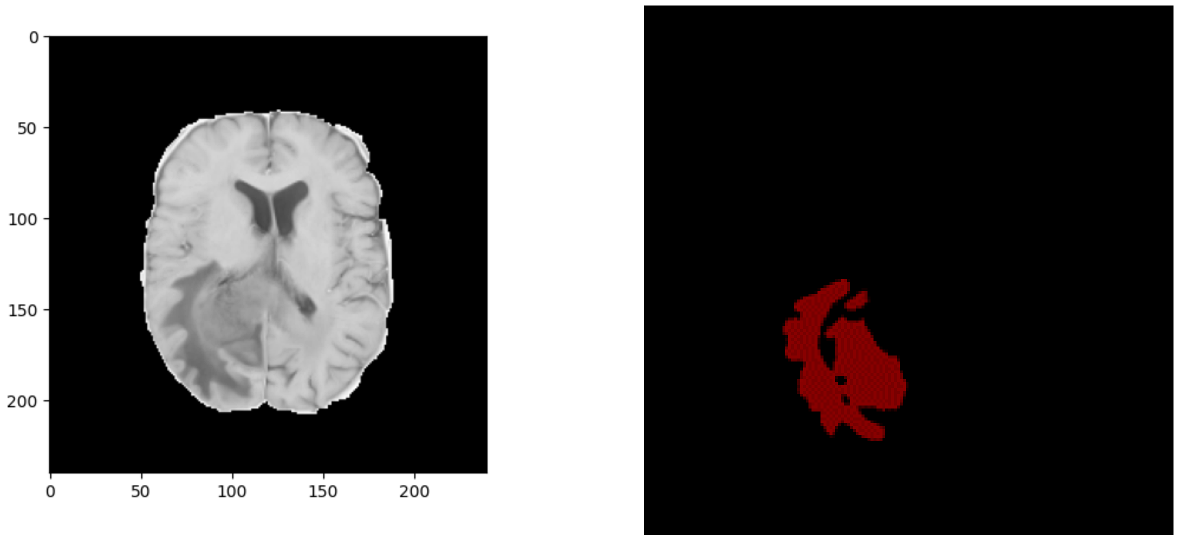


Figure 4. T1 segmentation result

6 Conclusion and future work

In this paper, we first attempt to study the generalizable cross-modality medical image segmentation task. We employ Bézier Curves to augment single source domain D^s into different styles and split them into source-similar domain D^{ss} and source-dissimilar domain D^{sd}. Moreover, we design a dual-normalization module to estimate domain distribution information. During the test stage, we select the nearest feature statistics according to style-embeddings in the dual-normalization module to normalize target domain features for generalization. Our method shows significant improvement compared to other state-of-the-art methods on BraTS, Cardiac and Abdominal Multi-Organ datasets.

References

[1] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *Advances in neural information processing systems*, 31, 2018.

[2] Wei-Lun Chang, Hui-Po Wang, Wen-Hsiao Peng, and Wei-Chen Chiu. All about structure: Adapting structural information across domains for boosting semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1900–1909, 2019.

[3] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 7354–7362, 2019.

[4] Prithvijit Chattopadhyay, Yogesh Balaji, and Judy Hoffman. Learning to balance specificity and invariance for in and out of domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 301–318. Springer, 2020.

[5] Cheng Chen, Qi Dou, Hao Chen, Jing Qin, and Pheng Ann Heng. Unsupervised bidirectional cross-modality adaptation via deeply synergistic image and feature alignment for medical image segmentation. *IEEE transactions on medical imaging*, 39(7):2494–2505, 2020.

[6] Yun-Chun Chen, Yen-Yu Lin, Ming-Hsuan Yang, and Jia-Bin Huang. Crdoco: Pixel-level domain transfer with cross-domain consistency. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1791–1800, 2019.

[7] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. *Advances in neural information processing systems*, 32, 2019.

[8] Yingjun Du, Jun Xu, Huan Xiong, Qiang Qiu, Xiantong Zhen, Cees GM Snoek, and Ling Shao. Learning to learn with variational information bottleneck for domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pages 200–216. Springer, 2020.

[9] Xinjie Fan, Qifei Wang, Junjie Ke, Feng Yang, Boqing Gong, and Mingyuan Zhou. Adversarially adaptive normalization for single domain generalization. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 8208–8217, 2021.

