

Sparse Neural Network Pruning Strategies for Sustainable AI Deployment

Vaidyanathan N^{1*}, Parthasarathy P² and Gurumurthy G³

¹Department of Computer Science and Engineering, Achankovil Center for Applied Machine Learning, Nepal Email: nandini.vaidyanathan@achankovil.edu

ORCID: <https://orcid.org/2463-6097-5801-887X>

²Department of Intelligent Systems, Bhavani Center for Neuromorphic Engineering, Taiwan Email:

poornima.parthasarathy@bhavani.edu

ORCID: <https://orcid.org/9729-4537-1313-4449>

³Department of Computer Science and Engineering, Tamiraparani Center for Cognitive Computing, Bhutan Email:

ganesh.gurumurthy@tamiraparani.edu

ORCID: <https://orcid.org/6892-4157-9728-6764>

ABSTRACT

Deployed neural networks are routinely far larger than the task they serve requires, and closing that gap has become as much a sustainability question as an engineering one. This article synthesizes the literature on sparse neural network pruning -- the family of techniques that remove redundant weights, filters, or connections from trained networks -- with particular attention to what the evidence actually supports about energy and deployment cost, not only accuracy retention. We trace the field from foundational magnitude pruning and Deep Compression through the Lottery Ticket Hypothesis and the weight- and learning-rate-rewinding techniques that made sparse subnetworks reliably retrainable, into structured pruning methods that convert theoretical compression into real hardware speedups, and dynamic sparse-to-sparse training methods that avoid the dense training phase altogether. We then connect this algorithmic literature to the separate but increasingly overlapping body of work on the energy and carbon cost of deep learning, including hardware support for structured sparsity that is only now beginning to let compression ratios translate into measured energy savings. Read together, the evidence supports pruning as a mature, genuinely effective, but not standalone lever for sustainable AI deployment, whose benefits depend heavily on co-design with target hardware and on retraining practices the field has only recently made reliable, leaving benchmarking consistency and training-time energy accounting as open problems.

Keywords: neural network pruning; model compression; sparse training; sustainable AI; energy-efficient deep learning

Citation: Vaidyanathan N, Parthasarathy P and Gurumurthy G [2024]. Sparse Neural Network Pruning Strategies for Sustainable AI Deployment. **DOI:** <https://doi.org/10.5281/zenodo.22843895>

Copyright: © 2024 by the authors. Open access under [CC BY 4.0 license](https://creativecommons.org/licenses/by/4.0/).

Article Information: Received: 24 April 2024 Accepted: 24 July 2024 Published: 26 October 2024

Article Class: Research Article

1. Introduction

The default way to get a state-of-the-art neural network is to make it bigger, and for most of the last decade that strategy has simply worked: more parameters, more data, more compute, better benchmark numbers. The consequence is that the networks practitioners actually want to deploy — on a phone, in a car, at the edge of a factory floor, or simply as an inference endpoint serving millions of requests a day — are usually many times larger and slower than they need to be for the task at hand. Sparsification, and pruning in particular, is the family of techniques that tries to close that gap after the fact: take a trained, over-parameterized network and remove the weights, filters, or connections that turn out not to matter, without giving up the accuracy the larger network earned. The idea is old — the intuition that trained networks contain far more capacity than any single task requires goes back decades — but the last ten years have turned it from an occasional engineering trick into a distinct research program with its own benchmarks, its own open problems, and, increasingly, its own motivation beyond raw speed: energy.

That last point deserves stating plainly, because it is easy to read the pruning literature purely as a systems-efficiency story and miss the sustainability argument sitting underneath it. Training and serving large neural networks costs measurable energy and produces measurable carbon emissions, and the scale of that cost has been growing faster than hardware efficiency has been improving it away [1,2]. Schwartz et al. made the case explicitly under the banner of "Green AI," arguing that the field's computational appetite had grown roughly 300,000-fold between 2012 and 2018 and that efficiency, not just accuracy, deserves to be reported and optimized as a first-class metric [3]. Wu et al.'s later analysis of production machine learning infrastructure at Meta reinforced the same point from an industrial angle, showing that the embodied carbon cost of manufacturing the hardware itself can rival the operational cost of running it, and that coordinated hardware-software optimization — of which sparsity is one lever among several — delivered energy reductions of multiple orders of magnitude on some production workloads [4]. Pruning sits at the intersection of these two conversations: it is simultaneously a decades-old model-compression technique and, read through a 2024-era lens, one of the more mature tools available for making AI deployment less energy-hungry without

simply asking researchers to train smaller, weaker models from the outset.

