Efficient population based hyperparameter scheduling for medical image segmentation

No Author Given

No Institute Given

Abstract. The training hyperparameters (learning rate, augmentation policies, e.t.c) are key factors affecting the performance of deep networks for medical image segmentation. Manual or automatic hyperparameter optimization (HPO) is used to improve the performance. However, manual tuning is infeasible for a large number of parameters, and existing automatic HPO methods like Bayesian optimization are extremely time consuming. Moreover, they can only find a fixed set of hyperparameters. Population based training (PBT) has shown its ability to find dynamic hyperparameters and has fast search speed by using parallel training processes. However, it is still expensive for large 3D medical image datasets with limited GPUs, and the performance lower bound is unknown. In this paper, we focus on improving the network performance using hyperparameter scheduling via PBT with limited computation cost. The core idea is to train the network with a default setting from prior knowledge, and finetune using PBT based hyperparameter scheduling. Our method can achieve $1\% \sim 3\%$ performance improvements over default setting while only taking $3\% \sim 10\%$ computation cost of training from scratch using PBT.

Keywords: Hyperparameter optimization, Population based training, Bayesian optimization, Medical image segmentation

1 Introduction

The success of deep networks relies heavily on the correct setting of training hyperparameters. Hyperparameters like learning rate, choice of optimizers, and augmentation policies can greatly affect the performance of 3D medical image segmentation networks. This has led to the automatic hyperparameter optimization (HPO) algorithms. The basic algorithms include random search [3], grid search, and Bayesian optimization [8, 18, 19]. Random search trains the deep network with the randomly sampled hyperparameters, which is inefficient. Bayesian optimization samples the hyperparameters based on a Bayesian model, which is updated during searching. For medical imaging, Tran et al. [20] applied bayesian optimization (Gaussian Processes) on 2D echocardiography segmentation and trained 100 jobs. Yang et al. [22] and Nath et al. [14] used reinforcement learning and searched on 50 GPUs for 2 days. Hoopes et al. [7] proposed a specific hypernetwork to generate hyperparameters for registration tasks. The core of those

methods are the same: explore and interpolate the manifold of hyperparameter θ and deep network performance p , as shown in Fig. 1 (a). The major drawback is that the evaluation for each sampled hyperparameter requires expensive deep network retraining. Many works focus on reducing the evaluation cost by early stopping the network retraining [4, 11, 12]. For example, Bayesian optimization hyperband (BOHB) [4] samples hyperparameters using a Tree Parzen estimators (TPE) [2], and the hyperparameters with good performances are allocated more training budget for more accurate evaluations (Hyperband [12]). However, if the network requires long training epochs to converge, aggressive early stopping may cause inaccurate evaluation. Moreover, the searched hyperparameters are fixed for the network training, which is sub-optimal.

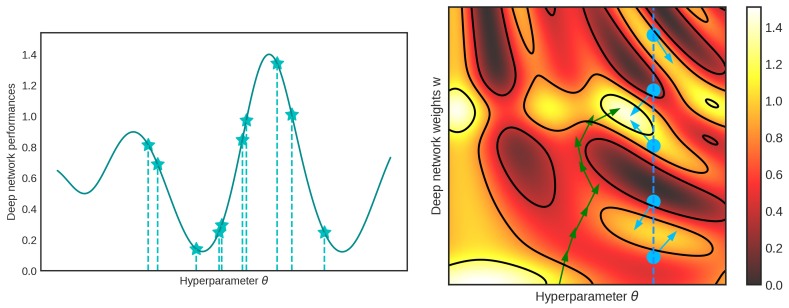

Fig. 1: (a) The hyperparameters θ and best validation performances p of the network trained with θ. Each dashed line represents a sample of hyperparameters. (b) The manifold when considering the deep network parameters (weights, w). The green arrow represents the PBT from scratch, and the blue dashed line represents training with a default setting. The blue dots are the validation checkpoints. The blue arrows represent PBT searching starting from validation checkpoints.

