SUPPLEMENTARY MATERIAL

Chuyan Zhang
Institute and Medical Robotics
Shanghai Jiao Tong University

Hao Zheng
Institute and Medical Robotics
Shanghai Jiao Tong University

Yun Gu
Institute and Medical Robotics
Shanghai Jiao Tong University

1 Implementation details for all tasks

Our attached SSL code base allows training and evaluation for various pretext tasks and downstream tasks in a unified framework. For training, implementations for multiple loss functions and optimization options are provided. It is easy for researchers to implement new datasets or tasks on our code base, thereby making it convenient to compare their proposed methods with existing SSL methods. Our implementations rely on Pytorch [Paszke et al., 2019] and some APIs for medical imaging. Details can be found in the README.md file. The following are the details of our implementations for pretraining SSL pretext tasks as well as fine-tuning the pretrained models in downstream tasks subsequently.

1.1 Pretext tasks

Following [Zhou et al., 2021a], we pretrain all the pretext tasks with the LUNA 2016 dataset [Setio et al., 2017] that is composed of 10 subsets with a total of 888 low-dose chest CT scans. It is noteworthy that several downstream tasks also utilize the images from LUNA 2016 dataset. To avoid test-image leakages, we should keep the test images for relevant target tasks unseen for proxy task training. Consistently, the 1-7 subsets (623 images) are used for pretraining and fine-tuning, with 1-5 subsets (445 images) as the training set and 6-7 (178 images)subsets as the validation set. The remaining 8-10 subsets (265 images) constitute the test set for target tasks. For each CT scan, we applied an intensity window at $[-1000, 1000]$ and then normalized it to $[0, 1]$. Even though the training data for all proxy tasks is the same, the input processing varies for different types of SSL. Below, we provide the implementation details about the datasets, data processing and optimization for each SSL method.

Predictive SSL The predictive SSL models are trained by Adam optimizer [Kingma and Ba, 2014]. The learning rate is initially set as 1e-3 and then decayed by a factor of 10 at 250 epochs. To prevent overfitting, the training stops if no improvement is observed in the validation set over a specified number of epochs. As for the input processing, we center-cropped each whole CT volume to a uniform size of [320, 320, 74] ([H, W, D]) in predictive tasks. This processing step ensures that: (a) the images of the same size allow batch training; (b)most of the empty areas were removed according to the CT image prior.

- **RPT:** We split each volume into $3 \times 3 \times 3$ patches. The central patch is regarded as a reference patch. In the training stage, we randomly select a patch other than the central patch. Both this patch and the central patch are taken as the input to the network. One potential problem is that the model might fail to learn useful representations due to some shortcuts such as edge continuity. Therefore, we leave a random gap of 3 pixels per axis between adjacent patches.
- **ROT:** The predictions for whole CT volumes are so simple for deep neural networks that the expressive power of learned features might be very limited. To increase the difficulty of the proxy task, we randomly crop a small patch in a volume as the input.
- **Jigsaw, RKB, RKB+:** In jigsaw-based methods, we split each volume into $2 \times 2 \times 2$ patches. Then, we iteratively select 100 permutations with the largest Hamming distance from all the 8! permutations. The jigsaw puzzle turns into a 100-way classification task.

Generative SSL In purely generative SSL tasks (AE, MG), we train the models by SGD optimizer [Zhang, 2004] with an initial learning rate of 1e0. Following Zhou et al. [2021a], we employ a learning rate annealing strategy namely

ReduceLROnPlateau. In specific, the learning rate is decayed by half when no improvement is observed in the validation set over a specified number of epochs. For each volume, we randomly cropped patches with the size of [64, 64, 32] and excluded those which are empty (air) or contain full tissues according to the intensity values. Finally, a training set consisting of 14159 patches with the size of [64, 64, 32] and a validation set consisting of 5640 patches with the size of [64, 64, 32] were generated for the reconstruction tasks.

Contrastive SSL In proxy tasks that involve contrast learning (PCRL, SimLCR, BYOL), we train the models by SGD optimizer [Zhang, 2004] with an initial learning rate of 1e-3. Following Zhou et al. [2021b], a cosine annealing strategy is adopted to schedule the learning rate. In typical contrastive learning for natural images, randomly cropping and some other augmentations are applied to one image to generate a positive pair. However, in the medical field, multiple objects always exist in one image so randomly cropping would lead to model confusion. For effective contrastive learning in medical images, a crucial trick is to crop positive pairs with overlaps. Specifically, we randomly crop a pair of patches with the size of [64 64, 32] in one volume and compute their IoU (Intersection over Union) score. Only pairs with the IoU score larger than 0.25 can be kept. For each training volume, such operations are repeated several times. Finally, we built a training dataset composed of 9968 positive pairs.

