ModHiFi: Identifying High Fidelity predictive components for Model Modification

Dhruva Kashyap
CSA, IISc
kdhruva@iisc.ac.in

Chaitanya Murti
HP Inc. AI Lab
chaitanya.murti@hp.com

Pranav Nayak
CSA, IISc
pranavk@iisc.ac.in

Tanay Narshana *
Google
tanaynarshana@google.com

Chiranjib Bhattacharyya
CSA, IISc
chiru@iisc.ac.in

Abstract

Modifying well-trained models for purposes such as pruning or unlearning, without access to training data or the original loss function, is a challenging problem. While techniques exist for such modification, they often require training data, are computationally expensive, or are architecture-specific. To address this, we investigate the fundamental question of identifying components that are critical to the model's predictive performance, without access to either gradients or the loss function, and with only distributional access such as synthetic data. We theoretically demonstrate that the global reconstruction error is linearly bounded by local reconstruction errors for Lipschitz-continuous networks such as CNNs and well-trained Transformers (which, contrary to existing literature, we find exhibit Lipschitz continuity). This motivates using the locally reconstructive behavior of component subsets to quantify their global importance, via a metric that we term *Subset Fidelity*. In the uncorrelated features setting, selecting individual components via their Subset Fidelity scores is optimal, which we use to propose **ModHiFi**, an algorithm for model modification that requires no training data or loss function access. **ModHiFi-P**, for structured pruning, achieves an 11% speedup over the current state of the art on ImageNet models and competitive performance on language models. **ModHiFi-U**, for classwise unlearning, achieves complete unlearning on CIFAR-10 without fine-tuning and demonstrates competitive performance on Swin Transformers.[2]

1 Introduction

Modern deep learning has made significant strides in a wide variety of tasks, such as classification [39, 40], image generation [23], and natural language processing [54]; moreover, *well-trained* models for such tasks are easily accessible. However, significant challenges remain in their deployment, such as inference in resource-constrained settings [65, 75], inference with unbalanced or biased data [30, 31], and interpretable inference [101]. These challenges have increased interest in methods that modify the parameters of well-trained models to alter their behaviour [67, 69, 71]. These methods include pruning [29], classwise unlearning [34, 38, 71], and debiasing [55, 71], among other model modifications. Moreover, recent work has studied model modification in the setting where the original training data and loss function are unavailable [58]; this is motivated by concerns related to privacy and security [96], and also the use of *synthetic data*, which has become critical in a variety of language modeling settings [11, 80]. Thus, we address the challenging problem of *altering well-trained models*

*Author primarily contributed to this work before joining Google.
[2]Our code is available at https://github.com/DhruvaKashyap/modhifi

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

without training data or the loss function, and only with distributional access to the original training distribution in the form of synthetic data, focusing specifically on structured pruning and classwise unlearning.

Modifying well-trained models without the loss function and only synthetic data requires answering a fundamental question: *which components in a model contribute significantly to its predictive performance*[3]*?* However, most methods that identify critical components for specific modifications (e.g., pruning) cannot be applied to others (e.g., unlearning) [58], often require expensive fine-tuning, and are architecture-specific. Moreover, most methods utilize gradients to assess the impact of a component on the loss objective, which is not feasible in the absence of the loss function and the training data. While the LLM pruning literature uses calibration datasets to mitigate the problem of the absence of datasets [2, 51], the problem of achieving sparsity in vision models without original training data is hard and unsolved [29]. Moreover, the problem of performing classwise unlearning without original training data is unattempted [35, 58].

Towards enabling the modification of well-trained models amidst these challenges, we make the following contributions:

(C1) **Local-to-Global with Lipschitzness**. An open question is the extent to which local model modifications impact the predictive performance of the model. In the absence of loss functions and training sets, estimating the impact of component modification by using gradients (as done in [35, 48, 51]) is infeasible. To address this, in Theorem 3.6, we show that for Lipschitz continuous networks, the reconstruction error at the final layer is at most linear in the local reconstruction errors. Moreover, contrary to the assertion that transformers are not Lipschitz continuous [66], in Corollary B.4, we show that this is not the case for *well-trained* transformers, allowing us to apply Theorem 3.6 to not just CNNs, but well-trained ViTs and LLMs as well.

(C2) **Identifying Subsets of Important Components**. Contrary to prior work that usually infer saliencies for single components, we propose measuring the importance of sets of components for understanding cumulative effects of groups of components on a model's predictive performance. Leveraging Theorem 3.6, we propose *Subset Fidelity*, which quantifies the extent to which a subset of components can reconstruct the output after modifying their weights. However, computing optimal subsets is NP-complete, motivating us to compute Subset Fidelity scores for singleton sets. Theorem 3.9 establishes that selecting singletons with the highest subset fidelity scores is optimal when the features are uncorrelated.

(C3) **Modifying Models with ModHiFi-X.** Motivated by Theorem 3.9, we propose the **ModHiFi** algorithm, which uses the subset fidelity of singletons to modify models for pruning and classwise unlearning; the algorithm identifies important components using the singleton scores, and removes them (for classwise unlearning, **ModHiFi-U**) or retains them (for structured pruning, **ModHiFi-P**). We show that ModHiFi-P achieves state-of-the-art speedup for Imagenet models, and is consistently competitive with current baselines for language models. For classwise unlearning, ModHiFi-U achieves complete unlearning on all CIFAR-10 classes without fine-tuning, and is competitive with baselines on Swin-Transformers that finetune. When allowing for a similar fine-tuning budget as said baselines, ModHiFi-U outperforms, particularly when given access to training data. These empirical results demonstrate the practical effectiveness of Subset Fidelity.

2 Background, Setup, and Related Work

In this section, we describe background relevant to this manuscript, formally set up our problem statement, and position our work to related prior work in literature.

2.1 Background and Notation

Notation For $p \in \mathbb{N}$, let $[p]$ denote $\{1, \ldots, p\}$. Let $\boldsymbol{v} \in \mathbb{R}^n$ denote an n-dimensional vector with entries v_i for $i \in [n]$. Let $\boldsymbol{B} \in \mathbb{R}^{n \times m}$ denote a matrix with rows $\boldsymbol{b}_i^\top \in \mathbb{R}^m$ for $i \in [n]$. The vectors $\mathbf{1}_d$ and $\mathbf{0}_d$ denote the all-ones and all-zeros vectors in $\mathbb{R}^d$, respectively. For a vector $\boldsymbol{v}$,

[3]We measure predictive performance using accuracy for classification tasks, and perplexity and other measures for language modeling tasks [58, 71].

$\text{diag}(\boldsymbol{v})$ denotes the diagonal matrix with diagonal entries v_i. Let $\|\boldsymbol{v}\|_2 = \sqrt{\boldsymbol{v}^\top \boldsymbol{v}}$ and $|\boldsymbol{v}|_0$ denote the number of nonzero entries of $\boldsymbol{v}$. Given matrices $\boldsymbol{C}, \boldsymbol{D} \in \mathbb{R}^{n \times m}$, we define the inner product $\langle \boldsymbol{C}, \boldsymbol{D} \rangle = \text{Tr}(\boldsymbol{C}^\top \boldsymbol{D})$,, the Frobenius norm as $\|\boldsymbol{C}\|_F = \sqrt{\langle \boldsymbol{C}, \boldsymbol{C} \rangle}$, and the spectral norm $\|\boldsymbol{C}\|_2$ is the largest singular value of $\boldsymbol{C}$. The submatrix of a matrix $\boldsymbol{C}$ with of set of rows $A \subset [n]$ and columns $B \subset [m]$ is denoted by $\boldsymbol{C}[A, B]$. We denote the expectation of the random variable X with $\mathbb{E}_X[X]$ and we drop the subscript when it is clear from context.

2D Convolution Let the l^{th} layer of the network be a 2D convolutional layer with c^l_{in} input channels, c^l_{out} output channels, and k^l kernel size be parameterized by weights $\mathbf{W}^l \in \mathbb{R}^{c^l_{\text{out}} \times c^l_{\text{in}} \times k^l \times k^l}$. The input to this layer is $\Phi^l(\boldsymbol{X}) \in \mathbb{R}^{c^l_{\text{in}} \times h^{l-1} \times w^{l-1}}$, and the output is $\mathbf{Y}^l(\boldsymbol{X}) \in \mathbb{R}^{c^l_{\text{out}} \times h^l \times w^l}$, where h^{l-1}, w^{l-1} and h^l, w^l denote the input and output spatial dimensions, respectively. Each output channel $c \in [c^l_{\text{out}}]$ is computed as

$$\boldsymbol{Y}^l_c(\boldsymbol{X}) = \sum_{i=1}^{c^l_{\text{in}}} \boldsymbol{\Phi}^l_i(\boldsymbol{X}) \star \boldsymbol{W}^l_{ci} = \sum_{i=1}^{c^l_{\text{in}}} \boldsymbol{A}^l_{ci}(\boldsymbol{X}) \tag{CONV}$$

where $\star$ denotes the 2D convolution operator. We define $\boldsymbol{A}^l_{ci}(\boldsymbol{X}) \coloneqq \boldsymbol{\Phi}^l_i(\boldsymbol{X}) \star \boldsymbol{W}^l_{ci} \in \mathbb{R}^{h^l \times w^l}$ as the *input contribution* from input channel i to output channel c. For simplicity, we omit the bias term.

Transformers Transformer models consist of transformer blocks connected by residual connections, with intermediate activations normalized using LayerNorm or RMSNorm [2, 3]. Each transformer block consists of a Multi-head Attention (MHA) module, and a Feed Forward Network (FFN) module consisting of two feedforward layers with weights W_U and W_D. Formally the FFN block operates on input $\boldsymbol{X}$ to the block as $\text{FFN}(\boldsymbol{X}) = \sigma(\boldsymbol{X}W^U)W^D$. Here $\sigma(\cdot)$ is an elementwise nonlinearity. In this work, we consider modifications of the Feed-Forward Network (FFN) block. We define *input contributions* for FFN blocks of transformers as follows:

$$\boldsymbol{A}_{ci}(\boldsymbol{X}) := \boldsymbol{\Phi}_i(\boldsymbol{X})\boldsymbol{W}^D_{ci} \quad \text{where} \quad \boldsymbol{\Phi}_i(\boldsymbol{X}) := (\sigma(\boldsymbol{X}W^U))_i \tag{LIN}$$

Unified Notation We describe a unified notation encompassing both CNN and transformer architectures which we use to simplify exposition in the rest of this work. Let $\text{N}_\theta : \mathbb{R}^{c_{\text{in}} \times d} \to \mathbb{R}^{c_{\text{out}}}$ denote a network parameterized by θ, mapping an input matrix to an output vector. The input $\boldsymbol{X} \sim \mathcal{D}$ is sampled from a distribution $\mathcal{D}$, and the output is $\text{N}_\theta(\boldsymbol{X})$ for some input sample $\boldsymbol{X}$. The network is a composition of L layers, $\text{N}_\theta = f^L \circ f^{L-1} \circ \cdots \circ f^1$, where each f^l has parameters $\mathcal{W}^l$. For layer l, let c^l_{in}, c^l_{out} be the input and output channel counts, and d^l_1 and d^l_2 represent the spatial (or sequence) dimension of layer l. The layer input and output are $\boldsymbol{\Phi}^l(\boldsymbol{X}) \in \mathbb{R}^{c^l_{\text{in}} \times d^l_1 \times d^l_2}$ and $\boldsymbol{Y}^l(\boldsymbol{X}) \in \mathbb{R}^{c^l_{\text{out}} \times d^{l+1}_1 \times d^{l+1}_2}$. Each output channel c is computed as $\boldsymbol{Y}^l_c(\boldsymbol{X}) = \sum_{i=1}^{c^l_{\text{in}}} \boldsymbol{A}^l_{ci}(\boldsymbol{X})$, where $\boldsymbol{A}^l_{ci}(\boldsymbol{X})$ is computed using $\mathcal{W}^l$ according to either Eq. (CONV) or Eq. (LIN) depending on the type of the model.

2.2 Modifying Models without Training Data or the Loss Function

We call methods that selectively modify the parameters of a trained model to alter its behavior, typically without re-training from scratch [35, 69, 71], *model modification*. These techniques are motivated by concerns around efficiency, privacy, and regulatory compliance (e.g., GDPR) [7, 29, 62]. A broad class of tasks are fundamentally model modification tasks, such as debiasing [34], selective unlearning [21], network scrubbing [41], and continual or lifelong learning [22, 67].

Modifying models in real-world settings is often challenging, especially when the original loss function or training data is unavailable [29, 71]. Recent works such as [58] have suggested relaxing the problem to that of modifying models with *distributional access*, wherein access to the underlying data distribution is available in the form of a few natural samples. However, synthetic data is a more prevalent form of distributional access which is widely used in a variety of downstream LLM-related modification or fine-tuning tasks [11, 80], particularly since original training corpora are rarely available for use. and curating natural samples is costly. Moreover, the addition of synthetic data to training data for use in compression pipelines has gained importance [32, 102] when compressing LLMs. Thus, in this work, we address the following question: *Can we effectively modify trained models, for structured pruning or unlearning, only with distributional access in the form of synthetic data?*

Formalizing Model Modification We aim to modify well-trained models by selectively modifying components that are relevant to a particular modification task. We use the term *well-trained model* to be a model that has been trained to accurately perform a specific task (e.g. classification, generation, etc). Each modification task can be characterized by a set of data distributions and associated weights $\{(\mathcal{D}_i, \alpha_i)\}$, as well as a family of allowable masks $\mathcal{M}$ applied to the learned parameters. Despite the term "mask," $\mathcal{M}$ need not consist solely of binary-valued elements.

