Projective Pruning for Decoupling Weights

Alexander Kovalenko (✉) and Tommy Chu

Faculty of Information Technology, Czech Technical University,
Thákurova 9, 160 00 Prague, Czech Republic
{chutommy,alexander.kovalenko}@fit.cvut.cz

Abstract. This paper proposes Projective Pruning, a structured deep neural network sparsification technique that removes highly correlated weights, as they provide a minimal contribution to the parameter subspace. Due to the inefficiencies in deep neural networks caused by excessive overparametrization and highly correlated weights, the method enables parameter compression while maintaining the high performance of the models. The approach incorporates a redistribution mechanism to preserve model performance and expressiveness. Evaluations on multiple vision and language benchmarks, including large language model architectures, demonstrate that, unlike most other pruning methods, Projective Pruning delivers reliable compression while ensuring stable model performance. Applying this method improves retrainability and achieves competitive results compared to existing structured pruning methods.

Keywords: Model compression · Structured pruning · Sparsification

1 Introduction

In recent years, deep learning has profoundly transformed many disciplines such as computer vision, natural language processing, speech recognition, recommendation systems, and computational biology. These advancements have enabled unprecedented capabilities in pattern recognition, content generation, and decision-making systems across various domains [?, ?].

The most performing models, however, often have billions of parameters and prohibitively high training costs [?, ?]. For instance, state-of-the-art large language models (LLM) such as GPT-4 are estimated to have over 1.8 trillion parameters [?], requiring thousands of GPU-years for training and costing tens of millions of dollars [?].

The environmental impact of such computational demands is highly concerning. Training a single large transformer model can produce carbon emissions equivalent to the lifetime emissions of five automobiles [?]. The energy consumption of modern AI systems has been increasing exponentially, with energy usage doubling approximately every 3.4 months between 2012 and 2022 [?]. This trajectory raises significant sustainability concerns about the future development of deep learning technologies.

This challenge gave rise to the paradigm of foundation models, which are pretrained on vast, generic datasets, and can be reusably fine-tuned for various downstream tasks [?]. This approach amortizes the initial training costs across multiple applications, improving overall efficiency.

Moreover, while training constitutes a substantial initial energy investment, it is the inference process—the deployment and repeated use of trained models—that accounts for the majority of the total energy consumption over a model's lifetime. Recent studies indicate that inference operations represent up to 90% of the total computational cost for widely deployed models [?, ?]. This imbalance is particularly pronounced in commercial applications of LLM that serve millions of queries daily.

Nevertheless, even after training, large models require substantial compute, and memory resources for inference [?, ?]. The memory footprint alone can exceed the capabilities of many edge devices, limiting AI deployment in resource-constrained environments such as mobile phones, IoT devices, and autonomous vehicles [?].

One of the possible solutions to reduce the inference and/or training cost of deep learning models is model pruning. The concept of pruning builds upon the observation that neural networks are typically over-parameterized, containing significantly more parameters than necessary to represent their learned functions [?]. This redundancy creates an opportunity for compression without substantial performance degradation.

Therefore, in this paper, we propose a new flexible neural pruning method that decouples weights in deep learning models. Projective Pruning uses a dependency graph [?] to group weights, and prune them based on redundancy computed by their relative distance to orthogonal projections on the subspace spanned by other parameters in the same group. The pruned weights are then used to redistribute lost signals to the remaining parameters to maintain the model's representational capacity. We test our method on various convolutional and attention-based models against other pruning algorithms. On average, Projective Pruning outperforms existing methods in terms of post-pruning retrainability, and raw performance retention.

2 Background

Model compression has a rich history, tracing back to the late 1980s with pioneering pruning techniques such as Optimal Brain Damage [?] and Optimal Brain Surgeon [?]. These early methods aimed to compress neural networks by systematically removing parameters based on their significance using second-derivative and magnitude-based criteria. Initially, such pruning approaches were driven primarily by necessity due to the severely limited computing resources available at the time.

As deep learning architectures and hardware capabilities evolved throughout the decades, the field of model compression expanded significantly, with Han et al.'s Deep Compression [?] marking a pivotal advancement in modern approaches. Contemporary research has refined these techniques and established a temporal classification framework for pruning methodologies. According to recent surveys

[?], modern pruning methods are typically applied at one of three distinct stages in a model's lifecycle:

- **Pruning before training** [?, ?]: The network is pruned at initialization, before training begins. This approach is motivated by improved convergence speed and reduced memory costs of sparse training.
- **Pruning during training** [?, ?]: Pruning is integrated into the training process, and the model's size is iteratively reduced as it learns. The large number of parameters initially helps the model recover from poor initialization [?], and quickly traverse to the low area of the loss landscape [?]. At the same time, the iterative pruning reduces the model's size, and improve its generalization.
- **Pruning after training** [?, ?, ?, ?]: The model is fully trained first, then pruned, and finally fine-tuned to regain any lost performance. This strategy targets existing pretrained models like vision and large language models with computationally intensive convolutional or transformer-based architectures.

Beyond pruning, the model compression landscape encompasses several complementary approaches. Knowledge distillation [?, ?] transfers information from a larger "teacher" model to a smaller "student" network by training the smaller network using the output logits of the fully converged larger model. Low-rank approximation [?] replaces large matrices with products of smaller matrices to reduce parameter counts while preserving essential representational capacity. Quantization [?] achieves speedups and memory savings by reducing the numerical precision of weights and activations without necessarily changing the network architecture.

The structural nature of pruning operations represents another important classification dimension. Unstructured pruning involves setting individual parameters to zero while maintaining the original network architecture. This approach yields a sparse network that often demonstrates improved generalization capabilities but primarily benefits from specialized hardware such as FPGAs [?, ?]. A significant limitation of unstructured pruning is that the overall dimensions of weight matrices remain unchanged, resulting in minimal speedup on conventional computing hardware despite the reduction in non-zero parameters.

Structured pruning, by contrast, systematically removes entire groups of parameters, such as neurons, layers, or channels. This approach directly reduces the model's dimensions, leading to concrete improvements in both inference speed and memory utilization across standard hardware platforms. Between these approaches lies semi-structured pruning, which removes individual parameters within organized structural groups. Given that structured sparsity patterns are increasingly supported by GPU manufacturers, these approaches can deliver substantial improvements in both training and inference performance [?].

