Short Paper for MICCAI FLARE21 Challenges

Joseph Nathanael Witanto
MEDICALIP
9F, Yeonkang Building, 15, Jong-ro 33-gil, Jongno-gu, Seoul, Republic of Korea
josephnw@medicalip.com

Kevin Pratama
MEDICALIP
9F, Yeonkang Building, 15, Jong-ro 33-gil, Jongno-gu, Seoul, Republic of Korea
kvpratama@medicalip.com

Doohee Lee
MEDICALIP
9F, Yeonkang Building, 15, Jong-ro 33-gil, Jongno-gu, Seoul, Republic of Korea
dooheelee@medicalip.com

Soon Ho Yoon
Seoul National University Hospital
Department of Radiology, Seoul National University Hospital, Seoul, Republic of Korea
yshoka@gmail.com

Sang Joon Park
Seoul National University
Department of Radiology, Seoul National University, Seoul, Republic of Korea
lunao78@snu.ac.kr

Abstract

Auto-segmentation of abdominal organs has been made possible by the advent of the deep learning technique. The developments of convolution neural network as an automatic segmentation method of main abdominal organs have achieved remarkable results on many benchmark datasets. However, most of the existing datasets are very uniform which means it only contains single-center, single-phase, single-vendor, or single-disease cases. Therefore, it remains to be seen whether this outstanding result can be generalized on diverse datasets. In this challenge, we utilized the nnU-Net as our baseline neural network. Taking into account the trade-off between accuracy and efficiency, we decided to go ahead with the 2D model which could run inference in a reasonable time while producing a comparable result with its 3D counterpart. Furthermore, 3D image isotropic and manual data augmentation using our in-house medical image software are introduced to simulate unseen data during training. We train the network and then computed Dice similarity coefficients for evaluating our estimates. Our experiments show that our approach gives strong results in terms of Dice similarity coefficients while requiring reasonable inference time.

1. Introduction

The development of non-invasive imaging technologies has expanded new horizons in studying and researching human anatomical structures. In the wake of this technology, image segmentation has become a critical task in image analysis with numerous applications, including but not limited to computer-assisted diagnosis, surgery planning, and image-guided intervention. Therefore, robust automatic image segmentation technique is needed to guide image interpretation and clinical decision making while as much as possible limiting and avoiding manual intervention efforts.

Deep learning, specifically Convolution Neural Net-

work, has emerged as the state of the art technique when it comes to image segmentation tasks. For example, it has achieved a remarkable result in abdominal organ segmentation which contains organs such as the liver, kidney, spleen, and pancreas. However, the extraordinary result was achieved from benchmark datasets that contain homogeneous cases. It remains to be seen whether it can be generalized to benchmark datasets that contain more diverse cases.

2021 FLARE Challenge: Fast and Low GPU Memory Abdominal Organ Segmentation has been proposed to drive further research on abdominal image segmentation. There are two main objectives in this challenge: the first is to develop a segmentation algorithm that can segment four abdominal organs (liver, kidney, spleen, and pancreas) simultaneously. The second objective for the participant is to develop an algorithm that not only can perform a high accuracy segmentation but also has high efficiency, meaning it requires low GPU memory and takes reasonable time to run prediction.

Participating in this challenge, we use the well-known nnU-Net [1] architecture as our baseline network. Besides the fact that it has performed very well on many medical image dataset, nnU-Net also perform inference in a reasonable time. Furthermore, our experiment in the provided dataset for this challenge shows that nnU-Net with 2D architecture has a much more efficient performance when being compared to its 3D counterpart while producing a comparable result. To simulate unseen data, we introduced a couple of data augmentation techniques using our in-house medical image software. This augmentation coupled with the data augmentation technique that is already available in the nnU-Net segmentation pipeline has proven to improve the segmentation performance on the training set.

2. Method

A deep learning architecture called U-Net [2, 3] has produced state-of-the-art results in segmentation tasks and has been widely used in various medical image segmentation tasks. Many attempts and variants of its vanilla version have been proposed to improve its performance. One example of those variants that come to our attention is nnU-Net. nnU-Net was proposed as a segmentation framework with an end-to-end pipeline that can automatically adapt preprocessing strategies and network hyperparameters (e.g. the size and the number of pooling layers, convolutional kernel size, and stride step) to a given medical image dataset. Without any manual intervention, nnU-Net can achieve high performances, even when being compared to specialized deep learning pipelines, and set the new state-of-the-art results in many tasks.

