Multi-Organ Segmentation with RepVGG-UNet

Tang Zhe[1]

[1]Keya Medical

zhet@keyamedical.com

Abstract. In this paper, we present a novel and efficient 2D segmentation algorithm on abdomen 3D multi-organ segmentation. Our method is based on the MONAI platform and implements our designed network with the RepVGG backbone. To balance the accuracy, memory usage, and running time, we select a 2D algorithm *RepVGG-UNet* rather than 3D. Our algorithm gets similar accuracy but with minor GPU memory usage and lower running time compare with 3D UNet based segmentation algorithm.

Keywords: 2D segmentation, RepVGG, MONAI

1. Introduction

Multi-organ segmentation in medical images is a common and significant technique for many clinical applications, especially in radiology, surgery. FLARE21 competition provides more than 500 abdominal cases from 12 medical centers, which is very useful for abdominal organ segmentation and vital in promoting multi-organ segmentation.

In this competition, according to the rules, not only accuracy but also efficiency should be considered. Thus, we tried serval methods to get a better result with balanced accuracy, memory usage, and running time. The 3D segmentation algorithm is most used and can get high accuracy with the abdomen organ. However, it uses significant GPU memory with patch-wise inference strategy and long-running time by sliding window step. To avoid these disadvantages, we use the 2D segmentation algorithm RepVGG-UNet instead of 3D. The 2D algorithm can get similar accuracy with smaller GPU memory and lower running time.

2. Method

Our method uses the 2D model RepVGG as our backbone to extract features and use UNet decoder architecture to design our model. We use ‘B2g4’ version structure of RepVGG.

Figure 1 illustrates the applied 2D RepVGG-UNet, where a U-Net [1] architecture is adopted.

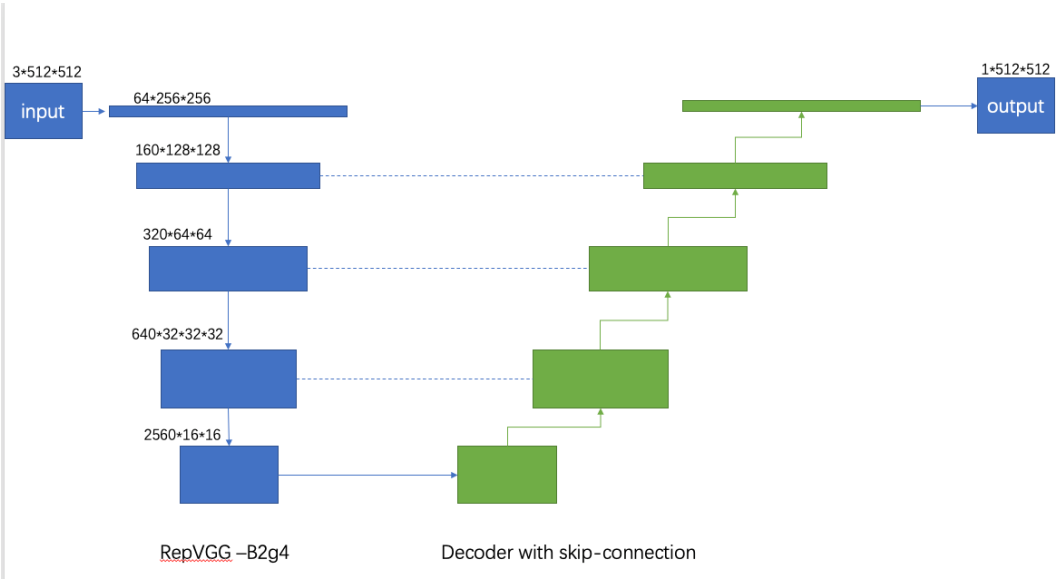


Figure 1 Network architecture

2.1. Preprocessing

The baseline method includes the following preprocessing steps:

- Resampling: firstly, we calculate the spacing of all cases, and use median value as the standard spacing, then we apply respacing with 3D trilinear interpolation on all datasets, 3D trilinear interpolation on all labels.