1.2 Target tasks

In downstream experiments, we utilize three medical tasks that are the most frequent in existing publications of SSL: NCC, NCS and LCS. Besides, we also use the MSD-liver dataset to explore multi-class segmentation. Note that we keep the dataset partitioning protocols the same as Zhou et al. [2021a]. The training details for each are provided as follows.

NCC The dataset for NCC is from LUNA 2016 [Setio et al., 2017], which contains 5,510,166 candidate locations for the lung nodule false positive reduction task. According to the location annotations, we can obtain candidate cubes with the size of [48, 48, 48] as inputs. The dataset splitting has been introduced at the beginning of this section, leading to a training set with 282568 cubes, a validation set with 109482 cubes and a test set with 166225 cubes. The false positive cubes are labelled as class 0 while the true positive cubes are tagged as class 1. The number of training samples for class 0 and class 1 is 274637 and 7931, respectively. Obviously, NCC is a severely imbalanced binary classification task. Thus, we perform a mean resampler to ensure the number of class 0 is equal to class 1 at each epoch. During training, we adopt the binary cross entropy loss to learn classifiers. Adam optimizer [Kingma and Ba, 2014] with a learning rate of 1e-3 is used for optimization.

NCS The dataset for NCS is from the Lung Image Database Consortium image collection (LIDC-IDRI) [Armato III et al., 2011], which consists of 1018 lung CT scans. For each scan, the nodules are marked as binary masks. Firstly, we re-sampled the volumes to 1-1-1 spacing and cropped a patch around the nodule with the size of $64 \times 64 \times 32$ as the input. We split the volumes into training (510), validation (100), and test (408) sets. We adopt the dice loss function to train the model. Adam optimizer [Kingma and Ba, 2014] with a learning rate of 1e-3 is used for optimization. The training stops if no improvement has been observed in the validation set for a specified number of epochs.

LCS The dataset for LCS is from MICCAI 2017 LiTS Challenge [Bilic et al., 2019], which is composed of 130 abdominal CT scans. In the data preprocessing stage, the volumes were first clipped with a HU window of [-200, 200] to optimize the organ contrast and subsequently were normalized to [0, 1]. Next, they were resampled to 1-1-1 spacing. Due to the limited GPU memory, we 0.5x down-sampled each image and used the sliding window cropping strategy along the z axis. The scans were split into training (100), validation (15) and test (15) sets. We combine the binary cross entropy with dice loss functions to train the model. Adam optimizer [Kingma and Ba, 2014] with an initial learning rate of 1e-2 is used for optimization. At the 50th and 150th epoch, the learning rate is reduced by a factor of 10. The training stops if no improvement has been observed in the validation set for a specified number of epochs.

MSD-Liver The dataset for MSD-Liver is from the Medical Segmentation Decathlon Challenge (MSD) [Antonelli et al., 2022], which consists of 130 labelled abdominal CT scans. The images are the same as the LCS dataset but the annotations include both livers and tumors, which makes MSD-liver task more challenging than LCS. We applied the data preprocessing scheme in nnUNet Isensee et al. [2021] to generate inputs. The data splitting is the same as LCS. We combine the cross entropy with dice loss functions to train the model. Adam optimizer [Kingma and Ba, 2014] with an initial learning rate of 1e-2 is used for optimization. At the 150th and 300th epoch, the learning rate is reduced by a factor of 10. The training stops if no improvement has been observed in the validation set for a specified number of epochs.

Table 1: Notations for data augmentations used in NCS. Here, p denotes the probability for augmentation.

