Energy Savings Playing the Lottery

Marco Schnell
marco.schnell@fau.de
Nora Gourmelon
nora.gourmelon@fau.de
Vincent Christlein
vincent.christlein@fau.de
Andreas Maier
andreas.maier@fau.de
Mathias Seuret
mathias.seuret@fau.de

Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg, 9058 Erlangen, Germany

Abstract

The rapid development of modern neural networks has led to highly over-parameterized models, resulting in excessive memory usage, computation, and energy consumption at inference time. In this paper, we propose a structured pruning method inspired by the Lottery Ticket Hypothesis, aiming to reduce the network size while preserving accuracy. Our method removes entire neurons based on a magnitude-based selection criterion – unlike the conventional unstructured approach of setting weights to zero via the utilization of binary masks. Hereby, we demonstrate the efficacy of magnitude-based layer and neuron selection techniques that guide our structured pruning algorithm without the necessity of complex search patterns. We validate our method on two distinct scenarios: the well known CIFAR-100 dataset, and a document image analysis task. We evaluate the benefits of our methodology using GPU-based energy measurements and show that our pruned networks can reduce the energy consumption per sample by more than 40 % [Wh/sample], with comparable or slightly superior test accuracies. These findings highlight the potential of structured pruning to create energy-efficient neural networks suitable for deployment in resource-constrained environments.

1 Introduction

Artificial Neural Networks (ANNs) have demonstrated significant success in real-world applications such as object recognition, image classification, autonomous driving, and natural language processing [2, 6, 22, 37]. Recent advancements, such as OpenAI's ChatGPT, have increased public interest in the possibilities enabled by ANNs [40]. However, the growing adoption of such models is accompanied by significantly higher energy requirements.

A core issue behind this inefficiency is the large number of parameters in state-of-the-art ANNs, which lead to impressive accuracy but also to substantial memory and computational resource demands [2, 22, 37, 42]. Consequently, energy consumption increased, particularly in over-parameterized Deep Neural Networks (DNNs) [6, 31, 40, 41, 43]. Considering that

the energy sector contributes approximately 34 % to global greenhouse gas emissions, there is a pressing need for more energy-efficient ANNs [24].

Among various compression techniques, knowledge distillation [14] and pruning [2, 6, 7, 9, 12, 23, 42] stand out. However, knowledge distillation may not sufficiently eliminate redundancy and achieve significant compression [1, 3], making pruning a more promising avenue for compression. The Lottery Ticket Hypothesis (LTH), introduced by Frankle and Carbin [7], demonstrated that sparse sub-networks derived from a larger network can achieve similar or better performance. While LTH traditionally applies unstructured pruning (i. e., setting weights to zero), our work instead explores the effect of completely removing parameters via structured pruning [31]. Moreover, the focus of this study is the measured reduction in energy consumption resulting from the implementation of structured pruning techniques. In addition, we focus on convolutional layers, known to be energy-intensive [41], and apply a novel combination of structured pruning and LTH principles.

Our methodology integrates the iterative pruning process with reinitialization of weights from the original network, as proposed by the LTH, with the concept of structured pruning to remove neurons, as outlined by Hu et al. [16], without necessitating the implementation of sophisticated layer and neuron selection strategies. Instead, we employ a straightforward approach of searching for the *lowest mean weights* between neurons, as these neurons have a limited impact on the network's output. We show that this strategy can produce sub-networks with significantly fewer neurons while retaining – or even improving test accuracy. Furthermore, we validate our method using real GPU energy measurements during inference, demonstrating that energy consumption per sample can be reduced by over 40 % without compromising accuracy. This reduction not only lowers the environmental footprint of Neural Networks (NNs), but also makes them more cost-effective for practical deployment.