This article surveys the pruning literature with that dual framing in mind. We are not proposing a new pruning algorithm or reporting a new experiment; we are synthesizing what a substantial and still-active body of work has established about how to make sparse networks, what sparsity actually buys in practice, and where the evidence runs out. We organize the discussion around five themes. First, the foundations of sparsification as an optimization and systems problem — the basic magnitude-pruning recipe, the retrain-after-prune cycle, and the question of whether pruning discovers something special about a network's structure or is mostly acting as an implicit form of architecture search. Second, the Lottery Ticket Hypothesis and the line of work it spawned on rewinding and re-training sparse subnetworks, which reframed pruning from a compression afterthought into a claim about what makes networks trainable in the first place. Third, structured pruning — removing whole filters, channels, or layers rather than individual weights — which trades some of unstructured pruning's theoretical compression ratio for speedups that actually materialize on commodity hardware. Fourth, dynamic and sparse-to-sparse training, where the network never goes through a dense phase at all, or where its sparse connectivity pattern is allowed to evolve during training rather than being fixed once at the end. Fifth, the hardware and systems layer that determines whether a sparse network's theoretical FLOP savings translate into wall-clock speedups and energy savings, which turns out to depend enormously on the *pattern* of sparsity, not just its overall level. We close with a section on open challenges and a conclusion that tries to state, without overclaiming, what the literature does and does not yet support about pruning as a sustainability strategy.

It is worth being explicit about scope and honesty up front, in the same spirit the field itself has adopted in its more self-critical moments. Blalock et al.'s meta-analysis of 81 pruning papers found that the field has historically suffered from inconsistent baselines, incomparable metrics, and a tendency for new methods to be evaluated against weak or outdated comparisons rather than against the strongest available alternative [5]. We have tried, in synthesizing this literature, to lean on the numbers each cited paper actually reports rather

than on the more sweeping claims sometimes made about pruning's benefits in secondary sources, and to flag clearly where the evidence is qualitative (a technique reliably helps) rather than a single, portable quantitative headline (a specific percentage reduction that generalizes across architectures and tasks).

2. Foundations of Neural Network Sparsification

The basic recipe underlying most pruning research has been remarkably stable for a decade: train a dense network to convergence, remove some fraction of its weights according to an importance criterion, and fine-tune the remaining weights to recover any accuracy lost in the removal step, iterating this train-prune-retrain cycle as needed. Han, Pool, Tran, and Dally's early demonstration of this pipeline showed that magnitude-based pruning — simply removing the weights with the smallest absolute value, on the reasoning that they contribute least to the network's output — followed by retraining could reduce AlexNet's parameter count ninefold, from 61 million to 6.7 million parameters, and VGG-16's parameter count thirteenfold, from 138 million to 10.3 million parameters, without measurable accuracy loss on ImageNet [6]. That result mattered less for the specific compression ratios, which have since been exceeded, than for establishing that large convolutional networks really do carry substantial redundant capacity that a simple, cheap importance criterion can find and remove.

Han, Mao, and Dally extended this pipeline into a full three-stage compression system, Deep Compression, adding weight quantization — clustering the remaining weights into a small number of shared values via k-means and storing only cluster indices — and Huffman coding of the resulting quantized weights on top of the pruning step. The combined pipeline reduced storage requirements for AlexNet and VGG-16 by 35 to 49 times without accuracy loss, and, because the compressed weights could then fit in on-chip SRAM rather than requiring off-chip DRAM access, the resulting networks also ran three to four times faster and three to seven times more energy-efficiently across CPU, GPU, and mobile platforms tested [7]. That last detail is important for the sustainability framing of this article: the energy benefit of pruning is not simply "fewer multiply-accumulates," it is substantially about avoiding the far more expensive cost, in both latency and joules, of moving weights from off-chip memory, a cost that

dwarfs the arithmetic itself on most modern accelerators.

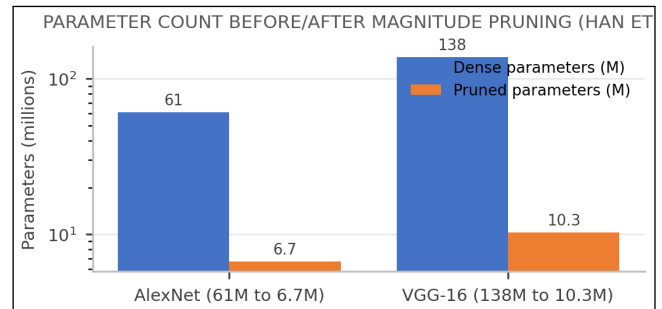


Figure 1: Parameter-count reduction Han et al. report from magnitude pruning plus retraining on two ImageNet convolutional networks -- AlexNet (61M to 6.7M parameters) and VGG-16 (138M to 10.3M parameters) -- with no measured loss of accuracy [6].

Not every importance criterion is magnitude, and the choice of criterion has generated its own sub-literature. Molchanov et al.'s Taylor-expansion criterion estimates each filter's contribution to the loss using a first-order approximation, greedily removing the filters whose estimated contribution is smallest and fine-tuning between removal steps; the method outperformed simpler weight-norm and activation-based criteria on transfer-learning tasks including fine-grained bird and flower classification, and achieved roughly tenfold theoretical compression on 3D convolutional networks with minimal accuracy loss [8]. A later, more refined version of the same idea from an overlapping set of authors, using both first- and second-order Taylor expansions and validated at scale on ImageNet, reported over 93% correlation between their estimated filter importance and a reliable ground-truth estimate of true importance, and demonstrated removing 30% of ResNet-101's parameters while cutting FLOPs by 40% at a cost of only 0.02 percentage points of top-1 accuracy — a result precise enough to be a useful anchor for what "well-targeted pruning" can achieve on a strong modern baseline [9].