Population based training (PBT) [10, 15] solves the HPO problems by introducing parallel workers and evolutionary strategies. Each parallel training process (worker) trains the network with different hyperparameters for a step (e.g. 1 epoch). The workers with top performances keep the hyperparameters unchanged. The rest workers shall load the hyperparameters and network weights from the top performing ones (exploit), then continue training with mutated hyperparameters (explore). All the workers will continue and repeat the process. If a deep network needs to be trained for N epochs with certain hyperparameters, the total training epochs for PBT are also N if ideally without computation resource limit, since the workers are running in parallel. Meanwhile, the searched hyperparameters are dynamic across training steps. This method is suitable for training large scale deep networks and shows superior performance in reinforcement learning. However, the computation resource is limited in real practice (e.g. a station with

8 GPUs), and data parallel is usually used to increase the batch size for 3D medical images (training one network using all 8 GPUs, and no GPUs left for parallel workers). The training cost becomes $N \times W$, where W is the number of PBT workers and needs to be large enough to reach certain performance. Can we reduce N and adapt the early stopping methods for evaluation mentioned above? PBT evaluates different hyperparameters every single step (e.g. 1 epoch) and already uses "early stopping". To understand the problem, we plot the network performance manifold related to both the hyperparameters θ and deep network weights w (simulated in 2D for intuitive explanation), as shown in Fig. 1 (b). Fig. 1 (a) is a sub-space of Fig. 1 (b), where the deep network performance (the y axis) is the maximum performance of all w given a θ. The blue dashed line represents the training of w using certain θ and each blue dot means a validation checkpoint. The green arrow represents the best worker in each step for the PBT algorithm. The large training cost comes from the large number of steps needed to converge from scratch. Meanwhile, the green arrow explores the manifold from random initialled states without any convergence guarantee [23], thus the performance lower bound cannot be estimated beforehand.

The key to reduce training costs is to reduce the steps the network needs to converge from scratch. However, the "No Free Lunch Theorem" [17, 21] suggests that no optimization method offers a "shortcut" unless using prior knowledge about the problem. We observed that a major difference between medical image and natural image analysis tasks is that the targets (brain, lung, e.t.c) and modalities (MRI, CT, e.t.c) for medical images are limited, and the expert knowledge might be obtained and transferred. For example, nn-UNet [9] shows that hyperparameters generated by an expert understanding of the medical data can achieve outstanding performance on most medical segmentation challenges. In many scenarios, we already have a good set of hyperparameters for a task, and what we really need is an efficient and robust method to further improve the performance (e.g. compete in challenges). If we train the network using the hyperparameters from expert knowledge (denoted as the default setting) and w is updated along the dotted blue line in Fig. 1 (b), we can start from the validation checkpoints and use PBT for finetuning (blue arrows). This greatly accelerates the training and the results is at least as good as the default setting. Based on this observation, we propose a fast performance improvement method using PBT and finetuning. The proposed method can be applied to any given task and achieve significant performance improvement with limited computation cost. As far as we know, our method is (1) the first work applying PBT for large scale 3D medical image segmentation, and can (2) reduce training cost to $3\% \sim 10\%$ of the original PBT and makes it computationally feasible, (3) the performance lower bound is bounded by the default setting performance, thus making the whole process controllable. (4) The mutation step (explore the hyperparameter space) in original PBT relies on random heuristics and lacks theoretical guarantees [15], and we implemented the TPE based Bayesian sampler which is theoretically sound and does not need a heuristic exploring algorithm as in [6, 10]. TPE model has shown its efficacy and simplicity as used by BOHB [4], so we use TPE instead

of the Gaussian process as used in [15]. Besides, we implemented the PBT with sequential workers, where each worker uses all GPUs for data parallel and is more suitable for 3D medical imaging tasks. We performed experiments on four datasets from the MSD challenge [1] using two network structures. The results show the efficacy and simplicity of our proposed method.

