Investigation of Energy-efficient AI Model Architectures and Compression Techniques for "Green" Fetal Brain Segmentation

Szymon Mazurek[1,2][0009−0006−7557−0157], Monika Pytlarz[1][0000−0001−5319−1769], Sylwia Malec[1][0000−0002−3603−9930], and Alessandro Crimi[1,2][0000−0001−5397−6363]

1. Sano Centre for Computational Personalized Medicine, Nawojki 11, 30-072 Cracow, Poland https://www.sano.science
2. AGH University of Krakow, Adam Mickiewicz Avenue 30, 30-059 Cracow, Poland https://www.agh.edu.pl
a.crimi@sanoscience.org

Abstract. Artificial intelligence has contributed to advancements across various industries. However, the rapid growth of artificial intelligence technologies also raises concerns about their environmental impact, due to associated carbon footprints to train computational models. Fetal brain segmentation in medical imaging is challenging due to the small size of the fetal brain and the limited image quality of fast 2D sequences. Deep neural networks are a promising method to overcome this challenge. In this context, the construction of larger models requires extensive data and computing power, leading to high energy consumption. Our study aims to explore model architectures and compression techniques that promote energy efficiency by optimizing the trade-off between accuracy and energy consumption through various strategies such as lightweight network design, architecture search, and optimized distributed training tools. We have identified several effective strategies including optimization of data loading, modern optimizers, distributed training strategy implementation, and reduced floating point operations precision usage with light model architectures while tuning parameters according to available computer resources. Our findings demonstrate that these methods lead to satisfactory model performance with the low energy consumption during deep neural network training for medical image segmentation.

Keywords: medical imaging · segmentation · green learning · fetal brain · sustainable AI

1 Introduction

1.1 Fetal Brain Segmentation

Magnetic resonance imaging (MRI) is a popular non-invasive method for evaluating the development of the central nervous system of the fetus during pregnancy. In recent years, neuroimaging has become popular for studying the fetal

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

brain. However, manual segmentation of the structures of the fetal brain is time-consuming and subject to variability between observers. Therefore, researchers have used artificial intelligence to automate and standardize this process [5]. Fetal brain segmentation faces challenges due to the small size of the fetal brain, extra tissues that need to be distinguished, motion artifacts, and limited image quality from fast 2D sequences. Additionally, only narrow datasets that vary in image acquisition parameters are publicly available, which complicates the training of the Deep Learning (DL) algorithm. In terms of DL methods, various approaches have been tested, including unsupervised training, atlas fusion, deformation, and parametric methods. Many methods incorporate preliminary steps, such as fetal brain location and region of interest (ROI) cropping. Super-resolution reconstruction algorithms have enabled 3D segmentation, but standardization of these techniques is lacking. The majority of techniques make use of convolutional neural networks (CNNs), especially the U-Net architecture. Reconstruction algorithms and transfer learning methods are widely used to address data issues. Due to their ability to minimize partial volume effects, 3D segmentation techniques are becoming more and more common; but their performance depends on image quality [5].

1.2 "Green" Deep Learning

Over the past decade, significant advances have been made in artificial intelligence and machine learning (ML), driven largely by the accessibility of large datasets and the rise of DL. Deep neural networks (DNNs) play a key role in DL, and a prominent focus while developing a new DNN architecture is on achieving state-of-the-art (SOTA) results. However, the progress of the current SOTA is often achieved by increasing the complexity of the model, leading to a 300,000-fold increase in computational load in a span of six years [24]. Although these networks deliver impressive results, improving their capabilities requires substantial data and computational resources, resulting in high energy usage with notable economic and environmental implications. Consequently, there is a critical need for energy-efficient DL to address concerns related to finances, ecology, and practical usability [15]. Moreover, these models are extremely data-hungry, making their direct application to tasks, such as fetal brain segmentation, difficult. As a result, the exploration of their lightweight counterparts becomes a reasonable step towards addressing climate change; they will also facilitate the application of SOTA medical image analysis techniques in scenarios where energy is limited or costly, such as in developing nations and on portable devices relying on batteries. Moreover, they will encourage the adoption of improved training methods to replace the current conventional approach, which typically involves extensive searching for optimal hyperparameters and a trial-and-error process.