[10] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pages 2551–2559, 2015.

[11] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. Dlow: Domain flow for adaptation and generalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2477–2486, 2019.

[12] Yunye Gong, Xiao Lin, Yi Yao, Thomas G Dietterich, Ajay Divakaran, and Melinda Gervasio. Confidence calibration for domain generalization under covariate shift. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8958–8967, 2021.

[13] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8129–8138, 2020.

[14] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr, 2018.

[15] Yen-Chang Hsu, Zhaoyang Lv, and Zsolt Kira. Learning to cluster in order to transfer across domains and tasks. *arXiv preprint arXiv:1711.10125*, 2017.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[17] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[18] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1446–1455, 2019.

[19] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5400–5409, 2018.

[20] Xiaomeng Li, Hao Chen, Xiaojuan Qi, Qi Dou, Chi-Wing Fu, and Pheng-Ann Heng. H-denseunet: hybrid densely connected unet for liver and tumor segmentation from ct volumes. *IEEE transactions on medical imaging*, 37(12):2663–2674, 2018.

[21] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 624–639, 2018.

[22] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

[23] Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1013–1023, 2021.

[24] Quande Liu, Qi Dou, and Pheng-Ann Heng. Shape-aware meta-learning for generalizing prostate mri segmentation to unseen domains. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part II 23*, pages 475–485. Springer, 2020.

[25] Quande Liu, Qi Dou, Lequan Yu, and Pheng Ann Heng. Ms-net: multi-site network for improving prostate segmentation with heterogeneous mri data. *IEEE transactions on medical imaging*, 39(9):2713–2724, 2020.

[26] Ziwei Liu, Zhongqi Miao, Xingang Pan, Xiaohang Zhan, Dahua Lin, Stella X Yu, and Boqing Gong. Open compound domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12406–12415, 2020.

[27] Bjoern H Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE transactions on medical imaging*, 34(10):1993–2024, 2014.

[28] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016.

[29] Michael E Mortenson. *Mathematics for computer graphics applications*. Industrial Press Inc., 1999.

[30] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5715–5725, 2017.

[31] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 464–479, 2018.

[32] Fengchun Qiao, Long Zhao, and Xi Peng. Learning to learn single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12556–12565, 2020.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[34] Mattia Segu, Alessio Tonioni, and Federico Tombari. Batch normalization embeddings for deep domain generalization. *Pattern Recognition*, 135:109115, 2023.

[35] Seonguk Seo, Yumin Suh, Dongwan Kim, Geeho Kim, Jongwoo Han, and Bohyung Han. Learning to optimize domain specific normalization for domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 68–83. Springer, 2020.

[36] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[37] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2517–2526, 2019.

[38] Shujun Wang, Lequan Yu, Kang Li, Xin Yang, Chi-Wing Fu, and Pheng-Ann Heng. Dofe: Domain-oriented feature embedding for generalizable fundus image segmentation on unseen datasets. *IEEE Transactions on Medical Imaging*, 39(12):4237–4248, 2020.

[39] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2100–2110, 2019.

[40] Sergey Zakharov, Wadim Kehl, and Slobodan Ilic. Deceptionnet: Network-driven domain randomization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 532–541, 2019.

[41] Ling Zhang, Xiaosong Wang, Dong Yang, Thomas Sanford, Stephanie Harmon, Baris Turkbey, Bradford J Wood, Holger Roth, Andriy Myronenko, Daguang Xu, et al. Generalizing deep learning for medical image segmentation to unseen domains via deep stacked transformation. *IEEE transactions on medical imaging*, 39(7):2531–2540, 2020.

[42] Yiheng Zhang, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei. Fully convolutional adaptation networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6810–6818, 2018.

[43] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Deep domain-adversarial image generation for domain generalisation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13025–13032, 2020.

[44] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 561–578. Springer, 2020.

[45] Zongwei Zhou, Vatsal Sodha, Md Mahfuzur Rahman Siddiquee, Ruibin Feng, Nima Tajbakhsh, Michael B Gotway, and Jianming Liang. Models genesis: Generic autodidactic models for 3d medical image analysis. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part IV 22*, pages 384–393. Springer, 2019.