For this challenge, we employ nnU-Net with its 2D Architecture. The patch size was set at 512x512 and batch size was set at 12. Stochastic gradient descent with nesterov momentum [4] ($\mu = 0.99$) was the chosen optimization algorithm. The polynomial learning rate scheduler was initialized at 0.01. Data augmentation techniques were introduced during training, such as rotation, scaling, Gaussian Noise, gamma correction, and mirroring. The loss function is the sum of Dice loss and cross-entropy loss. All the models are trained for 1000 epochs with 250 mini batches each on NVIDIA RTX 3090 GPUs.

2.1. Preprocessing

Our method includes the following preprocessing steps:

- Cropping strategy: None

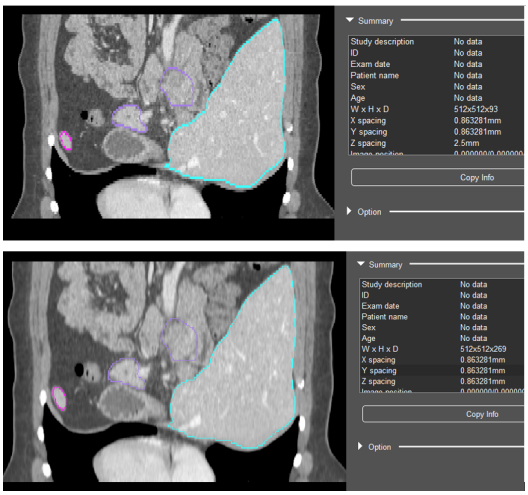

Figure 1. Example of preprocessing with z-isotropification result on training data. The image depth is increased from 93 to 269.

- Z-Isotropification:
  Voxel spacing represents the real physical space that a voxel cover. Typically, the medical domain dataset has a wide-ranging voxel spacing. Problem will arise given the fact that convolution neural networks operate on voxel grids and ignore this information. To tackle this problem, we preprocess the given dataset with our in-house medical image software to create isotropic data.

Figure 2. Example of preprocessing with disease simulation on training data.

- Disease Simulation:
  After running a few experiments, we notice that our

trained networks struggle with unseen data, specifically, data with diseases that are not part of the training data. We then decided to augment the diseases in the selected 50 cases of training data with our in-house medical image software.

- Intensity normalization:
  A global normalization scheme was applied to all images in the dataset. To initiate this process, 0.5 and 99.5 percentiles of the foreground voxels were clipped to remove outliers. Following the clipping, the global mean and standard deviation was calculated and used to determine the z-score normalization.

2.2. Proposed Method

In this subsection, we will describe the network architecture along with its hyperparameters that we use to participate in this challenge.

- As we have mentioned in the previous section, we employ nnU-Net as the chosen method. Specifically, we closely follow the 2D U-Net architecture that is available in the nnU-Net framework. We trained with a batch size of 12 with a patch size of 512. To stabilize the training process, we use Instance Normalization[5]. Leaky ReLU [6] (with negative slope = 0.01) replace the standard ReLU[7] as the activation function. The network is trained with deep supervision technique, meaning auxiliary losses were injected into all decoder layers except the two lowest resolution layers. The motivation behind this injected loss is to allow gradients to be propagated deeper into the network's layers, hence assisting the learning process.

- The objective function for the network to optimize is the combination of cross-entropy loss with dice loss [8]. For each deep supervision output, the loss was calculated for each resolution, and a weighted sum of the losses then becomes the final loss.

- The training is scheduled for 1000 epochs, with one epoch consist of 250 mini-batches. Stochastic gradient descent with Nesterov momentum ($\mu = 0.99$) is the chosen optimization algorithm. The initial learning is set to 0.01 and decayed with poly learning rate decay. $(1 - epoch/epoch_{max})^{0.9}$

- Number of model parameters: 41268192

- Number of flops: 61307143168

2.3. Post-processing

No Post-processing is used.