When looking for energy usage optimizations for DL, one could divide the areas of interest into the following categories: infrastructure and software, architectural design, efficient data use, optimization of training, and inference. The foundation for effective running of DL models is a reliable software and hardware

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

system, including a careful selection of libraries and the use of graphics processing units, tensor processing units, or neuromorphic computing [15,16]. Another crucial component in minimizing computational effort is the selection of the right architecture. Modeling efficiency can be achieved through the use of compact neural networks and various automation and assembly techniques [16,32]. Even lightweight and optimized architectures often require a large amount of data to achieve peak performance. To reduce the data-related cost of training, data augmentation and active learning can be used [16,32]. Various techniques have been suggested to reduce the expense of training itself, such as different initialization techniques, normalization, progressive training, and mixed precision training [16,32]. After successful training, the energy efficiency of inference should be considered. Common compression methods contain quantization, pruning, low-rank factorization, deployment sharing, and knowledge distillation [16,32]. The need for reducing energy usage in DL applications is recognized, also in the medical domain. Parsa et. al. [17] proposed an interesting approach that decomposes the diagnosis process into subtasks evaluated by separate networks of varying complexity, achieving remarkable energy savings. Yu et. al. [33] tested numerous machine learning techniques in clinical prediction tasks, exploring different approaches to reduce energy requirements for their training. Sathish et.al. [23] propose a model quantization technique in medical imaging tasks, achieving major energy usage reductions while performing inference on CPUs. The code is available in https://github.com/szmazurek/efficient_segmentation.

1.3 Contribution

We aim to evaluate the environmental impact of deep learning by looking at energy consumption during model training. Existing solutions concentrate only on one category, such as architecture design [30] or hardware acceleration [31]. In contrast, our study intends to thoroughly examine the impact of different optimization techniques, starting from architecture selection, efficient data usage, and training acceleration, leading to the evaluation of the model's environmental effect. We investigated a variety of energy-efficient techniques to apply to the segmentation of the fetal brain in MRI, to develop a model with the best ratio of Dice score to energy consumption. We draw several observations and recommendations for creating energy-aware DL algorithms from the obtained results. We hope that our findings will help guide future research in the domain, helping researchers and practitioners navigate the landscape of available techniques for energy-efficient medical DL.

2 Methods

2.1 Setup and hardware

All experiments were carried out using Python 3.11.5 with Pytorch 2.0.1 [18] and Lightning 2.0.3 libraries [7] with CUDA 11.7. Data loading and processing

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

were performed with the Monai 1.2.0 [3] library. Energy usage was tracked using Codecarbon 2.3.1 and logged with Weights and Biases 0.15.4. We use an HPC environment containing 4 Nvidia A100 GPUs with 40GB of RAM memory, 120 cores of 2 AMD EPYC 7742 64-core processors, and up to 500 GB of RAM memory.

2.2 Used datasets

For our project, we incorporated the Openneuro dataset, a library of 1241 manually traced fetal fMRI images of 207 fetuses [22]. To comply with BIDS standards, the authors merged 3D volumes (raw and mask) into a 4D time series file [27]. The masks were drawn in a single volume from a period of fetal stillness. Within pre-processing, we cleaned the data from files missing matching masks .nii or raw .nii, extracted 3D volumes from functional MRI (fMRI) times series, and sliced 3D volumes into 2D pairs of slices raw vs. mask. Furthermore, we used a second dataset of fMRI, T2-weighted, and diffusion-weighted MRI scans [6]. Then we merged these datasets, doing the subject-wise division. In total, we obtained 40945 2D slices, which were used for the experiments. Data were split by patient into training, validation, and test subsets. For the test 10% patients were randomly chosen. Another 10% was allocated from the remaining patients for validation. The rest was used during training.

2.3 Energy usage and performance measure

For tracking the relationship between performance and energy consumption, we measured the total energy consumed by the hardware used in training (CPUs, GPUs, and RAM memory) during the evaluation process. We chose Dice/kJ (kilojoules) as the metric describing this relationship.

2.4 Experimental design

We designed the experiments as follows: first, we aimed to establish a baseline performance. We chose U-Net [21] as our baseline due to its ubiquitous usage in medical image segmentation tasks. This model was optimized using the Adam optimizer with a learning rate of 0.001. DDP in the default version was chosen for the communication between GPUs. Floating point operations were reduced to mixed 16-bit bfloat precision. We decided to use an early stopping algorithm to prevent overfitting and stop training when no improvement in validation loss is achieved in 15 consecutive epochs. The best parameters of the model are saved and used later in the inference. The batch size was set to 128 images per GPU. Training with a larger batch size would be possible, however, using a too-large value can cause training instability and require tuning of other parameters that would offset the large gradient values effects.