2 Related Work

2.1 Pruning

The concept of pruning dates back to the 1960s [17, 18]. A major milestone was achieved by LeCun et al. [23], who introduced weight elimination using second-order derivatives. Later, Han et al. [12] proposed pruning small-magnitude weights to reduce model complexity. Recent works have explored various pruning strategies that achieve efficiency gains with minimal performance loss [16, 26, 28]. In particular, Hu et al. [16] developed a structured pruning technique based on the Average Percentage of Zeros (APoZ), identifying and removing neurons with minimal activation on a validation set. They found that reinitializing pruned networks with original weights prior to retraining helped retain accuracy, provided that pruning steps were not too aggressive.

In the context of transfer learning, Molchanov et al. [28] introduced pruning strategies that remove entire parameters from the network.

Pruning techniques are commonly classified as either unstructured or structured [2, 31, 32]. Unstructured pruning zeroes individual weights, which can reduce storage, but does not necessarily lead to computational gains unless the hardware supports sparse matrices [8]. Structured pruning, by contrast, removes neurons or entire layers, enabling real speedups and memory savings [26]. Structured pruning also impacts energy consumption, which in turn affects carbon emissions [38]. For instance, Widmann [38] report energy savings of 59 % with only 3 % accuracy loss when pruning 75 % of a network using structured APoZ

pruning. Conversely, unstructured pruning alone showed no energy benefit unless heavily optimized [33]. Timing of pruning also matters. While static pruning removes neurons after training, dynamic pruning eliminates them during training [6]. In the context of structured pruning, it is essential that the tensor sizes between layers are consistent, which may necessitate pruning multiple layers, e.g., in the case of batch-normalization or attention heads in transformers [15]. It is imperative to note that certain components may exhibit heightened sensitivity to pruning, potentially resulting in substantial performance implications [10].

2.2 Lottery Ticket Hypothesis

The LTH introduced by Frankle and Carbin [7] hypothesizes that dense, over-parameterized networks contain sparse sub-networks that can be trained to match original accuracy. The identification of these sub-networks is achieved through an iterative pruning process, resetting remaining weights to their initial (prior-training) values and retraining the network. This is achieved by masking zeroed connections from previous iterations. Therefore, LTH is traditionally unstructured.Recent studies have investigated stability enhancements, alongside the acquisition of insight into the pruning mechanism and the extension of applications to domains such as Natural Language Processing (NLP) and Spike Neural Networks (SNNs) [2, 8, 20, 27].A notable extension of LTH is the Early-Bird (EB) ticket concept [43], which seeks to identify winning tickets early in training through their proposed mask distance metric and reduced precision schemes, avoiding the need for full retraining cycles.

2.3 Quantization

Network quantization reduces the numerical precision of weights and / or activations, e.g., from 32-bit floats to 8-bit integers [6]. This has been shown to improve memory usage, computation time, and bandwidth, while concomitantly achieving a substantial reduction in energy consumption [6, 11]. Gholami [10] reported that the use of 8-bit integer quantization resulted in speed enhancements of up to 5× and energy savings of up to 30×. Because quantization works independently, it can be used in combination with other technologies such as pruning or knowledge distillation to optimize benefits [6].

2.4 Neural Architecture Search

Neural Architecture Search (NAS) automates the design of efficient network architectures within a predefined space [5]. Unlike pruning, which reduces an existing pre-trained model, NAS constructs a new model before training from the ground up [4, 5].

However, NAS and pruning are not mutually exclusive [4, 25]. For example, NAS can be used to design an initial compact and efficient architecture, which can then be further refined through pruning to eliminate redundancies and increase computational efficiency [4].

2.5 Energy Measurements

There is increasing attention on the energy consumption of neural networks, considering aspects such as network type (e.g., SNNs vs. DNNs) and the energy used during training, pruning, and inference [20]. Energy consumption can be assessed through estimation tools like the *CodeCarbon emissions tracker* [31, 33] or through real-world measurement devices [43]. In a study by Yang et al. [41], hardware-based energy measurements were

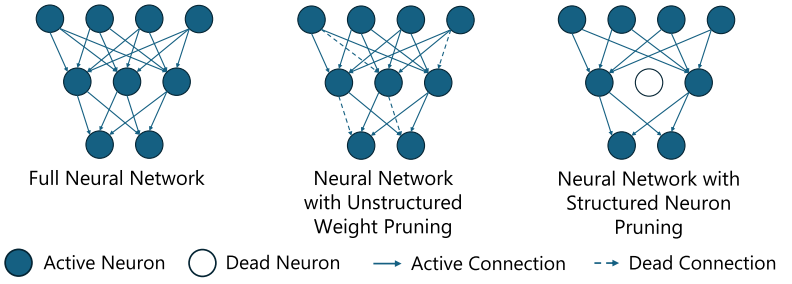


Figure 1: Unstructured and structured pruning variants. Inspired by Figures 4 and 5 from Hu et al. [16].