- Intensity normalization method: our network input has three channels. Thus we normalize data into three different target intensity range and concatenate them into 3-channel input data. The three methods as shown in follow
  - calculate maximum, minimum, mean, standard deviation foreground intensity of 4 classes target region with both training and validation data, the normalize image as:

$$clip\_low = minimum + (maximum - minimum) * 0.995$$
$$clip\_high = minimum + (maximum - minimum) * 0.5$$
$$image[image < clip\_low] = clip\_low$$
$$image[image > clip\_high] = clip\_high$$
$$image = (image - mean) / std$$

  - Normalize image by using default window level and window width, here we set WW= 40Hu, WL= 450Hu, then we have:

$$image[image > 265] = 265$$
$$image[image < -185] = -185$$
$$image = (image - image.min()) / (image.max() - image.min())$$

  - Directly normalize the image by the z-score method:

$$Image = (image - image.mean()) / image.std()$$

- Cropping strategy: Our 2D algorithm does not do any cropping but uses the whole slice as input.

2.2. Proposed Method

- Network architecture details: backbone use 2D RepVGG network with ‘B2g4’ type, features of each layer are:

  Input: 3*512*512

  Layer0: 64*256*256

  Layer1: 160*128*128

  Layer2: 320*64*64

Layer3: 640*32*32

Layer4: 2560*16*16

Dropout = 0.5

Decoder use UNet decoder with skip connection, we use nn.COnvTranspose2d operator instead of upsampling.

- Loss function: we use the summation between Dice loss and cross entropy loss because it has been proved to be robust in various medical image segmentation tasks.
- Number of model parameters: 175843957
- Number of flops: 658304794624

2.3. Post-processing

There is no post-processing step in our method.

3. Dataset and Evaluation Metrics

3.1. Dataset

- A short description of the dataset used:
- The dataset used of FLARE2021 is adapted from MSD [4] (Liver[5], Spleen, Pancreas), NIH Pancreas [6][7][8], KiTS[9][10], and Nanjing University under the license permission. For more detail information of the dataset, please refer to the challenge website and [11].

- Details of training / validation / testing splits:
  The total number of cases is 511. An approximate 70%/10%/20% train/validation/testing split is employed resulting in 361 training cases, 50 validation cases, and 100 testing cases. The detail information is presented in Table 1.

Table 1 Data splits of FLARE2021.

Data Split	Center	Phase	#Num.
Training (361 cases)	The National Institutes of Health Clinical Center	portal venous phase	80
	Memorial Sloan Kettering Cancer Center	portal venous phase	281
Validation (50 cases)	Memorial Sloan Kettering Cancer Center	portal venous phase	5
	University of Minnesota	late arterial phase	25
	7 Medical Centers	various phases	20
Testing (100 cases)	Memorial Sloan Kettering Cancer Center	portal venous phase	5
	University of Minnesota	late arterial phase	25
	7 Medical Centers	various phases	20
	Nanjing University	various phases	50

3.2. Evaluation Metrics

- Dice Similarity Coefficient (DSC)
- Normalized Surface Distance (NSD)
- Running time
- Maximum used GPU memory (when the inference is stable)

4. Implementation Details

4.1. Environments and requirements

The environments and requirements of the baseline method is shown in Table 2.

Table 2 Environments and requirements.

Windows/Ubuntu version	Ubuntu 20.04 LTS
CPU	Intel(R) Core(TM) i9-10920X CPU @ 3.50GHz
RAM	16×4GB;
GPU	Nvidia Geforce RTX 3090
CUDA version	11.1
Programming language	Python3.8
Deep learning framework	Pytorch (Torch 1.9.0, torchvision 0.10.0)
Specification of dependencies	MONAI
(Optional) code is publicly available at	

4.2. Training protocols

The training protocols of the baseline method is shown in Table 3.

Table 3 Training protocols.

Data augmentation methods	Rotations, scaling, Rand2DElastic.
Initialization of the network	"he" normal initialization
Patch sampling strategy	We don't use patch but with 512*512 slice with 3 channel input.
Batch size	48
Patch size	3*512*512
Total epochs	1000
Optimizer	AdamW with default value
Initial learning rate	0.01
Learning rate decay schedule	poly learning rate policy: ReduceLROnPlateau scheduler with patience = 10, factor = 0.9
Stopping criteria,and optimal model selection criteria	Stopping criterion is reaching the maximum number of epoch (1000).
Training time	48 hours
CO_2eq[1]	

4.3. Testing protocols

- Pre-processing steps of the network inputs: The same strategy is applied as training steps.
- Post-processing steps of the network outputs: No post-processing step is used.