Immediately after establishing the baseline, we switched to Attention-Squeeze-Unet, as it has shown a relatively small reduction in test Dice score compared to

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

U-Net, while drastically reducing training time, allowing us to conduct upcoming experiments faster. The rest of the configuration remained unchanged. We then proceeded to evaluate the techniques described in the later part of this section incrementally. That is, for a chosen technique, we ran the experiments and measured the performance. If it improved the Dice/kJ score, it was incorporated into the setup and the next technique was evaluated. It is important to note that these experiments were inspired by the MICCAI 2023 E2MIP [6], therefore with the experiments, we aimed to align with its evaluation criteria. Due to this fact, we chose the hardware setup used by the challenge authors. Additionally, we assumed that the algorithm we propose will be trained and evaluated on unknown data. We therefore avoid taking any solutions that require tuning specific to the used data distribution.

2.5 Evaluated techniques

Data caching and loader configuration During neural network training, the speed with which the data can be provided to the model is often a bottleneck. Training on large datasets involves a lot of I/O operations, as the data needs to be loaded from the memory. Also, the pre-processing applied to the data on the fly slows down the process. This can be alleviated by using data caching in memory, especially if computational resources allow it. We evaluated the potential solution to this problem, Monai's CacheDataset abstraction. It first preloads the data into RAM and applies deterministic transformations, resulting in gains in model throughput at the cost of increased memory consumption. Furthermore, we examined the impact of the configuration of data-loading utilities offered by popular machine-learning frameworks such as Pytorch:

- Number of workers determines the number of concurrent processes involved in accessing the data.
- Data prefetching is an operation of loading and pre-processing the data by each worker into a buffer, from where the data are immediately accessed when the model requests for it.
- Workers persistence is a term referring to the handling of parallel dataloading processes. When using it, worker processes are not destroyed upon the completion of an epoch, hence there is no re-spawn overhead.
- Memory pinning enables the allocation of a predetermined memory subspace from which the transfer of data to GPU is increased.

Hyperparameter search techniques We adopted the automatic learning rate tuning offered by the Lightning API to select the initial learning rate [7]. This was the only chosen automatic tuning method, as the full architecture and hyperparameter search are compute-costly procedures.

Data Augmentation Medical images can take advantage of number data augmentation techniques for natural image analysis such as geometric transformations (rotations, horizontal reflections, cropping, shifting), the addition of random noise, or gamma correction [13]. Medical image datasets should not have

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

been augmented via transformations influencing the color such as the modification of the saturation or the hue of the natural image; therefore, we applied simple geometric transforms, rotations, and flips, to reduce overfitting.

Efficient model architectures Architectural design and the overall size are the key determinants of model resource requirements. We explored the landscape of lightweight DL models. We chose architectures to evaluate based on their number of parameters and existing code implementations. Finally, we choose the following: MobileNetV3-small [12], MicroNet [4], EfficientNet [25], Squeeze-Unet [1] and Attention-Squeeze-Unet [19]. Several additional models were directly taken from or inspired by implementation in [29]: SQNet, LinkNet, SegNet, ENet, ERFNet, EDANet, ESPNetv2, FSSNet, ESNet, CGNet, DABNet, ContextNet and FPENet.

Quantization for Training and Inference Quantization is a technique for reducing model size that converts model weights from high-precision floating point to low-precision floating point or integer representations, such as 16-bit or 8-bit. By converting the weights of a model from a high-precision to a lower-precision representation, the model size and inference speed can be increased without sacrificing too much precision. Additionally, quantization improves the efficacy of a model by reducing memory bandwidth requirements and increasing cache utilization [2]. However, quantization can introduce new challenges and trade-offs between accuracy and model size, especially when using low-precision integer formats such as INT8.

Loss Functions The choice of the loss function can help the model both to achieve better final performance and to increase convergence speed. In the experiments, we evaluated the most popular loss functions used in training neural networks for segmentation, such as Dice, Binary Cross Entropy (BCE), and Matthews Correlation Coefficient (MCC).

Pruning Pruning is a known technique to reduce the model size by removing a subset of parameters. This is intended to increase throughput and lower the computation requirements. Usually, this reduction comes with the cost of reduced performance; therefore, the procedure usually involves iterations of applying pruning followed by fine-tuning the model to regain the performance. Pruning can be classified as unstructured or structured. Unstructured pruning aims to remove chosen weights without altering the network structure. The objective of structured pruning is to remove a group of parameters, thus reducing the size of neural networks. It also involves establishing the importance of the parameters to prune and the relationships between them. This is achieved using various algorithms, including the method evaluated in this study, a Dependency Graph (DepGraph). This method explicitly models the dependency between layers and comprehensively group coupled parameters for pruning, showing great results on benchmark tasks [8].

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

Gradient Averaging Gradient averaging stabilizes and speeds up training in distributed settings, when training CNNs with batch gradient descent. From the available methods, stochastic weight averaging (SWA) was chosen for evaluation [14]. This method tracks the learned parameters for every epoch as the training nears the end and replaces the final ones with the average of them. Research has consistently shown that performance gains were observed in various problems solved using neural networks with nearly no additional computational costs.