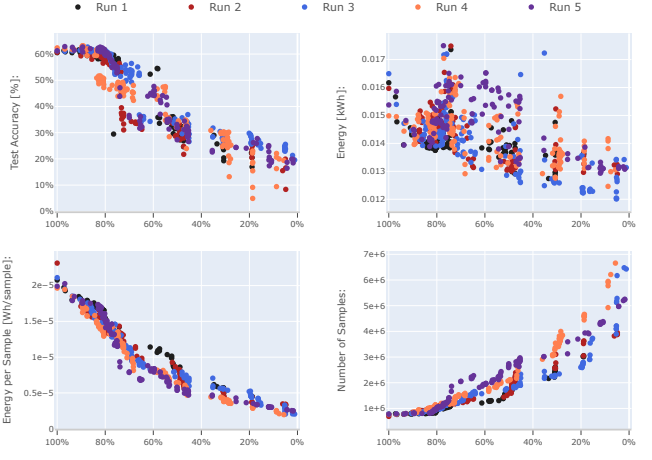


Figure 2: ResNet50-Bat on CIFAR-100: test accuracy, energy usage, and throughput across network sizes. The x-axis on all plots shows the network size ranging from 100 % to 0 %.

used to guide pruning decisions by analyzing multiply and accumulate (MAC) operations and memory access patterns of neurons or a group of weights. We do not measure any CO_2 reduction or cost saving, as these values are not constant: they change from one place to another, and might change much over time.

3 Method

This paper presents a method for structured, static, and iterative pruning based on a simple yet effective neuron relevance criterion: the *lowest mean weights*. The following subsections describe our pruning method, energy measurement setup, and the overall neuron removal strategy.

3.1 Pruning Method

Unlike the original LTH [7], which uses unstructured pruning by setting weights to zero, our approach removes neurons entirely from the network. This qualifies as structured pruning

and affects both the layer being pruned and the subsequent layers. We adapt the structured pruning principle from Hu et al. [16], who use the APoZ metric, but simplify the process by introducing the *lowest mean weight* metric to evaluate layer and neuron relevance. Previously, we investigated a method called *percentile masking*, inspired by APoZ. This technique masked neuron weights below a percentile threshold into binary values and counted zeros per neuron to determine importance. However, the *lowest mean weight* method produced more stable sub-networks with similar or improved performance, while being computationally more straightforward.

In the *lowest mean weight* method we define the mean of a neuron's weights as $mean_{l,n} = \frac{1}{k \cdot p} \sum_{c=0}^{p} \sum_{j,i=0}^{k} \|A_{l,n,c,j,i}\|$, where $A_{l,n,c,j,i}$ is the convolution weight at layer l, neuron n, input channel c, and spatial location (j,i).

This choice is grounded in the observation that neurons with higher *mean weights* often contribute more significantly to activations in subsequent layers.

The pruning process proceeds as follows in each pruning iteration. First, we select the layer by computing the *mean weights* for all layers and select the one with the overall *lowest mean value*. Second, we select the neurons by ranking neurons within the selected layer based on their *mean weights*. Neurons with the *lowest mean weights* are considered the least important. And third, we prune the selected neuron and adjust connected layers. We repeat this pruning process until a predefined *neuron pruning percentage* is achieved or until the layer is pruned to the maximum.

The *neuron pruning percentage* is introduced to establish a quantitative metric to define the degree of aggressiveness of pruning the selected layer in each pruning iteration.