5. Results

5.1. Quantitative results for 5-fold cross validation. (Optional)

We didn't perform 5-fold cross validation but only test with single fold. We select 72 cases from training set as our self-validation dataset. Table 4 self-validation result shows the result.

Table 4 self-validation result

Case	DSC_1	NSD-1mm_1	DSC_2	NSD-1mm_2	DSC_3	NSD-1mm_3	DSC_4	NSD-1mm_4
train_000.nii.gz	0.98265133	0.894940521	0.960920889	0.914636357	0.984037232	0.960695963	0.814890651	0.617548122
train_005.nii.gz	0.986605228	0.920294426	0.977806856	0.964278282	0.987154466	0.971767982	0.686875054	0.237093153
train_012.nii.gz	0.983372395	0.912808236	0.962401272	0.926267901	0.972864634	0.945995035	0.863347119	0.665841106
train_023.nii.gz	0.980177591	0.890456284	0.978313086	0.968110347	0.971794217	0.907715871	0.830775007	0.484304624
train_029.nii.gz	0.986255267	0.932111419	0.978049725	0.980152604	0.988222664	0.975005961	0.755833772	0.519283625
train_034.nii.gz	0.926269358	0.686832942	0.977371512	0.96555567	0.991862994	0.996119195	0.849779142	0.696935647
train_036.nii.gz	0.982600092	0.922721318	0.955860709	0.922769532	0.988560265	0.995505727	0.618508777	0.435605345
train_043.nii.gz	0.955935218	0.677320988	0.964346053	0.883525939	0.941973609	0.775055263	0.885040629	0.690873476
train_058.nii.gz	0.97739905	0.89725704	0.932452221	0.801813794	0.965436915	0.934135476	0.632251668	0.414111934
train_059.nii.gz	0.985408681	0.925753082	0.966582024	0.879098378	0.9787298	0.956502036	0.662539428	0.355260237
train_062.nii.gz	0.976359575	0.865155654	0.898323133	0.754472393	0.98838746	0.991323356	0.744318837	0.487751165
train_063.nii.gz	0.977194807	0.844985496	0.981260113	0.983391857	0.983523771	0.982057689	0.813496389	0.566722937
train_064.nii.gz	0.956221632	0.802822611	0.971104806	0.946465307	0.954662256	0.92886027	0.855158228	0.741934977
train_072.nii.gz	0.978397343	0.906876649	0.978386075	0.957472298	0.975588924	0.923969557	0.83954795	0.580855369
train_074.nii.gz	0.977871981	0.908807846	0.942402496	0.853578278	0.975359268	0.966110023	0.749311295	0.630035429
train_076.nii.gz	0.978222998	0.876892313	0.923141319	0.759750118	0.962395914	0.904184875	0.580338401	0.277912625
train_077.nii.gz	0.975327171	0.820087018	0.973749098	0.946083465	0.988933486	0.983344276	0.689845137	0.401512695
train_080.nii.gz	0.985800997	0.944602611	0.973318752	0.92153527	0.980567684	0.962777863	0.767719385	0.569372978
train_081.nii.gz	0.981468238	0.914749362	0.953343744	0.863592832	0.889314079	0.788166931	0.562424377	0.355307338
train_082.nii.gz	0.989028881	0.942627444	0.951222392	0.864668647	0.986716381	0.963242823	0.811564997	0.444477521
train_083.nii.gz	0.969897292	0.848103397	0.978838404	0.948190573	0.985557628	0.983958928	0.647517782	0.396173924
train_091.nii.gz	0.986115609	0.942538685	0.97192386	0.940481713	0.982054963	0.946229013	0.829673455	0.565788905
train_105.nii.gz	0.975617744	0.881026459	0.978815542	0.973243654	0.95440416	0.859857694	0.889997794	0.728015632
train_107.nii.gz	0.975165579	0.901628577	0.975525333	0.979533712	0.967930492	0.92027361	0.779715024	0.538294728
train_108.nii.gz	0.982051817	0.916902751	0.969443837	0.911397331	0.968775281	0.937833957	0.740751737	0.510575959
train_118.nii.gz	0.97850529	0.860277772	0.981646032	0.965476801	0.966236524	0.938915281	0.811618853	0.556945643
train_119.nii.gz	0.989165243	0.947765118	0.975727825	0.976322825	0.983664768	0.97022145	0.679670383	0.490150477
train_124.nii.gz	0.990263784	0.956020002	0.972630112	0.962521553	0.987680042	0.963177774	0.809464092	0.544298437
train_129.nii.gz	0.98961502	0.949949103	0.982915466	0.986643929	0.991816106	0.997430188	0.876670126	0.689177881
train_133.nii.gz	0.973939782	0.865736635	0.939221519	0.910274485	0.987929438	0.98687139	0.763833358	0.555904442
train_138.nii.gz	0.986420432	0.932291625	0.981728281	0.977082894	0.990973036	0.985803751	0.858100993	0.668050277
train_142.nii.gz	0.985362613	0.926344147	0.976106477	0.96803326	0.980315365	0.948549852	0.798713325	0.614193088
train_143.nii.gz	0.979363503	0.868640319	0.944447892	0.857152966	0.977214591	0.943567771	0.832623588	0.614852718
train_146.nii.gz	0.978608809	0.907509516	0.971194411	0.955762293	0.96367983	0.891562176	0.827028734	0.635945931