Through this systematic progression of techniques, model compression has evolved from a necessity imposed by hardware constraints to a sophisticated field focused on optimizing the efficiency-performance trade-off across diverse neural network architectures and application domains.

3 Related Work

Pruning has significantly advanced network acceleration, with numerous studies demonstrating its effectiveness across various architectural paradigms. This section outlines key pruning methodologies that form the foundation of modern network compression techniques.

3.1 Magnitude-Based Pruning Approaches

Magnitude-based pruning represents one of the most widely adopted approaches due to its simplicity and effectiveness. Han et al. introduced Deep Compression [?], which established simple structured magnitude pruning as a strong baseline by removing weights based on their L2 norms. Recent comprehensive surveys have confirmed that this straightforward approach remains competitive with many newer techniques [?].

Building upon this foundation, Layer-Adaptive Magnitude-based Pruning (LAMP) [?] offers a notable advancement by dynamically adjusting layerwise sparsity based on L2 distortion metrics. This approach eliminates the need for extensive hyperparameter tuning, making pruning more accessible and practical for widespread implementation.

3.2 Alternative Pruning Criteria

Interestingly, research has revealed that random pruning produces surprisingly effective results. Studies by Liu et al. [?] and Sui et al. [?] demonstrate that random weight elimination can offer competitive performance comparable to more sophisticated techniques, challenging assumptions about the necessity of complex pruning criteria.

Network Slimming [?] takes a different approach by enforcing channel-level sparsity through the addition of scaling factors to channel-wise parameters. This technique effectively identifies and removes entire channels with minimal impact on network performance.

Filter Pruning via Geometric Median (FPGM) [?] diverges from mainstream approaches by focusing on redundancy rather than magnitude. Instead of eliminating low-magnitude weights, FPGM identifies and removes filters that are most redundant within the representational space of each layer, preserving unique feature extractors.

3.3 Advanced Mathematical Frameworks

EigenDamage [?] introduces a more sophisticated mathematical framework by leveraging Kronecker-factored eigenbases for structured pruning. This approach aims for more precise weight removal by analyzing the eigenvalues of the Hessian matrix, targeting parameters that contribute minimally to the loss landscape.

3.4 Architecture-Agnostic Methods

DepGraph [?] represents a significant advancement by enabling general structural pruning across diverse architectures, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Graph Neural Networks (GNNs), and Transformers. By modeling inter-layer dependencies through a graph-based approach, DepGraph ensures structural integrity through grouped norm pruning while providing a flexible framework that can accommodate arbitrary pruning criteria.

3.5 Transformer-Specific Techniques

As transformer architectures have become increasingly dominant, specialized pruning methods have emerged. SliceGPT [?] employs Principal Component Analysis (PCA)-based projections to identify and remove weights associated with the smallest singular values in transformer models. While highly effective for large language models, this approach has limitations: it can only prune dimensions that do not directly interact with nonlinear structures and requires coordinated pruning across multiple layers to maintain architectural consistency.

These pruning methodologies collectively demonstrate the evolution from simple heuristic approaches to sophisticated, architecture-specific techniques that leverage advanced mathematical concepts to optimize neural network efficiency while preserving performance.

4 Methodology

We first introduce Projective Pruning for fully connected layers and then extend the concept to arbitrary learning structures. Projective Pruning decouples weights by identifying neurons with high redundancy based on their projection onto the subspace formed by other neurons in the same layer. If a neuron's signal can be closely represented as a linear combination of others, it is pruned, and its weights are redistributed across the remaining neurons. The strength of the redistributed signal varies for each parameter and is proportional to the coefficients of the linear combination. This approach minimizes the impact on the model's overall expressiveness while compensating for the lost information.

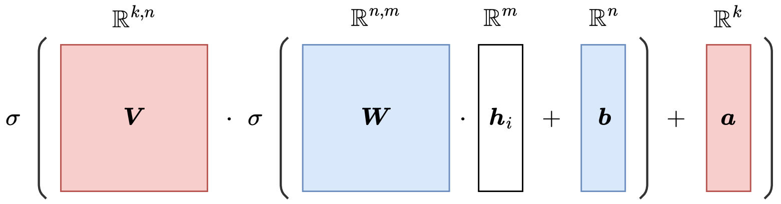
4.1 Fully Connected Layers

Consider two adjacent fully connected layers with weights $\boldsymbol{W} \in \mathbb{R}^{n,m}$, bias $\boldsymbol{b} \in \mathbb{R}^n$ in the first layer and $\boldsymbol{V} \in \mathbb{R}^{k,n}$, $\boldsymbol{a} \in \mathbb{R}^k$ in the second layer. The forward pass through these two layers is given by

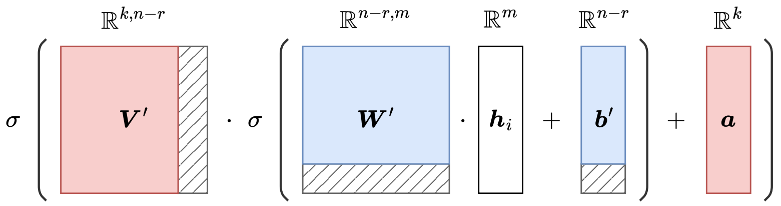
$$\boldsymbol{h}_{i+1} = \sigma(\boldsymbol{W}\boldsymbol{h}_i + \boldsymbol{b}), \tag{1}$$

$$\boldsymbol{h}_{i+2} = \sigma(\boldsymbol{V}\boldsymbol{h}_{i+1} + \boldsymbol{a}), \tag{2}$$

where σ is a nonlinear activation function, and $\boldsymbol{h}_i \to \boldsymbol{h}_{i+1} \to \boldsymbol{h}_{i+2}$ are hidden states that transition from $\mathbb{R}^m$ to $\mathbb{R}^n$ to $\mathbb{R}^k$. Projective Pruning decreases the shared dimension n by removing redundant neurons (output dimension) in the first layer and input dimension in the second.



(a) Two adjacent fully connected layers before pruning, with a shared dimension n.



(b) The same layers after pruning r redundant neurons, reducing the shared dimension to $n-r$ while preserving the input dimension m and output dimension k.