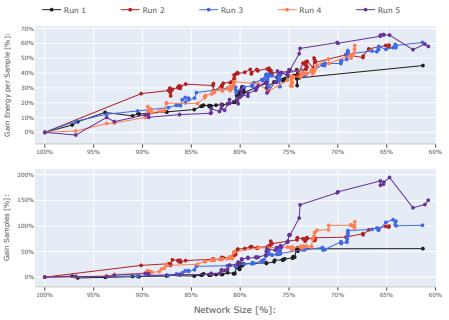
3.2 Energy Measurement

3.2.1 Nvidia-smi tool

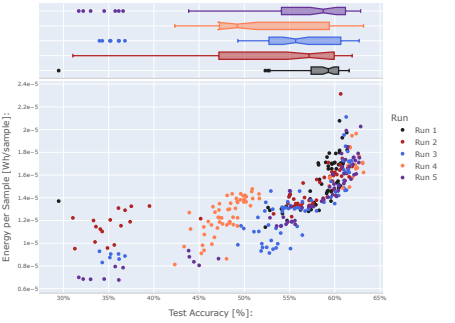
We use the `nvidia-smi` tool [30] to log GPU power draw every second. These values include baseline idle consumption, which we subtract from the raw readings to isolate network-related energy use. Since `nvidia-smi` outputs watt-seconds, conversion to kilowatt-hours is done using $1Ws = \frac{1}{3600000}$ kWh. We use this tool in our neural network measurements to obtain a more accurate approximation of the actual consumption of the network and its sub-networks running exclusively on the GPU. In a previous study, the validity of the `nvidia-smi` measurements was established through a comparison with the readings obtained from an external electricity meter. The accuracy is $\pm 2\,\%$ compared to the external meter, but does not require manual inspections of an electricity meter.

3.2.2 Measurement Setup

Energy measurements are conducted using preloaded batches of random images on the GPU memory. The random images have been established as equivalent in dimension to the original dataset. Batch sizes are optimized per sub-/network in order to maximize the throughput of the GPU. The duration of each measurement is fixed at 5 minutes to ensure consistent and representative energy readings, without incurring long runtimes via `nvidia-smi` or electricity meter. Optimizing batch size for throughput is critical, as smaller pruned networks are capable of processing a larger volume of data per second on a given amount of memory. This results in a lower energy consumption per sample, despite a comparable total power draw.



(a) ResNet50-Bat on CIFAR-100: gains in energy per sample and sample throughput. Gains are calculated by comparing performances to that of full networks. *Gain Energy per Sample* indicates the energy reduction used per sample. *Gain Samples* refers to the increase in the number of images that can be processed by the sub-networks.



(b) ResNet50-Bat on CIFAR-100: test accuracy vs. energy per sample. This plot displays the relationship between test accuracy and energy consumption per sample for all sub- and full-networks. The box plot at the top shows the median, lower and upper quantiles, with the whiskers representing the 1.5 inner quantile range and the dots indicating outliers in the data.

Figure 3: ResNet50-Bat on CIFAR-100.

3.3 Neuron Removal Algorithm

Our algorithm combines the LTH iterative pruning with structured pruning from Hu et al. [16], using the *lowest mean weight* metric for neuron removal. Our objectives of attaining accuracies analogous to those of the original network are congruent with the ones of of [16] and [7]. Disparities between unstructured and structured pruning are illustrated in Fig. 1.

We used the same experimental process for both dataset. We can summarize it as follows. We start by selecting the network architecture and dataset. Then, we train the network until the validation accuracy converges at a learning rate of 0.001. Once this is done, we identify the layer and the specific neurons for pruning based on the *lowest mean weight* method, in addition to those already pruned. Then, if the network has more than 5% of its initial weight count, we reset the model's weights to their *initial* values, and prune selected neurons by removing output channels (weights/bias) of the selected neurons, adjusting the input channels of the subsequent layer, and updating architecture-specific layers, e.g. batch normalization and resampling layers to preserve network integrity. We then evaluate accuracy on the test data and perform energy measurements via `nvidia-smi` on artificially generated random dataset. We repeat this process, starting from the training phase, until over 95 % of the network is pruned or until all convolutional layers have only one remaining neuron.