2 Method

Our method can be added to any training pipeline. Conventionally, given default hyperparameters, the network is trained for N epochs and validated for V times (network weights are saved as checkpoints). The checkpoint with the best validation accuracy is deployed. Our method tries to improve the performance of the network from this training pipeline, and we have two steps: 1) select checkpoints to finetune 2) apply PBT to the selected checkpoints.

Checkpoint Selection

If only finetune from the checkpoint with the highest validation accuracy, the final network can only reach the local maximum around that point. We can start from multiple checkpoints to explore more local maximums, as shown in Fig. 1 (b). If the checkpoint has low validation accuracy, the checkpoint may be located at a bad position, and explore around is a waste of computation. If two checkpoints are too close, they may be located around the same local maximum, and the exploration is redundant. Given checkpoints $c_{e_1}, c_{e_2}, \cdots, c_{e_V}$ validated at epoch $e_1, e_2, \cdots, e_V$ with performances $p_{e_1}, p_{e_2}, \cdots, p_{e_V}$, we select a list of checkpoints $S = \{c_{e_i} | p_{e_i} > 0.95 * p_{e_m}\}$ with good enough performances, where $p_{e_m} = max(p_{e_1}, p_{e_2}, \cdots, p_{e_V})$. We chose three checkpoints $c_{e_i}, c_{e_j}, c_{e_k}, e_i < e_j < e_k$ from S, which satisfies

$$\mathop{\mathbf{max}}_{i,j,k}(\mathbf{min}(e_j - e_i, e_k - e_j)) \quad \forall i, j, k \in S, m \in \{i, j, k\}. \tag{1}$$

The epoch differences $e_j - e_i$ and $e_k - e_j$ is used as distances between the checkpoints, and we want to keep checkpoints as far as possible, so we maximize the minimum value of these two distances. Meanwhile, we make sure the best checkpoint c_{e_m} is among these three selected checkpoints ($m \in \{i, j, k\}$). If several sets of $\{i, j, k\}$ have the same value in Eq. 1, the set with $\mathbf{max}_{i,j,k}(\mathbf{max}(e_j - e_i, e_k - e_j))$ is selected.

Population Based Training with TPE

We run PBT starting from $c_{e_i}, c_{e_j}, c_{e_k}$. The algorithm is shown in Alg. 1. We use W = 27 workers, and each worker runs sequentially. In the first step, the first worker loads the checkpoint c_{e_i} (the same procedure for c_{e_j} and c_{e_k}) and randomly samples hyperparameters h from configuration space H. The network Φ is trained for B = 1 epoch with h. The validation result and h will be saved into a set R, and a TPE model will be fitted if the elements in R is large enough (the details of TPE fitting and sampling can be found in BOHB[1] [4]). All the following workers in this step will load checkpoint c_{e_i}. If the TPE model is not fitted

[1] https://github.com/automl/HpBandSter

or a random number $p \sim \text{Uniform}(0, 1)$ is smaller than $\sigma = 0.3$, h is randomly sampled; otherwise the TPE model is used. After all workers finish training and validation in the step, the top $\eta = 3$ workers ranked by validation accuracies will not change their hyperparameters or load weights from other workers. The rest $\text{W} - \eta$ workers will randomly load checkpoints from these top workers and sample new h. Then all workers will continue to the next step for in total $\text{S} = 50$ steps.