Fig. 1: Illustration of pruning a dense layer.

Pruning a single neuron corresponds to removing a row in $\boldsymbol{W}$, its associated element in $\boldsymbol{b}$, and a column in $\boldsymbol{V}$. Initially, the overall mapping of these two layers is $\sigma(\boldsymbol{V} \cdot \sigma(\boldsymbol{W}\boldsymbol{h}_i + \boldsymbol{b}) + \boldsymbol{a})$, as illustrated in ??. After pruning r rows, we get $\sigma(\boldsymbol{V}' \cdot \sigma(\boldsymbol{W}'\boldsymbol{h}_i + \boldsymbol{b}') + \boldsymbol{a})$ with reduced weights $\boldsymbol{W}' \in \mathbb{R}^{n-r,m}$, $\boldsymbol{b}' \in \mathbb{R}^{n-r}$, $\boldsymbol{V}' \in \mathbb{R}^{k,n-r}$ as shown in ??. The pruned composite function maintains the same domain $\mathbb{R}^m$ and codomain $\mathbb{R}^k$, so no structural changes are required.

4.2 Projective Pruning

Our goal is to determine the reduced weights $\boldsymbol{W}', \boldsymbol{b}', \boldsymbol{V}'$, so that the new composite transformation approximates the original one ($\boldsymbol{a}$ is unaffected, and thus ignored):

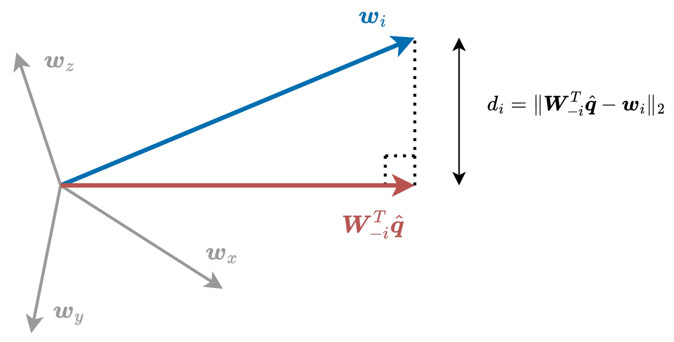
$$\forall \boldsymbol{h}_i \in \mathcal{D}: \qquad \boldsymbol{V}' \cdot \sigma(\boldsymbol{W}'\boldsymbol{h}_i + \boldsymbol{b}') \overset{\text{approx.}}{\approx} \boldsymbol{V} \cdot \sigma(\boldsymbol{W}\boldsymbol{h}_i + \boldsymbol{b}). \tag{3}$$

The nonlinear function σ prevents us from simply merging the two layers into one $(\boldsymbol{V}\boldsymbol{W})\boldsymbol{h}_i + (\boldsymbol{V}\boldsymbol{b} + \boldsymbol{a})$ and pruning the entire layer losslessly. Minimizing the difference between the left and the right side of the ?? requires either some training data or assumptions about its distribution (over which domain to minimize). Unlike data-dependent methods, Projective Pruning prioritizes general expressivity over data fitness, and does not use any data assumptions.

Criterion. Neurons are pruned iteratively one at a time. The most redundant neuron (row) in $\boldsymbol{W}$ is identified as the one which is closest to its projection onto the subspace spanned by the other neurons. This heuristic assumes that highly correlated neurons can be fully replaced by strengthening the signal of similar neurons.

Although this assumption holds only partially as neurons are not active all the time or the activation function may not have a linear form (similar to ReLU, GELU, Swish), pruning under this assumption encourages weights to better utilize the full parameter range. The reliance on this assumption can be regulated by the parameters α, β, γ (discussed in **??**) which control the amount of the pruned signal to be redistributed.

4.3 Orthogonal Projection



$\boldsymbol{w}_i$
$\boldsymbol{w}_z$
$d_i = \|\boldsymbol{W}_{-i}^T\hat{\boldsymbol{q}} - \boldsymbol{w}_i\|_2$
$\boldsymbol{W}_{-i}^T\hat{\boldsymbol{q}}$
$\boldsymbol{w}_x$
$\boldsymbol{w}_y$

Fig. 2: Assuming the vector set $(\boldsymbol{w}_i^T, \boldsymbol{w}_x^T, \boldsymbol{w}_y^T, \boldsymbol{w}_z^T, \ldots)$ in $\boldsymbol{W}$ is linearly independent: The figure shows the projection of $\boldsymbol{w}_i^T$ onto the subspace spanned by the other rows. The projection $\boldsymbol{W}_{-i}^T\hat{\boldsymbol{q}}$ resides within the subspace $\langle \boldsymbol{w}_x, \boldsymbol{w}_y, \boldsymbol{w}_z, \ldots\rangle$ and its distance from $\boldsymbol{w}_i$ defines the redundancy score d_i.

Let $\boldsymbol{w}_i^T$ denote the i-th row of $\boldsymbol{W}$, and let $\boldsymbol{W}_{-i}$ be $\boldsymbol{W}$ without this row. The projection of $\boldsymbol{w}_i^T$ onto the subspace spanned by the rows of $\boldsymbol{W}_{-i}$ is found by solving:

$$\hat{\boldsymbol{q}} = \arg\min_{\boldsymbol{q}} \|\boldsymbol{W}_{-i}^T\boldsymbol{q} - \boldsymbol{w}_i\|_2, \tag{4}$$

where $\hat{\boldsymbol{q}}$ represents the coefficients of the linear combination of the projection. Equivalently, $\boldsymbol{q}$ is the solution $\hat{\boldsymbol{q}}$ only if $\boldsymbol{W}_{-i}^T\boldsymbol{q} - \boldsymbol{w}_i$ is orthogonal to the span $\langle \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{i-1}, \boldsymbol{w}_{i+1}, \ldots, \boldsymbol{w}_n\rangle$. This yields the normal equation $\boldsymbol{W}_{-i}(\boldsymbol{W}_{-i}^T\hat{\boldsymbol{q}} -$

$\boldsymbol{w}_i) = \mathbf{0}$ with a closed-form solution $\hat{\boldsymbol{q}} = \left(\boldsymbol{W}_{-i}\boldsymbol{W}_{-i}^T\right)^{-1}\boldsymbol{W}_{-i}\boldsymbol{w}_i$. The redundancy is then quantified by the norm of the difference between the original vector and its projection $d_i = \|\boldsymbol{W}_{-i}^T\hat{\boldsymbol{q}} - \boldsymbol{w}_i\|_2$, as illustrated in **??**.