Augmentation scheme	Description
Aug-S1	random flipping (p = 0.5)
Aug-S2	rotating by 90 degree (p = 0.5)
Aug-S3	random flipping (p = 0.5), random rotating by degrees within [-10, 10] (p = 0.3)
Aug-S4	rotating by 90 degree (p = 0.5) random rotating by degrees within [-20, 20] (p = 0.3)
Aug-S5	random flipping (p = 0.5), rotating by 90 degree (p = 0.5), random rotating by degrees within [-30, 30] (p = 0.6)
Aug-S6	random flipping (p = 0.5), rotating by 90 degree (p = 0.5), random rotating by degrees within [-30, 30] (p = 0.6), random scaling in [0.8, 1.2] (p = 0.4)

2 Network architectures for all tasks

We use a 3D UNet [Ronneberger et al., 2015] encoder for all tasks. It includes four 3D convolutional blocks with 64, 128, 256 and 512 channels, respectively. On the top of the encoder, we add specific network heads or decoders for different tasks, detailed below.

ROT, NCC The network head of ROT or NCC is a classifier constituted by a global average pooling layer and two fully connected layers with 1024 and category number neurons, respectively.

RPL The network head of PRL consists of two branches, which share parameters but are intended for reference patch and random patch respectively, and a classifier. First, a global average pooling layer is added after the encoder to get a dense hidden layer. Then, in two branches, a fully connected layer followed by a Batch Normalization layer get two 64-dimensional representations separately from the reference patch and random patch. Next, these two vectors are concatenated together and input to the classifier that is the same as the ROT.

Jigsaw, RKB, RKB+ The network heads of Jigsaw, RKB and RKB+ include multiple shared branches for $2 \times 2 \times 2$ cubes and classifiers for several sub-tasks. Similar to RPL, a 64-dimensional representation is extracted for each cube through a global average pooling and a fully connected layer followed by a Batch Normalization layer. For each image, the representations of 8 cubes are then concatenated and subsequently input to the order, rotation and mask classifiers. Considering that the order task is more complex than the other two, the order classifier is composed of three fully connected layers with 1024, 1024 and category number neurons while the rotation and mask classifiers only contain two fully connected layers with 1024 and category number neurons.

MG, AE, NCS, LCS The network head of the reconstruction task or segmentation task is a symmetric decoder to the encoder. In contrast to the encoder, the channels of convolutional blocks in the decoder are 256, 128, 64 and class number. We provide two alternatives about whether use the skip connections between the decoder and encoder.

SimCLR, BYOL The network head of SimCLR and BYOL aims to project the input to the hidden representation space. For SimCLR, the feature projector consists of two fully connected layers with 512 and 256 neurons. BYOL requires two projectors and one predictor, where the structure of each is identical to the projector in SimCLR.

PCRL The network head of PCRL contains three branches. The first branch is the decoder same as MG, aiming to recover images. The second branch is constituted by two fully connected layers with 512 and 128 neurons, to extract representations for contrastive learning. The third branch consists of two fully connected layers with 256 and 512 neurons, targeting the controlling vector to perform an explicit transformation on the input.

3 Augmentation for NCC, NCS and Contrastive SSL tasks

NCC On account of the very limited number of the true positive class, we augment the original true positive samples by 10 times with random rotating, shifting and flipping.

NCS When exploring the additive effect of data augmentation with SSL in NCS, we implement six data augmentation schemes from “Aug-S1" to “Aug-S6" (see Table 1). Since random rotating by 90 degrees yields more different views than only flipping, we consider S2 to be stronger than S1. We can claim that the strength of data augmentation increases from S1 to S6.

Contrastive SSL Many data augmentation strategies commonly used in natural images are inappropriate for gray-scale medical images, such as color jitter and grayscale. Thus, we only apply random flipping, rotating, small-range scaling and Gaussian blur on the generated overlapped positive pairs.