Formally, given parameters $\theta^\star \in \mathbb{R}^D$ of a well-trained model, the modified model $\theta^E \in \mathbb{R}^D$ is:

$$\theta^E = \theta^\star \odot \mathsf{m}^\star, \quad \text{where} \quad \mathsf{m}^\star = \arg\min_{\mathsf{m} \in \mathcal{M}} \sum_i \alpha_i \, \mathbb{E}_{\mathcal{D}_i} \left[\mathcal{L}(\mathrm{N}_{\theta^\star \odot \mathsf{m}}(\mathrm{X})) \right], \tag{MODIFY}$$

where $\odot$ denotes the element-wise product. While a range of tasks may be formulated as problems involving model modification [71], this work focuses on two cases: **structured pruning** and **classwise unlearning**.

In *structured pruning*, we seek masks that retain at most B structured components (e.g., channels, neurons, layers). Structured pruning can be posed under the general formulation of Eq. (MODIFY) as:

$$\mathsf{m}^\star = \arg\min_{\mathsf{m} \in \mathcal{M}} \mathbb{E}_{\mathcal{D}} \left[\mathcal{L}(\mathrm{N}_{\theta^\star \odot \mathsf{m}}(\mathrm{X})) \right]$$

where $\mathcal{M}$ consists of elements with at most B non zero structured components.

In *classwise unlearning*, the objective is to reduce the model's performance on a target *forget class*, sampled from distribution $\mathcal{D}_f$, while preserving accuracy on the *remain classes* drawn from $\mathcal{D}_r$. Classwise unlearning may be posed under the general formulation of Eq. (MODIFY) as:

$$\mathsf{m}^\star = \arg\min_{\mathsf{m} \in \mathbb{R}^D} \mathbb{E}_{\mathcal{D}_r} \left[\mathcal{L}(\mathrm{N}_{\theta^\star \odot \mathsf{m}}(\mathrm{X})) \right] - \mathbb{E}_{\mathcal{D}_f} \left[\mathcal{L}(\mathrm{N}_{\theta^\star \odot \mathsf{m}}(\mathrm{X})) \right]$$

Objective We address the problem of solving Eq. (MODIFY) - modifying well-trained models through masking - *without access to the training loss or data used to obtain* $\theta^\star$.

2.3 Related Work

We position our work against relevant works modifying well-trained models for classification (CNNs & ViTs) and language generation (LLMs). A comprehensive related work is presented in Appendix A.

Modifying Vision Classification Models. Structured pruning algorithms have been developed for both kinds of models we perform experimental validation on - CNNs ([29] and the references therein) and ViTs [15, 97, 99]. Similarly, classwise unlearning has been explored in both CNNs and ViTs [12, 35, 38], and a few works address pruning and classwise unlearning jointly [58]. Also, performing classwise unlearning without training data is unattempted [35, 58].

Modifying Large Language Models. Modifying LLMs presents its own set of challenges, owing to the size of both models and datasets [9, 36]. A variety of structured pruning approaches have been proposed to address the challenge of efficient inference in LLMs [2, 51, 52, 74, 77, 92]. However, no contemporary works address the problem of modifying LLMs and other models jointly. We compare performance of our work on LLMs to other structured pruning methods for LLMs in addition to additional modification tasks.

3 Which Components Are Important for Modifying Well-Trained Models?

In this section, we address the problem of identifying components that strongly contribute to a model's predictive performance. To this end, we introduce *High-Fidelity (HiFi)* components and provide both theoretical and empirical evidence that these components are crucial to maintaining model accuracy.

3.1 High-Fidelity Components and the Subset Fidelity Score

Our objective is to estimate how removing a subset of input contributions affects the model's output, after optimally compensating for this removal. Directly quantifying this effect is difficult, so we introduce the *Subset Fidelity*, a measure of how well a subset of components can locally approximate the layer's output.

Definition 3.1 (Subset Fidelity)**.** The *fidelity* of a subset of components $C \subseteq [c_{in}^l]$ in layer l for output channel c is defined as

$$\mathrm{FS}_c^l(C) = \max_{\delta_c^l} \left(1 - \frac{d\big(\boldsymbol{Y}_c^l(\mathrm{X}), \sum_{i \in C} \delta_{ci}^l \boldsymbol{A}_{ci}^l(\mathrm{X})\big)}{d(\boldsymbol{Y}_c^l(\mathrm{X}), \mathbf{0})}\right) \tag{1}$$

where δ_c^l is a *compensation term* that scales only the components in C and $d(f(\mathrm{X}), g(\mathrm{X}))$ denotes $\mathbb{E}[\|f(\mathrm{X}) - g(\mathrm{X})\|_2^2]$ for appropriate functions f, g.

Eq. (1) generalizes the formulation of Halabi et al. [26]. We motivate this definition through the following properties.

Lemma 3.2 (Properties of Subset Fidelity)**.** *For any subset $C \subseteq [c_{in}^l]$ in layer l, the following hold: (1) $0 \leq \mathrm{FS}_c^l(C) \leq 1$, (2) if $D \subset C$, then $\mathrm{FS}_c^l(D) \leq \mathrm{FS}_c^l(C)$.*

The proof is provided in Appendix B.2. A larger Subset Fidelity indicates that the subset more effectively reconstructs the output. The second property shows that subset fidelities are monotone. Lemma 3.2 implies two key insights: (1) fidelity serves as a principled measure of component importance, and (2) monotonicity suggests that greedy selection strategies may be effective.

Remark 3.3. In this work, we focus only on the case where the subset fidelities are measured with the expected squared difference. We leave to future work an exploration of other possible measures of distributional similarity.

We now use Definition 3.1 to define subsets of components with high fidelity.

Definition 3.4 ((k,η)-HiFi Set)**.** Given a target subset size k and a threshold $\eta \in (0,1)$, the (k,η)-HiFi Set $S_c^{k,\eta}$ for output channel c is any subset in $[c_{in}^l]$ satisfying

$$\mathrm{FS}_c^l(S_c^{k,\eta}) \geq \eta, \quad |S_c^{k,\eta}| \leq k. \tag{HiFi}$$

Thus, attributing predictive performance to components reduces to finding the HiFi set for a given (k,η). We can reduce the identification of HiFi sets to solving an optimization problem, the solution of which yields the *Maximum Fidelity Subset*, which contains the components that best recover the layer's output.

Definition 3.5 (k-Maximum Fidelity Subset)**.** Given a target subset size k for layer l, the *Maximum Fidelity Subset* $S_c^{l\star}$ for channel c is defined as

$$S_c^{l\star} = \underset{S \subseteq [c_{in}^l],\, |S|=k}{\arg\max} \; \mathrm{FS}_c^l(S). \tag{K-MFS}$$

A simple algorithm for identifying a (k,η)-HiFi set is to solve Eq. (K-MFS) for the given k and check whether its fidelity exceeds η. If it does not, no such (k,η)-HiFi set exists.

3.2 Local Distributional Measures of Component Importance

Finding HiFi subsets corresponds to finding subsets that minimize the l_2 reconstruction error while accounting for weight compensation. Additionally, it enables the derivation of a closed-form expression for weight compensation, allowing for accuracy recovery without requiring fine-tuning.

Bounding Global Error via Local Modification We now show that the influence of a component on its immediate layer output provides a tractable proxy for its overall effect on model predictions. A proof is provided in Appendix B.1.

Theorem 3.6 (Local to Global)**.** *Consider a network as described in Section 2.1. Let all the parameters $\mathcal{W}^\ell \neq 0 \; \forall \ell \in [L]$. Let $\boldsymbol{M}^l$ represent a mask which modifies the parameters in layer l. If there exists a positive scalar r^ℓ for each layer $\ell > l$ such that $\|\boldsymbol{\Phi}_c^\ell(\mathrm{X})\|_F \geq r^\ell \;\; \forall c \in [c_{out}^\ell]$ almost surely (over the distribution of X), then there exist constants $\{C_\ell\}_{\ell=1}^L$ (dependent only on the architecture) such that:*

$$\mathbb{E}\big[\|\mathrm{N}_\theta(\mathrm{X}) - \mathrm{N}_{\theta \odot \boldsymbol{M}^l}(\mathrm{X})\|^2\big] \leq C_l^2 \sum_{c=1}^{c_{out}^l} \Big((\mathbf{1}_{c_{in}^l} - \boldsymbol{m}_c^l)^\top \boldsymbol{Q}_c^l (\mathbf{1}_{c_{in}^l} - \boldsymbol{m}_c^l)\Big) \tag{2}$$

where $\boldsymbol{Q}_c^l \in \mathbb{R}^{c_{in}^l \times c_{in}^l}$ is the component similarity matrix (CSM) *for channel c, with entries $(\boldsymbol{Q}_c^l)_{ij} = \mathbb{E}[\langle \boldsymbol{A}_{ci}^l(\mathrm{X}), \boldsymbol{A}_{cj}^l(\mathrm{X})\rangle]$.*

Remark 3.7. Theorem 3.6 implies that global error grows at most linearly with local error, making local fidelity a practical, architecture-agnostic proxy for component influence. The constant C_l quantifies the amplification of local errors through subsequent layers and activations, and is independent of the data distribution. The theorem also necessitates that networks discussed in this work under suitable conditions are Lipschitz continuous. While CNNs are Lipschitz continuous [98], transformers are not [66]. In Corollary B.4, we show that this is not the case for *well-trained* transformers. Empirical estimates of C_l reported in Appendix C.2 demonstrate the practicality of these constants.

Subset Fidelity for l_2 reconstruction error Next, we show that when the distributional similarity measure is the l_2 reconstruction error, both the compensation term and the singleton fidelity scores admit closed-form expressions, thus motivating their use in this work.

Proposition 3.8 (Compensation and Singleton Fidelity)**.** *For the l_2 reconstruction error, the optimal compensation term $\delta_c^\star$, which is the value at which the fidelity score is computed according to Eq. (1) for a subset C is given by,*

$$\delta_{ci}^{l\star}(C) = \begin{cases} 1 + ((\boldsymbol{Q}_c^l[C,C])^{-1})_i^\top \boldsymbol{Q}_c^l[C,\overline{C}]\mathbf{1}_{n-k} & \text{if } i \in C \\ 0 & \text{if } i \notin C \end{cases} \tag{FS}$$

where $\boldsymbol{Q}_c^l$ is the component similarity matrix as in Eq. (2). The singleton fidelity scores are:

$$s_{ci}^l = \mathrm{FS}_c^l(\{i\}) = 1 - \frac{\mathbb{E}[\|\boldsymbol{Y}_c^l(\mathrm{X}) - \alpha_{ci}^l \boldsymbol{A}_{ci}^l(\mathrm{X})\|^2]}{\mathbb{E}[\|\boldsymbol{Y}_c^l(\mathrm{X})\|^2]}, \quad \alpha_{ci}^l = \frac{\mathbb{E}[\langle \boldsymbol{Y}_c^l(\mathrm{X}), \boldsymbol{A}_{ci}^l(\mathrm{X})\rangle]}{\mathbb{E}[\|\boldsymbol{A}_{ci}^l(\mathrm{X})\|^2]}. \tag{3}$$

The proof is provided in Appendix B.3.

Note that solving Eq. (K-MFS) exactly is still equivalent to a constrained binary quadratic optimization problem, known to be NP-hard [1]. Viewing $\boldsymbol{Q}^c$ as the adjacency matrix of a weighted graph, maximizing Eq. (K-MFS) corresponds to identifying a clique of size k, the decision version of the MAXIMUM CLIQUE problem. Intuitively, such cliques correspond to groups of components whose joint removal maximally increases the reconstruction error. However, since fidelity is monotonic, a natural heuristic selects the k components with the highest singleton fidelities s_{ci}^l; we call this strategy: `NAIVE`.

Optimal Identification of HiFi Sets We identify conditions under which the `NAIVE` selection strategy is optimal.

Theorem 3.9. *Consider output channel c in the l^{th} layer of a network described in Section 2.1. Let the s_{ci}^l be defined according to Eq. (3). Let*

$$\hat{S}_c^l = \{i \mid s_{ci}^l \geq s_{(k)}\}$$

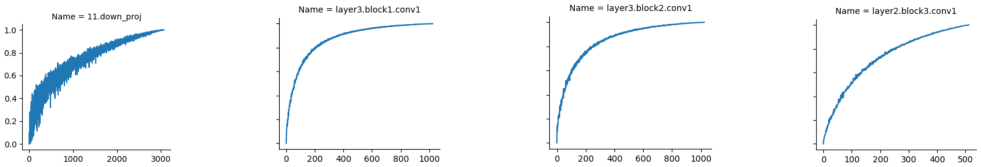
where $s_{(k)}$ is the k largest value of $\boldsymbol{s}_c^l$. Assuming that that there are no ties, $|\hat{S}_c^l| = k$. If $\mathbb{E}[\langle \boldsymbol{A}_{ci}^l(\mathrm{X}), \boldsymbol{A}_{cj}^l(\mathrm{X})\rangle] = 0 \ \ \forall i \neq j$, then

$$\hat{S}_c^l = \underset{S \subseteq [c_{in}^l],\, |S|=k}{\arg\max} \ \mathrm{FS}_c^l(S)$$

The proof is provided in Appendix B.4.

Remark 3.10. Theorem 3.9 connects a statistical property of the representations to the efficient discovery of HiFi components. It states that when the input contributions are pairwise uncorrelated, the optimal subset is the set of components with the highest fidelity score. Although the assumption of uncorrelated features rarely holds exactly in practice, it offers a useful theoretical justification for `NAIVE` HiFi selection. We demonstrate the practical effectiveness of `NAIVE` HiFi selection through our experiments in Section 5.

Existence of small HiFi components Our experiments in Section 5.2 empirically establish the existence of a small subset of components that can achieve high fidelity. Fig. 1 indicates a sample of the results indicating that fewer than 20% of components can achieve high fidelity (≥ 0.8). Moreover, in Section 5.3, we validate the effectiveness of HiFi components with the model's predictive performance.



(a) OPT-125M (WikiText) (b) ResNet50 (CIFAR-10) (c) ResNet50 (CIFAR-100) (d) ResNet50 (ImageNet)

Figure 1: Monte Carlo estimation of Eq. (K-MFS) across selected layers of various models. The x-axis indicates subset size k, and the y-axis the maximum subset fidelity across random samples.