4 Experimental Setup

4.1 CIFAR-100 Dataset

The CIFAR-100 dataset contains 60,000 images in 100 categories. Each image is 32×32 pixels and belongs to one of 600 images per class. Its compact size and broad use make it ideal for benchmarking.

Table 1: Comparison of our adapted ResNet-50 architectures to the original.

	Original ResNet-50	Our ResNet-50 variants
Convolutional layers	53	65
Fully-connected layers	1	1
Total layers	54	66

4.2 Dataset of Pages from Early Printed Books with Multiple Font Groups

This dataset includes over 35,600 images of book pages from the 15th to 18th century. These images have been captured in a variety of high resolutions and classified into ten font group classes plus two noise classes [34, 35, 36]. The labels are expert-annotated and often multi-label, with strong class imbalance [29]. The dataset was selected due to its substantial difference from CIFAR-100, both in content and volume, with the former exceeding 44 GB. Furthermore, it highlights the importance of energy-efficient network architectures, with direct applications in large-scale historical document classification tasks, such as those conducted in state libraries, where millions of unlabeled pages must be processed. To accommodate GPU memory constraints, we resize training images to 320×320 for the purpose of achieving better training throughput. In addition, the dimensions of the validation and test images are limited to a maximum width of 1000 pixels, thereby restricting validation and test batch sizes to 1. This dataset is denoted as *Fonts* in the following.

4.3 Network Architectures

Our models are based on ResNet-50 [13], a residual Convolutional Neural Network (CNN) that learns $H(x) - x$ using skip connections. To alleviate dimensionality dependency between building blocks, we added 1×1 resampling layers in every building block [1]. Table 1 shows the difference in the number of weight layers from our architectures to the original ResNet-50. We further use a modified version called *ResNet50-Bat*, which halves the neurons in the third convolutional layer of each block, and adds dropout before the fully connected layer. The *ResNet50-Bat* model was employed to attain a higher reference test accuracy than the standard *ResNet-50* on the *CIFAR-100* dataset. Both models use LeakyRectified Linear Unit (ReLU) activations [19, 39] to mitigate the dying ReLU problem. Furthermore, we only apply cropping and rotation as data augmentation.

5 Results

This section presents the results of our structured pruning method applied to two datasets: *CIFAR-100* and *Fonts*. We assess both classification accuracy and energy consumption, using sub-networks derived from *ResNet-50* and *ResNet50-Bat* models. All networks were trained using the Cross-Entropy (CE) criterion and the Adaptive Moment Estimation (ADAM) optimizer [21]. Training data was split 80/20 into training and validation sets. Pruning was applied iteratively with a *neuron pruning percentage* of at least 70 % to avoid excessively small pruning steps. No fine-tuning was applied after pruning; hence, reported results reflect raw performance of pruned networks.

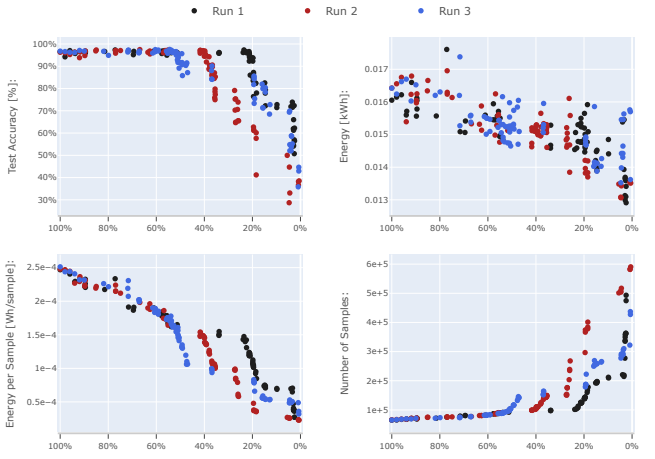
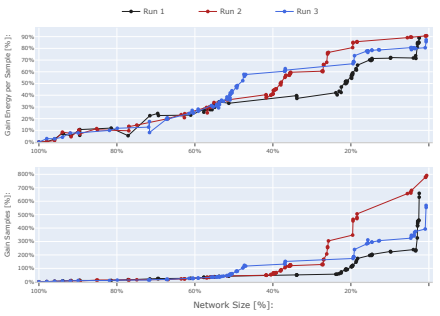