Algorithm 1 Population Based Training for Finutuning

Input: checkpoints $c_{e_i}, c_{e_j}, c_{e_k}$, network Φ, hyperparameter search space H, number of search steps S and epochs B in each step, number of workers W, number of top workers η, random rate σ

for $c \in \{c_{e_i}, c_{e_j}, c_{e_k}\}$ **do**
  Initialize TPE model $\text{M} = \emptyset$, performance record $\text{R} = \emptyset$
  Initialize top workers set $T_w = \emptyset$, checkpoints set $C_w = \emptyset$
  for s in $\{1, 2, \cdots, \text{S}\}$ **do**
    Initialize performance record $\text{SR} = \emptyset$
    for w in $\{1, 2, \cdots, \text{W}\}$ **do**
      if $w \notin T_w$ **then**
        if $\text{M} = \emptyset$ or $p \sim \text{Uniform}(0, 1) < \sigma$ **then**
          Sample random hyperparameters h from H
        else
          Sample h using M
        if $C_w \neq \emptyset$ **then**
          Load a random checkpoint from C_w into Φ
        else
          Load checkpoint c into Φ
      else
        Use the hyperparameters h of worker w
        Load checkpoint that corresponds to w from C_w
      Train Φ for B epochs with hyperparameters h
      Get validation performance p_w of Φ and save checkpoints
      $\text{SR} = \text{SR} \cup \{p_w\}$, $\text{R} = \text{R} \cup \{(p_w, h)\}$
      if $|R|_0 >$ Minimum samples needed for fitting TPE **then**
        Fit new TPE model M using R
    Find η top workers, $T_w = \{w | w \in$ top η workers in SR$\}$
    Define set C_w= {most recent checkpoint of $w | w \in T_w$}

3 Experiments

Datasets and Default Setting Four datasets from the MSD challenge [1] are used: Task01 Brain Tumour (484 multi-modal MR images), Task05 Prostate (32 MR/ADC images), Task06 Lung Tumour (64 CT images), and Task07 Pancreas Tumour (282 CT images). We train the standard U-Net [16] and the SegResNet [13] (1st in Brats18 challenge) with a sophisticated training pipeline with hyperparameters from DiNTS [5][2] (2st place in MSD live challenge). Each

[2] https://github.com/Project-MONAI/research-contributions/tree/master/DiNTS

Table 1: Hyperparamter search space H. The values in "Range" are the choices for "Categorical", and are the min and max values for others. {Aug} contains nine random augmentation probabilities. Details about augmentations and losses can be found at https://monai.io/ and DiNTS[2].

Parameter	Learning rate	Optimizer	Weight Decay	Loss Function	Background Crop Ratio	{Aug}
Distribution	LogUniform	Categorical	LogUniform	Categorical	Uniform	Uniform
Range	[1e-4, 0.2]	[Sgd, Adam]	[1e-5, 1e-1]	[DiceLoss, DiceCELoss, DiceFocalLoss]	[0.1, 0.9]	[0, 1]

dataset is equally split into five folds, and the network is trained on the first four (5000 epochs for Task05, Task06, and 1500 epochs for Task01, Task07) and validated (every 100 epochs) on the last fold. Our method tries to find the optimal hyperparameter schedule to fit data, and we report validation results.

Search Setting The hyperparameter search space H is shown in Table. 1. H defines how h is sampled using random sample and TPE model. We search 14 parameters: learning rate, optimizer, weight decay, loss function, background crop ratio (background patches sampling ratio), and 9 random augmentations probabilities. PBT starts from three checkpoints, and run with 27 workers (W=27) for S=50 steps (B=1, one epoch per step) using 8 NVIDIA V100 GPU.

PBT Finetuning Results The validation curve (average Dice score for all segmentation classes) of the U-Net and SegResNet on the four datasets are shown in Fig. 3. We show the best PBT validation score among W=27 workers in each step. The training using the default setting validates every 100 epochs, while PBT validates every 1 epoch for 50 epochs. We resume the default setting training (including optimizer states) from these three checkpoints and validates every epoch for 50 epochs ("default-continue"). The best Dice score of PBT finetuning in all the steps and the Dice score of the default setting are also listed in Fig. 3. We can see that PBT finetuning can achieve $1\% \sim 3\%$ Dice improvement over the default setting rapidly for most experiments, while the "default-continue" follows the trend of the default setting validation curve. Intuitively, the "default-continue" explores along the blue dashed line (y axis), while PBT can explore both x and y axes thus can reach better performance, as shown in Fig. 1 (b).