4 Modifying Model Behavior using HiFi Sets

In this section, we describe how the HiFi framework can be used to edit model behavior via structured pruning and class unlearning. The central idea is to identify high-fidelity (HiFi) components and then modify them in a targeted manner using a unified algorithmic procedure, Algorithm 1. Additional details including complexity and implementation specifics are provided in Appendix D.

Structured Pruning In structured pruning, the objective is to remove entire input channels (or features) that contribute minimally to the model's predictive performance. In convolutional architectures, we identify and remove input channels across all layers that do not appear in the HiFi sets of any output channel of the residual-coupled layers. For CNNs, pruning is applied to the input channels of convolutional layers. For LLMs, we target the input features of the MLP down-projection matrices (W^D). After pruning, the compensation term defined in Definition 3.1 is computed using the remaining weights to restore fidelity.

Algorithm 1 ModHiFi-X

Require: Model parameters θ, layer l, k components, threshold η, data $\mathcal{D}$
Ensure: Modified parameters θ^E
Identify (k, η)-HiFi Set via Eq. (3) using $\mathcal{D}$
if $X =$ Prune **then**
  for $i \in [c^l_{in}] \setminus \{i \mid (c, i) \in H_l\}$ **do**
    $\boldsymbol{W}^l_{c,i} \leftarrow \mathbf{0} \quad \forall c \in [c^l_{out}]$
else if $X =$ Unlearn **then**
  for $(c, i) \in H_l$ **do**
    $\boldsymbol{W}^l_{c,i} \leftarrow \mathbf{0}$
return $\hat{\theta}$

Class Unlearning To perform unlearning, we first compute HiFi sets using only samples from the class we wish to forget. The components in these sets are then zeroed out, effectively erasing the influence of that class. This causes the model's predictive performance on the forgotten class to degrade without significantly impacting other classes.

Fidelity Estimation For vision models, the singleton fidelity score $\mathrm{FS}^l_c(\cdot)$ can be estimated efficiently using distributional access to the input data, i.e. synthetic samples. In practice, we compute scalar values α^l_{ci} which correlate with the importance of component i to output channel c. A large α^l_{ci} indicates a high-fidelity component. For LLMs, we develop a tractable Cholesky-based heuristic to estimate the score and provide details in Appendix D.2.

5 Experiments

In this section, we present experimental validation of our work to answer the following questions.

(Q1) **Existence of HiFi components**. Do a small subset of components exist that can achieve high fidelity?

(Q2) **Effectiveness of HiFi components**. Do HiFi components accurately represent those components important for the predictive performance?

(Q3) **Effectiveness of using HiFi components for pruning using ModHiFi-P**. Does ModHiFi-P result in better accuracy-sparsity tradeoff compared to structured pruning algorithms for vision tasks and language modeling tasks?

(Q4) **Effectiveness of using HiFi components for machine unlearning using ModHiFi-U**. Is it possible to perform machine unlearning, as posed by Jia et al. [35], without fine-tuning? If yes, how does ModHiFi-U fare against their method?

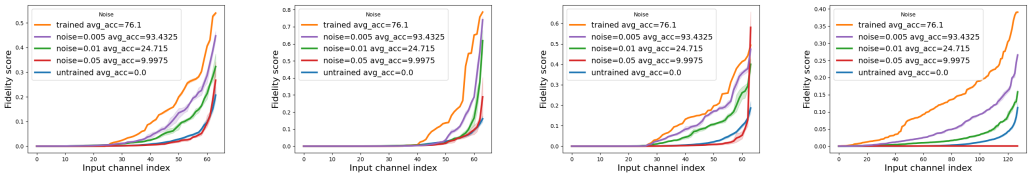


Figure 2: Fidelity score of selected layers of a ResNet-50 model on CIFAR10 and the effect of noise on the fidelity score.

5.1 Details of the experimental setup

Models, Datasets, and Evaluation We conduct experiments on ResNet-50/101 [27], VGG19 [76], Swin-Transformer [49] and Llama-2-7B [84], benchmarking against relevant experiments from related literature [2, 51]. For vision tasks, we measure the classification accuracy, and for NLP tasks, we use EleutherAI's `lm-eval-harness` [20].

Distributional Access For CIFAR10/100 [39], we use synthetically generated images as detailed in Appendix C.3. We use Alpaca [82] (a synthetic dataset) and WikiText-2 [53] as calibration data for NLP tasks following related literature [2, 51]. We provide **ablations** to measure the impact of synthetic data quality in Appendix C.3.3.

Compute platform and implementation details We discuss the compute platform, implementation details and hyperparameters used for our experiments in Appendix C.6.

5.2 Existence of HiFi components: Exploring (Q1)

To empirically assess whether small subsets can achieve high fidelity, we estimate $S_c^\star$ by sampling random subsets of size k across different architectures and selecting the subset with the highest fidelity. Detailed results are presented in Appendix C.1.

Observation 1. *Across all evaluated models, each layer typically contains a small subset of input channels (fewer than 20%) that achieves high subset fidelity* (≥ 0.8).

This empirical observation indicates that in trained models, there are only a small set of components in each layer that are responsible for the model prediction. This observation aligns with the success of structured pruning algorithms in constructing small subnetworks with high statistical performance.

5.3 Effectiveness of HiFi components: Exploring (Q2)

To answer (Q2), and verify whether HiFi components are the components that matter for the final predictive performance, we measure the effects of the fidelity of a component getting destroyed by noising. For a ResNet-50 on CIFAR-10, when 20% of the HiFi components are perturbed with a zero mean Gaussian noise with standard deviation of 0.01, the accuracy of the model drops by around 12%, whereas perturbing 80% of the non-HiFi components identically results in an accuracy drop of only 1%. At 50% of components with a noise of standard deviation 0.02, the accuracy drops by 85% when HiFi components are noise compared to only around 1.4% when non-HiFi components are noised. In Appendix C.1.3, we make similar observations across various models and tasks. In Appendix C.1.2, we additional experiments where we compare the *removal* of HiFi, non HiFi, and random sets of the same size and make similar observations.

5.4 Structured Pruning Experiments: (Q3)

5.4.1 Vision Models

Baselines We compare against the state of the art structured pruning algorithms specialized for pruning vision models [8, 50, 61, 87], and present additional results on other architectures and datasets in Appendix C.4 where we make similar observations. Following [18], we update the batch norm statistics using the data from distributional access.

Table 1: Comparison of pruning methods on ResNet50 evaluated on ImageNet.

Algorithm	Accuracy	FLOP Reduction	Param Reduction	CPU Speedup	GPU Speedup
Unpruned	76.1	1x	1x	1x	1x
GReg-2 [87]	73.9	3.02x	2.31x	1.36x	1.53x
OTO [8]	74.7	2.86x	2.81x	1.25x	1.45x
DepGRAPH [14]	75.83	2.07x	-	-	-
ThiNet [50]	71.6	3.46x	2.95x	1.38x	1.50x
DFPC (30) [61]	75.9	1.98x	1.84x	1.42x	1.53x
DFPC (54) [61]	73.80	3.46x	2.65x	2.37x	**2.38x**
Ours	**76.70**	2.17x	1.47x	1.69x	1.70x
Ours	73.82	**3.66x**	**3.05x**	**2.42x**	**2.38x**

Table 2: Comparison of pruning methods on ResNet50 with CIFAR10 *(ST: Synthetic Tuning)*.

Algorithm	Accuracy	FLOP Reduction	Param Reduction
Unpruned	94.99	1x	1x
DFPC [61]	90.25	1.46x	2.07x
L_2 [44]	15.91	**4.07x**	4.71x
L_2 w/ ST [44]	90.12	**4.07x**	4.71x
Ours	**91.02**	**4.07x**	**5.36x**

Observations We find that our method results in better accuracy-vs-sparsity trade off when compared to other algorithms across datasets. We also *train* a model obtained with L_2 norm-based structured pruning *using the synthetic set* based on CIFAR10 for comparison. We observe that for the *same FLOP sparsity*, our method obtains higher accuracy than the model finetuned on synthetic samples, indicating that our method is able to *outperform fine-tuning in some cases* using synthetic samples for the same sparsity. For the ImageNet dataset, we compare against various state-of-the-art structured pruning algorithms for networks with complex interconnections with training. In the training regime, we observe that for models of similar accuracy, our algorithm obtains the best accuracy-speedup tradeoff with fewer epochs of finetuning. Details of pre-trained networks and post-training are given in Appendix C.7.2. Our study of the effect of the quality of synthetic samples on our algorithm in Appendix C.3.3 indicates that the sparsity-accuracy tradeoff of our algorithm degrades with lower quality samples, but it does not degrade as much as L_2 pruning + finetuning on synthetic samples.

5.4.2 Large Language Models

Table 3: Comparison of pruning methods on Llama-2-7B, measured with PPL and task accuracy

Sparsity	Algorithm	WikiText PPL ↓	ARC-e ↑	ARC-c ↑	PIQA ↑	WinoG.↑	HellaS. ↑	Average
0%	Dense	5.12	74.58	46.25	79.11	69.06	75.99	69.00
10%	SliceGPT [2]	6.46	56.14	35.33	69.53	64.80	59.02	59.96
	ModHiFi-P-WikiText (**ours**)	**5.97**	68.1	41.89	75.89	65.43	69.92	64.23
	ModHiFi–P-Alpaca (**ours**)	6.36	**71.42**	**42.06**	**76.44**	**68.19**	**71.67**	**65.96**
20%	ShortGPT [52]	14.32	58.33	38.05	72.58	**65.51**	**65.27**	59.95
	SliceGPT [2]	8.13	50.08	31.14	64.85	62.04	48.84	51.39
	ModHiFi-P-WikiText (**ours**)	**7.91**	60.1	34.89	70.62	61.48	58.7	57.16
	ModHiFi-P-Alpaca (**ours**)	9.38	**64.73**	**38.22**	**72.79**	64.64	62.7	**60.62**
30%	ShortGPT [52]	33.21	48.65	**32.85**	64.31	**64.33**	**56.13**	**53.25**
	SliceGPT [2]	**10.96**	44.19	27.47	58.71	57.46	41.27	45.82
	ModHiFi-P-WikiText (**ours**)	11.53	48.98	28.07	64.03	55.88	46.19	48.63
	ModHiFi-P-Alpaca (**ours**)	14.78	**53.15**	32.5	**66.59**	59.35	50.61	52.44

Baselines We evaluate ModHiFi on Llama-2-7B, comparing it against state-of-the-art algorithms for structured pruning [2, 52]. The use of calibration datasets to compute statistics is in line with our framing of distributional access to data, since LLMs do not make their training data openly accessible. Unless otherwise specified, the algorithms use WikiText-2 for calibration, with 128 samples of length 1024[4]. None of the algorithms perform post-pruning recovery fine-tuning. Additional details about our choice of baselines can be found in Appendix C.4.3.

[4]ShortGPT's calibration data is not publicly available.

Table 4: Comparison of class unlearning methods on CIFAR10.

Model	Algorithm	Forget Acc.	Remain Acc.	Time (sec)
ResNet-50	Base	94.99	94.99	-
	Gradient Ascent	6.59	93.44	30
	Jia et al. [35]	3.54	**94.14**	363
	Ours	**0.2**	92.98	**10**
Swin-T [49]	Base	92.31	92.31	-
	Jia et al. [35]	**1.20**	**90.69**	235
	Ours	8.83	73.57	**2**

Evaluation We also measure the performance of the model via its zero-shot accuracy on a suite of standard NLP tasks [5, 10, 68, 100] and WikiText perplexity. In Table 3, we see that our method is competitive, with consistently high average and task-specific performance, and outperforming at moderate sparsities. We find that the quality of the calibration set plays an important role, with the performance of ModHiFi-P-Alpaca being consistently higher than that of ModHiFi-P-WikiText. This tells us that retaining only HiFi components offers a **model-agnostic** approach to structured pruning, with its application to LLMs requiring no changes from its application to vision models.

5.5 Class Unlearning Experiments: (Q4)

Baselines and Metrics We report the forget and retain accuracy averaged across 10 classes of the CIFAR10 dataset on ResNet-50 and Swin-T models. We benchmark against Gradient Ascent and Jia et al. [35] which are both retraining-based technique for Unlearning.

Unlearning Results We report the results of our algorithm in Table 4. To answer (Q4), we observe that it is possible to perform unlearning **without finetuning** in a general editing framework $10\times$ faster than our baseline. In Appendix C.5, we compare results with finetuning using synthetic and training data. We note that the results for Swin-Transformer without finetuning fail to achieve state of the art. However, as reported in Appendix C.5, we observe a drastic improvement with only 3 epochs of finetuning on synthetic samples. With 10 epochs of finetuning with our algorithm, we find that compared to [35] (who use full training), our forget accuracy is superior when using synthetic samples, and both forget and remain accuracies are superior when using training samples. Experiments with VGG-19 are present in Appendix C.5 where we make similar observations.

6 Discussion and Conclusion

In this work, we identify the problem of model modification without access to the loss function or training data, and address it by using synthetic samples to isolate important, HiFi components. These components are few in number, are robust to sample noise, and strongly correlate with the model's predictive performance. We leverage their existence to develop the ModHiFi algorithm for pruning and classwise unlearning. Our work differs from prior art in three ways: first, our method is uniquely suited to synthetic data, the use of which is crucial to modern machine learning. Second, it enables model modification for a wide variety of tasks, of which we focus on pruning and classwise unlearning. Last, our method is architecture- and domain-agnostic, being applicable to both CNNs and transformers, for both vision and language tasks.

Limitations. The bound in Theorem 3.6 cannot be applied to networks at initialization, and we leave it to future work to explore models that are not well-trained.

Acknowledgments and Disclosure of Funding

We authors gratefully acknowledge AMD for their support. The authors thank Ramaswamy Govindarajan (IISc) and Prakash Raghavendra (AMD) for their insight and assistance in this work. The authors are also grateful to the reviewers of this work for improve its content.