We introduce a parameter λ to stabilize the matrix inverse. The full expression of the criterion for selecting the neuron to prune is therefore given by

$$\arg\min_i \|\boldsymbol{W}_{-i}^T\left(\boldsymbol{W}_{-i}\boldsymbol{W}_{-i}^T + \lambda\mathbb{I}\right)^{-1}\boldsymbol{W}_{-i}\boldsymbol{w}_i - \boldsymbol{w}_i\|_2. \tag{5}$$

4.4 Redistribution Mechanism

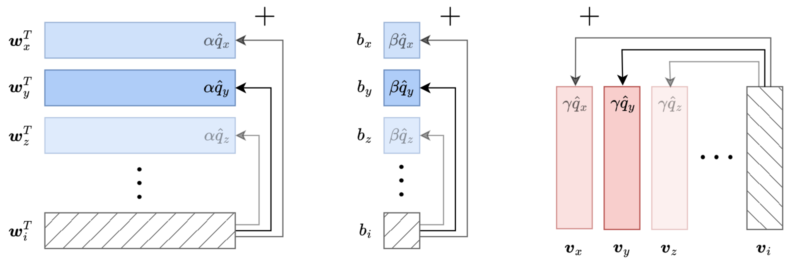


Fig. 3: The redistribution schema in which $\boldsymbol{w}_i$ is pruned, with its weights readjusting unpruned parameters using the coefficients $\hat{\boldsymbol{q}}$ of the orthogonal projection.

After identifying the neuron, we remove its row from $\boldsymbol{W}$, its bias element from $\boldsymbol{b}$, and its associated column from $\boldsymbol{V}$. This keeps the unpruned signals properly aligned while maintaining the model's overall structure. We also reuse the coefficients of the linear combination to proportionally reinforce the remaining weights (**??**):

$$\forall j \neq i: \quad \boldsymbol{w}_j \leftarrow \boldsymbol{w}_j \cdot (1 + \alpha\hat{\boldsymbol{q}}_j), \tag{6}$$

$$b_j \leftarrow b_j \cdot (1 + \beta\hat{\boldsymbol{q}}_j), \tag{7}$$

$$\boldsymbol{v}_j \leftarrow \boldsymbol{v}_j \cdot (1 + \gamma\hat{\boldsymbol{q}}_j), \tag{8}$$

where $\boldsymbol{w}_j^T$, b_j, $\boldsymbol{v}_j$ are elements of $\boldsymbol{W}$, $\boldsymbol{b}$, $\boldsymbol{V}$, and α, β, and γ determines the amount of signal redistributed back.

4.5 Generalization to Other Structures

Projective Pruning can also be applied to 2D convolutional layers by reshaping the weights into a matrix, where each row represents a flattened filter. The pruning criterion and the mathematical framework stay the same: redundancy is measured across filters within the same layer (**??**).

Conceptually, this can be viewed as shifting the vector space from $\mathbb{R}^n$ to $\mathbb{R}^{d,h,w}$, where d is the input channel, and (h, w) is the kernel size. The pruned dimension corresponds to the shared channel dimension between consecutive layers. This concept extends naturally to other types of convolutions.

More generally, flattening weights allows this analogy to extend to any deep learning structure where prunable units form a vector space of the same dimension. For example, in attention layers, Projective Pruning targets the key-query matrices. Projections are computed on joint key-query pairs, and pruning is applied to the shared key-query dimension.

5 Efficient Implementation

A naive implementation of the algorithm is computationally prohibitive. It requires inverting a large matrix for each neuron at every pruning step. To make this approach feasible, we introduce three key optimizations: a fast method for matrix inversion, a lazy update scheme for iterative pruning, and the use of weight masking.

5.1 Fast Matrix Inversion

To identify which neuron to prune, we evaluate expression (**??**), which requires the inverse of a different matrix $\mathbf{G}_{-i} :- \boldsymbol{W}_{-i}\boldsymbol{W}_{-i}^T + \lambda\mathbb{I} \in \mathbb{R}^{n-1,n-1}$ for each of the n candidate rows. However, these inverses do not need to be computed independently.

Instead, we first compute the inverse of the full matrix $\mathbf{G} :- \boldsymbol{W}\boldsymbol{W}^T + \lambda\mathbb{I}$. We then derive each required sub-matrix inverse $\mathbf{G}_{-i}^{-1}$ from the blocks of $\mathbf{G}^{-1}$. Let the symmetric positive-definite matrices $\mathbf{G}$ and $\mathbf{G}^{-1}$ be permuted and partitioned as follows:

$$\mathbb{P}_{i,n}\mathbf{G}\mathbb{P}_{i,n} = \begin{pmatrix} \mathbf{G}_{-i} & \mathbf{G}_i \\ \mathbf{G}_i^T & \mathbf{G}_{ii} \end{pmatrix} =: \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{pmatrix}, \tag{9}$$

$$\mathbb{P}_{i,n}\mathbf{G}^{-1}\mathbb{P}_{i,n} = \begin{pmatrix} \mathbf{g}_{-i} & \mathbf{g}_i \\ \mathbf{g}_i^T & \mathbf{g}_{ii} \end{pmatrix} =: \begin{pmatrix} \mathbf{a} & \mathbf{b} \\ \mathbf{b}^T & \mathbf{c} \end{pmatrix}, \tag{10}$$

where $\mathbb{P}_{i,n}$ is a permutation matrix. The block $\mathbf{G}_{-i}$ is the matrix $\mathbf{G}$ with the i-th row and column removed, and $\mathbf{G}_i \in \mathbb{R}^{n-1}$ is the removed column excluding the i-th element $\mathbf{G}_{ii} \in \mathbb{R}$. The blocks of $\mathbf{G}^{-1}$ are defined analogously.