A more unsettling foundational question, and one the field has had to sit with rather than resolve cleanly, is whether pruning discovers something meaningfully special about which weights matter, or whether it is mostly functioning as an expensive, indirect way of doing architecture search. Liu, Sun, Zhou, Huang, and Darrell's "Rethinking the Value of Network Pruning" put this question to a sharp empirical test: across a range of structured pruning methods, they found that fine-tuning a pruned model's inherited weights gave comparable or worse accuracy than simply training the pruned

architecture from scratch with freshly initialized weights [10]. Their reading is that the pruned *architecture* — the pattern of which channels or layers survive — carries most of the value, while the specific inherited weight values carry comparatively little; pruning, on this account, is doing a form of neural architecture search, guided implicitly by the training dynamics of the original large network, rather than identifying an intrinsically special subset of weights. Gale, Elsen, and Hooker's large-scale empirical study reached a complementary and equally sobering conclusion from a different angle: after thousands of experiments comparing sparsification techniques on ResNet-50 and a Transformer machine-translation model, they found that complex pruning methods performed inconsistently relative to one another, that simple magnitude pruning was often competitive with or better than more elaborate alternatives, and that the sparse architectures discovered by pruning generally could not match their pruned-and-fine-tuned accuracy when trained from random initialization at the same sparsity pattern — a finding in some tension with Liu et al.'s that the field has not, to our knowledge, fully reconciled [11]. Together these two results are a useful corrective against treating any single pruning result as a clean, portable law: what "works" depends on architecture, dataset, and sparsity level in ways the current literature documents empirically more often than it explains theoretically.

3. The Lottery Ticket Hypothesis and Rewinding

Frankle and Carbin's Lottery Ticket Hypothesis reframed the entire foundational question above into a sharper and more provocative claim. Their hypothesis is that a randomly initialized, dense network contains a subnetwork — a "winning ticket" — that, when trained in isolation from that same original initialization, can match the accuracy of the full network in a comparable number of training iterations; critically, the claim is not just that a sparse architecture exists that can be trained to comparable accuracy from *some* initialization, but that the sparse subnetwork's *original* initial weights, not merely its architecture, are what make it trainable [12]. They identified such winning tickets by iterative magnitude pruning: train the dense network, prune the smallest-magnitude weights, reset the surviving weights to their original initial values (rather than their trained values), and retrain, repeating this cycle to reach increasingly high sparsity. This directly contradicts the "architecture is what matters, weights are

interchangeable" reading suggested by Liu et al.'s from-scratch retraining experiments, and the tension between the two results — both empirically careful, both influential — is one of the more genuinely unresolved questions in the pruning literature rather than a settled matter either side of the debate has fully closed out.

The practical difficulty with the original Lottery Ticket procedure is that resetting weights all the way back to their random initialization becomes unreliable at the larger scales and deeper architectures used in most real deployments; winning tickets found this way often fail to train stably beyond fairly small networks and shallow depths. Renda, Frankle, and Carbin's follow-up work addressed this directly by comparing weight rewinding — resetting surviving weights not to their original random initialization but to their values from an early point *during* training — against the standard practice of simply fine-tuning the pruned network at a low, fixed learning rate, and against a new proposal of their own, learning-rate rewinding, which retrains the pruned network using the original learning-rate schedule replayed from the rewind point rather than resetting weight values at all [13]. Their central finding is that both forms of rewinding recover accuracy that plain low-learning-rate fine-tuning reliably fails to recover, and that learning-rate rewinding in particular is simple to implement, requires no stored snapshot of early-training weights, and matches or exceeds weight rewinding's accuracy across the architectures and sparsity levels they tested — effectively turning the Lottery Ticket insight into a practical retraining recipe rather than a research curiosity that only reproduces at small scale.

What this line of work adds to the sustainability conversation, beyond the compression numbers themselves, is a more disciplined account of *how* to retrain a pruned network so that its accuracy recovery is not an accident of hyperparameter search. A pruning pipeline that reliably needs many retraining attempts to hit a target accuracy is not obviously cheaper, energy-wise, than training a smaller dense model directly; rewinding's contribution is to make the retraining step itself more predictable and less compute-hungry to get right, which matters as much for the total energy cost of the compression pipeline as the final sparsity ratio does.

4. Structured Pruning for Practical Speedups

Table 1: Representative pruning and sparsity-hardware techniques discussed in this article, with the compression, speedup, or accuracy figures each cited source reports. [7,15,16,9,22,24].