3. Dataset and Evaluation Metrics

3.1. Dataset

- The dataset used in FLARE2021 challenge is adapted from MSD [9] (Liver [10], Spleen, Pancreas), NIH Pancreas [11, 12, 13], KiTS [14, 15], and Nanjing University under the license permission. For more detail information of the dataset, please refer to the challenge website and [16].

- 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.

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

We implemented our method in Python 3.8.10 using PyTorch framework 1.8.1[17]. Batchgenerators 0.21[18] is used for data augmentation inside nnU-Net framework. Some common python libraries that were also used include tqdm 4.61.1, dicom2nifti 2.3.0, scikit-image 0.18.1, MedPy 0.4.0, SciPy 1.7.0, NumPy 1.20.2, scikit-learn 0.24.2, SimpleITK 2.0.2 and pandas 1.2.5.

For the complete list of environments and requirements of our method is please refer to Table 2.

4.2. Training Protocols

Our training approach closely follows the training method of the nnU-Net framework. During training, we employ data augmentation techniques to prevent overfitting and simulate unseen data. Training will run for 1000 epochs with every epoch has 250 mini-batches iterations. 33.3% samples in a batch are guarantee to contain at least one randomly chosen foreground class, and the rest are random locations.

The training details of our method are shown in Table 3.

4.3. Testing Protocols

We apply the same preprocessing strategy as training data to the validation and testing data. However, we did not use any patch-based strategy or post-processing steps.

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

Table 2. Environments and requirements.

OS	Ubuntu 18.04.5 LTS
CPU	Intel(R) Core(TM) i9-10980X CPU@3.00GHz
RAM	8×32GB
GPU	Nvidia RTX3090
CUDA version	11.1
Programming language	Python3.8
Deep learning framework	Pytorch (Torch 1.8.1, torchvision 0.9.1)
Python Library	batchgenerators 0.21, dicom2nifti 2.3.0, hiddenlayer 0.2, matplotlib 3.4.2, MedPy 0.4.0, nibabel 3.2.1, NumPy 1.20.2, pandas 1.2.5, pillow 8.2.0, pydicom 2.1.2, scikit-image 0.18.1, scikit-learn 0.24.2, SciPy 1.7.0, SimpleITK 2.0.2, tqdm 4.61.1

Table 3. Training protocols.

Data augmentation methods	Rotations, scaling, Gaussian noise, Gaussian blur, brightness, contrast, simulation of low resolution, gamma correction and mirroring.
Initialization of the network	"he" initialization
Patch sampling strategy	33.3 % of the samples in a batch is guarantee to contain at least one randomly chosen foreground class while the rest are random location
Batch size	12
Patch size	512×512
Total epochs	1000
Optimizer	Stochastic gradient descent with nesterov momentum ($\mu = 0.99$)
Initial number of feature map	32
Kernel size	3x3
Initial learning rate	0.01
Learning rate decay schedule	poly learning rate policy: $(1 - epoch/1000)^{0.9}$
Stopping criteria, and optimal model selection criteria	Stopping criterion is reaching the maximum number of epoch (1000).
Training time	72 hours

5. Results

5.1. Quantitative Results on Training Set.

The provided results analysis is based on three metrics (Dice, Sensitivity, and Precision) on the training set. Table 4 illustrates the results of training set. Figure 3, Figure 4, and Figure 5 are the corresponding box plot of the organ segmentation performance.

High dice, sensitivity, and precision scores are obtained for the liver, kidney, spleen, and pancreas. However, this result was obtained from the training set which could indicate overfitting. For the final submission, we decided not to use cross-validation. Instead, we use all available data as a training set and approximate the best network from the prediction result on the validation set. We believe that this will be beneficial for our network performance on the validation set and testing set which have not been released at the time of writing of this short paper.

Table 4. Quantitative results on training set.