From the identity $\mathbb{I}_n = (\mathbb{P}_{i,n}\mathbf{G}\mathbb{P}_{i,n})(\mathbb{P}_{i,n}\mathbf{G}^{-1}\mathbb{P}_{i,n})$, we can establish a relationship between the blocks:

$$\begin{pmatrix} \mathbb{I}_{n-1} & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{Aa} + \mathbf{Bb}^T & \mathbf{Ab} + \mathbf{Bc} \\ \mathbf{B}^T\mathbf{a} + \mathbf{Cb}^T & \mathbf{B}^T\mathbf{b} + \mathbf{Cc} \end{pmatrix} \tag{11}$$

The upper blocks give $\mathbf{Aa} + \mathbf{Bb}^T = \mathbb{I}_{n-1}$, and $\mathbf{Ab} + \mathbf{Bc} = 0$, from which we derive:

$$\mathbf{Bb}^T = \mathbf{B}(\mathbf{cc}^{-1})\mathbf{b}^T = (\mathbf{Bc})\mathbf{c}^{-1}\mathbf{b}^T = -\mathbf{Abc}^{-1}\mathbf{b}^T \tag{12}$$

$$\mathbf{Aa} + \mathbf{Bb}^T = \mathbf{Aa} - \mathbf{Abc}^{-1}\mathbf{b}^T = \mathbf{A}(\mathbf{a} - \mathbf{bc}^{-1}\mathbf{b}^T) = \mathbb{I}_{n-1} \tag{13}$$

The first equation in (**??**) is valid because $\mathbf{c}$, a diagonal element of a positive-definite matrix, is guaranteed to be strictly positive ($\mathbf{c} \neq 0$). The final equation in (**??**) provides a direct formula for the required inverse:

$$\mathbf{G}_{-i}^{-1} = \mathbf{A}^{-1} = \mathbf{a} - \frac{1}{\mathbf{c}}\mathbf{bb}^T = \mathbf{g}_{-i} - \frac{1}{\mathbf{g}_{ii}}\mathbf{g}_i\mathbf{g}_i^T. \tag{14}$$

This efficient formulation allows us to derive all necessary inverses from the sub-blocks of the single inverse $\mathbf{G}^{-1}$, and avoid explicit matrix inversions inside a loop.

5.2 Iterative Pruning with Lazy Updates

After pruning a neuron, the weight matrix $\boldsymbol{W}$ changes, which in turn invalidates the previously computed matrix $\mathbf{G}$ and its inverse. Recomputing $\mathbf{G}^{-1}$ and the associated $\mathbf{G}_{-i}^{-1}$ matrices at every step remains expensive. The lazy update scheme avoids this by tracking changes without modifying the underlying weight matrices until the end.

We track the cumulative multiplicative updates for each neuron using coefficient vectors, $\boldsymbol{u}_\phi^{(t)}$ for $\phi \in \{\alpha, \beta, \gamma\}$, initialized as vectors of ones. At each pruning step t, the coefficient vector for the unpruned neurons is updated in-place:

$$\boldsymbol{u}_\phi^{(t+1)} \leftarrow \boldsymbol{u}_\phi^{(t)} \odot \left(1 + \phi\hat{\boldsymbol{q}}_i^{(t)}\right), \tag{15}$$

where $\odot$ is the element-wise product and $\hat{\boldsymbol{q}}_i^{(t)}$ is the coefficient vector of the linear combination of the projection of the pruned row $\boldsymbol{w}_i^T$. To maintain its validity for subsequent steps, we apply a similar update:

$$\forall j \text{ not pruned}: \quad \hat{\boldsymbol{q}}_j^{(t+1)} \leftarrow \hat{\boldsymbol{q}}_j^{(t)} \odot \frac{1}{1 + \phi\hat{\boldsymbol{q}}_i^{(t)}} \tag{16}$$

The final weight updates are applied only once after all r pruning iterations are complete:

$$\forall j \text{ not pruned}: \quad \boldsymbol{w}_j' \leftarrow \boldsymbol{w}_j \cdot \left(\boldsymbol{u}_\alpha^{(r)}\right)_j, \tag{17}$$

$$\boldsymbol{b}_j' \leftarrow \boldsymbol{b}_j \cdot \left(\boldsymbol{u}_\beta^{(r)}\right)_j, \tag{18}$$

$$\boldsymbol{v}_j' \leftarrow \boldsymbol{v}_j \cdot \left(\boldsymbol{u}_\gamma^{(r)}\right)_j. \tag{19}$$

5.3 Masking and Views

To support the lazy update scheme, pruned rows are not physically removed from data structures during the iteration. Instead, a binary mask tracks the status of each neuron.

Operations with weights and coefficients are performed efficiently on unpruned elements via masking (eq. **??**, **??**) and views (eq. **??**). By preserving the shape of the underlying matrices, this method avoids costly memory reallocations while maintaining the relevant coefficients up-to-date.

6 Experimental Results

We evaluate Projective Pruning on various deep learning architectures and datasets covering both computer vision (image and point cloud classification) and language modeling (next-token prediction). All experiments are conducted in a pruning-after-training scenario, with results reported either with or without recovery fine-tuning. The former evaluates model trainability after the compression, while the latter evaluates the pruning method's ability to retain accuracy on its own.

Additionally, we analyze the impact of redistribution parameters α, β, γ, and the regularization term λ on model performance. Overall, the results indicate that Projective Pruning is a viable alternative to existing structured pruning algorithms. The code is available at github.com/bnjpm/projective-pruning.

6.1 Vision Tasks

We evaluate Projective Pruning on image classification tasks using the CIFAR-10/100 and ImageNet datasets [**?**, **?**], and on point cloud classification with ModelNet40 [**?**]. Architectures include VGG-19 [**?**], ResNet56 [**?**], MobileNetV2 [**?**], ViT [**?**], and PointNet [**?**]. Projective Pruning is compared against eight baselines: random pruning, MagnitudeL2, Slimming [**?**], FPGM [**?**], EigenDamage [**?**], LAMP [**?**], DepGraph, and DepGraph-SL [**?**]. Models are pruned globally at 2× and 3× FLOPs reduction using dependency graph-based grouping [**?**].