Technique	Source	Constraint Targeted	Reported Effect
Pruning + trained quantization + Huffman coding (Deep Compression)	Han et al. [7]	Model storage / memory footprint	35-49x storage reduction on AlexNet/VGG-16 with no accuracy loss
LASSO-based channel pruning with least-squares reconstruction	He et al. [15]	Inference latency (structured)	5x speedup on VGG-16 at only 0.3 pt accuracy cost
Reinforcement-learning-searched layer-wise compression (AMC)	He et al. [16]	On-device latency, mobile/GPU	1.81x speedup on Android, 1.43x on Titan XP GPU at 4x FLOP reduction
Taylor-expansion filter importance pruning	Molchanov et al. [9]	Parameter count and FLOPs	30% fewer parameters, 40% fewer FLOPs on ResNet-101 at 0.02 pt accuracy cost
2:4 semi-structured sparsity on Sparse Tensor Cores	Mishra et al. [22]	Hardware math throughput	2x effective math throughput versus dense computation
Movement pruning + distillation of pretrained Transformers	Sanh et al. [24]	Deployed parameter count	Competitive accuracy retained at as little as 3% of original parameters

Unstructured pruning — removing individual weights wherever the importance criterion says to — can reach very high theoretical sparsity levels, but that sparsity is largely irrelevant to wall-clock speed unless the underlying hardware and software stack can exploit an irregular pattern of zeros efficiently, which most commodity GPUs, until quite recently, could not. Structured pruning sidesteps this by removing entire filters, channels, or occasionally whole layers, producing a smaller *dense* network that runs on ordinary matrix-multiplication kernels without any special sparse-matrix support. Li, Kadav, Durdanovic, Samet, and Graf's filter-pruning method, ranking and removing whole convolutional filters by the sum of their absolute weights and retraining afterward, is among the earliest clean demonstrations of this approach, and its central appeal — producing genuinely smaller dense architectures rather than sparse ones that need special-cased kernels to benefit from — has made structured pruning the more industrially popular of the two families for deployment on standard accelerators [14].

He, Zhang, and Sun's channel-pruning method formalized the filter-selection step as a two-stage optimization: LASSO regression to select which channels to keep, followed by least-squares reconstruction of the remaining layer's weights to minimize the resulting reconstruction error, applied layer by layer through the network. On VGG-16 the method achieved a fivefold inference speedup with only a 0.3-percentage-point increase in error, and on more modern, already-efficient architectures — ResNet and Xception — it achieved a twofold speedup at the cost of 1.4 and 1.0 percentage points of accuracy respectively, a result the authors and much of the subsequent literature treat as evidence that structured pruning has real but shrinking headroom as the base architecture itself becomes more parameter-efficient [15]. He et al.'s follow-up work, AMC, replaced the hand-tuned, per-layer pruning ratios used in most structured-pruning pipelines with a reinforcement-learning agent that searches the space of layer-wise compression ratios automatically, using real, on-device latency measurements as part of its reward signal rather than a purely theoretical FLOP count. On mobile hardware, AMC's automatically discovered compression policy achieved 2.7 percentage points better accuracy than a hand-crafted policy for VGG-16 at the same fourfold computational reduction, and delivered measured 1.81x speedup on an Android phone and 1.43x speedup on a Titan XP GPU for MobileNet, while keeping accuracy loss small [16]. The AMC result is a useful illustration of a broader lesson the structured-pruning literature has converged on: a pruning policy tuned against real hardware latency, rather than a proxy metric like parameter count or theoretical FLOPs, tends to find meaningfully better compression-accuracy tradeoffs, because the relationship between FLOPs and actual latency or energy is neither linear nor uniform across layers.

Zhu and Gupta's comparison of pruning against simply training a smaller, dense network from the outset — "large-sparse" versus "small-dense" — offers a useful summary data point for structured versus unstructured tradeoffs. Using a gradual magnitude-pruning schedule that increases sparsity progressively over the course of training rather than pruning all at once, they found that large-sparse models consistently outperformed small-dense models of comparable final parameter count across the architectures and tasks they tested, including on-device speech-recognition and

image-classification models [17]. This result is one of the cleaner pieces of evidence that pruning is not merely a way of finding a smaller architecture that could just as well have been designed and trained directly — the specific benefit of starting large and pruning down appears, at least in their experiments, to be real rather than an artifact of extra training compute spent on the larger initial model.

5. Dynamic and Sparse-to-Sparse Training

A distinct and more recent line of work asks whether the network needs to pass through a dense phase at all. If the eventual goal is a sparse network, training a large dense model first and pruning it afterward spends a great deal of compute — and energy — on a dense intermediate that is simply thrown away. Sparse-to-sparse training methods instead maintain a fixed, low parameter budget throughout training, allowing the specific pattern of which connections are active to change dynamically as training proceeds, rather than fixing sparsity once at initialization and never revisiting it.