References

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009. ISBN 0521424267. 3.2

[2] Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024. 1, 2.1, 2.3, 5.1, 5.1, 3, 5.4.2, A, C.4.3

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. 2.1

[4] Cenk Baykal, Lucas Liebenwein, Igor Gilitschenski, Dan Feldman, and Daniela Rus. Data-dependent coresets for compressing neural networks with applications to generalization bounds. In *International Conference on Learning Representations*, 2018. A

[5] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641. 5.4.2

[6] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020. A

[7] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE, 2021. 2.2, A

[8] Tianyi Chen, Bo Ji, Tianyu Ding, Biyi Fang, Guanyi Wang, Zhihui Zhu, Luming Liang, Yixin Shi, Sheng Yi, and Xiao Tu. Only train once: A one-shot neural network training and pruning framework, 2021. URL https://arxiv.org/abs/2107.07467. 5.4.1, 1

[9] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 2.3

[10] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457. 5.4.2

[11] Bosheng Ding, Chengwei Qin, Ruochen Zhao, Tianze Luo, Xinze Li, Guizhen Chen, Wenhan Xia, Junjie Hu, Luu Anh Tuan, and Shafiq Joty. Data augmentation using llms: Data perspectives, learning paradigms and challenges. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 1679–1705, 2024. 1, 2.2

[12] Chongyu Fan, Jiancheng Liu, Yihua Zhang, Eric Wong, Dennis Wei, and Sijia Liu. Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. *arXiv preprint arXiv:2310.12508*, 2023. 2.3

[13] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16091–16101, 2023. A

[14] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning, 2023. URL https://arxiv.org/abs/2301.12900. 1

[15] Gongfan Fang, Xinyin Ma, Michael Bi Mi, and Xinchao Wang. Isomorphic pruning for vision models. In *European Conference on Computer Vision*, pages 232–250. Springer, 2024. 2.3

[16] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018. A

[17] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR, 2023. A

[18] Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning, 2022. URL https://openreview.net/forum?id=ksVGCOlOEba. 5.4.1, A

[19] Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2426–2436, 2023. A

[20] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602. 5.1

[21] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312, 2020. 2.2, A

[22] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*, 2019. 2.2

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 1

[24] Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Calian, and Timothy Mann. Improving robustness using generated data. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713845393. C.3.2

[25] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11516–11524, 2021. A

[26] Marwa El Halabi, Suraj Srinivas, and Simon Lacoste-Julien. Data-efficient structured pruning via submodular optimization. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36613–36626. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ed5854c456e136afa3faa5e41b1f3509-Paper-Conference.pdf. 3.1, A

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5.1, D.2

[28] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper%5Ffiles/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf. C.3.1

[29] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021. 1, 2.2, 2.3, A

[30] Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248*, 2019. 1, A

[31] Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. Characterising bias in compressed models. *arXiv preprint arXiv:2010.03058*, 2020. 1

[32] Cheng-Yu Hsieh, Chun-Liang Li, CHIH-KUAN YEH, Hootan Nakhost, Yasuhisa Fujii, Alex Jason Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023. 2.2

[33] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pages 2008–2016. PMLR, 2021. A

[34] Saachi Jain, Hannah Lawrence, Ankur Moitra, and Aleksander Madry. Distilling model failures as directions in latent space. *arXiv preprint arXiv:2206.14754*, 2022. 1, 2.2

[35] Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model sparsity can simplify machine unlearning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=0jZH883i34. 1, (C1), 2.2, 2.3, (Q4), 4, 5.5, 5.5, A, C.5, 12

[36] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*, 2023. 2.3

[37] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022. C.3.3

[38] Sangamesh Kodge, Gobinda Saha, and Kaushik Roy. Deep unlearning: Fast and efficient gradient-free class forgetting. *Transactions on Machine Learning Research*, 2024. 1, 2.3

[39] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf. 1, 5.1

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper%5FFfiles/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. 1

[41] Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. *Advances in Neural Information Processing Systems*, 36, 2024. 2.2, A

[42] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989. URL https://proceedings.neurips.cc/paper%5Ffiles/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf. A

[43] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. A

[44] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rJqFGTslg. 2, A

[45] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. In *International Conference on Learning Representations*, 2019. A

[46] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1529–1538, 2020. A

[47] Jing Liu, Bohan Zhuang, Zhuangwei Zhuang, Yong Guo, Junzhou Huang, Jinhui Zhu, and Mingkui Tan. Discrimination-aware network pruning for deep model compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. A

[48] Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression. In *International Conference on Machine Learning*, pages 7021–7032. PMLR, 2021. (C1), A

[49] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. URL https://arxiv.org/abs/2103.14030. 5.1, 4, D.2

[50] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017. 5.4.1, 1

[51] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023. 1, (C1), 2.3, 5.1, 5.1, A

[52] Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*, 2024. 2.3, 3, 5.4.2, A, C.4.3

[53] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843. 5.1, C.3

[54] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2): 1–40, 2023. 1

[55] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021. 1

[56] P Molchanov, S Tyree, T Karras, T Aila, and J Kautz. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings*, 2019. A

[57] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019. A

[58] Chaitanya Murti and Chiranjib Bhattacharyya. DisCEdit: Model editing by identifying discriminative components. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=tuiqq1G8I5. 1, 3, 2.2, 2.3, A, C.5, C.5

[59] Chaitanya Murti, Tanay Narshana, and Chiranjib Bhattacharyya. Tvsprune-pruning non-discriminative filters via total variation separability of intermediate representations without fine tuning. In *The Eleventh International Conference on Learning Representations*, 2022. A

[60] Preetum Nakkiran, Behnam Neyshabur, and Hanie Sedghi. The deep bootstrap framework: Good online learners are good offline generalizers, 2021. URL https://arxiv.org/abs/2010.08127. C.3.1

[61] Tanay Narshana, Chaitanya Murti, and Chiranjib Bhattacharyya. Dfpc: Data flow driven pruning of coupled channels without data. In *The Eleventh International Conference on Learning Representations*, 2023. 5.4.1, 1, 2, 3, A, C.4.2, C.6, D.3

[62] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*, 2022. 2.2, A

[63] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper%5Ffiles/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf. C.6, C.7.2

[64] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703. C.6

[65] Prafull Prakash, Chaitanya Murti, Saketha Nath, and Chiranjib Bhattacharyya. Optimizing dnn architectures for high speed autonomous navigation in gps denied environments on edge devices. In *Pacific Rim International Conference on Artificial Intelligence*, pages 468–481. Springer, 2019. 1, A

[66] Xianbiao Qi, Jianan Wang, Yihao Chen, Yukai Shi, and Lei Zhang. Lipsformer: Introducing lipschitz continuity to vision transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cHf1DcCwcH3. (C1), 3.7

[67] Sabyasachi Sahoo, Mostafa Elaraby, Jonas Ngnawe, Yann Pequignot, Frédéric Precioso, and Christian Gagné. Layerwise early stopping for test time adaptation. *arXiv preprint arXiv:2404.03784*, 2024. 1, 2.2

[68] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641. 5.4.2

[69] Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules. *Advances in Neural Information Processing Systems*, 34:23359–23373, 2021. 1, 2.2

[70] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021. A

[71] Harshay Shah, Andrew Ilyas, and Aleksander Madry. Decomposing and editing predictions by modeling model computation. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=rTBR0eqE4G. 1, 3, 2.2, 2.2

[72] Maying Shen, Hongxu Yin, Pavlo Molchanov, Lei Mao, Jianna Liu, and Jose Alvarez. Structural pruning via latency-saliency knapsack. In *Advances in Neural Information Processing Systems*, volume 35, pages 12894–12908, 2022. C.6

[73] Maying Shen, Hongxu Yin, Pavlo Molchanov, Lei Mao, Jianna Liu, and Jose M Alvarez. Structural pruning via latency-saliency knapsack. *Advances in Neural Information Processing Systems*, 35:12894–12908, 2022. A

[74] Xuan Shen, Zhao Song, Yufa Zhou, Bo Chen, Jing Liu, Ruiyi Zhang, Ryan A. Rossi, Hao Tan, Tong Yu, Xiang Chen, Yufan Zhou, Tong Sun, Pu Zhao, Yanzhi Wang, and Jiuxiang Gu. Numerical pruning for efficient autoregressive models. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i19.34249. URL https://doi.org/10.1609/aaai.v39i19.34249. 2.3, A

[75] Md Maruf Hossain Shuvo, Syed Kamrul Islam, Jianlin Cheng, and Bashir I Morshed. Efficient acceleration of deep learning inference on resource-constrained edge devices: A review. *Proceedings of the IEEE*, 111(1):42–91, 2022. 1

[76] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1409.1556. 5.1, D.2

[77] Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *arXiv preprint arXiv:2402.09025*, 2024. 2.3

[78] Yang Sui, Miao Yin, Yi Xie, Huy Phan, Saman Aliari Zonouz, and Bo Yuan. Chip: Channel independence-based pruning for compact neural networks. *Advances in Neural Information Processing Systems*, 34, 2021. A

[79] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023. A

[80] Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. Large language models for data annotation and synthesis: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 930–957, 2024. 1, 2.2

[81] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33:6377–6389, 2020. A

[82] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023. 5.1, C.3

[83] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 303–319. IEEE, 2022. A

[84] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. 5.1

[85] Murad Tukan, Loay Mualem, and Alaa Maalouf. Pruning neural networks via coresets and convex geometry: Towards no assumptions. *Advances in Neural Information Processing Systems*, 35:38003–38019, 2022. A

[86] Enayat Ullah and Raman Arora. From adaptive query release to machine unlearning. In *International Conference on Machine Learning*, pages 34642–34667. PMLR, 2023. A

[87] Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Neural pruning via growing regularization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=o966%5FIs%5FnPA. 5.4.1, 1

[88] Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. Federated unlearning via class-discriminative pruning. In *Proceedings of the ACM Web Conference 2022*, pages 622–632, 2022. A

[89] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. A

[90] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. *arXiv preprint arXiv:2108.11577*, 2021. A

[91] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020. URL https://arxiv.org/abs/1910.03771. C.6

[92] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023. 2.3, A

[93] Zuobin Xiong, Wei Li, Yingshu Li, and Zhipeng Cai. Exact-fun: an exact and efficient federated unlearning approach. In *2023 IEEE International Conference on Data Mining (ICDM)*, pages 1439–1444. IEEE, 2023. A

[94] Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2024. A

[95] Haonan Yan, Xiaoguang Li, Ziyao Guo, Hui Li, Fenghua Li, and Xiaodong Lin. Arcane: An efficient architecture for exact machine unlearning. In *IJCAI*, volume 6, page 19, 2022. A

[96] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724, 2020. 1

[97] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3143–3151, 2022. 2.3

[98] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018. 3.7, A, B.1

[99] Shixing Yu, Tianlong Chen, Jiayi Shen, Huan Yuan, Jianchao Tan, Sen Yang, Ji Liu, and Zhangyang Wang. Unified visual transformer compression. *arXiv preprint arXiv:2203.08243*, 2022. 2.3

[100] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830. 5.4.2

[101] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021. 1

[102] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12: 1556–1577, 2024. 2.2, A

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We link each contribution in the paper's contents when we theoretically or empirically justify the claims.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Yes, for each theoretical result, we provide a complete set of assumptions and a correct proof. Proofs are attached in the appendix. We link the relevant appendices in the main body for the reader.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Yes, we fully disclose all the information to reproduce the main experimental results in Section 5 and the appendices mentioned within that section. Moreover, we provide our code, and the data sets used are open-sourced.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code and the instructions to reproduce the experiments are provided in our Anonymous GitHub link provided in the paper.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All details to understand the results and reproduce the results are provided in Section 5 and the appendices mentioned in this section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For experiments where error bars are relevant and computationally feasible, we report them.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We sufficiently describe the compute resources used for our experiments in Section 5 and the appendices referred to within the section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We go through the NeurIPS Code of Ethics and confirm that we adhere to them.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: We do not discuss the societal impact of the work performed in this manuscript since this is foundational research and not tied to particular applications.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release models or data in this work.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We credit original owners of assets used in this work appropriately through citations.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release any new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We do not use LLMs to develop any methods presented in this work. We clarify further in Appendix E.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

APPENDIX

This appendix and their takeaways are summarized below for ease of navigation:

1. Appendix A contains additional related work and description of the gaps in existing literature addressed in our work.
2. Appendix B contains proofs and discussions not presented in the main body. In particular, we present the following:
   - In Appendix B.1, we provide the proof for Theorem 3.6. We also state and justify the assumption used towards proving the Theorem.
   - In Appendix B.2, we prove the properties of Subset Fidelity stated in Lemma 3.2.
   - In Appendix B.4, we prove the optimality of the naive algorithm in selecting the k-MFS Optimal set. While the assumption of uncorrelated features might not hold under practical scenarios, this result provides an indication that the method could result in effective identification of critical model components in practical settings. Our experiments in Section 5 and Appendix C practically demonstrate the empirical efficacy of the methodology.
3. Appendix C contains additional experimental validation that make the following points:
   - In Appendix C.1 we validate the existence of HiFi sets.
     - In Appendix C.1.1, we show the results of the full Monte-Carlo experiments on more models and datasets to strengthen our answer for Question (Q1).
     - In Appendix C.1.2, we conduct counterfactual experiments to validate if the sets computed by our proposed method are disproportionately responsible for predictive performance. **This emphasizes the effectiveness of Subset Fidelity in addition to our theoretical results in Theorem 3.9**.
     - In Appendix C.1.3, we empirically discuss the sensitivity of the estimation of $\boldsymbol{Q}^c$. This ablation study shows that the Subset Fidelity score is robust to the number of samples used for estimation. Among the datasets used, we see that we require at most 200 synthetic samples per class for accurate estimation.
   - In Appendix C.3, we study the effect of quality of synthetic samples on our proposed method. We find that higher quality data leads to an improved sparsity accuracy tradeoff.
   - In Appendix C.4 we provide pruning results on additional datasets strengthen our validation of Question (Q3).
     - In Appendix C.4.1 we show that each sub-module of our model modification is critical towards successful model modification.
     - In Appendix C.4.2 we compare the pruned ImageNet models of ModHiFi-P against SoTA data-free structured pruning DFPC [61] to compare layer-wise sparsity to see where is improved speedup coming from.
     - In Appendix C.4.3, we justify the appropriateness of our baselines for the LLM pruning baselines.
   - In Appendix C.5 we provide additional unlearning experiments with different models to strengthen our validation of Question (Q4) and discuss unlearning with finetuning. We validate that ModHifi-U performs competitively without finetuning against baselines. Moreover, with very few epochs of finetuning over synthetically generated samples, we achieve complete unlearning, as opposed to our baselines who fully-finetune on entire training set.
   - In Appendix C.6, we discuss implementation and compute platform details and additional timing measurements, to improve replicability of our empirical results.
   - In Appendix C.7 we discuss hyperparameters used for training, to improve replicability of our empirical results.
4. Appendix D contains additional algorithmic details including
   - Appendix D.1 clarifies of the role of Lipschitz Constants in our algorithms since they are critical to Theorem 3.6.
   - Appendix D.2 presents practical details as to how we implement fidelity estimation in our model modification algorithms.