Results in **??** demonstrate that Projective Pruning achieves state-of-the-art performance on common benchmark computer vision datasets. On CIFAR-10, it outperforms competitors for ResNet56 at 3× FLOPs reduction (0.9402 vs. 0.9401 for DepGraph-SL) and MobileNetV2 at 3× FLOPs reduction (0.9051 vs. 0.9041 for MagnitudeL2). For CIFAR-100, it attains the highest accuracy on VGG-19 at 3× FLOPs speedup (0.7387 vs. 0.7352 for DepGraph-SL) and ResNet56 at 2× FLOPs reduction (0.7256 vs. 0.7239 for DepGraph-SL). On ModelNet40, Projective Pruning dominates PointNet at both 2× (0.8966) and 3× (0.8963) reduction, surpassing all baselines.

Key strengths of Projective Pruning include its superiority on lightweight architectures (e.g., MobileNetV2) and high pruning ratios (3× speedup). For example, on MobileNetV2 (CIFAR-100, 3×), it achieves 0.6911 accuracy vs.

Table 1: Models are pretrained, globally pruned, and fine-tuned on CIFAR-10/100 and ModelNet40. The results show the top-1 accuracy of pruned models with 2/3× FLOPs reduction.

(a) CIFAR-10

Model	VGG-19		ResNet56		MobileNetV2	
Accuracy	0.9368		0.9392		0.8938	
Method	2×	3×	2×	3×	2×	3×
Random	0.7538	0.6338	0.9295	0.8963	0.8561	0.8884
MagnitudeL2	<u>0.9398</u>	0.9356	0.9346	0.9315	0.8997	<u>0.9041</u>
Slimming	0.9387	0.9367	**0.9408**	0.9389	**0.9064**	0.9030
FPGM	0.9388	0.9363	0.9340	0.9241	0.9025	0.9011
EigenDamage	0.9367	0.9361	0.9292	0.9108	0.9005	0.8981
LAMP	0.9389	0.9383	0.9297	0.9133	0.9019	0.9021
DepGraph	**0.9402**	0.9370	0.9362	0.9313	0.9010	0.9000
DepGraph-SL	0.9392	**0.9387**	<u>0.9407</u>	<u>0.9401</u>	0.9045	0.8977
Projective (ours)	0.9388	<u>0.9384</u>	0.9403	**0.9402**	<u>0.9061</u>	**0.9051**

(b) CIFAR-100

Model	VGG-19		ResNet56		MobileNetV2	
Accuracy	0.7377		0.7269		0.6699	
Method	2×	3×	2×	3×	2×	3×
Random	0.7049	0.6558	0.7099	0.6708	**0.7068**	0.6042
MagnitudeL2	0.7318	0.7227	0.7143	0.6900	0.6843	0.6823
Slimming	0.7380	0.7350	0.7213	**0.7257**	0.6892	<u>0.6865</u>
FPGM	0.7357	0.7181	0.7076	0.6919	0.6856	0.6848
EigenDamage	<u>0.7403</u>	0.7284	0.6739	0.6369	0.6792	0.6848
LAMP	**0.7414**	0.7308	0.6750	0.6378	0.6824	0.6862
DepGraph	0.7326	0.7237	0.7177	0.6925	0.6847	0.6849
DepGraph-SL	0.7366	<u>0.7352</u>	<u>0.7239</u>	<u>0.7253</u>	0.6873	0.6129
Projective (ours)	0.7396	**0.7387**	**0.7256**	0.7242	<u>0.6995</u>	**0.6911**

(c) ModelNet40

Model / Accuracy	PointNet / 0.9238	
Method	2×	3×
Random	0.8517	0.8840
MagnitudeL2	0.8902	0.8886
Slimming	0.8936	<u>0.8934</u>
FPGM	0.8878	0.8870
EigenDamage	0.8912	0.8851
LAMP	0.8906	0.8849
DepGraph	0.8898	0.8886
DepGraph-SL	<u>0.8938</u>	0.8926
Projective (ours)	**0.8966**	**0.8963**

0.6865 for the Slimming method. Additionally, without fine-tuning, Projective Pruning retains expressiveness better than heuristic methods, as shown by its post-pruning accuracy on ViT models (**??**). This underscores its mathematical advantage in preserving critical signal pathways during redundancy removal, even for complex architectures.

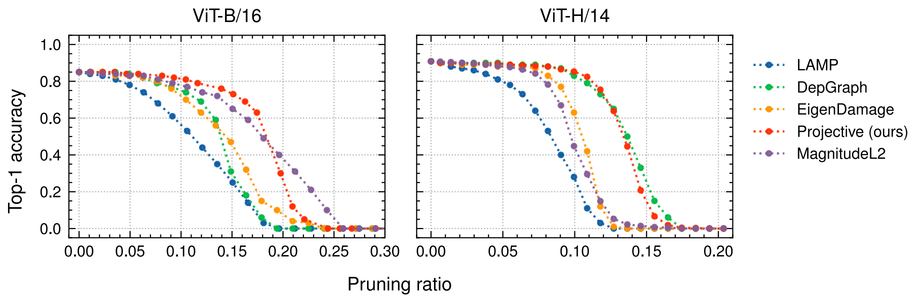


Fig. 4: Pretrained (PytorchHub) vision transformers evaluated on ImageNet. The models are pruned uniformly across all layers. Top-1 accuracy is reported with no recovery fine-tuning.

6.2 Language Modeling

We evaluate the effectiveness of Projective Pruning on the GPT-2 model using the Penn Treebank [**?**] and WikiText-2 [**?**] datasets with pre-trained weights (OpenAI) [**?**]. **??** presents the perplexity scores of the pruned models without fine-tuning. The results demonstrate that Projective Pruning consistently outperforms other methods, particularly for small to medium-sized models (124M, 355M, and 774M parameters) at moderate pruning rates (10% and 20%). For instance, on the Penn Treebank dataset with a 20% pruning ratio, Projective Pruning achieves a perplexity of 46.66 for the 124M model, significantly lower than Random (70.32), MagnitudeL2 (459.13), and DepGraph-SL (85.97). Similarly, on WikiText-2, Projective Pruning maintains strong performance, with perplexities of 39.13 (124M) and 26.26 (355M) at 20% pruning, showing its ability to retain model expressiveness after significant weight reduction.

However, at higher pruning rates (30%) and for larger models (1.5B parameters), random pruning shows competitive performance, occasionally surpassing Projective Pruning. This suggests that for highly overparametrized models, Random Pruning is seemingly a valid option due to the high level of redundancy in weights.