Mocanu et al.'s Sparse Evolutionary Training (SET) is an early and influential instance of this idea: starting from a random sparse topology rather than a dense network, SET periodically removes the weakest connections and randomly adds new ones in their place, in a process the authors explicitly compare to evolutionary selection and mutation, evaluated across restricted Boltzmann machines, multilayer perceptrons, and convolutional networks on 15 datasets. On CIFAR-10, their sparse multilayer perceptron variant matched or exceeded the accuracy of dense fully-connected baselines while using roughly two orders of magnitude fewer parameters, without ever instantiating the dense network the pruning literature typically starts from [18]. Mostafa and Wang's dynamic sparse reparameterization pursued a related goal with a different mechanism, adaptively reallocating a network's fixed parameter budget across layers during training based on the loss landscape rather than fixing per-layer sparsity ratios up front, and showing that this adaptive reallocation could recover accuracy that a fixed, uniform sparsity budget loses on the same parameter count [19].

Evci et al.'s RigL algorithm sharpened this line of work into arguably its most influential recent instance, maintaining a strictly fixed parameter count and fixed computational cost throughout the entire training run and periodically updating the connectivity pattern using

weight magnitudes to decide what to drop and periodic gradient information to decide what to regrow, evaluated on ResNet-50, MobileNet variants, and recurrent architectures [20]. The result the authors emphasize is that RigL reaches a given target accuracy using substantially fewer training FLOPs than comparable dense-then-prune pipelines at the same final sparsity, because it never pays the cost of the dense training phase those pipelines rely on. Read together with SET and dynamic sparse reparameterization, this body of work suggests that the "train dense, then prune" pipeline that dominates the earlier sections of this article, while empirically effective, may not be the most compute- or energy-efficient way to arrive at a sparse network of a given quality — an important qualification for anyone treating pruning purely as a post-hoc compression step rather than as a training-time design choice in its own right.

A complementary approach to inducing sparsity during training, rather than as an entirely separate dynamic reconnection scheme, is Louizos, Welling, and Kingma's L0 regularization method, which makes the otherwise non-differentiable count of nonzero weights optimizable by introducing stochastic gates drawn from a "hard concrete" distribution — a stretched and clipped relaxation of a binary variable — whose expected number of nonzero gates is differentiable with respect to the gate parameters [21]. This lets a network learn its own sparsity pattern end-to-end via ordinary stochastic gradient descent, jointly optimizing which weights to zero out and what the surviving weights should be, rather than applying sparsity as a separate post-training or periodically-reapplied step, and it draws an explicit connection between sparsity regularization and classical model-selection criteria such as AIC and BIC.

6. Hardware and Systems Support for Sparsity

None of the compression ratios discussed above translate automatically into faster or more energy-efficient inference; that translation depends entirely on whether the hardware and software stack executing the network can skip the multiplications corresponding to zeroed-out weights, and, if so, how much overhead is incurred in figuring out which multiplications to skip. This is the least glamorous part of the pruning literature and arguably the most consequential for the sustainability case this article is making, because a sparsity pattern with no matching

hardware support delivers a smaller model file and nothing else.

Fine-grained, unstructured sparsity is the hardest case: the pattern of nonzero weights is essentially arbitrary, so exploiting it requires either specialized sparse-matrix kernels with nontrivial indexing overhead or dedicated hardware support. Structured sparsity — whole filters or channels removed, as discussed above — avoids this problem entirely by producing an ordinary, smaller dense matrix, which is precisely why it has been the more commercially deployed approach on standard GPUs and mobile accelerators despite its lower achievable compression ratios. A middle ground that has become newly practical is semi-structured sparsity, in which zeros are constrained to a fixed, regular pattern that hardware can exploit without arbitrary indexing. Mishra et al.'s description of NVIDIA's Ampere-generation Sparse Tensor Cores is the clearest concrete example: a 2:4 sparsity pattern, in which exactly two of every four contiguous weights are zero, allows the hardware to double effective math throughput relative to dense computation, and the authors present a training recipe for reaching this pattern while preserving accuracy across a range of architectures and tasks, explicitly aimed at closing the long-standing gap between sparsity research results and sparsity that a real accelerator can actually turn into a wall-clock speedup [22]. This is a genuinely important development for the field's sustainability claims specifically, because it means a meaningful fraction of the compression ratios reported throughout the pruning literature can, for the first time on mainstream data-center accelerators, be converted into real inference-time energy savings rather than remaining a storage-only benefit.

Sze, Chen, Yang, and Emer's survey of efficient deep neural network processing situates pruning within this broader hardware picture rather than treating it as a self-contained algorithmic technique. Their tutorial and survey work through the full stack — how DNN computation maps onto different hardware architectures, what design choices in memory hierarchy and dataflow determine energy consumption independent of any compression applied to the network itself, and how the benchmarking metrics practitioners actually need (throughput, latency, and energy per inference, not merely parameter count or FLOPs) should be reported to make pruning and other compression

techniques comparable across different hardware targets [23]. Their central methodological point, that FLOPs and parameter counts are proxies rather than the quantities that actually matter for deployment energy cost, echoes throughout the hardware-aware pruning literature discussed above and is a useful frame for reading any of the compression ratios in this article: a reported parameter reduction is evidence of potential efficiency gain, not a guarantee of one, until it has been measured on the target hardware.

