M3D-NCA: Robust 3D Segmentation with Built-In Quality Control

John Kalkhof(✉) and Anirban Mukhopadhyay

Darmstadt University of Technology, Karolinenplatz 5, 64289 Darmstadt, Germany
john.kalkhof@gris.tu-darmstadt.de

Abstract. Medical image segmentation relies heavily on large-scale deep learning models, such as UNet-based architectures. However, the real-world utility of such models is limited by their high computational requirements, which makes them impractical for resource-constrained environments such as primary care facilities and conflict zones. Furthermore, shifts in the imaging domain can render these models ineffective and even compromise patient safety if such errors go undetected. To address these challenges, we propose M3D-NCA, a novel methodology that leverages Neural Cellular Automata (NCA) segmentation for 3D medical images using *n-level* patchification. Moreover, we exploit the variance in M3D-NCA to develop a novel quality metric which can automatically detect errors in the segmentation process of NCAs. M3D-NCA outperforms the two magnitudes larger UNet models in hippocampus and prostate segmentation by 2% Dice and can be run on a Raspberry Pi 4 Model B (2 GB RAM). This highlights the potential of M3D-NCA as an effective and efficient alternative for medical image segmentation in resource-constrained environments.

Keywords: Neural Cellular Automata · Medical Image Segmentation · Automatic Quality Control

1 Introduction

Medical image segmentation is ruled by large machine learning models which require substantial infrastructure to be executed. These are variations of UNet-style [17] architectures that win numerous grand challenges [9]. This emerging trend raises concerns, as the utilization of such models is limited to scenarios with abundant resources, posing barriers to adoption in resource-limited settings. For example, conflict zones [10], low-income countries [3], and primary care facilities in rural areas [1] often lack the necessary infrastructure to support the deployment of these models, impeding access to critical medical services. Even when the infrastructure is in place, shifts in domains can cause the performance of deployed models to deteriorate, posing a risk to patient treatment decisions.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-43898-1_17.

H. Greenspan et al. (Eds.): MICCAI 2023, LNCS 14222, pp. 169–178, 2023.
https://doi.org/10.1007/978-3-031-43898-1_17

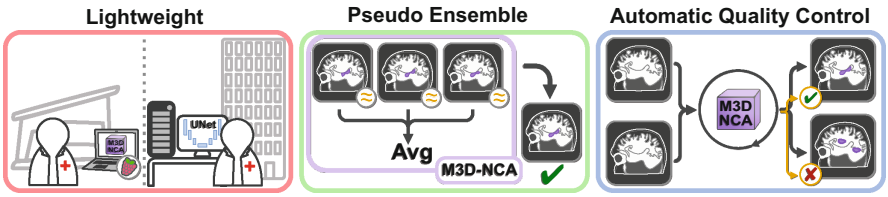

Fig. 1. M3D-NCA is lightweight, with a parameter count of less than 13k and can be run on a Raspberry Pi 4 Model B (2 GB RAM). The stochasticity enables a pseudo-ensemble effect that improves prediction performance. This variance also allows the calculation of a score that indicates the quality of the predictions.

To address this risk, automated quality control is essential [6], but it can be difficult and computationally expensive.

Neural Cellular Automata (NCA) [5] diverges strongly from most deep learning architectures. Inspired by cell communication, NCAs are one-cell models that communicate only with their direct neighbours. By iterating over each cell of an image, these relatively simple models, with often sizes of less than *13k parameters*, can reach complex global targets. By contrast, UNet-style models quickly reach *30m parameters* [11], limiting their area of application. Though several minimal UNet-style architectures with backbones such as EfficientNet [22], MobileNetV2 [18], ResNet18 [7] or VGG11 [20] exist, their performance is generally restricted by their limited size and still require several million parameters.

With Med-NCA, Kalkhof et al. [11] have shown that by iterating over two scales of the same image, high-resolution 2D medical image segmentation using NCAs is possible while reaching similar performance to UNet-style architectures. While this is a step in the right direction, the limitation to two-dimensional data and the fixed number of downscaling layers make this method inapplicable for many medical imaging scenarios and ultimately restricts its potential.