Choosing optimizer and its configuration To perform optimization in neural networks, various algorithms were proposed. The choice of the optimizer and its hyperparameters has a significant impact on the convergence of the model, training time, and generalizability. We oriented ourselves towards adaptive optimization algorithms, which can tune the learning rates per parameter during training based on gradients. The initial choice of learning rate less influences them, as it is only used mostly as an upper limit value for the aforementioned choice of the one specific for a given parameter. Therefore, we sought to examine the Novograd optimizer, a relatively novel method for adaptive optimization[10]. This algorithm allows for adaptive parameter updates and reduces the memory footprint of the Adam optimizer by half. It was also shown to be more robust to the choice of learning rates, therefore being the go-to choice when the data are unknown or when the cost of hyperparameter search is to be avoided. It can also be paired with AMSGrad [20], a solution to problems with convergence of Adam optimizer where learning rate updates were too aggressive, the closer the optima. We also evaluated the influence of the exponential decay rate for the first and second gradient moments estimates (β_1 and β_2).

PowerSGD PowerSDG is a low-rank gradient compressor based on power iteration that compresses gradients quickly, aggregates them using all-reduce, and achieves test performance comparable to stochastic gradient descent (SDG) [28].

Training parallelization Distributed data parallel (DDP) is a method for training parallelization of DL models based on copying the model to each computing device, performing the forward pass on a subset of training batches, and accumulating the results in the main process to update the model parameters. We evaluated its performance as well as the available configuration options offered by Lightning API implementation, such as static

We also evaluated Bagua, a set of distributed training algorithms[9]. The authors have demonstrated that these algorithms benefit the training speed compared to other strategies, including Pytorch DDP implementation. The solutions offered were incorporated via the lightning-bagua library, a plugin that allows us to use these algorithms with the Pytorch Lightning framework [7]. The algorithms offered by Bagua are related to distributed communication (bytegrad, asynchronous model average, improved all-reduce, low-precision decentralized SGD) and optimization algorithm (QAdam [26]).

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

3 Results and Discussion

Data caching was the first efficiency technique explored as the solutions were evaluated in an HPC cluster environment with significant available compute resources. Storing training and validation data in RAM after first loading from the disk led to shortening the time of epochs by nearly 400% - a significant improvement that offsets the initial cost of loading cache within a few epochs.

The use of the Novograd optimizer versus the Adam originally used has shown improvements in training speed.

We also experimented with setting different parameters β_1 and β_2. This led to slower training, drastically increasing the energy cost. These parameters could probably be tuned via an extensive hyperparameter search; however, energy constraints do not allow this in this setting.

The learning rate was chosen by an automatic search included in Pytorch Lightning. We decided to incorporate this step despite the benefits of using Novograd optimizer, which led to 2- 3% improvements in final performance. However, increasing the search time above 100 steps did not improve performance while increasing execution time. We also decided to incorporate learning rate scheduling and performing the learning rate decay by a factor of 10 when no improvement in validation loss was observed for the past 5 epochs, up to a minimum value of 10^{-6}. This also led to a slight improvement in the final performance, increasing the test Dice score by another 1-2%.

The next step was to choose the number of workers to load the data. We decided to check 4, 6, and 8 workers for the data loader to avoid unnecessary profiling runs. These experiments were done with every worker pre-fetching 2 batches of data. Six workers provided the best performance to energy consumed ratio. Going beyond a certain number of workers seems to decrease performance due to communication overhead, even if a suitable number of processor cores are available. Finally, we also decided to use pinned memory and persistent workers, as the memory resources allow for. This can also increase the throughput by removing the need to spawn worker processes every epoch and allocating memory for CPU to GPU data transfer.

Next, we add augmentation of the training images via rotation and random flipping. It led to improved performance at virtually no cost, increasing the test Dice score by nearly 5%. Up to this point, the experiments were conducted using a Dice loss. We evaluated the usefulness of MCC and BCE losses. They led to worse performance without any reduction in energy usage.

To further improve communication with DDP, we disabled the search for unused parameters with every training step and set the computational graph to static. These changes allowed to increase the model's throughput and reduce training time. Using a bucket view for gradient reduction between devices led to decreased performance without energy benefits.