The key strength of Projective Pruning lies in its ability to preserve model performance at moderate pruning levels, especially for smaller architectures, as its mathematical foundation in redundancy identification and weight redistribution ensures minimal loss of information. This makes it particularly suitable for resource-constrained environments where maintaining accuracy while reducing model size is critical.

Table 2: Perplexity of the pretrained GPT-2 model after pruning on the Penn Treebank and WikiText-2 datasets. Projective Pruning on average outperforms other pruning methods across S/M/L model sizes, while random pruning shows the best performance retention in the XL model.

		Penn Treebank				WikiText-2			
Ratio	Method	124M	355M	774M	1.5B	124M	355M	774M	1.5B
0%	None	35.86	27.18	23.14	21.10	29.94	21.71	19.44	17.40
10%	Random	49.75	35.39	24.45	22.05	45.32	25.74	20.88	18.14
	MagnitudeL2	248.64	466.72	59.79	37.80	217.69	540.25	50.62	29.84
	DepGraph-SL	53.67	**28.31**	**23.52**	24.94	57.91	29.86	21.39	18.31
	Projective	**38.50**	29.43	23.92	**21.91**	**31.79**	**22.75**	**20.14**	**17.89**
20%	Random	70.32	43.34	28.18	23.57	67.29	42.30	22.48	**18.83**
	MagnitudeL2	459.13	685.59	101.99	56.20	494.08	911.08	93.29	44.18
	DepGraph-SL	85.97	41.92	26.19	26.01	81.71	44.19	22.71	18.91
	Projective	**46.66**	**35.69**	**26.10**	**23.27**	**39.13**	**26.26**	**21.81**	18.96
30%	Random	131.55	86.41	31.91	**26.49**	118.68	81.58	27.43	**20.65**
	MagnitudeL2	687.52	1053.79	152.41	91.55	858.36	1399.66	154.61	70.28
	DepGraph-SL	121.88	79.21	**29.16**	32.72	104.12	51.49	**24.19**	21.13
	Projective	**66.31**	**51.77**	31.64	26.66	**62.37**	**37.90**	25.78	21.48

6.3 Hyperparameter Analysis

Redistribution parameters. α, β, and γ regulate the redistribution of pruned signals. To study their impact, we use a simple feedforward network with two convolutional layers and four dense layers (extended LeNet) on the FashionMNIST dataset [**?**]. In **??**, pruning is applied equally across dense layers, and performance is compared against Magnitude-L_2 pruning, random pruning, and variations of Projective Pruning with different weight readjustments. Pruning is assessed without retraining to measure accuracy retention directly.

Regularization term. λ stabilizes the inverse of $\boldsymbol{W}_{-i}\boldsymbol{W}_{-i}^T$. While this matrix is typically well-conditioned, it can become ill-conditioned when rows are highly correlated. In widening fully connected layers, $\boldsymbol{W}_{-i}\boldsymbol{W}_{-i}^T$ is always singular, in which case the inverse is undefined. Adding a small value to the diagonal introduces a bias towards $\mathbf{0}$ for $\hat{\boldsymbol{q}}$ (flattens redundancy scores), but it ensures that the matrix is always full rank, and the inverse is well-defined. λ represents a trade-off between bias and stability (**??**).

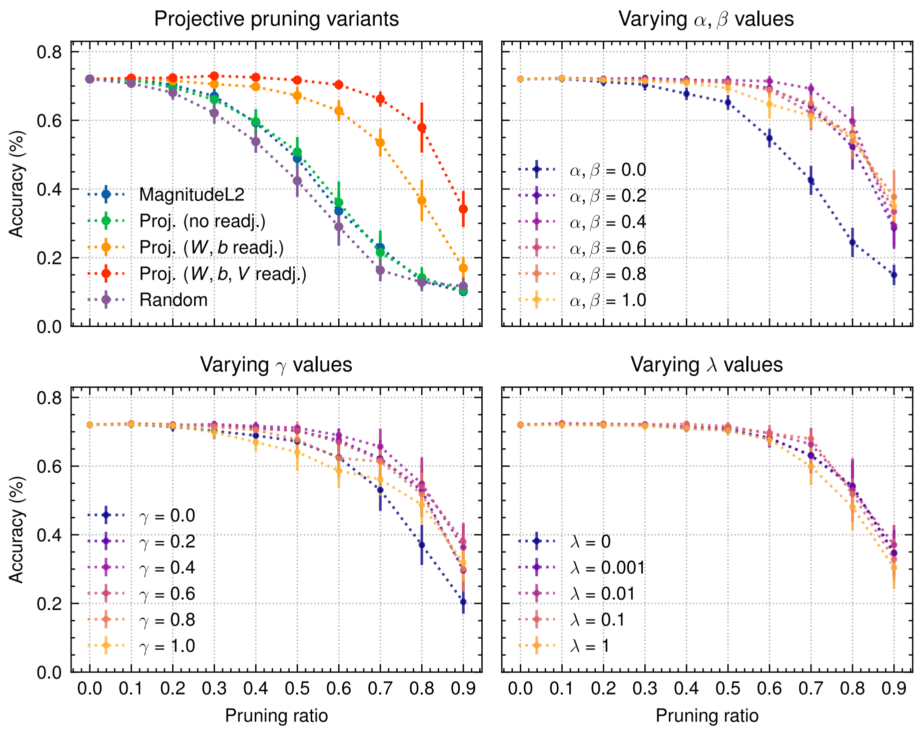


Fig. 5: (upper left) Comparison of Projective Pruning variants on FashionMNIST. Accuracy is reported after training and pruning with no retraining. Error bars show the 95% confidence interval. Other subfigures show impact of hyperparameters on model accuracy with varying α/β (upper right), γ (lower left), and λ (lower right). Non-varying parameters are set to default values: 0.5 for α, β, γ, and 10^{-3} for λ.