train_155.nii.gz	0.988468945	0.947428872	0.983418273	0.966001021	0.971381493	0.934059647	0.738527673	0.487586139
train_166.nii.gz	0.981957765	0.895091625	0.949078718	0.862844351	0.970338314	0.904285463	0.706596177	0.465467117
train_171.nii.gz	0.975487867	0.831419063	0.964632885	0.89298926	0.980009077	0.956996725	0.758569054	0.430218651
train_172.nii.gz	0.984233023	0.93185122	0.976922042	0.979298651	0.978973509	0.978888186	0.811704016	0.645707155
train_178.nii.gz	0.984107387	0.906904795	0.966634133	0.949012306	0.981485704	0.948043743	0.802895337	0.417884325
train_185.nii.gz	0.984463539	0.905664992	0.969497625	0.900156798	0.989433655	0.979342995	0.858707328	0.635862642
train_189.nii.gz	0.947084529	0.705616119	0.961646862	0.859822116	0.96854035	0.887050511	0.803042154	0.615563263
train_201.nii.gz	0.985200909	0.9368725	0.965094386	0.906355235	0.977905891	0.940718522	0.751917949	0.458746511
train_208.nii.gz	0.987802875	0.939489521	0.977685451	0.966315627	0.960683284	0.854844084	0.878083092	0.701605524
train_210.nii.gz	0.987226908	0.931960539	0.952512952	0.862168383	0.984573988	0.955514515	0.77283598	0.421645415
train_218.nii.gz	0.939770669	0.663904193	0.880743921	0.72957369	0.969672224	0.937530465	0.521180076	0.381616574
train_226.nii.gz	0.977283368	0.886029245	0.909196077	0.723776689	0.971904909	0.926095351	0.443061697	0.223807099
train_227.nii.gz	0.987648618	0.936495648	0.970521881	0.950638141	0.967097608	0.874659387	0.725240348	0.497034085
train_231.nii.gz	0.944591505	0.790017568	0.949750647	0.878588069	0.791447389	0.647311045	0.691942023	0.548242076
train_235.nii.gz	0.976716893	0.91053421	0.970919239	0.965835908	0.984386718	0.981637523	0.764647739	0.630506377
train_247.nii.gz	0.989710137	0.936988501	0.975886658	0.963817027	0.987156594	0.94952741	0.751300728	0.578121296
train_248.nii.gz	0.978960971	0.905929829	0.953906956	0.868590264	0.970853727	0.956855613	0.651501715	0.48559003
train_251.nii.gz	0.938446494	0.761241374	0.950138548	0.902042518	0.974675431	0.941445133	0.623519774	0.468913089
train_252.nii.gz	0.976173054	0.877185399	0.973898424	0.951063627	0.97270788	0.912700482	0.889140716	0.712823164
train_257.nii.gz	0.978947231	0.874863663	0.97722447	0.960195972	0.983409103	0.944767405	0.795390594	0.588863521
train_261.nii.gz	0.964060134	0.849244027	0.868903117	0.744315002	0.94626859	0.866614529	0.687090532	0.424278307
train_263.nii.gz	0.967278901	0.862414111	0.922215923	0.781223979	0.985126922	0.982331959	0.64658394	0.456513094
train_267.nii.gz	0.984323494	0.920564099	0.940803701	0.836088557	0.985958086	0.96767547	0.659327434	0.44405886
train_278.nii.gz	0.976587189	0.875777824	0.948671178	0.820069599	0.943488795	0.836674649	0.675091143	0.488626489
train_279.nii.gz	0.975762824	0.909612752	0.896791211	0.715202652	0.973417703	0.960367194	0.795973444	0.649238166
train_285.nii.gz	0.982652171	0.928999577	0.975975092	0.959254765	0.979587602	0.957729819	0.643341018	0.550443217
train_287.nii.gz	0.980242116	0.933500754	0.964821506	0.934904546	0.963851895	0.943436264	0.791157238	0.712291898
train_288.nii.gz	0.986053067	0.944207755	0.976368637	0.974703561	0.975275062	0.93306288	0.834223557	0.697049722
train_294.nii.gz	0.954646619	0.754954526	0.966760825	0.960128527	0.937096451	0.882748425	0.16298476	0.212598592
train_297.nii.gz	0.980809049	0.928015435	0.97116308	0.933617854	0.963551359	0.949711069	0.841042046	0.675401007
train_309.nii.gz	0.976700599	0.889340117	0.981824541	0.988721102	0.968720736	0.912300726	0.831057722	0.663578181
train_315.nii.gz	0.979192408	0.902252822	0.975014048	0.95679082	0.969170537	0.971887277	0.828146445	0.696740033
train_316.nii.gz	0.981994989	0.904646691	0.979278749	0.954207864	0.964457224	0.902882359	0.772887788	0.567461478
train_322.nii.gz	0.961651635	0.820436622	0.941525533	0.785083399	0.967689601	0.912927599	0.824055212	0.667723542
train_340.nii.gz	0.978908984	0.872719199	0.973363782	0.939083234	0.972745727	0.949331382	0.77215405	0.552019423
train_356.nii.gz	0.976085354	0.87564941	0.923611909	0.753215569	0.948374104	0.869696333	0.691055902	0.524279191
train_357.nii.gz	0.976610263	0.891130215	0.977480212	0.963298223	0.97222559	0.927625148	0.798281519	0.722494063
train_358.nii.gz	0.970460984	0.858277245	0.93997656	0.838484603	0.975276585	0.97541179	0.753835502	0.629969565