Organ	Dice (%)	Sensitivity (%)	Precision (%)
Liver	98.14±0.005	98.10±0.007	98.18±0.006
Kidney	96.74±0.011	97.22±0.013	96.29±0.018
Spleen	97.47±0.009	97.39±0.012	97.56±0.010
Pancreas	91.17±0.024	91.57±0.026	90.84±0.033

5.2. Quantitative Results on Validation Set

In this section, we will report our result on the validation test that we receive from this challenge organizer. As mentioned in Section 3, the evaluation metrics in this challenge are Dice Similarity Coefficient (DSC) and Normalized Surface Distance (NSD). Alongside those two quantitative metrics, the running time of the algorithm and GPU memory usage are also evaluated. We present the comparison of our result with the baseline result in Table 5

As we can see from Table 5, our result is comparable

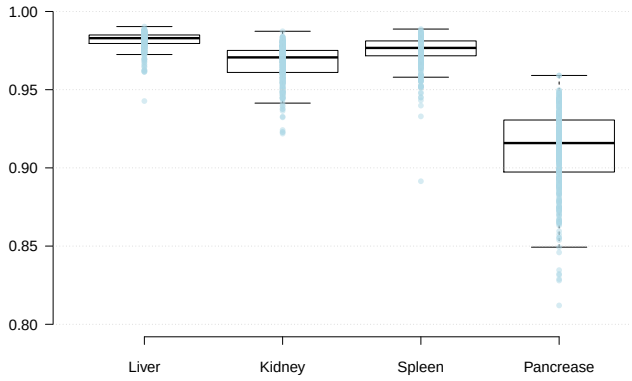

Figure 3. Box plot of the Dice score result obtained from the training set.

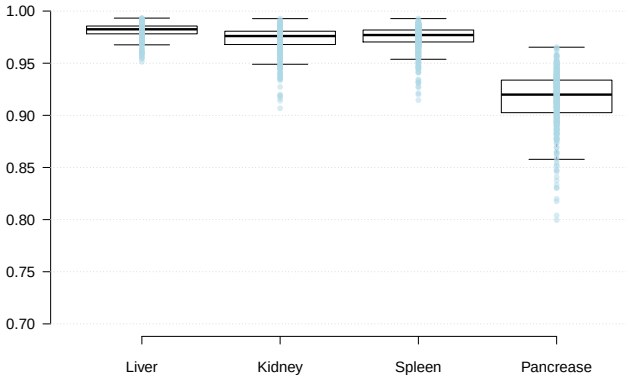

Figure 4. Box plot of the Sensitivity score result obtained from the training set.

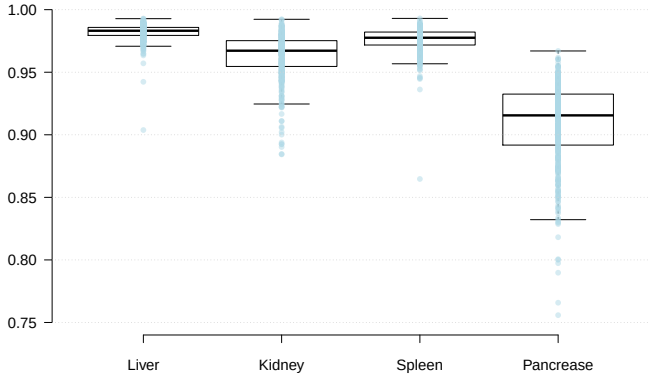

Figure 5. Box plot of the Precision score result obtained from the training set.

with the baseline while requiring shorter running time and less GPU memory. Looking at each class, our method outperforms the baseline on Kidney and Pancreas while performs slightly worse on Liver and Spleen. Our Liver segmentation result has 0.1 and 3.1 points less than segmentation produced by the baseline method on DSC and NSD metric respectively. On a similar note, our Spleen segmentation result has 0.2 and 2.0 points less than the baseline on DSC and NSD metrics respectively. On the other hand, our Kidney segmentation produces 1.2 and 0.9 points better on DSC and NSD metrics respectively. Similarly, for Pancreas segmentation, we outperform the baseline by 8.8 and 4.0 points on DSC and NSD metrics respectively.

Despite the fact that we underperform the baseline on two classes, the difference is quite negligible while producing noticeably better results on the other two classes. If we calculate the average of each metric for all classes, the baseline has an average Dice Similarity Coefficient of 81.125 while our method has 83.55. Furthermore, the baseline has an average Normalized Surface Distance of 70.7 while our method has 70.9. All these advantages are obtained with faster running time and less GPU memory.