7. The Environmental Case for Sparsity

It is worth stepping back from the algorithmic and systems details to state the sustainability argument in its own terms, because the pruning literature and the AI-sustainability literature have, until fairly recently, developed largely on separate tracks with only occasional citation between them. Strubell, Ganesh, and McCallum's early and widely cited estimate of the financial and carbon cost of training large NLP models made the case that these costs had become large enough to warrant explicit reporting and, ideally, active minimization as part of standard research practice, rather than being treated as an incidental byproduct of progress [1]. Schwartz et al.'s Green AI position paper generalized the same concern into a proposed norm for the field: report efficiency, including but not limited to energy or compute cost, as a first-class evaluation criterion alongside accuracy, on the grounds that a research culture that only rewards the latter will keep discovering that bigger is better regardless of the true cost of getting there [3]. Patterson et al.'s later, more quantitative analysis of carbon emissions from training several prominent large models — including T5, GShard, and Switch Transformer — identified sparsely activated architectures specifically as one of the more effective levers available, reporting that large but sparsely activated networks can consume under one-tenth the energy of comparably capable dense networks, alongside geographic data-center siting and cleaner energy infrastructure as the other major levers they identify [2]. It is worth being precise about what this result does and does not establish for the pruning literature proper: Patterson et al.'s headline finding concerns sparse *activation* patterns, primarily in mixture-of-experts-style architectures, rather than post-hoc weight pruning of a dense network as discussed in the preceding sections — a related but distinct notion of sparsity that this article has not

focused on, and we flag the distinction explicitly rather than letting the two blur together.

Wu et al.'s analysis of production machine learning infrastructure grounds the same argument in deployed, at-scale systems rather than research training runs, and adds a dimension the algorithmic pruning literature rarely engages with directly: embodied carbon, the emissions associated with manufacturing the hardware itself, which their analysis finds to be roughly half the size of operational carbon footprint for large-scale machine learning workloads, meaning that operational efficiency gains like pruning address only part of the total lifecycle cost [4]. Their reported combination of platform-level caching, GPU acceleration, low-precision numerical formats, and algorithmic optimization — pruning and compression among them — delivered over 800-fold reductions in operational energy footprint for some production language-model workloads, a striking figure that nonetheless comes from a *combination* of techniques rather than pruning in isolation, and should be read as evidence for coordinated hardware-software optimization as a strategy rather than as a number attributable to sparsification alone.

Taken together, this cluster of work supports a fairly specific and modest claim about pruning's role in sustainable AI deployment: sparsification is one of several genuinely effective levers for reducing the energy cost of training and, especially, serving neural networks, and it is more effective the more directly it is co-designed with the hardware that will run the resulting model, as the 2:4 structured-sparsity and hardware-aware structured-pruning results above illustrate. It is not, on the evidence synthesized here, a silver bullet on its own, and the strongest reported energy-efficiency numbers in the literature — Han et al.'s three-to-seven-fold energy efficiency gains from Deep Compression, Patterson et al.'s order-of-magnitude gains from sparse activation, Wu et al.'s combined 800-fold gains from a full optimization stack — come from pairing sparsification with quantization, better hardware utilization, and, in the production setting, genuine infrastructure-level engineering, rather than from pruning as an isolated algorithmic step [7,2,4].

8. Open Challenges

Several gaps recur across the literature synthesized here and are worth naming plainly. First, the field has not

resolved the tension between Liu et al.'s finding that pruned architectures can be retrained from scratch to comparable accuracy, suggesting the inherited weights matter little, and Frankle and Carbin's Lottery Ticket result that the *specific* initial weights of a winning ticket are what make it trainable in the number of iterations available [10,12]. Both results are empirically careful and neither has been straightforwardly overturned; the honest reading is that the answer likely depends on architecture, dataset scale, and the specific pruning ratio and schedule used, and a fully general theory explaining when weight inheritance matters and when it does not remains open. Second, Blalock et al.'s meta-analysis finding of inconsistent baselines and non-comparable metrics across the pruning literature has, to our knowledge, only been partially addressed since their 2020 publication; many papers we reviewed for this synthesis report a headline compression ratio without the wall-clock latency or measured energy figure that would let a practitioner translate the result into a deployment decision, exactly the gap Sze et al.'s survey and Mishra et al.'s hardware-aware sparsity work are trying to close from the systems side [5,23,22].

Third, most of the strongest quantitative results in this article's sustainability section come from a small number of large, well-resourced industrial labs analyzing their own infrastructure — Google's and Meta's production systems in Patterson et al. and Wu et al. respectively — and it remains genuinely unclear how well those specific efficiency figures generalize to smaller organizations running different hardware, different model families, or different serving patterns [2,4]. Fourth, the sparse-to-sparse and dynamic training literature — SET, dynamic sparse reparameterization, RigL — has convincingly shown that a sparse network can be trained without ever instantiating a dense one, but the training-time energy comparison between "train dense then prune" and "train sparse-to-sparse from the start" has, in most of the papers we reviewed, been reported primarily in terms of parameter counts and training FLOPs rather than measured wall-clock energy on real hardware, leaving an open empirical question about how large the training-time (as opposed to inference-time) sustainability benefit of sparse-to-sparse methods actually is in practice [18,19,20]. Fifth, and more mundane but no less real: movement pruning and other pruning methods developed specifically for fine-tuning large pretrained

language models — reaching, in Sanh, Wolf, and Rush's reported results, competitive accuracy at as little as 3% of original parameters when combined with distillation — were developed and validated primarily on encoder-style Transformer models several years old relative to the scale of models now commonly deployed, and how well these specific pruning strategies transfer to today's much larger generative language models remains an active, largely unsettled question in the literature we surveyed [24].