Naively adapting Med-NCA for three-dimensional inputs exponentially increases VRAM usage and *convergence becomes unstable*. We address these challenges with M3D-NCA, which takes NCA medical image segmentation to the third dimension and is illustrated in Fig. 1. Our **n-level architecture** addresses VRAM limitations by training on patches that are precisely adaptable to the dataset requirements. Due to the one-cell architecture of NCAs the **inference can be performed on the full-frame image**. Our *batch duplication scheme* stabilizes the loss across segmentation levels, enabling segmentation of high-resolution 3D volumes with NCAs. In addition, we propose a *pseudo-ensemble* technique that exploits the stochasticity of NCAs to generate multiple valid segmentations masks that, when averaged, improve performance by 0.5–1.3%. Moreover, by calculating the variance of these segmentations we obtain a quality assessment of the derived segmentation mask. Our *NCA quality metric (NQM)* detects between 50% (prostate) and 94.6% (hippocampus) of failure cases. M3D-NCA is *lightweigth* enough to be run on a Raspberry Pi 4 Model B (2 GB RAM).

We compare our proposed M3D-NCA against the UNet [17], minimal variations of UNet, Seg-NCA [19] and Med-NCA [11] on the medical segmentation decathlon [2] datasets for hippocampus and prostate. M3D-NCA consistently

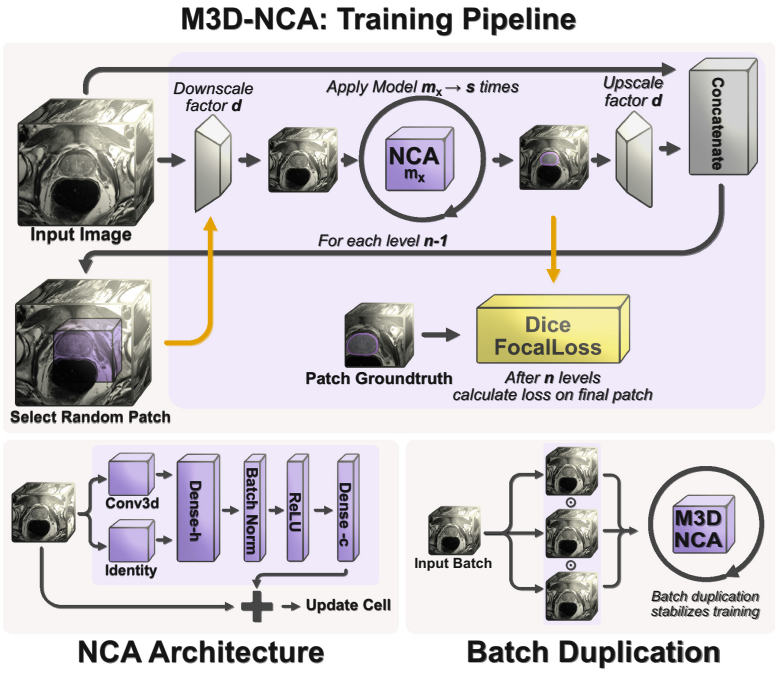

Fig. 2. The n-level M3D-NCA architecture uses *patchification* and *batch duplication* during training.

outperforms minimal UNet-style and other NCA architectures by at least 2.2% and 1.1% on the hippocampus and prostate, respectively, while being at least *two magnitudes smaller* than UNet-style models. However, the performance is still lower than the nnUNet by 0.6% and 6.3% Dice, the state-of-the-art auto ML pipeline for many medical image segmentation tasks. This could be due to the additional pre-and post-processing steps of the pipeline, as well as the extensive augmentation operations.

We make our complete framework available under github.com/MECLabTUDA/M3D-NCA, including the trained M3D-NCA models for both anatomies as they are only *56* KB in size.

2 Methodology

Cellular Automata (CA) are sets of typically hand-designed rules that are iteratively applied to each cell of a grid. They have been actively researched for decades, *Game Of Life* [4] being the most prominent example of them. Recently, this idea has been adapted by Gilpin et al. [5] to use neural networks as a representation of the update rule. These **Neural Cellular Automata** (NCA) are minimal and interact only locally (illustration of a 2D example can be found in the supplementary). Recent research has demonstrated the applicability of

NCAs to many different domains, including image generation tasks [12], self-classification [16], and even 2D medical image segmentation [11].