Figure 4: ResNet-50 on Fonts: test accuracy, energy usage, and throughput across network sizes. The x-axis on all plots shows the network size ranging from 100 % to 0 %.

For the CIFAR-100 dataset, five pruning runs were performed using *ResNet50-Bat* on the *CIFAR-100* dataset. Runs 1 and 2 used 40 training epochs; the remaining runs used 50 to ensure convergence in smaller sub-networks. Most runs used a 90 % *neuron pruning percentage*, with adjustments down to 70 % for finer-grained pruning below 83 % network size. Figure 2 shows that sub-networks around 75 % retain comparable test accuracy. However, pruning beyond this threshold causes performance to drop significantly. The pruning steps at ∼45 % network size led to a sharp drop in accuracy, suggesting the removal of critical neurons. The energy consumed per sample reduced by ∼50 % at around 70 % network size. This is achieved because the absolute energy consumption slightly decreases and the throughput in smaller networks increases. Figure 3(a) compares gains in energy and throughput. Sub-networks around 75 % network size reduce energy per sample by ∼40 % and increase sample count by approximately 50 %, while maintaining most of the test accuracy. Figure 3(b) summarizes pruning performance: lower energy per sample with only minor losses in test accuracy is preferable. For example, in *Run 3*, the energy usage decreased from $2.1e^{-5}$ to $0.9e^{-5}$ [Wh/sample] with only 7 percentage points lost in accuracy.

For the font dataset, three pruning runs were conducted using our adapted ResNet-50 on the *Fonts* dataset. Each sub-network was retrained for four epochs between pruning steps. A 90 % *neuron pruning percentage* was used throughout.Figure 4 shows that networks pruned below ∼40 % network size exhibit unstable accuracy, but reduce energy per sample by almost 50 %. Total energy use remains relatively flat due to constant 5-minute GPU runtime at high utilization.Gains shown in Fig. 5(a) confirm that 40 % networks can cut energy per sample by about 40 % while increasing throughput by 60 % or more and considering only a small decrease in test accuracy.Figure 5(b) provides the final accuracy/energy trade-off visualization. Sub-networks with test accuracies above 90 % consumed less than half the energy per sample, with *Run 1* delivering particularly strong results.



(a) ResNet-50 on Fonts: Gains in energy per sample and throughput are measured relative to the full network. *Gain Energy per Sample* denotes reduced energy use per sample, and *Gain Samples* indicates higher processing throughput.

(b) ResNet-50 on Fonts: test accuracy vs. energy per sample. The plot shows test accuracy against energy use per sample for all networks. The top box plot marks medians, quantiles, whiskers (1.5 IQR), and outliers.