- Appendix D.3 presents the computational complexity of estimating fidelity. The fidelity is linear in number of layers as opposed to the component identification module of SoTA in data-free structured pruning, which is quadratic in the number of layers.

A Related Work

We present a short literature review in Section 2. In this section, we discuss recent related work on structured pruning and unlearning not discussed in the main body.

Structured Pruning Structured pruning has been widely researched, with a wide variety of methods proposed for it [29]. Unlike unstructured pruning, which sparsifies the weight matrices without changing the architecture of the model [6, 16, 18, 42, 81], structured pruning enables immediate improvements in real-world performance measures such as inference time and memory footprint without requiring specialized hardware or software [29, 56, 65]. A variety of methods have been proposed for structured pruning of convolutional networks, including using norms of weight tensors [43, 44], directional derivative scores [56, 57, 73], feature map ranks [46, 78], coresets [4, 45, 85], discriminative ability of filters [47, 59], and reconstruction error [26, 74, 98]. However, modern neural networks possess complex interconnections, making them difficult to structurally compress [13, 48, 61], for which some recent algorithms have been proposed that use gradient information [48] or bounds on the reconstruction error [61, 98]. Moreover, pruning without access to either the training data or the loss function is an increasingly important area of research, for which some works have been proposed that use the discriminative ability of filters as a saliency [47, 59]. However, none of these works address the problem of pruning large language models.

Pruning of Large Language Models (LLMs) has garnered significant interest in recent years [102]. A variety of unstructured pruning methods have been proposed, such as [17, 79]. However, these methods do not provide direct improvements on inference time and memory footprint. Thus, the problem of pruning models with structural interconnections has naturally been applied to pruning LLMs as well, in works such as [2, 51, 52, 92]. A key drawback of these works is that most are not applicable to CNNs or other kinds of models. Our work proposes a unified framework for both pruning models with complex interconnections, including transformers and ResNets, as well as classwise unlearning.

Classwise Unlearning Machine unlearning has gained significant interest in recent years, both for data privacy concerns as well as connections to continual learning [7, 30, 62, 89]. Machine unlearning is typically categorized into exact and approximate unlearning [94]. Exact unlearning involves training models from scratch without the forget data (the data to be forgotten), or by training modules or experts on subsets of data [93, 95]. Approximate unlearning, on the other hand, refers to techniques that approximate exact unlearning via various approaches [33, 94]. Machine unlearning can be further classified into sample unlearning (wherein individual samples or random subsets of samples are unlearned)[70, 86] or classwise unlearning (where classes or concepts are unlearned) [19, 25]. In this work, we focus on classwise unlearning.

A variety of approaches have been proposed for classwise unlearning [25]. Popular methods include fine-tuning the model without data from the forget class [21, 90], gradient ascent on the forget set [25, 83], distillation-based approaches [41], and influence function based methods [33]. More recent work studies using sparsity for machine unlearning, such as [35], which first sparsifies the model, and then applies a fine-tuning-based unlearning algorithm, or [58, 88], which identify class-discriminative filters in CNNs, and removes them for unlearning. Two key drawbacks of prior art, however, are: first they exclusively address classwise unlearning, and do not address wider problems of model modification. Second, all prior art assumes access to the original training data. Our proposed approach for classwise unlearning differs from prior art because it only requires synthetic class data, uses a variety of granularities for sparsity in unlearning, and is part of a unified approach to model modification.

B Proofs

In this section, we restate the formal statements made in the main body of the paper and present the proofs ommitted in the main body.

B.1 Proof of Theorem 3.6

We now provide a proof of Theorem 3.6. We first state our assumptions and provide empirical evidence to justify their validity, followed by a restatement of the theorem, and its proof. The assumptions we make are on the norms of the activations of RMSNorm and LayerNorm.

Definition B.1 (RMSNorm and LayerNorm)**.** Let the l^{th} layer be an RMSNorm layer (or equivalently LayerNorm) with parameters $\boldsymbol{\gamma^l} \in \mathbb{R}^d$ and $\boldsymbol{\beta}^l \in \mathbb{R}^d$. For an input $\boldsymbol{\Phi}^l(\boldsymbol{x}) \in \mathbb{R}^d$ to the layer, the output $\boldsymbol{y}^l \in \mathbb{R}^d$ is given by

$$\text{RMS}^l(\boldsymbol{x}) = \boldsymbol{y}^l = \boldsymbol{\gamma}^l \odot \frac{\boldsymbol{z}^l}{||\boldsymbol{z^l}||_2} + \boldsymbol{\beta}^l \;\text{ where }\; \boldsymbol{z}^l = \boldsymbol{M}\boldsymbol{\Phi}^l(\boldsymbol{x})$$

where $\odot$ indicates the element-wise product and $\boldsymbol{M} = \boldsymbol{I}_d$ for RMSNorm and $\boldsymbol{M} = \boldsymbol{I}_d - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top$ for LayerNorm.

We now state some properties of RMSNorm and LayerNorm layers when the norms of the input are lower bounded.

Definition B.2. A function $f : \mathbb{R}^m \to \mathbb{R}^n$ is Lipschitz continuous if there exists a positive scalar constant L such that

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\|_2 \leq L\|\boldsymbol{x} - \boldsymbol{y}\|_2 \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$$

We now state a useful lemma which we use for showing Lipschitzness of RMSNorm.

Lemma B.3. *Let r be some positive scalar. Let the function $f_r : C_r \to \mathbb{R}^d$ where $C_r \subset \mathbb{R}^d = \{x \in \mathbb{R}^d \mid \|\boldsymbol{x}\|_2 \geq r\}$ be defined as*

$$f_r(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$$

f_r is a Lipschitz continuous function with Lipschitz constant $\frac{1}{r}$.

Proof. We have

$$\begin{aligned}\|f_r(\boldsymbol{x}) - f_r(\boldsymbol{y})\|_2^2 &= \left\| \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} - \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|} \right\|_2^2 \\ &= 2 - 2\frac{\boldsymbol{x}^\top \boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}\end{aligned}$$

and

$$\begin{aligned}\|\boldsymbol{x} - \boldsymbol{y}\|^2 &= \|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - 2\boldsymbol{x}^\top \boldsymbol{y} \\ &= \|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - 2\|\boldsymbol{x}\|\|\boldsymbol{y}\| + 2\|\boldsymbol{x}\|\boldsymbol{y}\| - 2\boldsymbol{x}^\top \boldsymbol{y} \\ &= (\|\boldsymbol{x}\| - \|\boldsymbol{y}\|)^2 + 2(\|\boldsymbol{x}\|\boldsymbol{y}\| - \boldsymbol{x}^\top \boldsymbol{y}) \\ &= (\|\boldsymbol{x}\| - \|\boldsymbol{y}\|)^2 + 2\|\boldsymbol{x}\|\boldsymbol{y}\|(1 - \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\|\boldsymbol{x}\|\boldsymbol{y}\|}) \\ &\geq 2\|\boldsymbol{x}\|\boldsymbol{y}\|(1 - \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\|\boldsymbol{x}\|\boldsymbol{y}\|}) \quad \text{since } (\|\boldsymbol{x}\| - \|\boldsymbol{y}\|)^2 \geq 0 \\ &\geq 2r^2(1 - \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\|\boldsymbol{x}\|\boldsymbol{y}\|}) \quad \text{since } \|\boldsymbol{x}\| \geq r, \|\boldsymbol{y}\| \geq r \\ &= r^2\|f_r(\boldsymbol{x}) - f_r(\boldsymbol{y})\|_2^2\end{aligned}$$

Thus,

$$\|f_r(\boldsymbol{x}) - f_r(\boldsymbol{y})\|_2 \leq \frac{1}{r}\|\boldsymbol{x} - \boldsymbol{y}\|$$

□

Using the above lemma, we now show that RMSNorm and Layer Norm layers are also Lipschitz continuous under suitable assumptions.

Corollary B.4. *When the norms of the input to l^{th} layer which is a RMSNorm (or LayerNorm) layer, $\|\mathbf{\Phi}(\boldsymbol{x})\|_2$ are lower bounded by some positive scalar r, for all inputs from a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \dots \boldsymbol{x}_n\}$, the layer is $\frac{\max_i |\gamma_i|}{r}$-Lipschitz continuous over the dataset $\mathcal{D}$.*

Proof. We prove here for RMSNorm, with the proof for LayerNorm following similarly. From Definition B.1,

$$\begin{aligned}\|\mathrm{RMS}^l(\boldsymbol{x}) - \mathrm{RMS}^l(\boldsymbol{y})\| &= \|\gamma^l \odot f_r(\boldsymbol{M}\Phi^l(\boldsymbol{x})) - \gamma^l \odot f_r(\boldsymbol{M}\Phi^l(\boldsymbol{y}))\\ &\leq (\max_i |\gamma_i^l|)\|f_r(\boldsymbol{M}\Phi^l(\boldsymbol{x})) - f_r(\boldsymbol{M}\Phi^l(\boldsymbol{y}))\|\end{aligned}$$

Under the conditions of the corollary, we may apply Lemma B.3 to complete the proof. □

Although the condition on lower bounded inputs holds true if every $\boldsymbol{z}^l$ is not zero, we make this assumption explicit since we do not want the lower bound to be vanishingly small. To justify this assumption, we show the layer-wise minimum norm of the pre-LayerNorm representations in Fig. 3, estimated on 100 samples from the Alpaca dataset, where for various models we observe the lower bound to be between 0.2 to 60. For clarity of exposition, we only show the layers with the largest and smallest values along with 5 randomly selected layers. Code for generating these plots can be found in Appendix C. We also observe that this value tends to increase for layers deeper in the network, and leave the utilization of this observation to future work.

Fact 1. *A function $f = f^L \circ f^{L-1} \circ \dots \circ f^1$ where each f^i is Lipschitz continuous with Lipschitz constant L^i, is Lipschitz continuous with Lipschitz constant $\prod_{i=1}^L L^i$.*

We now restate and prove Theorem 3.6.

Theorem 3.6 (Local to Global)**.** *Consider a network as described in Section 2.1. Let all the parameters $\mathcal{W}^\ell \neq 0\ \forall \ell \in [L]$. Let $\boldsymbol{M}^l$ represent a mask which modifies the parameters in layer l. If there exists a positive scalar r^ℓ for each layer $\ell > l$ such that $\|\mathbf{\Phi}_c^\ell(\mathrm{X})\|_F \geq r^\ell \ \ \forall c \in [c_{out}^\ell]$ almost surely (over the distribution of X), then there exist constants $\{C_\ell\}_{\ell=1}^L$ (dependent only on the architecture) such that:*

$$\mathbb{E}\big[\|\mathrm{N}_\theta(\mathrm{X}) - \mathrm{N}_{\theta\odot\boldsymbol{M}^l}(\mathrm{X})\|^2\big] \leq C_l^2 \sum_{c=1}^{c_{out}^l} \Big((\mathbf{1}_{c_{in}^l} - \boldsymbol{m}_c^l)^\top \boldsymbol{Q}_c^l (\mathbf{1}_{c_{in}^l} - \boldsymbol{m}_c^l)\Big) \tag{2}$$

where $\boldsymbol{Q}_c^l \in \mathbb{R}^{c_{in}^l \times c_{in}^l}$ is the component similarity matrix (CSM) *for channel c, with entries $(\boldsymbol{Q}_c^l)_{ij} = \mathbb{E}[\langle \boldsymbol{A}_{ci}^l(\mathrm{X}), \boldsymbol{A}_{cj}^l(\mathrm{X})\rangle]$.*

Proof. Consider a network as defined in Section 2.1. Let $\mathrm{N}_\theta = f^L \circ \dots \circ f^l \circ f^{l-1:1}$ where $f^{l-1:1} = f^{l-1} \circ \dots \circ f^1$. Under standard assumptions on the smoothness of activations [98], each layer f^l is Lipschitz continuous with Lipschitz constant L_f^l. From Fact 1,

$$\mathbb{E}\big[\|\mathrm{N}_\theta(\mathrm{X}) - \mathrm{N}_{\theta\odot\boldsymbol{M}^l}(\mathrm{X})\|^2\big] \leq (\prod_{\ell>l}^{L} L_f^\ell) \sum_{c=1}^{c_{out}^l} \mathbb{E}[\|\boldsymbol{Y}^l(\mathrm{X}) - \sum_i m_{ci} \boldsymbol{A}_{ci}(\mathrm{X})]\|^2$$

By taking an upper bound on the Lipschitz constants of each layer in the composition, we see that the subnetwork after layer l has a Lipschitz constant of at least $C^l = \prod_{\ell>l}^L L_f^\ell$. Where, for convolution based networks,

$$C_l = \left(\max_i \frac{\gamma_i^l}{\sigma_i^l}\right) \eta^{L-l} \prod_{\ell>l} \|\mathcal{W}^\ell\|_2 \cdot \max_i \frac{|\gamma_i^\ell|}{\sigma_i^\ell}$$

and for transformer models,

$$C_l = \eta^{L-l} \prod_{\ell>l} \|\mathcal{W}^\ell\|_2 \cdot \max_i \frac{|\gamma_i^\ell|}{r^\ell}$$

The theorem then follows from the definition of expected square loss. □

B.2 Proof of Lemma 3.2

In this section, we prove the properties of Subset Fidelity stated in Lemma 3.2.