NCA segmentation in medical images faces the problem of high VRAM consumption during training. Our proposed M3D-NCA described in Sect. 2.1 solves this problem by performing segmentation on different scales of the image and using patches during training. In Sect. 2.3 we introduce a score that indicates segmentation quality by utilizing the variance of NCAs during inference.

2.1 M3D-NCA Training Pipeline

Our core design principle for M3D-NCA is to minimize the VRAM requirements. Images larger than 100×100, can quickly exceed 40 GB of VRAM, using a naive implementation of NCA, especially for three-dimensional configurations.

The training of M3D-NCA operates on different scales of the input image where the same model architecture m is applied, as illustrated in Fig. 2. The input image is first downscaled by the factor d multiplied by the number of layers n. If we consider a setup with an input size of $320 \times 320 \times 24$, a downscale factor of $d = 2$, and $n = 3$, the image is downscaled to $40 \times 40 \times 3$. As d and n exponentially decrease the image size, big images become manageable. On this smallest scale, our first NCA model m_1, which is constructed from our core architecture (Sect. 2.2), is iterated over for s steps, initializing the segmentation on the smallest scale. The output of this model gets upscaled by factor d and appended with the according higher resolution image patch. Then, a random patch is selected of size $40 \times 40 \times 3$, which the next model m_2 iterates over another s times. We repeat this patchification step $n - 1$ times until we reach the level with the highest resolution. We then perform the dice focal loss over the last remaining patch and the according ground truth patch. Changing the downscaling factor d and the number of layers n allows us to precisely control the VRAM required for training.

Batch Duplication: Training NCA models is inherently more unstable than classical machine learning models like the UNet, due to two main factors. First, stochastic cell activation can result in significant jumps in the loss trajectory, especially in the beginning of the training. Second, patchification in M3D-NCA can cause serious fluctuations in the loss function, especially with three or more layers, thus it may never converge properly.

The solution to this problem is to duplicate the batch input, meaning that the same input images are multiple times in each batch. While this limits the number of images per stack, it greatly improves convergence stability.

Pseudo Ensemble: The stochasticity of NCAs, caused by the random activation of cells gives them an inherent way of predicting multiple valid segmentation masks. We utilize this property by executing the trained model 10 times on the same data sample and then averaging over the outputs. We visualize the variance between several predictions in Fig. 3.

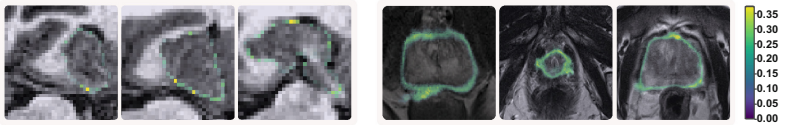

Fig. 3. Variance over 10 predictions on different samples of the hippocampus (left) and prostate dataset (right).

Once the model is trained, inference can be performed directly on the full-scale image. This is possible due to the one-cell architecture of NCAs, which allows them to be replicated across any image size, even after training.

2.2 M3D-NCA Core Architecture

The core architecture of M3D-NCA is optimized for simplicity. First, a convolution with a kernel size k is performed, which is appended with the identity of the current cell state of depth c resulting in state vector v of length $2 * c$. v thus contains information about the surrounding cells and the knowledge stored in the cell. v is then passed into a dense layer of size h, followed by a 3D BatchNorm layer and a ReLU. In the last step, another Dense layer is applied, which has the output size c, resulting in the output being of the same size as the input. Now the cell update can be performed, which adds the model's output to the previous state. Performing a full execution of the model requires it to be applied s times. In the standard configuration, the core NCA sets the hyperparameters to $k = 7$ for the first layer, and $k = 3$ for all the following ones. $c = 16$ and $h = 64$ results in a model size of 12480 parameters. The bigger k in the first level allows the model to detect low-frequency features, and c and h are chosen to limit VRAM requirements. The steps s are determined per level by $s = max(width, height, depth)/((k-1)/2)$, allowing the model to communicate once across the whole image.