For comparison with DDP, experiments were conducted with algorithms offered by Bagua. The speedup in our case was not significant. Compared to a properly configured basic DDP strategy, it has shown slightly lower performance and higher energy consumption.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

Next, we evaluated the usage of PowerSGD. We have observed significant performance drops, as the compression is lossy and its parameters need to be properly tuned to avoid these drops while maintaining the benefits of faster communication. Similar results were observed when we evaluated the SWA. Research has shown the benefits of increased performance with no performance loss; however, once again the length of the training had to be known to properly tune the algorithm's hyperparameters. We did not observe any performance benefits in our tests.

We also examined the effects of pruning. Iterative pruning of the least important parameters evaluated using DeGraph was performed. During tests, we observed drastic drops in model performance. We, therefore, concluded that pruning on models that are already relatively small (less than 10M parameters) has to be done with caution, as the capacity of the model to learn the patterns in the data can turn out to be too low after applying the technique.

3.1 Final model choice

As the pipeline and methods used were configured, we evaluated all the models listed. On the basis of these findings, we chose Attention-Squeeze-Unet as the final. The model has reached a high Dice score on the test data set relative to the baseline Unet. It has also shown the best energy per epoch ratio. To justify our choice, we present graphs of the performance of the model relative to energy consumption in Figures 2 to 4. We also report the segmentation to highlight qualitatively how noisy were the data, as depicted in Figure 1, where the overlay between an original MRI slice and a segmented results are showing, pointing out how cumbersome the data were and how challenging would be to obtain higher Dice score.

3.2 Conclusions and recommendations

We have shown that proper use of available methods can lead to satisfactory model performance while maintaining low energy consumption when training DNNs for medical image segmentation. In this study, we focused mostly on the training part of the process, since inference uses negligible amounts of energy when done only once. If the model is to be deployed and perform inference many times, then methods that were deemed too costly to optimize, such as pruning and quantization, can be considered, as the gains made for long-term operations can be substantial.

Recommendations for Energy-Efficient Fetal Brain Segmentation:

- Optimization of data loading - configuration of the data loading pipeline should be considered every time. Caching provides the most significant speedups but should be used with caution only in environments with no memory constraints.
- Choice of optimizer - modern optimizers reach the performance on par with the established ones while allowing for memory footprint reduction. Adaptive

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

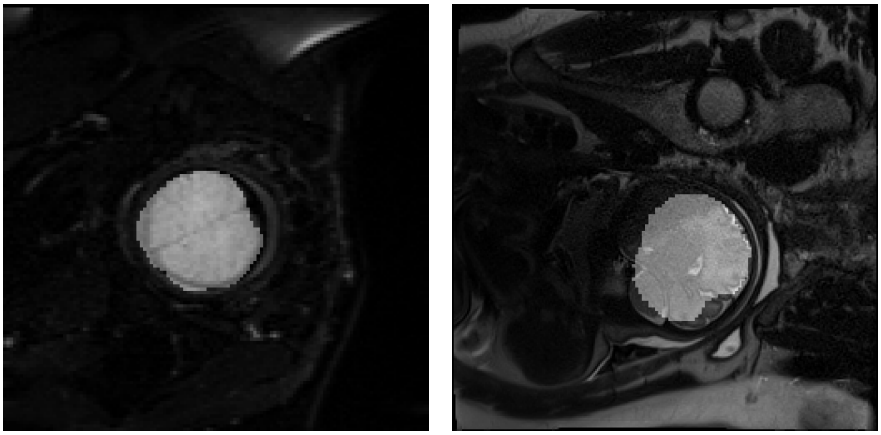

Fig. 1: Example DWI (left) and T2W (right) slices with segmentation mask obtained from final Attention-Squeeze-Unet trained with the optimized pipeline. Samples come from the E2MIP testing samples shared in the initial challenge announcement as an example.

methods also seem to be robust to the hyperparameter choice, leading to lower energy used for their configuration.
- Optimal distributed strategy and reducing floating point operation precision can offer significant throughput improvements without loss in performance.
- Model architecture - using already existing or creating custom architecture that uses a small number of parameters leads to faster training, smaller compute requirements, and potentially still maintains satisfying levels of performance.
- Usage of methods requiring parameter tuning should be considered when computing resources allow for their tuning, otherwise they may result in suboptimal performance.

This work can be further expanded on several levels. At first, the robustness of the presented techniques could be examined by applying them to different datasets and separate problems, such as classification or regression. This would allow us to see the generality of presented solutions and their tuning requirements for specific use cases.

In this study, due to the nature of the examined problem, we focused more on the training process. However, as we note, there are other techniques such as pruning that can dramatically impact the energy consumption of the inference process. Examination of such techniques and the creation of general guidelines would be especially relevant for applied DL in medicine, as the created models operate mostly in the inference mode.