Table 5. Quantitative results on validation set.

Metrics	Baseline (Avg ± std)	Ours (Avg ± std)
Liver-DSC	94.5 ± 8.09	94.4±5.2
Liver-NSD	79.3 ± 14.9	76.2±13.9
Kidney-DSC	80.4 ± 17.0	81.6±18.7
Kidney-NSD	70.9 ± 18.4	72.8±18.9
Spleen-DSC	89.5 ± 18.0	89.3±18.3
Spleen-NSD	82.0 ± 19.3	80.0±21.9
Pancreas-DSC	60.1 ± 23.1	68.9±20.5
Pancreas-NSD	50.6 ± 17.7	54.6±19.3
Running Time	145	102.7
GPU Memory	2298	2220.8

5.3. Qualitative Results

Figure 6 presents the worst segmentation result on one of the training data in terms of dice score. In this case, the dice score is 94.27, 94.14, 98.48, 92.98 for liver, kidney, spleen, and pancreas organs, respectively. As shown in the image, the network manages to segment the organ quite well. However, readers need to be aware that this result comes from training data with no guarantee that it could be generalized to unseen data. Figure 7 presents what we think is the worst segmentation result on one of the validation data. We could not justify this with an official metric since the ground truth for the validation set has not been released at the time of writing. However, we could justify it visually by looking at the segmentation result. Figure 7 clearly shows that our trained network struggle to segment the pancreas area. It also labels the non-liver area as the liver area while almost failed to detect the spleen area.

6. Discussion and Conclusion

Given the limited amount of time to explore and run the experiment on the provided dataset, we have found a segmentation network that works reasonably well while pro-

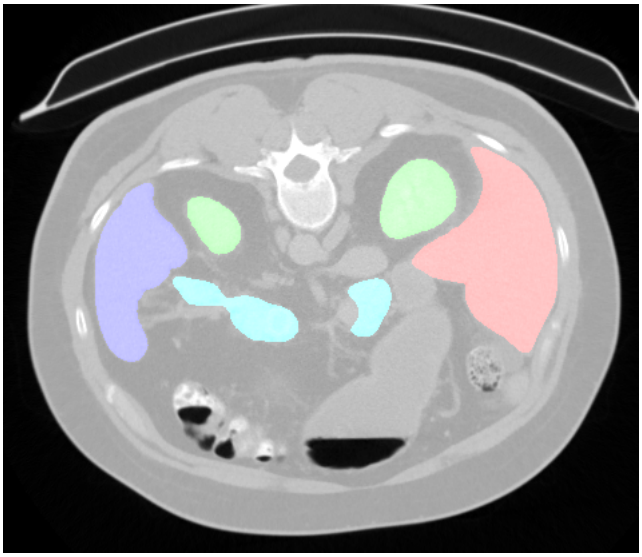

Figure 6. Example of segmentation result on training data by our trained network.

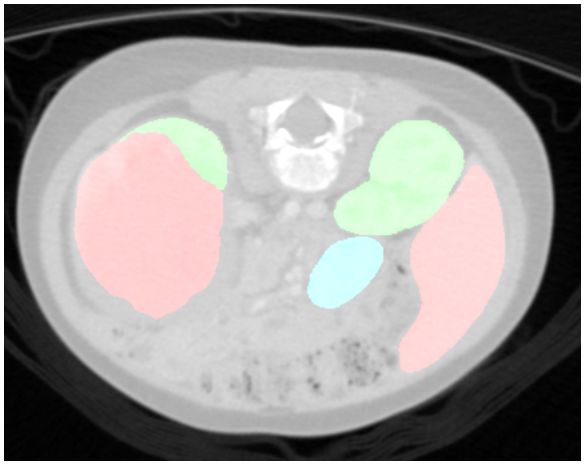

Figure 7. Example of segmentation result on validation data by our trained network.

ducing segmentation masks. Before deciding to go ahead with the network that we submitted, we have run a couple of experiments with U-Net variants such as 3D U-Net[3] and U-Net Attention[19]. However, considering the trade-off between model accuracy and model efficiency, we finally concluded that 2D U-Net is the best network for this challenge. Besides being the most efficient in terms of resources and time consumption, the segmentation result is comparable with its 3D and Attention network counterparts.