2.3 Inherent Quality Control

The variance observed in the derived segmentation masks serves as a quantifiable indicator of the predicted segmentation. We expect that a higher variance value indicates data that is further away from our training domain and consequently may lead to poorer segmentation accuracy. Nevertheless, relying solely on this number is problematic, as the score obtained is affected by the size of the segmentation mask. To address this issue, we normalize the metric by dividing the sum of the standard deviation by the number of segmentation pixels.

The *NCA quality metric (NQM)* where v is an image volume and v_i are $N = 10$ different predictions of M3D-NCA for v is defined as follows:

$$NQM = \frac{\sum_{s \in SD}(s)}{\sum_{m \in \mu}(m)}, \quad SD = \sqrt{\frac{\sum_{i=1}^{N}(v_i - \mu)^2}{N}}, \quad \mu = \frac{\sum_{i=1}^{N} v_i}{N} \tag{1}$$

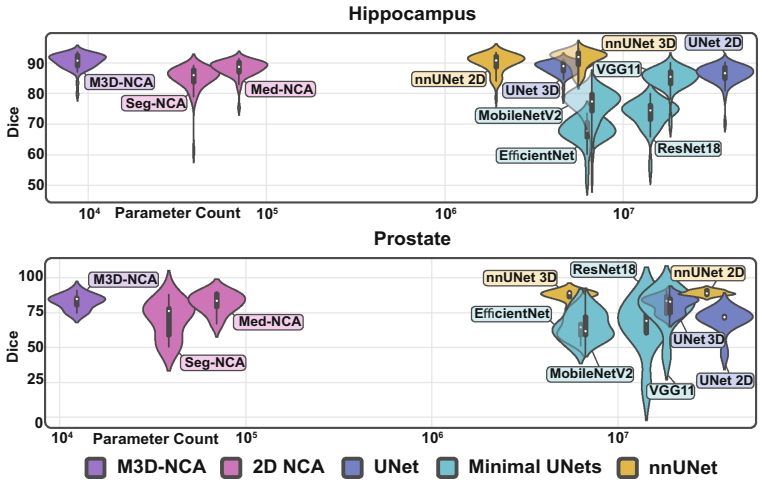

Fig. 4. Comparison of the Dice segmentation performance versus the number of parameters of NCA architectures, minimal UNets and the nnUNet (check supplementary for detailed numbers).

We calculate the relation between Dice and NQM by running a linear regression on the training dataset, which has been enriched with spike artifacts to extend the variance range. Using the regression, we derive the detection threshold for a given Dice value (e.g., $Dice > 0.8$). In clinical practice, this value would be based on the task and utility.

3 Experimental Results

The evaluation of the proposed M3D-NCA and baselines is performed on hippocampus (198 patients, $\sim 35 \times 50 \times 35$) and prostate (32 patients, $\sim 320 \times 320 \times 20$) datasets from the medical segmentation decathlon (medicaldecathlon.com) [2,21]. All experiments use the same 70% training, and 30% test split and are trained on an *Nvidia RTX 3090Ti* and an *Intel Core i7-12700*. We use the standard configuration of the *UNet* [14], *Segmentation Models Pytorch* [8] and *nnUNet* [9] packages for the implementation in PyTorch [13].

3.1 Comparison and Ablation

Our results in Fig. 4 show that despite their compactness, M3D-NCA performs comparably to much larger UNet models. UNet-style models instead tend to underperform when parameter constraints are imposed. While an advanced training strategy, such as the auto ML pipeline nnUNet, can alleviate this problem, it involves millions of parameters and requires a minimum of 4 GB of VRAM [9].