The effect of hardware-specific customization could also be examined further. HPC environments are notoriously heterogeneous, with different accelerators, node connection links, and filesystems. Thus, when using such systems,

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

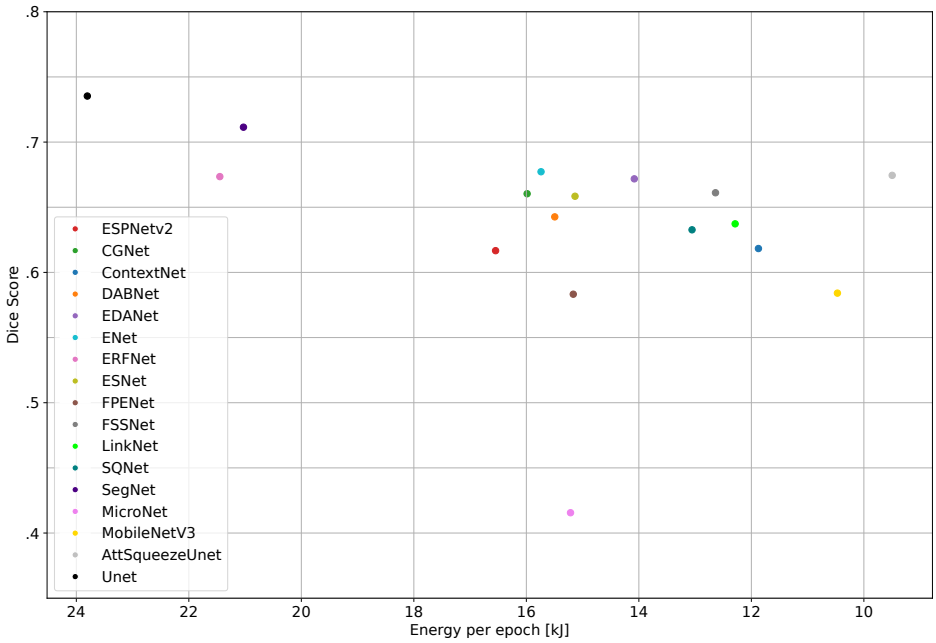

Fig. 2: Relationship of test Dice score in relation to the energy consumed per epoch of training.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

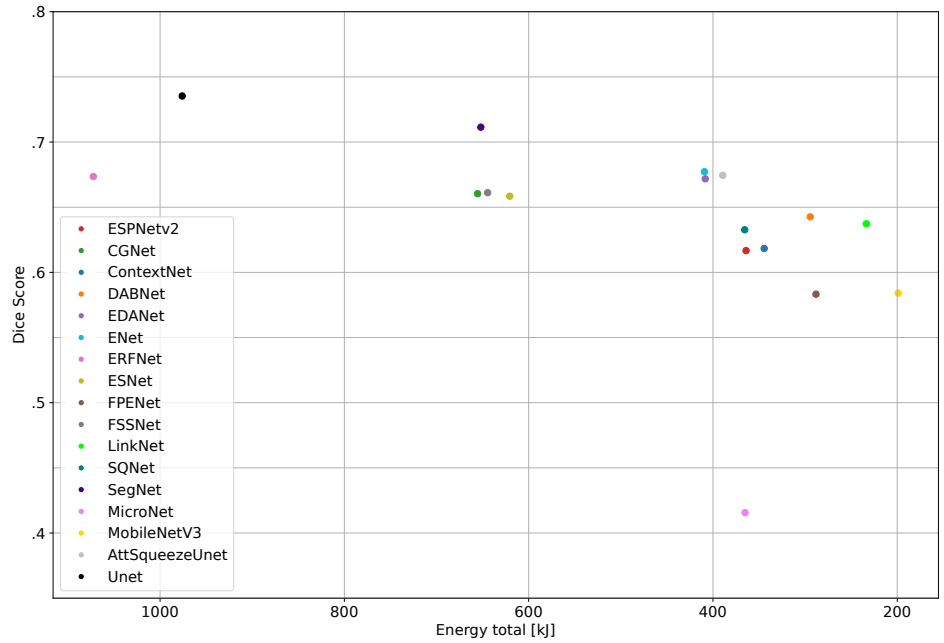

Fig. 3: Relationship of test Dice score in relation to the energy consumed during training.