Comparison with PBT from scratch We also show a comparison with PBT from scratch. The validation curve of PBT from scratch for Task05 Prostate and Task06 Lung Tumour using U-Net are shown in Fig. 2 (total 5000 epochs, S=100, B=50, W=9 and 27 for Task05, W=9 only for Task06 due to the training cost). We can see that the PBT performance is influenced by the number of workers. PBT from scratch shows faster convergence and achieves better results with more workers (W=27) than the default setting. However, it is worse than the default setting if the worker number is small (W=9). This is also observed on the Task01 Brain Tumour (PBT W=9 Dice:0.68, default Dice:0.73) and Task07 Pancreas Tumour (PBT W=9 Dice:0.49 default Dice:0.53, details are in the supplementary material). Our PBT finetuning (W=27) greatly reduces the performance gap between default setting and PBT from scratch (W=27). However, our method

is also influenced by the worker number. A smaller worker number (W=9) will provide less improvement, but is still better than PBT from scratch (W=9) and the default setting. Compared with our PBT finetuning, PBT training from scratch can potentially find better local performance maximums than searching around default settings, however, the major obstacle is the computation cost. For large datasets like Task01 and Task07, training with default setting takes 9 hours on an 8 GPU station (1500 epochs), and a 27 worker PBT process will cost ten days (27×1500 epochs), while PBT finetuning only takes 20 hours ($3 \times 27 \times 50$ epochs) and achieves 10 times speedup. For Task05 and Task06, the default setting takes 0.67 and 9 hours respectively for 5000 epochs, PBT finetuning achieves 33 times speedup ($3 \times 27 \times 50$ epochs) compared to PBT (W=27) from scratch (27×5000 epochs).

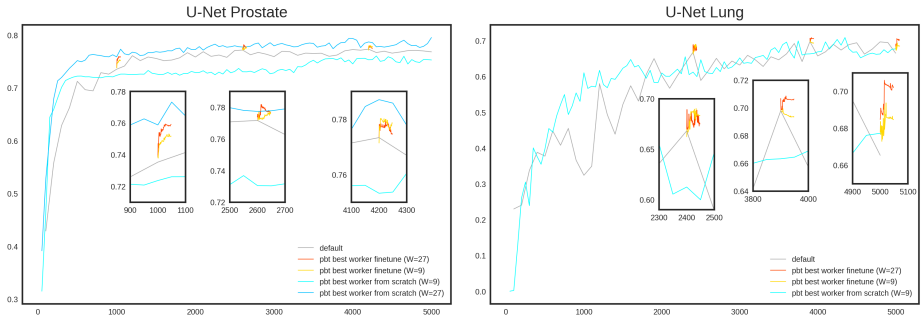

Fig. 2: Validation curve of PBT training from scratch and our PBT finetuning on prostate and lung datasets, with varying worker number W=9 and W=27.

4 Discussion and Conclusion

In this paper, we proposed an efficient HPO method using PBT. The method is simple, robust, and widely applicable to most training pipelines for a quick performance improvement. Our experiments show a $1\% \sim 3\%$ performance gain with $10 \sim 33$ times acceleration. PBT from scratch does not need any default setting and can potentially find much better results, but the performance is sensitive to settings like worker number, and the lower bound is unknown. More importantly, the training cost for large datasets is too expensive with limited GPUs (e.g. a typical 8 GPU station). Our method utilizes the prior knowledge to locate good starting points for PBT, and the performance lower bound is usually bounded by the starting points. The advantage is also its limitation, the short search time makes it hard to find a much better result than its starting point. Future work will include applying the methods to larger networks like DiNTS [5] and nn-UNet [9] and obtaining challenge leaderboard results.