Figure 5: ResNet-50 on Fonts

6 Discussion and Conclusion

Our structured pruning method achieved over 40 % reduction in energy consumption per sample across both architectures and datasets, while maintaining competitive test accuracy. Some sub-networks even slightly outperformed the full models – an effect also reported in Frankle and Carbin [7] – though the margins are small. Despite potential modest accuracy drops, they may still be viable for deployment depending on application constraints, particularly when energy efficiency is prioritized. Both *ResNet-50* and *ResNet50-Bat* yielded comparable energy savings, with stronger benefits on the *Fonts* dataset. In this case, *ResNet-50* was more stable under aggressive pruning. From Figures 2 and 4, we note that *ResNet50-Bat* peaked at roughly $2.3e^{-5}$ [Wh/sample], whereas *ResNet-50* reached approximately $2.51e^{-4}$ [Wh/sample]. This means that while both models show a 40 % gain, the absolute benefit in raw energy usage is higher for *ResNet-50*. The improvement amounts to $1.3e^{-5}$ [Wh/sample] for *ResNet50-Bat* and $1.5e^{-4}$ [Wh / sample] for *ResNet-50*. One limitation observed is the handling of the *neuron pruning percentage*. In instances where heavily pruned layers are subsequently selected, minimal alterations in network size occur while potentially crucial neurons are removed. This can be seen in Fig. 5(a), where denser sub-network point clouds sometimes exhibit unexpected drops in accuracy. Overall, these results serve as a baseline for the energy-saving potential of structured pruning methods. It is important to emphasize that no fine-tuning was applied after pruning, suggesting further improvements are likely achievable. Future work may incorporate methods like EB tickets [43] or pruning mechanisms as proposed in Rachwan [31], which dynamically select the *neuron pruning percentage* and may address current limitations. We envision broader adoption of structured pruning in real-world scenarios. A practical example would be national or academic libraries, where neural networks are applied at scale for classification tasks in digitized historical archives. Energy-efficient models can play a crucial role in reducing operational costs and environmental impact, especially as the energy sector remains responsible for approximately 34 % of global greenhouse gas emissions [24]. For our future research, we plan to explore quantization alongside pruning [6], as well as dynamic pruning strategies where metrics like the *neuron pruning percentage* are adjusted iteratively based on feedback during pruning.

References

[1] Nima Aghli and Eraldo Ribeiro. Combining weight pruning and knowledge distillation for cnn compression. In *CVPR Workshops*, pages 3191–3198, 2021.

[2] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? In *Proceedings of Machine Learning and Systems*, volume 2, pages 129–146, 2020.

[3] Baiyun et al. Cui. Joint structured pruning and dense knowledge distillation for efficient transformer model compression. *Neurocomputing*, 458:56–69, 2021. doi: 10.1016/j.neucom.2021.05.084.

[4] Yadong Ding, Yu Wu, Chengyue Huang, Siliang Tang, Fei Wu, Yi Yang, Wenwu Zhu, and Yueting Zhuang. Nap: Neural architecture search with pruning. *Neurocomputing*, 477:85–95, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.12.002. URL https://www.sciencedirect.com/science/article/pii/S0925231221018361.

[5] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019. URL http://jmlr.org/papers/v20/18-598.html.

[6] Tailin Liang et al. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021. doi: 10.1016/j.neucom.2021.07.045.

[7] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

[8] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Stabilizing the Lottery Ticket Hypothesis. *arXiv e-prints*, art. arXiv:1903.01611, March 2019.

[9] Richard C. Gerum, André Erpenbeck, Patrick Krauss, and Achim Schilling. Sparsity through evolutionary pruning prevents neuronal networks from overfitting. *Neural Networks*, 128:305 – 312, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2020.05.007. URL http://www.sciencedirect.com/science/article/pii/S0893608020301696.

[10] Amir et al. Gholami. A survey of quantization methods for efficient neural network inference. pages 291–326, 2022.

[11] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115, 2014. URL http://arxiv.org/abs/1412.6115.

[12] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 1135–1143, 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/pdf/1503.02531.

[15] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021. URL http://jmlr.org/papers/v22/21-0366.html.

[16] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures, 2016. URL http://arxiv.org/abs/1607.03250.

[17] A. G. Ivakhnenko. Polynomial theory of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-1(4):364–378, 1971. doi: 10.1109/TSMC.1971.4308320.

[18] Alexey Grigorevich Ivakhnenko. The group method of data handling, a rival of the method of stochastic approximation. *Soviet Automatic Control*, 13(3):43–55, 1968.

[19] Muhammad et al. Khalid. Empirical evaluation of activation functions in deep convolution neural network for facial expression recognition. In *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, pages 204–207, 2020. doi: 10.1109/TSP49548.2020.9163446.

[20] Youngeun et al. Kim. Exploring lottery ticket hypothesis in spiking neural networks. In *ECCV 2022*, pages 102–120, 2022.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[22] Vinnie Ko, Stefan Oehmcke, and Fabian Gieseke. Magnitude and uncertainty pruning criterion for neural networks. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2317–2326, 2019. doi: 10.1109/BigData47090.2019.9005692.