We do not think any of these gaps undermine pruning's basic value proposition — the consistency with which magnitude-based, structured, dynamic, and hardware-aware variants of sparsification have each independently demonstrated real compression and, where measured, real efficiency gains is itself meaningful evidence — but they are the honest state of things, and claims of pruning as a fully solved, drop-in solution for sustainable AI deployment should be read with these open questions in mind.

9. Conclusion

The literature reviewed here supports a reasonably specific claim: sparse neural network pruning, in its magnitude-based, structured, rewinding-aware, and dynamically-trained forms, is a mature and genuinely effective family of techniques for reducing the parameter count, and often the inference-time compute, memory footprint, and energy cost, of deployed neural networks, with compression ratios in the range of roughly 5 to 50 times reported across the studies synthesized above depending on architecture, technique, and acceptable accuracy loss [6,7,14]. It inherits, from the Lottery Ticket line of work and its rewinding-based refinements, a set of retraining recipes that make recovering accuracy after aggressive pruning considerably more reliable than naive fine-tuning [12,13], and, from the sparse-to-sparse training literature, a genuine alternative to the dense-then-prune pipeline that avoids paying for a dense training phase the final model never needs [18,20]. Whether any of this compression converts into an actual reduction in deployed energy use depends, more than the pruning algorithm itself, on whether the resulting sparsity pattern matches what the target hardware can exploit — a lesson the field has learned largely through the practical gap between unstructured pruning's high theoretical compression and its historically limited real-world speedup, and one that newer hardware

support for structured and semi-structured sparsity is only now beginning to close [22,23].

Placed alongside the broader AI-sustainability literature, pruning reads not as a standalone solution but as one well-evidenced lever among several — quantization, efficient hardware, data-center siting, and infrastructure-level engineering among the others — that together have produced some genuinely large reported efficiency gains in production settings [1,2,4]. The field's own self-assessment, via Blalock et al.'s meta-analysis, is that it needs more consistent benchmarking and more hardware-grounded metrics before individual results can be compared confidently across papers [5], and we share that assessment rather than treating any single number cited in this article as a fully general law. What we can say with reasonable confidence, having read this literature closely, is that sparsification is neither hype nor a solved problem: it is a well-evidenced, still-developing set of techniques whose benefits for sustainable AI deployment are real, contingent on hardware co-design, and worth the continued, honestly-benchmarked attention the field is increasingly giving them.

10. References

1. Strubell, E., Ganesh, A., McCallum, A. (2019) Energy and Policy Considerations for Deep Learning in NLP. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL). DOI: [10.48550/arXiv.1906.02243](https://doi.org/10.48550/arXiv.1906.02243)
2. Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., Dean, J. (2021) Carbon Emissions and Large Neural Network Training. arXiv preprint. DOI: [10.48550/arXiv.2104.10350](https://doi.org/10.48550/arXiv.2104.10350)
3. Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O. (2020) Green AI. Communications of the ACM, 63(12): 54-63. DOI: [10.1145/3381831](https://doi.org/10.1145/3381831)
4. Wu, C.-J., Raghavendra, R., Gupta, U., Acun, B., Ardalani, N., Maeng, K., Chang, G., Aga, F., Huang, J., Bai, C., et al. (2022) Sustainable AI: Environmental Implications, Challenges and Opportunities. Proceedings of Machine Learning and Systems (MLSys) 4. DOI: [10.48550/arXiv.2111.00364](https://doi.org/10.48550/arXiv.2111.00364)
5. Blalock, D., Ortiz, J.J.G., Frankle, J., Gutttag, J. (2020) What is the State of Neural Network Pruning? Proceedings of Machine Learning and Systems (MLSys) 2. DOI: [10.48550/arXiv.2003.03033](https://doi.org/10.48550/arXiv.2003.03033)
6. Han, S., Pool, J., Tran, J., Dally, W.J. (2015) Learning both Weights and Connections for Efficient Neural Networks.