4 Detailed experiments

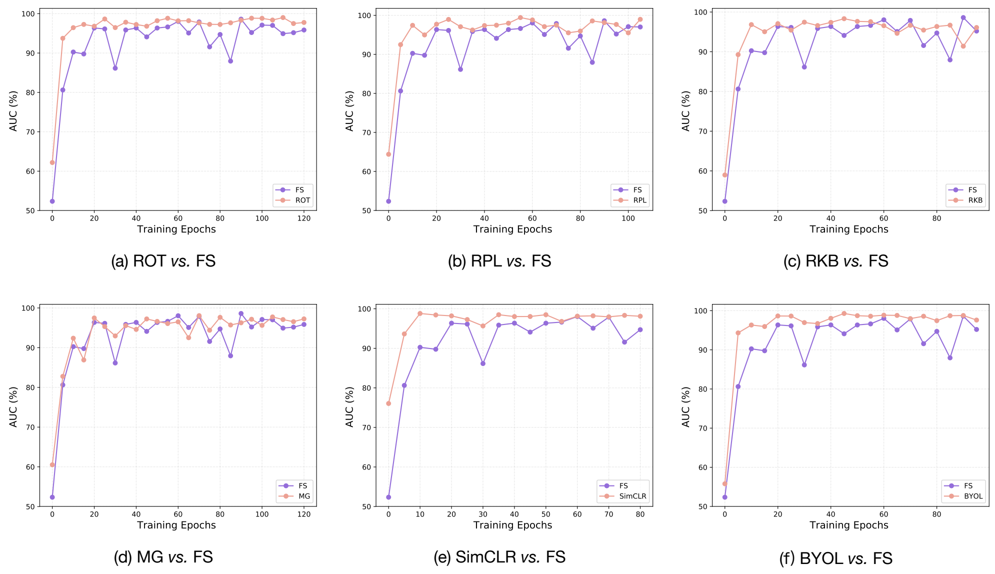

Figure 1: Comparisons of convergence speed between self-supervised pretraining methods (pink) and the from-scratch baseline (purple) in NCC.

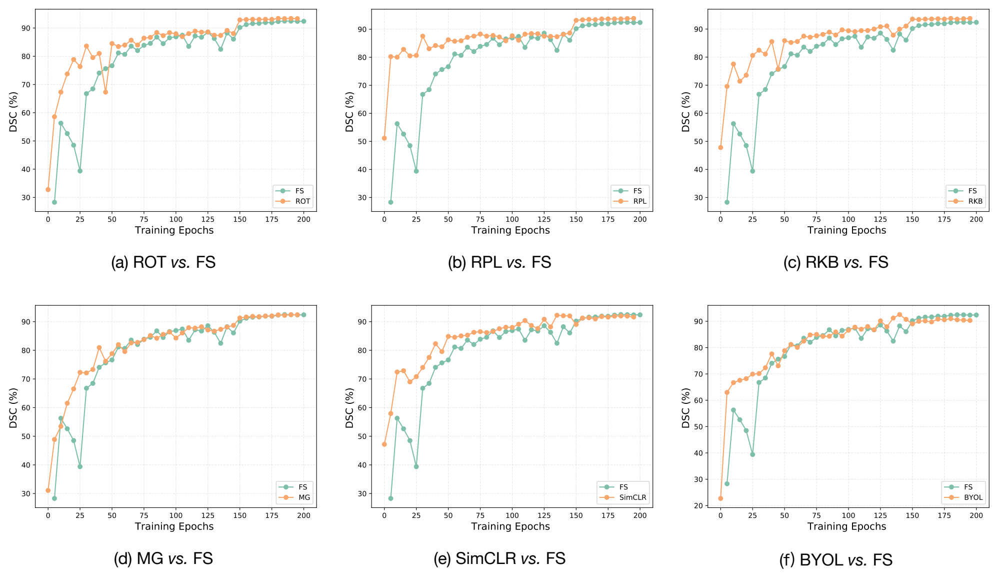

Figure 2: Comparisons of convergence speed between self-supervised pretraining methods (green) and the from-scratch baseline (orange) in LCS.

Table 2: The effect of network architectures in RKB. The number after "fc" stands for the number of hidden neurons. The "Rot-ver" and "Rot-hor" are short for vertically rotation task and horizontal rotation task respectively.

Network heads (# of params / M)			ACC (%) on the validation set		
Order classifier	Rot-ver classifier	Rot-hor classifier	Order	Rot-ver	Rot-hor
fc-1024-1024-100 (1.68)	fc-1024-1024-8 (1.58)	fc-1024-1024-8 (1.58)	35.52	49.09	51.29
fc-1024-1024-100 (1.68)	fc-1024-8 (0.53)	fc-1024-8 (0.53)	94.94	98.31	96.28

Table 3: The comparison of the online encoder and target encoder in BYOL across three target tasks.

Pretrained encoder	NCC (AUC / %)	NCS (DSC / %)	LCS (DSC / %)
Online	99.52	76.13	93.33
Target	98.64	75.37	94.43

4.1 Detailed convergence speed results

From Fig. 1 and Fig. 2, we find that most of the SSL methods facilitate more steady training and faster convergency speed, especially PRL and SimCLR. At the beginning epochs, most of the SSL methods show higher performance than the baseline, manifesting the benefits of pretrained representations in downstream tasks.