mean	0.976541164	0.883195409	0.95978307	0.906205788	0.970461634	0.933353862	0.748708874	0.54182885

We also compared with other networks; Table 5 shows our method performs better than the others.

Table 5 Compare with VNet and UNet

method	dice					surface dice				
	liver	kindey	spleen	pancreas	mean	liver	kindey	spleen	pancreas	mean
VNet (no aug.)	0.9426	0.9394	0.9033	0.5490	0.8336	-	-	-	-	-
VNet	0.9716	0.9499	0.9588	0.7547	0.9087	0.8329	0.8619	0.8948	0.48970	0.769825
UNet++2D	0.9729	0.9547	0.9645	0.7429	0.9088	-	-	-	-	-
RepVGG	0.9765	0.9597	0.9704	0.7487	0.9138	0.9078	0.9214	0.9468	0.6498	0.8564

5.2. Quantitative results on validation set.

Table 6 and Table 7 illustrates the results on validation cases.

Table 6 Quantitative results on validation set.

Organ	DSC (%)	NSD (%)
Liver	89.5±15.2	63.8±21
Kidney	75±23.2	65.7±20.4
Spleen	75.5±27	53.6±31.6
Pancreas	48±25.2	32.4±20.8

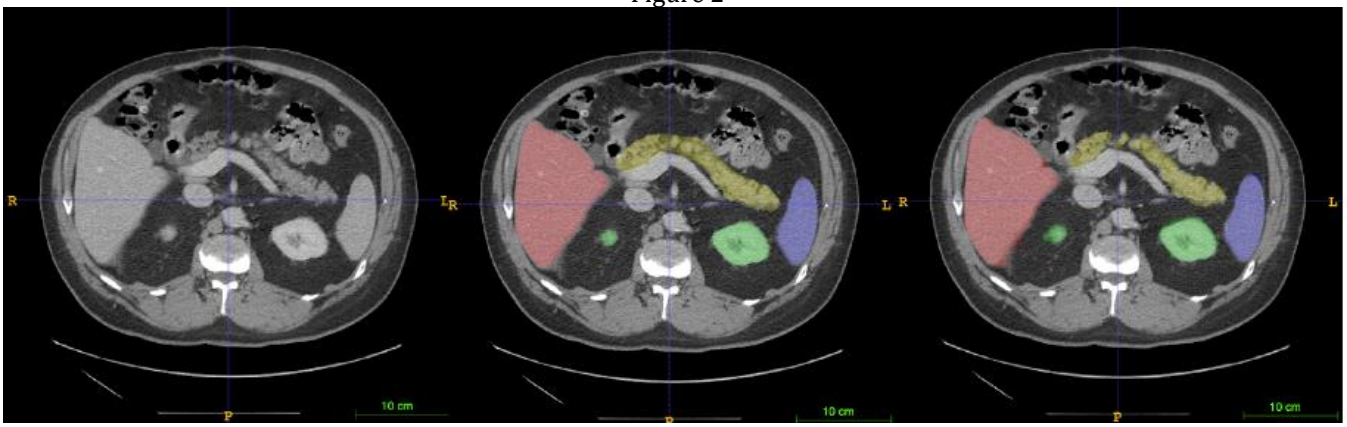
Table 7 Running time results on validation set.