Definition 3.1 (Subset Fidelity)**.** The *fidelity* of a subset of components $C \subseteq [c_{in}^l]$ in layer l for output channel c is defined as

$$\mathrm{FS}_c^l(C) = \max_{\delta_c^l} \left(1 - \frac{d\big(\boldsymbol{Y}_c^l(\mathrm{X}), \sum_{i\in C} \delta_{ci}^l \boldsymbol{A}_{ci}^l(\mathrm{X})\big)}{d(\boldsymbol{Y}_c^l(\mathrm{X}), \mathbf{0})}\right) \tag{1}$$

where δ_c^l is a *compensation term* that scales only the components in C and $d(f(\mathrm{X}), g(\mathrm{X}))$ denotes $\mathbb{E}[\|f(\mathrm{X}) - g(\mathrm{X})\|_2^2]$ for appropriate functions f, g.

Lemma 3.2 (Properties of Subset Fidelity)**.** *For any subset $C \subseteq [c_{in}^l]$ in layer l, the following hold: (1) $0 \leq \mathrm{FS}_c^l(C) \leq 1$, (2) if $D \subset C$, then $\mathrm{FS}_c^l(D) \leq \mathrm{FS}_c^l(C)$.*

Proof. We first show the boundedness of the fidelity score. Let $\delta_c^\star(C)$ is the value at which the maximum is attained for all vectors where indices of elements are contained in set C. Since $\mathbf{0}_{c_{in}^l}$ is a feasible solution to this optimization problem,

$$\min_{\delta_c} d(\boldsymbol{Y}_c^l(\mathrm{X}), \sum_{i\in C} \delta_{ci} \boldsymbol{A}_{ci}^l(\mathrm{R})) \leq d(\boldsymbol{Y}_c^l(\mathrm{X}), \mathbf{0})$$

From the definition of a metric, $d(\boldsymbol{Y}_c^l(\mathrm{X}), \mathbf{0}) \geq 0$. Thus completing the proof of boundedness.

To prove monotonicity, we show that $D \subset C \implies \mathrm{FS}_c^l(D) \leq \mathrm{FS}_c^l(C)$. We again utilize the optimality of $\delta_c^\star$. $\delta_c^\star(D)$ will always be feasible for the optimization problem for the subset C since $D \subset C$. Hence, $d(\boldsymbol{Y}_c^l(\mathrm{X}), \sum_{i\in C} \delta_{ci}^\star(D) \boldsymbol{A}_{ci}^l(\mathrm{R})) \geq d(\boldsymbol{Y}_c^l(\mathrm{X}), \sum_{i\in C} \delta_{ci}^\star(C) \boldsymbol{A}_{ci}^l(\mathrm{R}))$. The proof then follows from the definition of Subset Fidelity. □

B.3 Proof of Proposition 3.8

Proposition 3.8 (Compensation and Singleton Fidelity)**.** *For the l_2 reconstruction error, the optimal compensation term $\delta_c^\star$, which is the value at which the fidelity score is computed according to Eq. (1) for a subset C is given by,*

$$\delta_{ci}^{l\star}(C) = \begin{cases} 1 + ((\boldsymbol{Q}_c^l[C,C])^{-1})_i^\top \boldsymbol{Q}_c^l[C,\overline{C}]\mathbf{1}_{n-k} & \text{if } i \in C \\ 0 & \text{if } i \notin C \end{cases} \tag{FS}$$

where $\boldsymbol{Q}_c^l$ is the component similarity matrix as in Eq. (2). The singleton fidelity scores are:

$$s_{ci}^l = \mathrm{FS}_c^l(\{i\}) = 1 - \frac{\mathbb{E}[\|\boldsymbol{Y}_c^l(\mathrm{X}) - \alpha_{ci}^l \boldsymbol{A}_{ci}^l(\mathrm{X})\|^2]}{\mathbb{E}[\|\boldsymbol{Y}_c^l(\mathrm{X})\|^2]}, \quad \alpha_{ci}^l = \frac{\mathbb{E}[\langle \boldsymbol{Y}_c^l(\mathrm{X}), \boldsymbol{A}_{ci}^l(\mathrm{X})\rangle]}{\mathbb{E}[\|\boldsymbol{A}_{ci}^l(\mathrm{X})\|^2]}. \tag{3}$$

Proof. The expected square loss is given by as

$$d\left(\boldsymbol{Y}_c^l(\mathrm{X}), \sum_{i\in C} v_i \boldsymbol{A}_{ci}^l(\mathrm{X})\right) = (\mathbf{1}_{c_{in}^l} - \boldsymbol{v})^\top \boldsymbol{Q}_c^l (\mathbf{1}_{c_{in}^l} - \boldsymbol{v})$$

Consider minimizing the reconstruction error without components in a given set C. Then, the optimal reconstruction error is obtained at

$$\boldsymbol{v}^\star = \underset{\boldsymbol{v} \;\; s.t \;\; v_i = 0 \;\; \forall i \notin C}{\arg\min} (\mathbf{1}_{c_{in}^l} - \boldsymbol{v})^\top \boldsymbol{Q}_c^l (\mathbf{1}_{c_{in}^l} - \boldsymbol{v})$$

Let $\boldsymbol{z} = \mathbf{1}_{c_{in}^l} - \boldsymbol{v}$ and $z_i = 1 \;\; \forall i \notin C$. We have now reduced the problem to a an unconstrained quadratic. The objective is given by

$$\boldsymbol{z}_C^\top \boldsymbol{Q}_c^l[C,C]\boldsymbol{z}_C + 2 * \boldsymbol{z}_C \boldsymbol{Q}_c^l[C,\overline{C}]\mathbf{1}_m + \mathbf{1}_m^\top \boldsymbol{Q}_c^l[\overline{C},\overline{C}]\mathbf{1}_m$$

where $m = c_{in}^l - |C|$ and $\boldsymbol{z}_C$ is the sub-vector of $\boldsymbol{z}$ containing indices of C. This is a quadratic optimization problem with solution,

$$\boldsymbol{z}_C^\star = -(\boldsymbol{Q}_c^l[C,C])^{-1}(\boldsymbol{Q}_c^l[C,\overline{C}])\mathbf{1}_m.$$

Substituting $\boldsymbol{v}^\star = \mathbf{1}_{c_{in}^l} - \boldsymbol{z}^\star$ completes the proof. □

B.4 Proof of Theorem 3.9

In this section, we prove the optimality of the naive algorithm in selecting the k-MFS Optimal set.

Theorem 3.9. *Consider output channel c in the l^{th} layer of a network described in Section 2.1. Let the s^l_{ci} be defined according to Eq. (3). Let*

$$\hat{S}^l_c = \{i \mid s^l_{ci} \geq s_{(k)}\}$$

where $s_{(k)}$ is the k largest value of $\boldsymbol{s}^l_c$. Assuming that that there are no ties, $|\hat{S}^l_c| = k$. If $\mathbb{E}[\langle \boldsymbol{A}^l_{ci}(\mathrm{X}), \boldsymbol{A}^l_{cj}(\mathrm{X})\rangle] = 0 \ \ \forall i \neq j$, then

$$\hat{S}^l_c = \underset{S \subseteq [c^l_{in}],\, |S|=k}{\arg\max} \ \mathrm{FS}^l_c(S)$$

Proof. The assumption $\mathbb{E}[\langle \boldsymbol{A}_{ci}, \boldsymbol{A}_{cj}\rangle] = 0 \quad \forall i \neq j$ implies $((\boldsymbol{Q}^l_c)_{ij})_{ij} = 0$ and $(\boldsymbol{Q}^l_c)_{ii} = \mathbb{E}[||\boldsymbol{A}_{ci}||^2] \geq 0$. This implies that the component similarity matrix is diagonal. The Subset fidelity of a set is then given by

$$\begin{aligned}
\mathrm{FS}^l_c(C) &\overset{(a)}{=} 1 - \frac{\sum_{i=1}^{c^l_{in}} \boldsymbol{Q}^l_{cii} 1[i \notin C]}{\sum_{i=1}^{c^l_{in}} \boldsymbol{Q}^l_{cii}} \\
&= \frac{\sum_{i=1}^{c^l_{in}} \boldsymbol{Q}^l_{cii} 1[i \in C]}{\sum_{i=1}^{c^l_{in}} \boldsymbol{Q}^l_{cii}} = \sum_{i=1}^{c^l_{in}} \left(\frac{\boldsymbol{Q}^l_{cii}}{\sum_{i=1}^{c^l_{in}} \boldsymbol{Q}^l_{cii}} \right) 1[i \in C] \\
&= \sum_{i=1}^{c^l_{in}} \zeta^l_{ci} 1[i \in C]
\end{aligned}$$

where $1[i \in C]$ is an indicator function that takes 1 for when index i is in C and 0 otherwise. $\zeta^l_{ci} = \frac{\boldsymbol{Q}^l_{cii}}{\sum_{i=1}^{c^l_{in}} \boldsymbol{Q}^l_{cii}}$ is the normalized score for each component with $\sum_{i=1}^{c^l_{in}} \zeta^l_{ci} = 1$. (a) comes from the fact that $\delta^\star_i(C) = 1[i \in C]$ and substitution into the definition of expected square loss.

Observe now that the fidelity scores for each component can be written as

$$s^l_{cj} = \mathrm{FS}^l_c(\{j\}) = \zeta^l_{cj}$$

The Eq. (K-MFS) objective is then,

$$S^\star_c = \underset{S \subseteq [c^l_{in}],\, |S|=k}{\arg\max} \sum_{i \in S} \zeta^l_{ci}$$

Clearly, this is a linear objective that is maximized when the set S contains the k largest vales of $\boldsymbol{\zeta}^l_c$, which in turn correspond to the k largest values of the fidelity scores. □

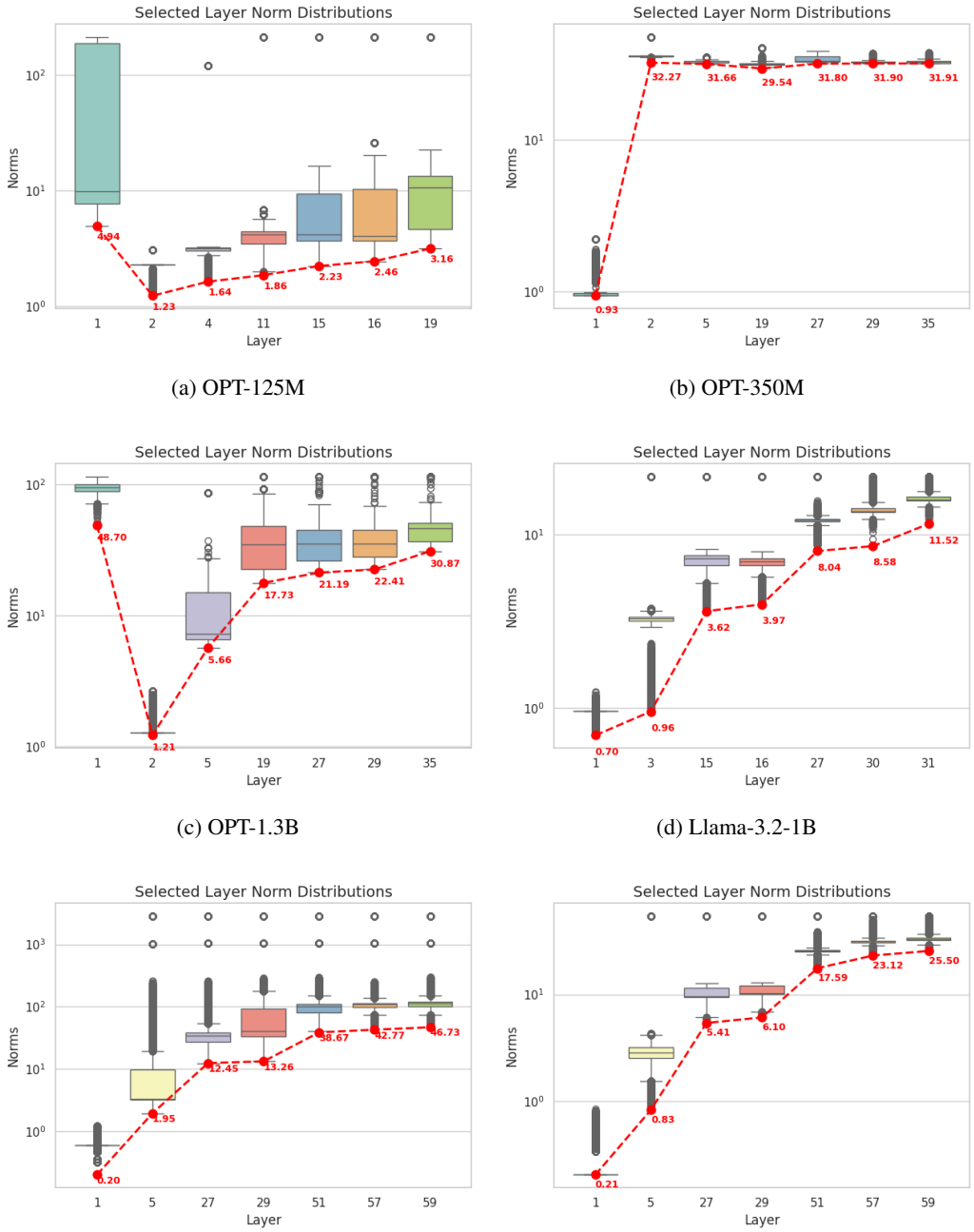
Remark B.5. While the assumption of uncorrelated features might not hold under practical scenarios, this result provides an indication that the method could result in effective identification of critical model components in practical settings. Our experiments in Section 5 and Appendix C practically demonstrate the empirical efficacy of the methodology.