4.2 The architectures of Jigsaw, RKB and RKB+

In the process of implementing Jigsaw-puzzling-based methods, we find that the depth of network heads needs to be chosen carefully otherwise inappropriate architecture could cause model collapse. Taking RKB as an example, we report the head architectures and the corresponding validation results in Table 2. Even a little difference in the head architectures yields completely different outcomes. An obvious consequence is that the model with deeper classifier layers does not converge and achieves a high error rate on the validation set. Such a phenomenon suggests that the manually designed SSL multi-task might lay many pitfalls for user training.

4.3 The online and target encoder in BYOL.

There are two encoders in BYOL, where the online encoder is updated by backward propagation and the target encoder is updated through exponential momentum average. It is a common consensus that the features in the target encoder are smoother and preferable when transferred to the downstream tasks. However, our results in Table 3 suggest that the online encoder outperforms the target encoder in NCC and NCS.

5 Proof of the classification theoretical model

5.1 Proof of Theorem 1

Given two Gaussian distributions $Z^+ \sim N(\mu_1, \sigma^2)$ and $Z^- \sim N(\mu_2, \sigma^2)$, the sum of independent Gaussian variables follows the below Gaussian distribution:

$$\frac{\sum_{k=1}^{N_+} Z_k^+}{N^+} + \frac{\sum_{k=1}^{N_-} Z_k^-}{N^-} \sim N(\mu_1 + \mu_2, (\frac{1}{N_+} + \frac{1}{N_-})\sigma^2) \tag{1}$$

By applying Gaussian concentration inequality, we have:

$$\mathbb{P}(|\frac{\sum_{k=1}^{N_+} Z_k^+}{N^+} + \frac{\sum_{k=1}^{N_-} Z_k^-}{N^-} - (\mu_1 + \mu_2)| \geq \delta) \leq 2e^{\frac{-\delta^2}{2\sigma^2} \frac{1}{\frac{1}{N_+} + \frac{1}{N_-}}} \tag{2}$$

in which $\delta \geq 0$. Therefore, the estimated $\hat{\theta}_1 = \frac{\sum_{k=1}^{N^+} Z_k^+/N^+ + \sum_{k=1}^{N^-} Z_k^-/N^-}{2}$ obeys:

$$\mathbb{P}(|\hat{\theta}_1 - \frac{(\mu_1 + \mu_2)}{2}| \geq \frac{\delta}{2}) \leq e^{\frac{-\delta^2}{2\sigma^2} \frac{1}{\frac{1}{N_+} + \frac{1}{N_-}}} \tag{3}$$

Let $t = \frac{\delta}{2}$, we finally obtain the Theorem 1.

5.2 Proof of Theorem 2

Consider a binary classification problem between two classes ω_1 and ω_2, we set $t = \ln \frac{p(\omega_1)}{p(\omega_2)}$ and then the Bayesian decision function is:

$$\begin{cases} h(x) = -\ln \frac{p(x|w_1)}{p(x|w_2)} < t, x \in \omega_1 \\ h(x) = -\ln \frac{p(x|w_1)}{p(x|w_2)} > t, x \in \omega_2 \end{cases} \tag{4}$$

in which h is the so-called minus-log-likelihood ratio. Then, we introduce the moment generating function of h:

$$\phi_1(s) = \int_{-\infty}^{\infty} e^{sh} p(h|\omega_1) dh, \quad s \in [0, 1] \tag{5}$$

where we can regard $p(h|\omega_1)$ as the density function of h for x from ω_1. We note that $u(s) = -\ln \phi_1(s)$ and define another function:

$$p_g(g = h|\omega_1) = \frac{e^{sh} p(h|\omega_1)}{\phi_1(s)} \tag{6}$$

Obviously, $\int_{-\infty}^{\infty} p_g(g = h|\omega_1) dh = 1$. Rearranging Eq. 6 and replacing $\phi_1(s)$ with $u(s)$ gives:

$$p(h|\omega_1) = e^{-u(s)-sh} p_g(g = h|\omega_1) \tag{7}$$

From Eq. 4, we can see that x is classified as ω_2 when $h \in (t, \infty)$. The error rate ϵ_1 for ω_1 is defined as the probability of classifying x actually from ω_1 into ω_2:

$$\epsilon_1 = \int_t^{\infty} p(h|\omega_1) dh = e^{-u(s)} \int_t^{\infty} e^{-sh} p_g(g = h|\omega_1) dh \tag{8}$$