[23] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In *Advances in neural information processing systems*, volume 2, 1989.

[24] Hoesung et al. Lee. Ipcc, 2023: Climate change 2023: Synthesis report, 2023. URL https://www.ipcc.ch/report/ar6/syr/downloads/report/IPCC_AR6_SYR_LongerReport.pdf.

[25] Xin Li, Yiming Zhou, Zheng Pan, and Jiashi Feng. Partial order pruning: For best speed/accuracy trade-off in neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[26] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2755–2763, 2017. doi: 10.1109/ICCV.2017.298.

[27] Eran Malach, Gilad Yehudai, Shai Shalev-shwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: pruning is all you need. In *Proceedings of the 37th International Conference on Machine Learning*, pages 6682–6691, 2020.

[28] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJGCiw5gl.

[29] Konstantina Nikolaidou, Mathias Seuret, Hamam Mokayed, and Marcus Liwicki. A survey of historical document image datasets. *Int. J. Document Anal. Recognit.*, 25(4): 305–338, 2022. URL https://doi.org/10.1007/s10032-022-00405-8.

[30] NVIDIA Corporation. Nvidia system management interface nvidia-smi. URL https://developer.nvidia.com/nvidia-system-management-interface. Licensed by the NVIDIA-Corporation (08.01.2023).

[31] John et al. Rachwan. Winning the lottery ahead of time: Efficient early network pruning. In *International Conference on Machine Learning*, pages 18293–18309, 2022.

[32] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1gSj0NKvB.

[33] Victor Schmidt, Kamal Goyal, Aditya Joshi, Boris Feld, Liam Conell, Nikolas Laskaris, Doug Blank, Jonathan Wilson, Sorelle Friedler, and Sasha Luccioni. Codecarbon: Estimate and track carbon emissions from machine learning computing, 2021.

[34] Mathias Seuret, Saskia Limbach, Nikolaus Weichselbaumer, Andreas Maier, and Vincent Christlein. Dataset of pages from early printed books with multiple font groups. In *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing*, HIP '19, page 1–6, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450376686. doi: 10.1145/3352631.3352640.

[35] Mathias Seuret, Saskia Limbach, Nikolaus Weichselbaumer, Andreas Maier, and Vincent Christlein. Dataset of Pages from Early Printed Books with Multiple Font Groups, August 2019.

[36] Mathias Seuret, Anguelos Nicolaou, Dalia Rodríguez-Salas, Nikolaus Weichselbaumer, Dominique Stutzmann, Martin Mayr, Andreas Maier, and Vincent Christlein. Icdar 2021 competition on historical document classification. In Josep Lladós, Daniel Lopresti, and Seiichi Uchida, editors, *Document Analysis and Recognition – ICDAR 2021*, pages 618–634, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86337-1.

[37] Sunil Vadera and Salem Ameen. Methods for pruning deep neural networks. *IEEE Access*, 10:63280–63300, 2022. doi: 10.1109/ACCESS.2022.3182659.

[38] Thomas et al. Widmann. Pruning for power: Optimizing energy efficiency in iot with neural network pruning. In *Engineering Applications of Neural Networks*, pages 251–263, 2023.

[39] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network, 2015.

[40] Jingfeng et al. Yang. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Trans. Knowl. Discov. Data*, 2024. doi: 10.1145/3649506.

[41] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6071–6079, 2017. doi: 10.1109/CVPR.2017.643.

[42] Seul-Ki Yeom, Kyung-Hwan Shim, and Jee-Hyun Hwang. Toward compact deep neural networks via energy-aware pruning. *CoRR*, abs/2103.10858, 2021. URL https://arxiv.org/abs/2103.10858.

[43] Haoran et al. You. Drawing early-bird tickets: Towards more efficient training of deep networks. *CoRR*, abs/1909.11957, 2019. URL http://arxiv.org/abs/1909.11957.