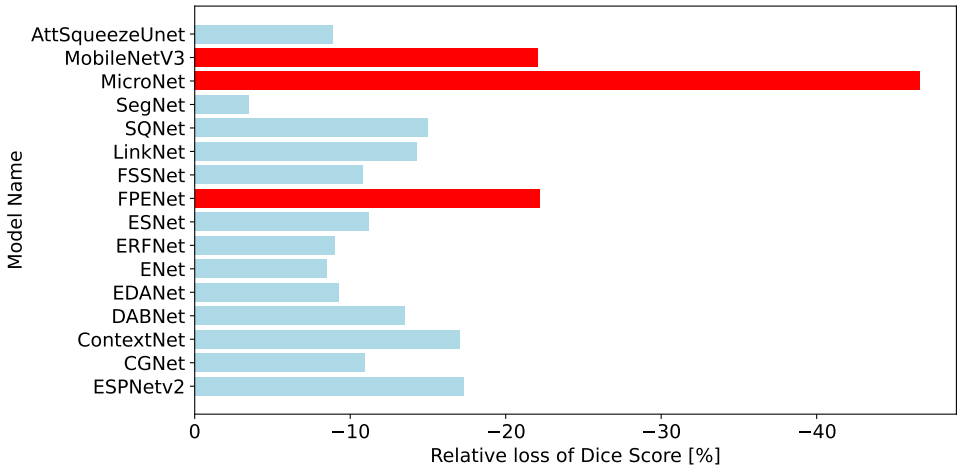

Fig. 4: Percentage of lost test Dice score for a given model tested relative to baseline value obtained by Unet.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

it would be beneficial to establish an approach for estimating optimal training configuration before starting full-scale experiments. Additionally, modern GPU accelerators have varying efficiency/energy curves, which means that there exists a "sweet spot" of clock frequency for a given GPU model that results in the best FLOPS to used energy ratio [11]. Furthermore, using custom ASIC platforms like TPU could lead to further energy consumption reductions, especially in the inference phase.

In summary, by enhancing the efficiency of machine learning algorithms through optimization techniques, the computational demands can be significantly reduced. This not only can accelerate model training but also minimizes the carbon footprint associated with the vast computational resources required. Taking into consideration this goal, we investigated and recommended some optimization ideas to reduce energy consumption and carbon emissions. We would like to stress the importance of sustainable computational approaches, and we would like to invite the scientific community to focus further on caring about carbon footprints.

Acknowledgments. This publication is supported by the European Union's Horizon 2020 research and innovation programme under grant agreement Sano No 857533. This publication is supported by Sano project carried out within the International Research Agendas programme of the Foundation for Polish Science, co-financed by the European Union under the European Regional Development Fund. This research was supported in part by the PLGrid infrastructure on the Athena computer cluster.

References

1. Beheshti, N., Johnsson, L.: Squeeze u-net: A memory and energy efficient image segmentation network. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. pp. 364–365 (2020)
2. Cai, Z., He, X., Sun, J., Vasconcelos, N.: Deep learning with low precision by half-wave gaussian quantization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5918–5926 (2017)
3. Cardoso, M.J., et al.: Monai: An open-source framework for deep learning in healthcare (2022). https://doi.org/10.48550/ARXIV.2211.02701, https://arxiv.org/abs/2211.02701
4. Chen, S.Y., Chen, G.S., Jing, W.P.: A miniaturized semantic segmentation method for remote sensing image. arXiv preprint arXiv:1810.11603 (2018)
5. Ciceri, T., Squarcina, L., Giubergia, A., Bertoldo, A., Brambilla, P., Peruzzo, D.: Review on deep learning fetal brain segmentation from magnetic resonance images. Artificial Intelligence in Medicine **143**, 102608 (Sep 2023). https://doi.org/10.1016/j.artmed.2023.102608
6. Faghihpirayesh, R.: E2MIP Challenge, MICCAI 2023. https://github.com/Faghihpirayesh/E2MIP_Challenge_FetalBrainSegmentation (2023)
7. Falcon, W., The PyTorch Lightning team: PyTorch Lightning (2019). https://doi.org/10.5281/zenodo.3828935
8. Fang, G., Ma, X., Song, M., Mi, M.B., Wang, X.: Depgraph: Towards any structural pruning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16091–16101 (2023)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