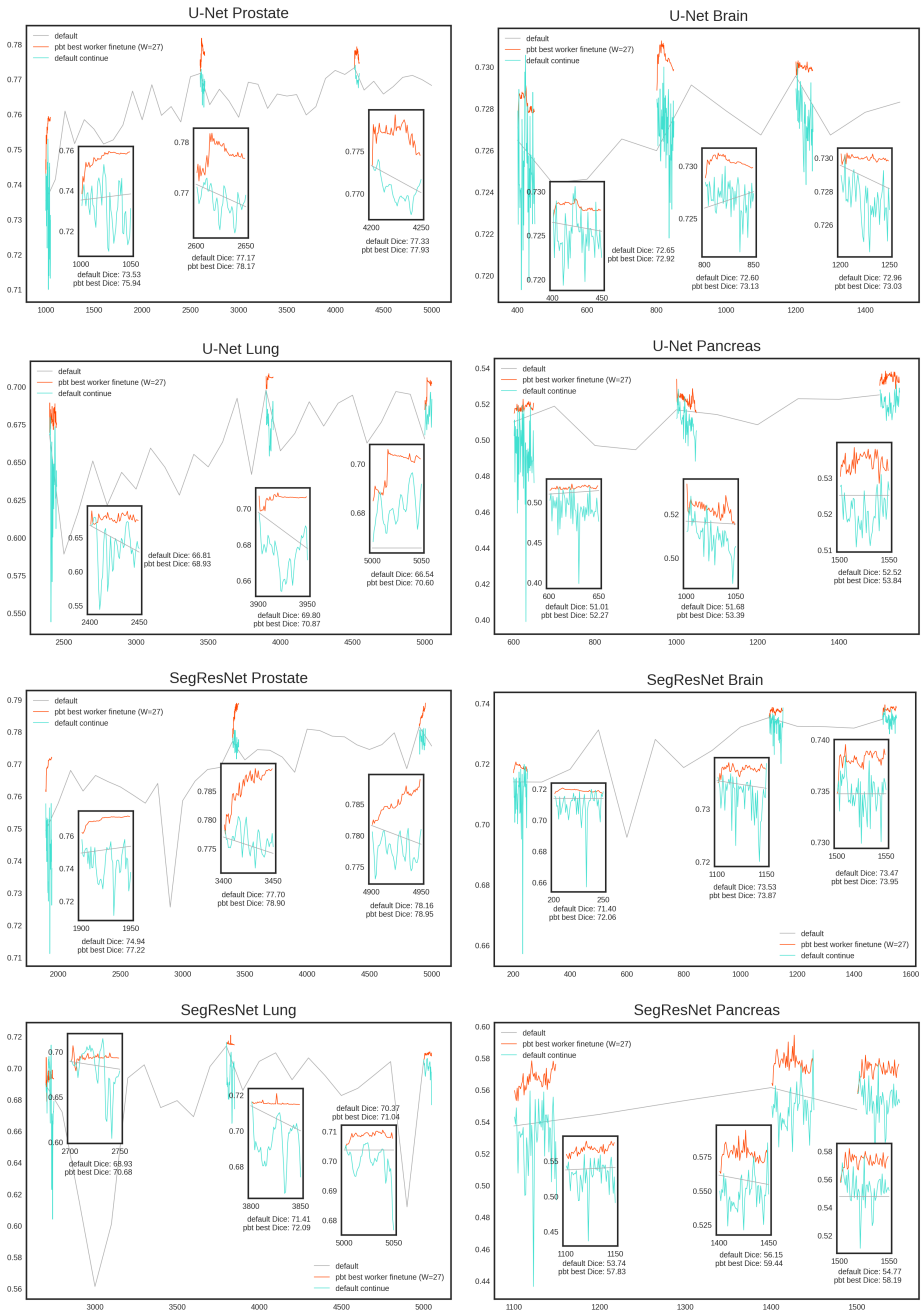

Fig. 3: Validation curve (Dice score, y-axis) of default hyperparameters ("default"), PBT finetuning ("pbt best worker finetune", results of the best worker in each step), and continue training using default hyperparameters ("default-continue"). x-axis is the training epoch number. The best Dice score of PBT finetuning in all the steps and the Dice score of the default training are listed besides the zoomed in windows for each checkpoint.