C Additional Experiments

In this appendix we detail additional results and ablations.

1. We elaborate on the Monte Carlo simulations of Eq. (K-MFS) across multiple models, as well as the efficiency with which Subset Fidelity estimates this while being robust to data samples. We present noising and counterfactual results to demonstrate this.
2. We discuss the synthetic samples used in our vision experiments and the effect of their quality on the algorithm.
3. We provide additional pruning and unlearning experiments for a variety of architectures.
4. We provide details of our compute platform, hyperparameters, and training procedure for our experiments.

Our code is available at `https://github.com/DhruvaKashyap/modhifi`.



(a) OPT-125M

(b) OPT-350M

(c) OPT-1.3B

(d) Llama-3.2-1B

(e) Llama-2-7B

(f) Llama-3.1-8B

Figure 3: Boxplots for the distribution of norms of inputs to normalization layer. Minimum value indicated in Red, showing that $\frac{1}{r}$ is at most 5. Y-axis is log scale.

C.1 Validating Subset Fidelity and H1F1 Sets

C.1.1 Monte Carlo Experiments

In this section, we provide additional details regarding Fig. 1 and demonstrate this behaviour across architectures.

Ideally, one would solve Eq. (K-MFS) exactly to compute those sets that have optimal reconstructive ability at different sizes. However, since enumerating across subsets is a combinatorial problem, we

instead approximate a solution by randomly sampling 1000 sets of the given size and compute the maximum across these samples. *This will always provide a lower bound for the "true" curve*.

Solving Eq. (κ-MFS) allows us to compute the optimal (k, η)-HiFi sets, since this captures the relation between η and k, i.e. the tradeoff between sparsity and accuracy. We observe that in many layers (at least 50% of the model), across multiple models, there are sets which contain at most 20% of components but have a subset fidelity of around 0.8. We also observe that as the difficulty of the task increases (CIFAR10 to ImageNet), fewer layers exhibit this sparsity, validating the assertion that networks trained on harder problems are less overparameterized.

Figs. 4 to 6 show this for a ResNet-50 trained on CIFAR10, CIFAR100, and ImageNet. Fig. 7 shows this for a OPT-125m model using 128 samples of WikiText and 100 random subsets instead of a 1000. Due to the expensive nature of this experiment, we are forced to use only 100 random subsets for each size, leading to a noisier curve. However, it is clear to see that the general trend continues to hold for several layers, especially for the "down-projection" weight matrices, which are the focus of our pruning algorithm for LLMs.

C.1.2 Counterfactual study of HiFi sets

We compare the effect of removing HiFi components from a layer with the effect of removing a random subset of the same size. For a ResNet-50 model trained on CIFAR-10, when around 22% of HiFi components are removed, the accuracy drops by around 70%, whereas removing a random subset of the same size decreases the accuracy by 32%. Note that there is a roughly 1% decrease in accuracy when only 22% of the non-HiFi components are removed. This indicates that components classed as "High Fidelity" have a significantly higher impact on the model's predictive performance than those with lower fidelity scores.

C.1.3 Robustness of the Fidelity Score

In this section, we perform ablations on the number of samples required for estimating the fidelity score and show how it reacts to additive noise on the model's weights.

In Figs. 8 and 9 we show how different data sizes affect different layers in a ResNet50 model trained on CIFAR10 and ImageNet, respectively. Each data size is selected over 3 random seeds with error bars shown. For clarity, we show only a subset of layers and provide plots and code to generate them.

We observe that the values are stable for 0.2%, 0.5% and 2% of data selected, indicating that it is robust to number of samples selected.

We also observe the effect of training in these graphs. In untrained models, almost all components have very small fidelity scores with a sharp increase for some values. This indicates that HiFi components are a function of training, with the well-trainedness of the network being a prerequisite for their presence

When investigating the effect of adding noise to the weights and its effect on accuracy and the fidelity score, we observe that adding zero mean noise of larger standard deviations, starting from 0.005 to 0.05, decreases the fidelity of components, with noisier weights behaving more like untrained models. We test this on a ResNet-50 trained on CIFAR10 and present the results in Fig. 10. Again, we present only a random subset of the layers for clarity.

C.2 Constants in Theorem 3.6

In this section, we provided worst case and average case estimates of the constant C_l in Theorem 3.6.

In Fig. 11, we plot the constants obtained in the proof of Theorem 3.6 in Appendix B.1 for a ResNet-50 trained on ImageNet, and observe that the values can be very large (10^{38}). However, it is important to note that these are worst case guarantees, and that these constants are much smaller in practice.

In Fig. 12, we compute the ratio $\frac{\mathbb{E}\left[\|\boldsymbol{y}(\mathrm{X})-\boldsymbol{y}(\mathrm{X};\boldsymbol{M}^l)\|^2\right]}{\sum_{c=1}^{c^l_{out}} d\left(\boldsymbol{Y}^l_c(\mathrm{X}),\sum_{i\in C} m^l_{ci}\boldsymbol{A}^l_{ci}(\mathrm{X})\right)}$ and observe that these values are indeed much smaller (10-50) for random values of $\boldsymbol{M}^l$ for the expected square loss.

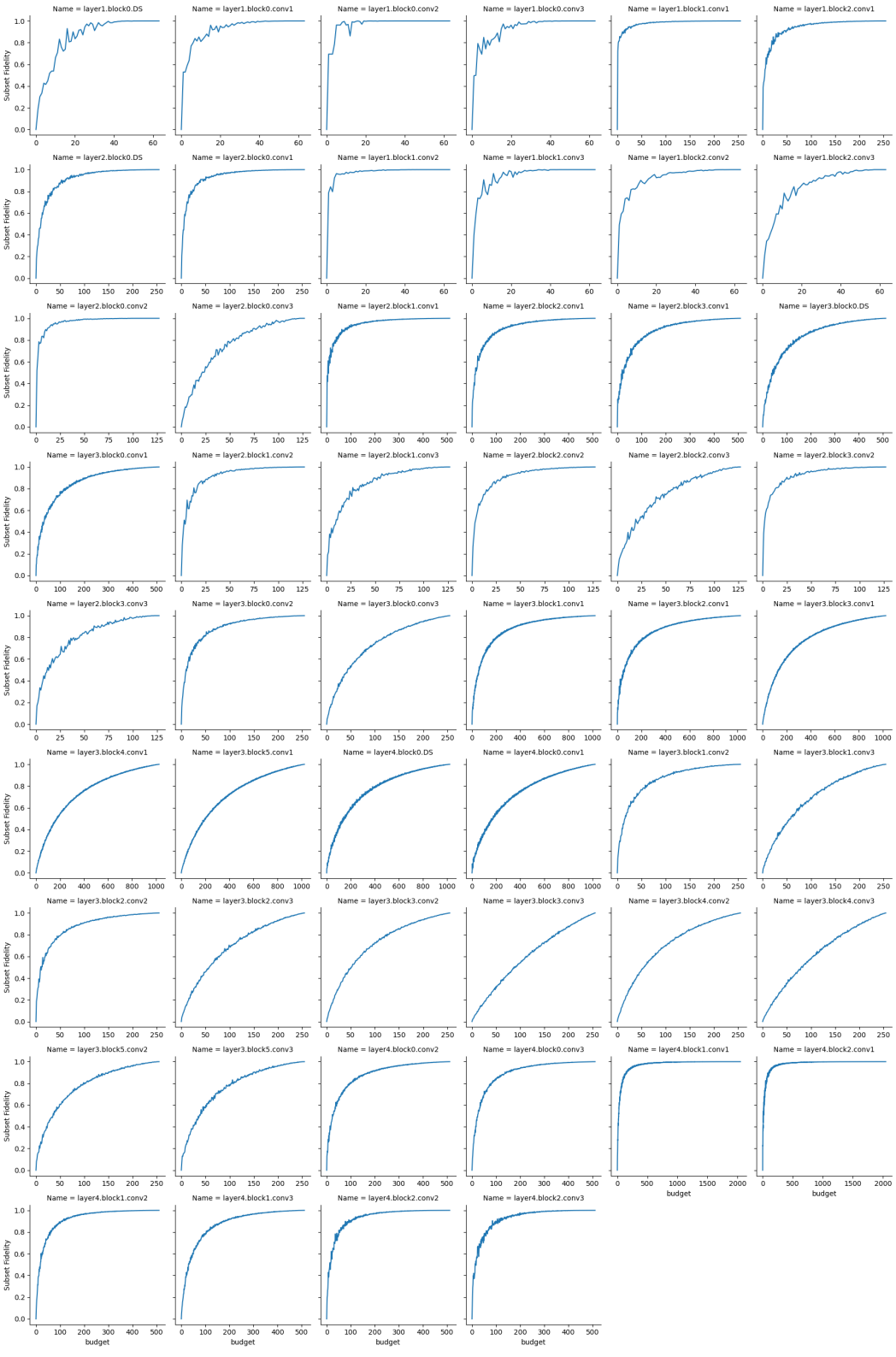


Figure 4: Estimates of Optimal subset fidelity for ResNet-50 on CIFAR10.

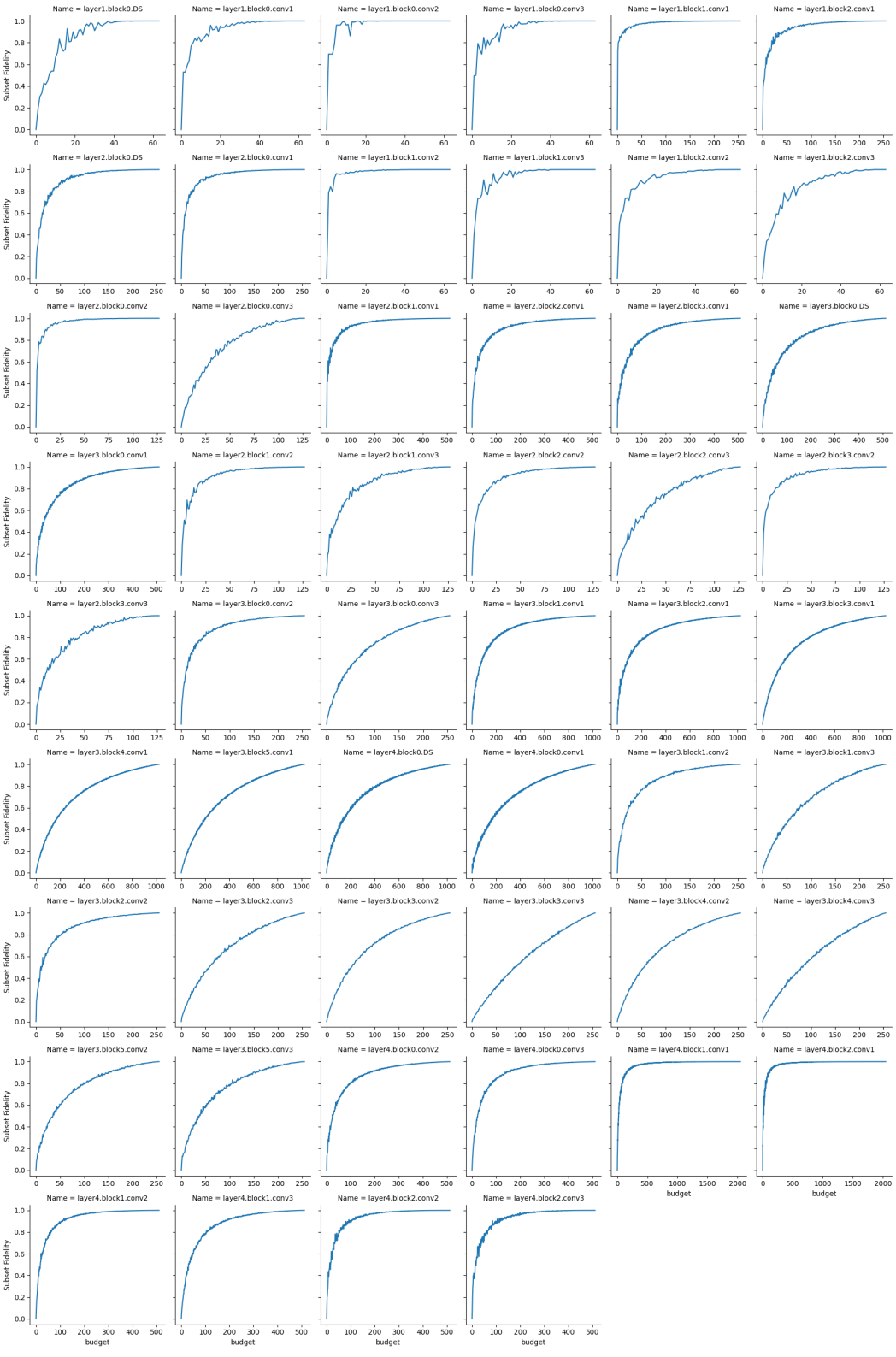


Figure 5: Estimates of Optimal subset fidelity for ResNet-50 on CIFAR100.