7 Conclusion

We introduced Projective Pruning, a structured pruning algorithm for neural compression that minimizes weight co-adaptation by removing linearly redundant parameters. Additionally, we designed a redistribution scheme that preserves model performance by reallocating the pruned weights' signal. Our method was evaluated against state-of-the-art pruning techniques across diverse domains, including standard vision and language modeling benchmarks. The results show that Projective Pruning retains model expressiveness effectively, the method performs consistently across different model sizes and pruning ratios, with particularly strong performance in scenarios involving relatively small parameter counts and aggressive pruning conditions.

Acknowledgments. This work was supported by the 2024 Student Summer Research Program at the Faculty of Information Technology, Czech Technical University in Prague. We extend our gratitude for this opportunity and the resources provided to make this work possible.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Ashkboos, S., Croci, M.L., do Nascimento, M.G., Hoefler, T., Hensman, J.: Slicegpt: Compress large language models by deleting rows and columns (2024)
2. Bommasani, R., Hudson, D.A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M.S., Bohg, J., Bosselut, A., Brunskill, E., et al.: On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258 (2021)
3. Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S.: Once-for-all: Train one network and specialize it for efficient deployment. International Conference on Learning Representations (2020)
4. Cheng, H., Zhang, M., Shi, J.Q.: A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–20 (2024)
5. Cottier, B., Rahman, R., Fattorini, L., Maslej, N., Besiroglu, T., Owen, D.: The rising costs of training frontier ai models. arXiv preprint arXiv:2405.21015 (2024)
6. Denil, M., Shakibi, B., Dinh, L., Ranzato, M., De Freitas, N.: Predicting parameters in deep learning. Advances in Neural Information Processing Systems **26** (2013)
7. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale (2021)
8. Evci, U., Gale, T., Menick, J., Castro, P.S., Elsen, E.: Rigging the lottery: Making all tickets winners (2021)
9. Fang, G., Ma, X., Song, M., Mi, M.B., Wang, X.: Depgraph: Towards any structural pruning (2023)
10. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. International Conference on Learning Representations (2019)
11. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks (2019)
12. Frantar, E., Alistarh, D.: SparseGPT: Massive language models can be accurately pruned in one-shot. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 10323–10337. PMLR (2023)
13. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT Press (2016)
14. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding (2016)
15. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems. vol. 28 (2015)
16. Hassibi, B., Stork, D.: Second order derivatives for network pruning: Optimal brain surgeon. In: Hanson, S., Cowan, J., Giles, C. (eds.) Advances in Neural Information Processing Systems. vol. 5. Morgan-Kaufmann (1992)

17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)
18. He, Y., Liu, P., Wang, Z., Hu, Z., Yang, Y.: Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
19. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. CoRR **abs/1503.02531** (2015)
20. Idelbayev, Y., Carreira-Perpinan, M.A.: Low-rank compression of neural nets: Learning the rank of each layer. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8046–8056 (2020)
21. Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L.R., Lachaux, M.A., Stock, P., Scao, T.L., Lavril, T., Wang, T., Lacroix, T., Sayed, W.E.: Mistral 7b (2023)
22. Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 (canadian institute for advanced research)
23. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012)
24. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
25. LeCun, Y., Denker, J., Solla, S.: Optimal brain damage. In: Touretzky, D. (ed.) Advances in Neural Information Processing Systems. vol. 2. Morgan-Kaufmann (1989)
26. Lee, J., Park, S., Mo, S., Ahn, S., Shin, J.: Layer-adaptive sparsity for the magnitude-based pruning. In: International Conference on Learning Representations (2021)
27. Lee, N., Ajanthan, T., Torr, P.H.S.: Snip: Single-shot network pruning based on connection sensitivity (2019)
28. Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the loss landscape of neural nets. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018)
29. Li, Y., Adamczewski, K., Li, W., Gu, S., Timofte, R., Gool, L.V.: Revisiting random channel pruning for neural network compression (2022)
30. Liu, S., Chen, T., Chen, X., Atashgahi, Z., Yin, L., Kou, H., Shen, L., Pechenizkiy, M., Wang, Z., Mocanu, D.C.: Sparse training via boosting pruning plasticity with neuroregeneration. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. NIPS '21, Curran Associates Inc., Red Hook, NY, USA (2024)
31. Liu, S., Chen, T., Chen, X., Shen, L., Mocanu, D.C., Wang, Z., Pechenizkiy, M.: The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. In: International Conference on Learning Representations (2022)
32. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 2755–2763 (2017)
33. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics **19**(2), 313–330 (1993)

34. Merity, S., Xiong, C., Bradbury, J., Socher, R.: Pointer sentinel mixture models (2016)
35. Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.M., Rothchild, D., Yinan, D., Diker, E., Corrado, G.: Carbon emissions and large neural network training. arXiv preprint arXiv:2104.10350 (2021)
36. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017)
37. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
38. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI (2019)
39. Raiaan, M.A.K., Mukta, M.S.H., Fatema, K., Fahad, N.M., Sakib, S., Mim, M.M.J., Ahmad, J., Ali, M.E., Azam, S.: A review on large language models: Architectures, applications, taxonomies, open issues and challenges. IEEE access **12**, 26839–26874 (2024)
40. Sabih, M., Hannig, F., Teich, J.: Utilizing explainable ai for quantization and pruning of deep neural networks. CoRR **abs/2008.09072** (2020)
41. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
42. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. ArXiv **abs/1910.01108** (2019)
43. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)
44. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in nlp. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 3645–3650 (2019)
45. Sun, M., Liu, Z., Bair, A., Kolter, J.Z.: A simple and effective pruning approach for large language models. In: The Twelfth International Conference on Learning Representations (2024)
46. Thompson, N.C., Greenewald, K., Lee, K., Manso, G.F.: Deep learning's diminishing returns. IEEE Spectrum **59**(10), 34–39 (2022)
47. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and efficient foundation language models (2023)
48. Wang, C., Grosse, R.B., Fidler, S., Zhang, G.: Eigendamage: Structured pruning in the kronecker-factored eigenbasis. ArXiv **abs/1905.05934** (2019)
49. Wu, C.J., Brooks, D., Chen, K., Chen, D., Choudhury, S., Dukhan, M., Hazelwood, K., Isaac, E., Jia, Y., Jia, B., et al.: Sustainable ai: Environmental implications, challenges and opportunities. In: Proceedings of Machine Learning and Systems. vol. 4, pp. 795–813 (2022)
50. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1912–1920 (2015)
51. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)