During the data exploration phase, we notice that diseases exist in the validation set while almost lacking in the training set. This unseen variability on the training set will hinder the network from performing well on the validation set, and most likely, the test set as well. Furthermore, a standard data augmentation technique in deep learning literature

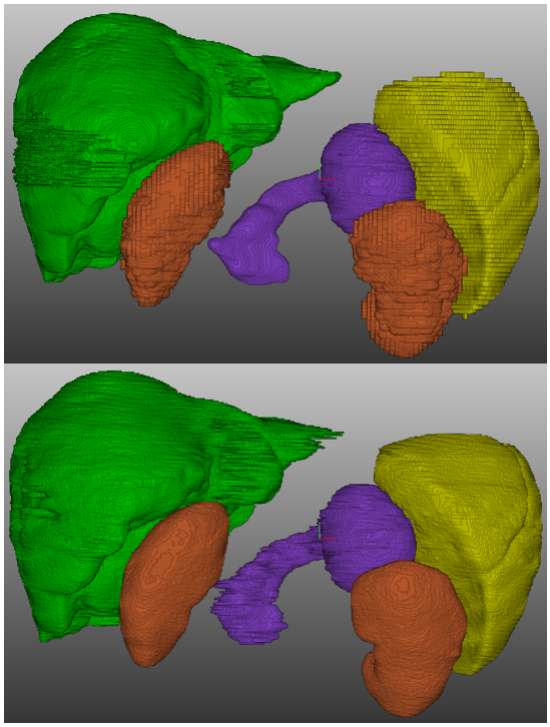

Figure 8. 3D comparison of label and prediction result from one of training data. Top row is label bottom row is prediction.

is not designed to simulate disease in the human organ. In order to overcome this problem, we decided to manually augment the disease in the training set with our in-house medical image software. By doing this disease simulation, we hope our network will be able to recognize the disease found in the validation set and segment it correctly.

We have discussed in Section 5 that we decided not to use cross-validation set to find the best network configuration. This decision is mainly due to the reason that we have explained in the previous paragraph that there is a huge variability between the training set and validation set. Hence performing well on the cross-validation set will not guarantee generalization on the validation set. Instead, we use all the provided training set for network training, and visually justify the network's performance on the validation set. We believe that using all training data will benefit our network's ability to generalize to unseen data as it is a well-known knowledge in the deep learning domain that more data will most likely yield a better result.

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] 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. 2

[2] 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. 2

[3] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: learning dense volumetric segmentationfrom sparse annotation," *CoRR*, vol. abs/1606.06650, 2016. 2, 6

[4] A. Botev, G. Lever, and D. Barber, "Nesterov's accelerated gradient and momentum as approximations to regularised update descent," *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017. 2

[5] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *CoRR*, vol. abs/1607.08022, 2016. 3

[6] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013. 3

[7] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010, pp. pp. 807–814,. 3

[8] 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, p. 102035, 2021. 3

[9] 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. 3

[10] 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. 3

[11] H. Roth, A. Farag, E. Turkbey, L. Lu, J. Liu, and R. Summers, "Data from pancreas-ct. the cancer imaging archive (2016)." 3

[12] 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. 3

[13] 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. 3

[14] 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. 3

[15] 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. 3

[16] 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. 3

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. 3

[18] F. Isensee, P. Jäger, J. Wasserthal, D. Zimmerer, J. Petersen, S. Kohl, J. Schock, A. Klein, T. Roß, S. Wirkert, P. Neher, S. Dinkelacker, G. Köhler, and K. Maier-Hein, "batchgenerators - a python framework for data augmentation," *Zenodo*, 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3632567 3

[19] O. Oktay, J. Schlemper, L. L. Folgoc, M. C. H. Lee, M. P. Heinrich, K. Misawa, K. Mori, S. G. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention u-net: Learning where to look for the pancreas," *CoRR*, vol. abs/1804.03999, 2018. [Online]. Available: http://arxiv.org/abs/1804.03999 6