In contrast, our proposed M3D-NCA uses **two orders of magnitude fewer parameters**, reaching 90.5% and 82.9% Dice for hippocampus and prostate

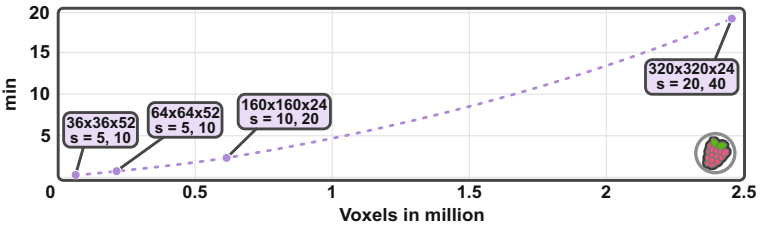

Fig. 5. Example inference times of a 2-level M3D-NCA architecture across different image scales on a *Raspberry Pi 4 Model B (2 GB RAM)*, where s defines the number of steps in each layer.

respectively. M3D-NCA outperforms all basic UNet-style models, falling short of the nnUNet by only 0.6% for hippocampus and 6.3% for prostate segmentation. Utilizing the 3D patient data enables M3D-NCA to outperform the 2D segmentation model Med-NCA in both cases by 2.4% and 1.1% Dice. The Seg-NCA [19] is due to its one-level architecture limited to small input images of the size 64×64, which for prostate results in a performance difference of 12.8% to our proposed M3D-NCA and 5.4% for hippocampus. We execute M3D-NCA on a Raspberry Pi 4 Model B (2 GB RAM) to demonstrate its suitability on resource-constrained systems, as shown in Fig. 5. Although our complete setup can be run on the Raspberry Pi 4, considerably larger images that exceed the device's 2 GB memory limit require further optimizations within the inference process. By asynchronously updating patches of the full image with an overlapping margin of $(k-1)/2$ we can circumvent this limitation while ensuring identical inference.

Table 1. Ablation results of M3D-NCA on the prostate dataset.

Lay.	Scale F.	# Param. ↓	Standard Dice ↑	w/o Batch Dup. Dice ↑	w/o Pseudo E. Dice ↑
2	4	12480	$\mathbf{0.829 \pm 0.051}$	$\mathbf{0.811 \pm 0.045}$	$\mathbf{0.824 \pm 0.051}$
3	2	16192	0.802 ± 0.038	0.723 ± 0.103	0.789 ± 0.041
4	2	**8880**	0.747 ± 0.112	0.704 ± 0.211	0.734 ± 0.117

The ablation study of M3D-NCA in Table 1 shows the importance of batch duplication during training, especially for larger numbers of layers. Without batch duplication, performance drops by 1.8–7.9% Dice. Increasing the number of layers reduces VRAM requirements for larger datasets, but comes with a trade-off where each additional layer reduces segmentation performance by 2.7–5.5% (with the 4-layer setup, a kernel size of 5 is used on the first level, otherwise the downscaled image would be too small). The pseudo-ensemble setup improves the performance of our models by 0.5–1.3% and makes the results more stable.

Fig. 6. Qualitative Results of M3D-NCA on hippocampus (left) and prostate (right).

The qualitative evaluation of M3D-NCA, illustrated in Fig. 6, shows that M3D-NCA produces accurate segmentations characterized by well-defined boundaries with no gaps or random pixels within the segmentation volume.

3.2 Automatic Quality Control

To evaluate how well M3D-NCA identifies failure cases through the NQM metric, we degrade the test data with artifacts using the *TorchIO* package [15]. More precisely, we use noise ($std = 0.5$), spike ($intensity = 5$) and ghosting ($num\_ghosts = 6$ and $intensity = 2.5$) artifacts to force the model to collapse (prediction/metric pairs can be found in the supplementary). We effectively identify 94.6% and 50% of failure cases (below 80% Dice) for hippocampus and prostate segmentation, respectively, as shown in Fig. 7. Although not all failure cases are identified for prostate, most false positives fall close to the threshold. Furthermore, the false negative rates of 4.6% (hippocampus) and 8.3% (prostate), highlights its value in identifying particularly poor segmentations.