For $h \in (t, \infty)$ and $s \in [0, 1]$, $e^{-sh} \leq e^{-st}$. We can obtain the upper bound on ϵ_1 by applying this inequality to Eq. 8:

$$\epsilon_1 \leq e^{-u(s)-st} \tag{9}$$

Similarly, we can derive the upper bound on ϵ_2 as:

$$\epsilon_2 \leq e^{-u(s)+(1-s)t} \tag{10}$$

Unlike Eq. 5, we re-express $\phi_1(s)$ using x as the variable:

$$\phi_1(s) = \int_{\Gamma} e^{sh} p(x|\omega_1) dx = \int_{\Gamma} p(x|\omega_1)^{1-s} p(x|\omega_2)^s dx \tag{11}$$

in which, Γ is the distribution space of x. From Eq. 11, we obtain a new expression of $u(s)$:

$$u(s) = -\ln \phi_1(s) = -\ln \int_{\Gamma} p(x|\omega_1)^{1-s} p(x|\omega_2)^s dx \tag{12}$$

Since the optimal value of s is complicated to acquire, we set $s = 1/2$. As the result, Eq. 12 turns to the Bhattacharyya distance between ω_1 and ω_2, i.e., $u(1/2) = D_B(\omega_1, \omega_2)$. For a continuous distribution, the definition of $D_B(\omega_1, \omega_2)$ is:

$$D_B(\omega_1, \omega_2) = -\ln \int_{\Gamma} \sqrt{p(x|\omega_1)p(x|\omega_2)} dx \tag{13}$$

This means that we can use the Bhattacharyya distance to compute the error upper bound of the Bayesian classifier. If $x|\omega_1 \sim N(\mu_1, \sigma^2)$ and $x|\omega_2 \sim N(\mu_2, \sigma^2)$, $D_B(\omega_1, \omega_2) = \frac{1}{8}\frac{(\mu_2-\mu_1)^2}{\sigma^2}$ according to [Fukunaga, 2013]. Let $t = \ln \lambda$, the Eq. 9 and Eq. 10 can be expressed as:

$$\begin{cases} \epsilon_1 \leq e^{-u(s)-st} = \lambda^{-1/2} e^{-D_B} \\ \epsilon_2 \leq e^{-u(s)+(1-s)t} = \lambda^{1/2} e^{-D_B} \end{cases} \tag{14}$$

Finally, we prove Theorem 2 above.

References

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Zongwei Zhou, Vatsal Sodha, Jiaxuan Pang, Michael B Gotway, and Jianming Liang. Models genesis. *Medical image analysis*, 67:101840, 2021a.

Arnaud Arindra Adiyoso Setio, Alberto Traverso, Thomas De Bel, Moira SN Berens, Cas Van Den Bogaard, Piergiorgio Cerello, Hao Chen, Qi Dou, Maria Evelina Fantacci, Bram Geurts, et al. Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: the luna16 challenge. *Medical image analysis*, 42:1–13, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116, 2004.

Hong-Yu Zhou, Chixiang Lu, Sibei Yang, Xiaoguang Han, and Yizhou Yu. Preservational learning improves self-supervised medical image models by reconstructing diverse contexts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3499–3509, 2021b.

Samuel G Armato III, Geoffrey McLennan, Luc Bidaut, Michael F McNitt-Gray, Charles R Meyer, Anthony P Reeves, Binsheng Zhao, Denise R Aberle, Claudia I Henschke, Eric A Hoffman, et al. The lung image database consortium (lidc) and image database resource initiative (idri): a completed reference database of lung nodules on ct scans. *Medical physics*, 38(2):915–931, 2011.

Patrick Bilic, Patrick Ferdinand Christ, Eugene Vorontsov, Grzegorz Chlebus, Hao Chen, Qi Dou, Chi-Wing Fu, Xiao Han, Pheng-Ann Heng, Jürgen Hesser, et al. The liver tumor segmentation benchmark (lits). *arXiv preprint arXiv:1901.04056*, 2019.

Michela Antonelli, Annika Reinke, Spyridon Bakas, Keyvan Farahani, Annette Kopp-Schneider, Bennett A Landman, Geert Litjens, Bjoern Menze, Olaf Ronneberger, Ronald M Summers, et al. The medical segmentation decathlon. *Nature Communications*, 13(1):1–13, 2022.

Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2):203–211, 2021.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.