9. Gan, S., Lian, X., Wang, R., Chang, J., Liu, C., Shi, H., Zhang, S., Li, X., Sun, T., Jiang, J., Yuan, B., Yang, S., Liu, J., Zhang, C.: BAGUA: scaling up distributed learning with system relaxations (2021), https://arxiv.org/abs/2107.01499
10. Ginsburg, B., Castonguay, P., Hrinchuk, O., Kuchaiev, O., Lavrukhin, V., Leary, R., Li, J., Nguyen, H., Cohen, J.M.: Stochastic gradient methods with layer-wise adaptive moments for training of deep networks (2019)
11. Hodak, M., Gorkovenko, M., Dholakia, A.: Towards power efficiency in deep learning on data center hardware. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 1814–1820 (2019). https://doi.org/10.1109/BigData47090.2019.9005632
12. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1314–1324 (2019)
13. Hussain, Z., Gimenez, F., Yi, D., Rubin, D.: Differential data augmentation techniques for medical imaging classification tasks. In: AMIA annual symposium proceedings. vol. 2017, p. 979 (2017)
14. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D.P., Wilson, A.G.: Averaging weights leads to wider optima and better generalization (2018)
15. Mehlin, V., Schacht, S., Lanquillon, C.: Towards energy-efficient deep learning: An overview of energy-efficient approaches along the deep learning lifecycle. arXiv preprint arXiv:2303.01980 (2023)
16. Menghani, G.: Efficient deep learning: A survey on making deep learning models smaller, faster, and better. ACM Comput. Surv. **55**(12) (mar 2023)
17. Parsa, M., Panda, P., Sen, S., Roy, K.: Staged inference using conditional deep learning for energy efficient real-time smart diagnosis. In: 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). pp. 78–81 (2017). https://doi.org/10.1109/EMBC.2017.8036767
18. Paszke, A., et al.: Pytorch: An imperative style, high-performance deep learning library (2019)
19. Pennisi, A., Bloisi, D.D., Suriani, V., Nardi, D., Facchiano, A., Giampetruzzi, A.R.: Skin lesion area segmentation using attention squeeze u-net for embedded devices. Journal of Digital Imaging **35**(5), 1217–1230 (2022)
20. Phuong, T.T., Phong, L.T.: On the convergence proof of amsgrad and a new version (2019), http://arxiv.org/abs/1904.03590
21. 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.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. pp. 234–241. Springer International Publishing (2015)
22. Rutherford, S., Sturmfels, P., Angstadt, M., Hect, J., Wiens, J., van den Heuvel, M.I., Scheinost, D., Sripada, C., Thomason, M.: Automated brain masking of fetal functional MRI with open data. Neuroinformatics **20**(1), 173–185 (jun 2021). https://doi.org/10.1007/s12021-021-09528-5
23. Sathish, R., Khare, S., Sheet, D.: Verifiable and energy efficient medical image analysis with quantised self-attentive deep neural networks. In: Albarqouni, S., Bakas, S., Bano, S., Cardoso, M.J., Khanal, B., Landman, B., Li, X., Qin, C., Rekik, I., Rieke, N., Roth, H., Sheet, D., Xu, D. (eds.) Distributed, Collaborative, and Federated Learning, and Affordable AI and Healthcare for Resource Diverse Global Health. pp. 178–189. Springer Nature Switzerland, Cham (2022)
24. Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green ai. Commun. ACM **63**(12), 54–63 (nov 2020). https://doi.org/10.1145/3381831, https://doi.org/10.1145/3381831

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5

25. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114 (2019)
26. Tang, H., Gan, S., Awan, A.A., Rajbhandari, S., Li, C., Lian, X., Liu, J., Zhang, C., He, Y.: 1-bit adam: Communication efficient large-scale training with adam's convergence speed (2021), https://arxiv.org/abs/2102.02888
27. Turk, E., et al.: Functional connectome of the fetal brain. The Journal of Neuroscience **39**(49), 9716–9724 (2019)
28. Vogels, T., Karimireddy, S.P., Jaggi, M.: Powersgd: Practical low-rank gradient compression for distributed optimization. Advances in Neural Information Processing Systems **32** (2019)
29. Wang, Y.: Efficient-segmentation-networks pytorch implementation. https://github.com/xiaoyufenfei/Efficient-Segmentation-Networks (2019)
30. Wu, D., et al.: Lightnet: A novel lightweight convolutional network for brain tumor segmentation in healthcare. IEEE Journal of Biomedical and Health Informatics (2023). https://doi.org/10.1109/JBHI.2023.3297227
31. Xiong, S., Wu, G., Fan, X., Feng, X., Huang, Z., Cao, W., Zhou, X., Ding, S., Yu, J., Wang, L., Shi, Z.: Mri-based brain tumor segmentation using FPGA-accelerated neural network. BMC bioinformatics **22**(1), 421 (September 2021)
32. Xu, J., et al.: A survey on green deep learning. arXiv preprint arXiv:2111.05193 (nov 2021)
33. Yu, J.R., Chen, C.H., Huang, T.W., Lu, J.J., Chung, C.R., Lin, T.W., Wu, M.H., Tseng, Y.J., Wang, H.Y.: Energy efficiency of inference algorithms for clinical laboratory data sets: Green artificial intelligence study. J Med Internet Res **24**(1), e28036 (Jan 2022). https://doi.org/10.2196/28036

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63772-8_5