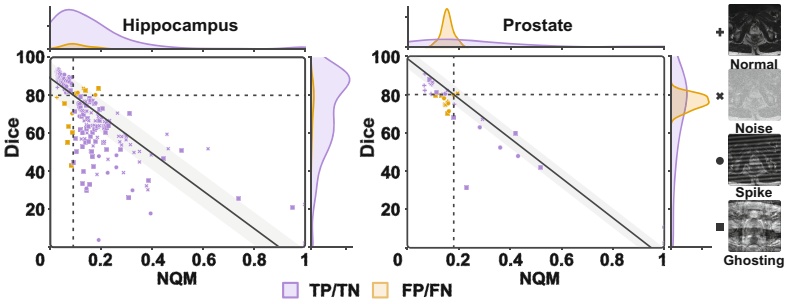

Fig. 7. The variance of NCAs during inference encapsulated in the NQM score indicates the quality of segmentation masks. In this example, the calculated threshold should detect predictions worse than 80% Dice. The distribution of FP/FN cases shows that most fall close to the threshold.

4 Conclusion

We introduce M3D-NCA, a Neural Cellular Automata-based training pipeline for achieving high-quality 3D segmentation. Due to the small model size with

under 13k parameters, M3D-NCA can be run on a Raspberry Pi 4 Model B (2 GB RAM). M3D-NCA solves the VRAM requirements for 3D inputs and the training instability issues that come along. In addition, we propose an NCA quality metric (NQM) that leverages the stochasticity of M3D-NCA to detect 50–94.6% of failure cases without additional overhead. Despite its small size, M3D-NCA outperforms UNet-style models and the 2D Med-NCA by 2% Dice on both datasets. This highlights the potential of M3D-NCAs for utilization in primary care facilities and conflict zones as a viable lightweight alternative.

References

1. Ajani, T.S., Imoize, A.L., Atayero, A.A.: An overview of machine learning within embedded and mobile devices-optimizations and applications. Sensors **21**(13), 4412 (2021)
2. Antonelli, M., et al.: The medical segmentation decathlon. Nat. Commun. **13**(1), 1–13 (2022)
3. Frija, G., et al.: How to improve access to medical imaging in low-and middle-income countries? EClinicalMedicine **38**, 101034 (2021)
4. Gardner, M.: The fantastic combinations of Jhon Conway's new solitaire game'life. Sci. Am. **223**, 20–123 (1970)
5. Gilpin, W.: Cellular automata as convolutional neural networks. Phys. Rev. E **100**(3), 032402 (2019)
6. González, C., et al.: Distance-based detection of out-of-distribution silent failures for COVID-19 lung lesion segmentation. Med. Image Anal. **82**, 102596 (2022)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
8. Iakubovskii, P.: Segmentation models pyTorch (2019). https://github.com/qubvel/segmentation_models.pytorch
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. Nat. Methods **18**(2), 203–211 (2021)
10. Jaff, D., Leatherman, S., Tawfik, L.: Improving quality of care in conflict settings: access and infrastructure are fundamental. Int. J. Qual. Health Care (2019)
11. Kalkhof, J., González, C., Mukhopadhyay, A.: Med-NCA: robust and lightweight segmentation with neural cellular automata. arXiv preprint arXiv:2302.03473 (2023)
12. Mordvintsev, A., Randazzo, E., Niklasson, E., Levin, M.: Growing neural cellular automata. Distill **5**(2), e23 (2020)
13. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
14. Perez-Garcia, F.: fepegar/unet: First published version of PyTorch U-Net, October 2019. https://doi.org/10.5281/zenodo.3522306
15. Pérez-García, F., Sparks, R., Ourselin, S.: Torchio: a python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. Comput. Methods Programs Biomed. 106236 (2021). https://doi.org/10.1016/j.cmpb.2021.106236, https://www.sciencedirect.com/science/article/pii/S0169260721003102

16. Randazzo, E., Mordvintsev, A., Niklasson, E., Levin, M., Greydanus, S.: Self-classifying mnist digits. Distill **5**(8), e00027-002 (2020)
17. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
18. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv 2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
19. Sandler, M., et al.: Image segmentation via cellular automata. arXiv e-prints pp. arXiv-2008 (2020)
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
21. Simpson, A.L., et al.: A large annotated medical image dataset for the development and evaluation of segmentation algorithms. arXiv preprint arXiv:1902.09063 (2019)
22. Tan, M., Le, Q.: Efficientnet: rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, pp. 6105–6114. PMLR (2019)