References

1. Antonelli, M., Reinke, A., Bakas, S., Farahani, K., Landman, B.A., Litjens, G., Menze, B., Ronneberger, O., Summers, R.M., van Ginneken, B., et al.: The medical segmentation decathlon. arXiv preprint arXiv:2106.05735 (2021)
2. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. Advances in Neural Information Processing Systems **24** (2011)
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research **13**(2) (2012)
4. Falkner, S., Klein, A., Hutter, F.: Bohb: Robust and efficient hyperparameter optimization at scale. In: International Conference on Machine Learning. pp. 1437–1446. PMLR (2018)
5. He, Y., Yang, D., Roth, H., Zhao, C., Xu, D.: Dints: Differentiable neural network topology search for 3d medical image segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5841–5850 (2021)
6. Ho, D., Liang, E., Chen, X., Stoica, I., Abbeel, P.: Population based augmentation: Efficient learning of augmentation policy schedules. In: International Conference on Machine Learning. pp. 2731–2741. PMLR (2019)
7. Hoopes, A., Hoffmann, M., Fischl, B., Guttag, J., Dalca, A.V.: Hypermorph: amortized hyperparameter learning for image registration. In: International Conference on Information Processing in Medical Imaging. pp. 3–17. Springer (2021)
8. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: International Conference on Learning and Intelligent Optimization. pp. 507–523. Springer (2011)
9. Isensee, F., Jaeger, P.F., Kohl, S.A., Petersen, J., Maier-Hein, K.H.: nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. Nature Methods **18**(2), 203–211 (2021)
10. Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W.M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al.: Population based training of neural networks. arXiv preprint arXiv:1711.09846 (2017)
11. Jamieson, K., Talwalkar, A.: Non-stochastic best arm identification and hyperparameter optimization. In: Artificial Intelligence and Statistics. pp. 240–248. PMLR (2016)
12. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. The Journal of Machine Learning Research **18**(1), 6765–6816 (2017)
13. Myronenko, A.: 3d mri brain tumor segmentation using autoencoder regularization. In: MICCAI Brainlesion Workshop. pp. 311–320. Springer (2018)
14. Nath, V., Yang, D., Hatamizadeh, A., Abidin, A.A., Myronenko, A., Roth, H.R., Xu, D.: The power of proxy data and proxy networks for hyper-parameter optimization in medical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 456–465. Springer (2021)
15. Parker-Holder, J., Nguyen, V., Roberts, S.J.: Provably efficient online hyperparameter optimization with population-based bandits. Advances in Neural Information Processing Systems **33**, 17200–17211 (2020)
16. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 234–241. Springer (2015)

17. Schaffer, C.: A conservation law for generalization performance. In: Machine Learning Proceedings, pp. 259–265. Elsevier (1994)
18. Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., Adams, R.: Scalable bayesian optimization using deep neural networks. In: International Conference on Machine Learning. pp. 2171–2180. PMLR (2015)
19. Srinivas, N., Krause, A., Kakade, S., Seeger, M.: Gaussian process optimization in the bandit detting: no tegret and experimental design. In: Proceedings of the 27th International Conference on Machine Learning. No. CONF, Omnipress (2010)
20. Tran, T., Stough, J.V., Zhang, X., Haggerty, C.M.: Bayesian optimization of 2d echocardiography ssgmentation. In: International Symposium on Biomedical Imaging (ISBI). pp. 1007–1011. IEEE (2021)
21. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (1997)
22. Yang, D., Roth, H., Xu, Z., Milletari, F., Zhang, L., Xu, D.: Searching learning strategy with reinforcement learning for 3d medical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 3–11. Springer (2019)
23. Yu, T., Zhu, H.: Hyper-parameter optimization: A review of algorithms and applications. arXiv preprint arXiv:2003.05689 (2020)