- Advances in Neural Information Processing Systems (NeurIPS) 28. DOI: [10.48550/arXiv.1506.02626](https://doi.org/10.48550/arXiv.1506.02626)
7. Han, S., Mao, H., Dally, W.J. (2016) Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. International Conference on Learning Representations (ICLR). DOI: [10.48550/arXiv.1510.00149](https://doi.org/10.48550/arXiv.1510.00149)
 8. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J. (2017) Pruning Convolutional Neural Networks for Resource Efficient Inference. International Conference on Learning Representations (ICLR). DOI: [10.48550/arXiv.1611.06440](https://doi.org/10.48550/arXiv.1611.06440)
 9. Molchanov, P., Mallya, A., Tyree, S., Frosio, I., Kautz, J. (2019) Importance Estimation for Neural Network Pruning. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). DOI: [10.48550/arXiv.1906.10771](https://doi.org/10.48550/arXiv.1906.10771)
 10. Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T. (2019) Rethinking the Value of Network Pruning. International Conference on Learning Representations (ICLR). DOI: [10.48550/arXiv.1810.05270](https://doi.org/10.48550/arXiv.1810.05270)
 11. Gale, T., Elsen, E., Hooker, S. (2019) The State of Sparsity in Deep Neural Networks. arXiv preprint. DOI: [10.48550/arXiv.1902.09574](https://doi.org/10.48550/arXiv.1902.09574)
 12. Frankle, J., Carbin, M. (2019) The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. International Conference on Learning Representations (ICLR). DOI: [10.48550/arXiv.1803.03635](https://doi.org/10.48550/arXiv.1803.03635)
 13. Renda, A., Frankle, J., Carbin, M. (2020) Comparing Rewinding and Fine-tuning in Neural Network Pruning. International Conference on Learning Representations (ICLR). DOI: [10.48550/arXiv.2003.02389](https://doi.org/10.48550/arXiv.2003.02389)
 14. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P. (2017) Pruning Filters for Efficient ConvNets. International Conference on Learning Representations (ICLR). DOI: [10.48550/arXiv.1608.08710](https://doi.org/10.48550/arXiv.1608.08710)
 15. He, Y., Zhang, X., Sun, J. (2017) Channel Pruning for Accelerating Very Deep Neural Networks. IEEE International Conference on Computer Vision (ICCV). DOI: [10.48550/arXiv.1707.06168](https://doi.org/10.48550/arXiv.1707.06168)
 16. He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., Han, S. (2018) AMC: AutoML for Model Compression and Acceleration on Mobile Devices. European Conference on Computer Vision (ECCV). DOI: [10.48550/arXiv.1802.03494](https://doi.org/10.48550/arXiv.1802.03494)
 17. Zhu, M., Gupta, S. (2017) To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression. arXiv preprint. DOI: [10.48550/arXiv.1710.01878](https://doi.org/10.48550/arXiv.1710.01878)
 18. Mocanu, D.C., Mocanu, E., Stone, P., Nguyen, P.H., Gibescu, M., Liotta, A. (2018) Scalable Training of Artificial Neural Networks with Adaptive Sparse Connectivity Inspired by Network Science. Nature Communications, 9: 2383. DOI: [10.1038/s41467-018-04316-3](https://doi.org/10.1038/s41467-018-04316-3)
 19. Mostafa, H., Wang, X. (2019) Parameter Efficient Training of Deep Convolutional Neural Networks by Dynamic Sparse Reparameterization. Proceedings of the 36th International Conference on Machine Learning (ICML), PMLR 97. DOI: [10.48550/arXiv.1902.05967](https://doi.org/10.48550/arXiv.1902.05967)
 20. Evci, U., Gale, T., Menick, J., Castro, P.S., Elsen, E. (2020) Rigging the Lottery: Making All Tickets Winners. Proceedings of the 37th International Conference on Machine Learning (ICML), PMLR 119. DOI: [10.48550/arXiv.1911.11134](https://doi.org/10.48550/arXiv.1911.11134)
 21. Louizos, C., Welling, M., Kingma, D.P. (2018) Learning Sparse Neural Networks through L0 Regularization. International Conference on Learning Representations (ICLR). DOI: [10.48550/arXiv.1712.01312](https://doi.org/10.48550/arXiv.1712.01312)
 22. Mishra, A., Latorre, J.A., Pool, J., Stosic, D., Stosic, D., Venkatesh, G., Yu, C., Micikevicius, P. (2021) Accelerating Sparse Deep Neural Networks. arXiv preprint. DOI: [10.48550/arXiv.2104.08378](https://doi.org/10.48550/arXiv.2104.08378)

23. Sze, V., Chen, Y.-H., Yang, T.-J., Emer, J.S. (2017) Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE*, 105(12): 2295-2329. DOI: [10.1109/JPROC.2017.2761740](https://doi.org/10.1109/JPROC.2017.2761740)
24. Sanh, V., Wolf, T., Rush, A.M. (2020) Movement Pruning: Adaptive Sparsity by Fine-Tuning. *Advances in Neural Information Processing Systems (NeurIPS)* 33. DOI: [10.48550/arXiv.2005.07683](https://doi.org/10.48550/arXiv.2005.07683)

11. Declarations

Funding

This literature synthesis received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflict of Interest

The authors declare that they have no conflict of interest.

Data Availability Statement

This article is a literature synthesis and generated no new data. All figures and results discussed are drawn from the cited sources, which readers should consult for the underlying data and code availability statements.

Ethical Approval

Not applicable. This work did not involve human participants, animal subjects, or the collection of primary experimental data.