Metrics	Result
Running	22.893s
GPU Memory	7800Mb

5.3. Qualitative results

Figure 2 presents some challenging examples. The left image is the original CT slice, the middle is the ground truth, and the right is our segmentation result. It can be found that our method cannot segment organs well. The liver (red), spleen(blue), and kidney(green) have similar segment result with ground truth, while the pancreas segmentation is a little complicated but still can get a good result.

Figure 2



6. Discussion and Conclusion

Our method performs well with most cases, particularly segments with the liver, spleen, and kidney. However, some cases with lesion-affected or heavy noise are hard to perform well. Also, our method performs a little weak with the pancreas, and pancreas segmentation is a challenging task in this competition.

We think the main reason is the sampling rate with different target organs. To improve pancreas segmentation performance, it needs to sample more with that, and perhaps an induvial model for pancreas segmentation is a better choice.

Acknowledgment

The authors of this paper declare that the segmentation method they implemented for participation in the FLARE challenge has not used any pre-trained models nor additional datasets other than those provided by the organizers.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation, in International Conference on Medical image computing and computer-assisted intervention, 2015, pp. 234–241. 1

[2] Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., & Sun, J. (2021). Repvgg: Making vgg-style convnets great again. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 13733-13742).

[3] MONAI Consortium. (2020). MONAI: Medical Open Network for AI (Version 0.5.0) [Computer software]. https://doi.org/10.5281/zenodo.4323058

[4] A. L. Simpson, M. Antonelli, S. Bakas, M. Bilello, K. Farahani, B. Van Ginneken, A. Kopp-Schneider, B. A. Landman,G. Litjens, B. Menze *et al.*, "A large annotated medical image dataset for the development and evaluation of segmentation algorithms," *arXiv preprint arXiv:1902.09063*, 2019. 2

[5] 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 preprint arXiv:1901.04056*, 2019. 2

[6] H. Roth, A. Farag, E. Turkbey, L. Lu, J. Liu, and R. Summers, "Data from pancreas-ct. the cancer imaging archive (2016)." 2

[7] H. R. Roth, L. Lu, A. Farag, H.-C. Shin, J. Liu, E. B. Turkbey, and R. M. Summers, "Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2015, pp. 556–564. 2

[8] K. Clark, B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle *et al.*, "The cancer imaging archive (tcia): maintaining and operating a public information repository," *Journal of digital imaging*, vol. 26, no. 6, pp. 1045–1057, 2013. 2

[9] 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 contrastenhanced ct imaging: Results of the kits19 challenge," *Medical Image Analysis*, vol. 67, p. 101821, 2021. 2

[10] N. Heller, S. McSweeney, M. T. Peterson, S. Peterson,J. Rickman, B. Stai, R. Tejpaul, M. Oestreich, P. Blake,J. Rosenberg et al., "An international challenge to use artificial intelligence to define the state-of-the-art in kidney and kidney tumor segmentation in ct imaging." American Society of Clinical Oncology, vol. 38, no. 6, pp. 626–626, 2020. 2

[11] J. Ma, Y. Zhang, S. Gu, C. Zhu, C. Ge, Y. Zhang, X. An,C. Wang, Q. Wang, X. Liu, S. Cao, Q. Zhang, S. Liu,Y. Wang, Y. Li, J. He, and X. Yang, "Abdomenct-1k: Is abdominal organ segmentation a solved problem?" IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021. 2, 3, 4