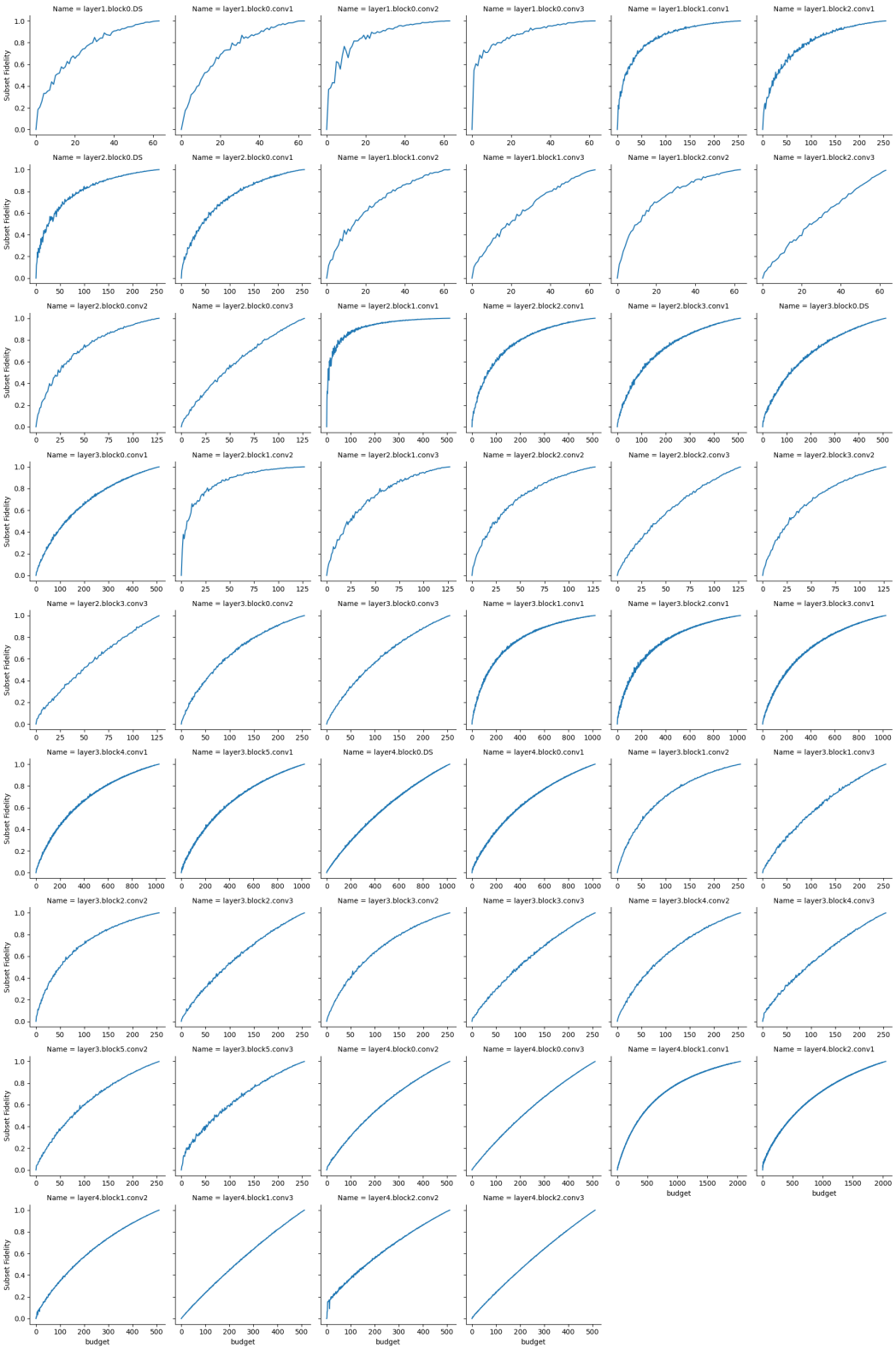


Figure 6: Estimates of Optimal subset fidelity for ResNet-50 on ImageNet.

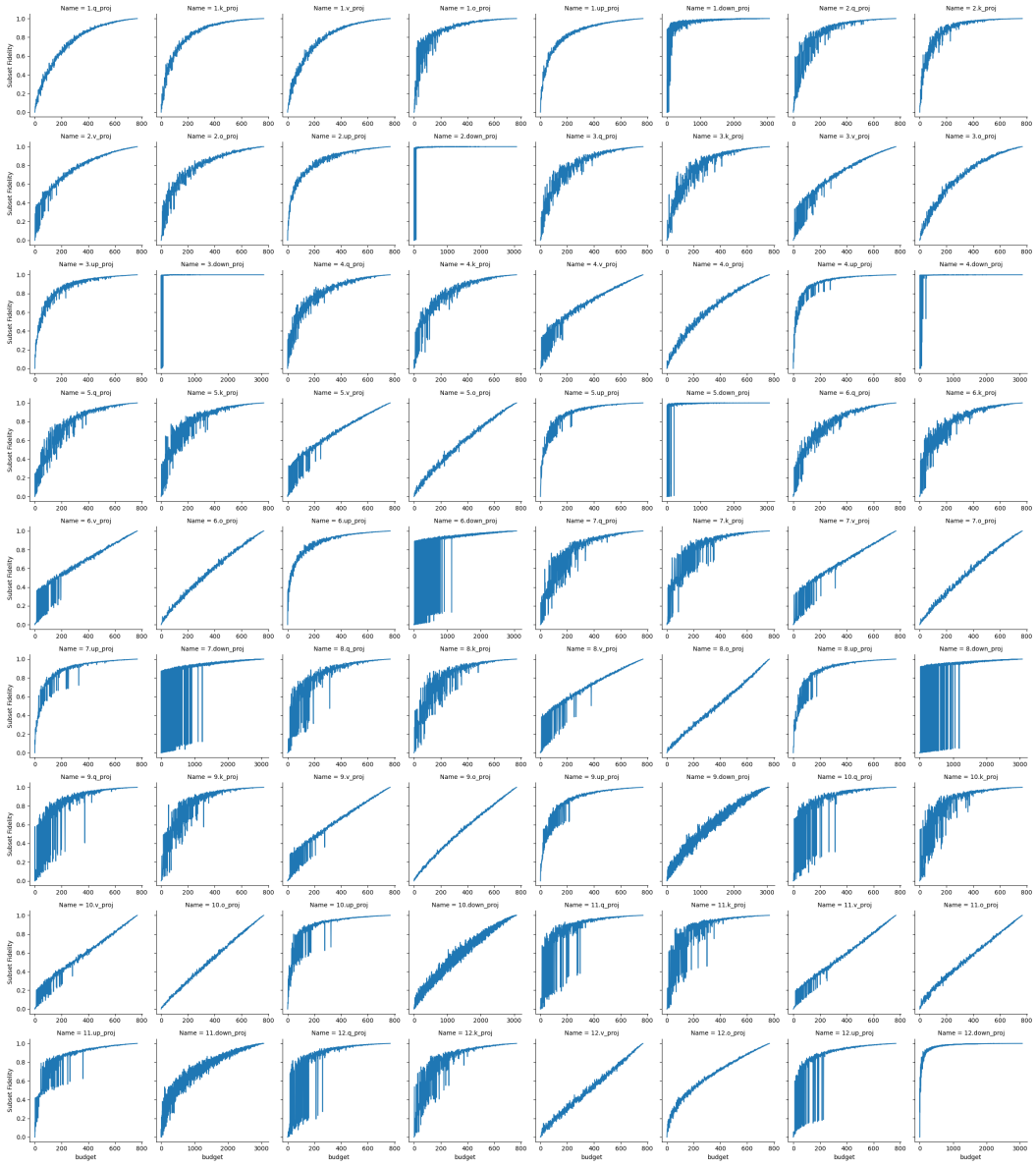


Figure 7: Estimates of Optimal subset fidelity for OPT-125M.

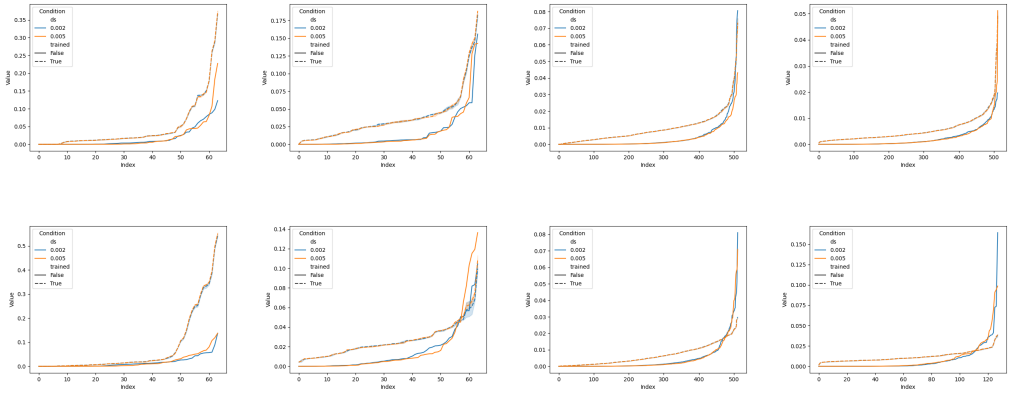


Figure 8: Fidelity scores for select layers of ResNet50 trained on ImageNet showing the effect of training and data set size.

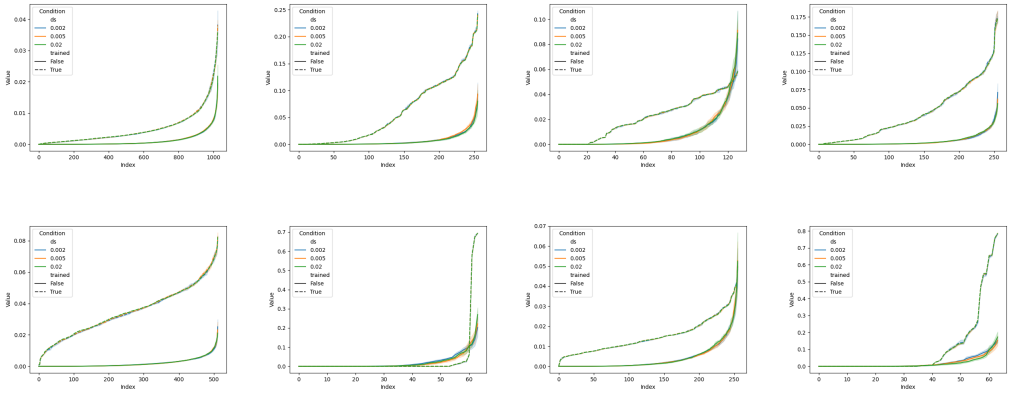


Figure 9: Fidelity scores for select layers of ResNet50 trained on CIFAR10 showing the effect of training and data set size.

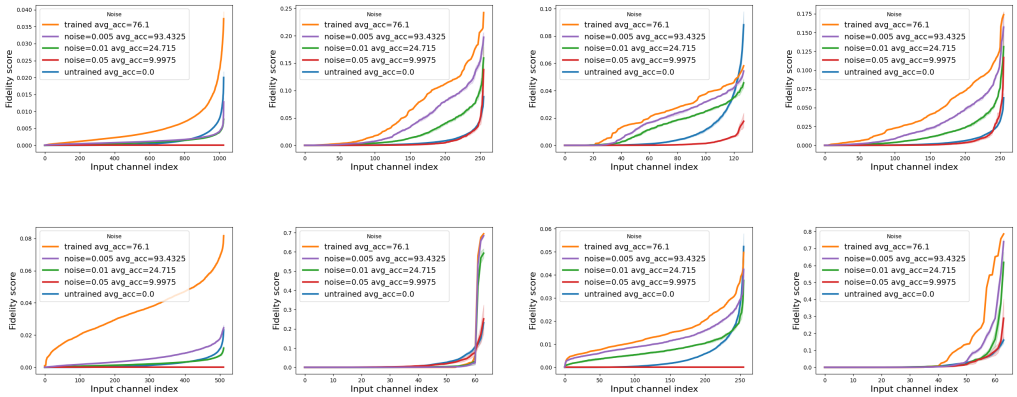


Figure 10: Fidelity scores for select layers of ResNet50 trained on CIFAR10 showing the effect of adding noise

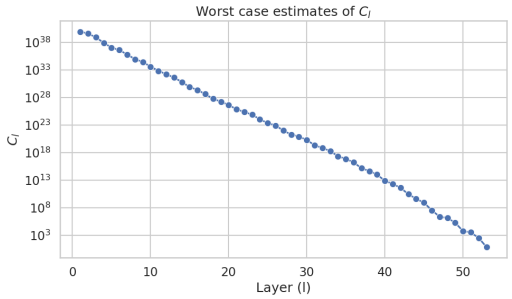


Figure 11: Worst case estimates of constants C_l for a ResNet-50 trained on ImageNet

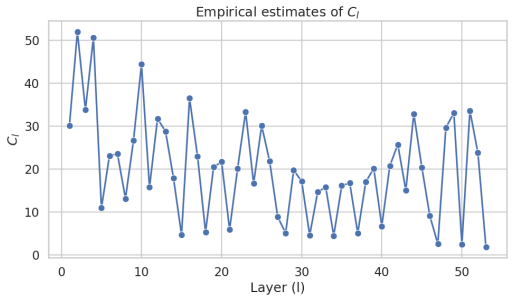


Figure 12: Empirical estimates of constants C_l for a ResNet-50 trained on ImageNet

C.3 Discussion on the synthetic samples used in the experiments

We describe the synthetic datasets used in our vision experiments to simulate distributional access. Randomly selected example images are provided in Fig. 13. For NLP tasks, we use WikiText and Alpaca datasets [53, 82] which are standard in this field.

C.3.1 CIFAR5M

For experiments with the CIFAR10 dataset, we use CIFAR5M, a dataset containing 6 million synthetic CIFAR-10-like images sampled from a Diffusion model and labeled by a Big-Transfer model [60], which we randomly sample 10,000 samples from each of the 10 classes to create our dataset. This dataset has an FID [28] of 15.95 with respect to the CIFAR10 training set. This dataset is obtained from here.

C.3.2 CIFAR100-DDPM

For experiments with the CIFAR100 dataset, we use CIFAR100-DDPM [24], which we randomly downsample to contain 1,000 samples from each of the 100 classes. This dataset has an FID of 4.74

(a) Automobile (5M) (b) Deer (5M) (c) Beaver (DDPM) (d) Forest (DDPM)

Figure 13: Randomly selected images from the synthetic sets

Table 5: Effect of data quality when pruning a ResNet-50 on CIFAR10

FID	Diffusion Steps	Accuracy	FLOP Reduction	Param Reduction
85.80	4	86.27	2.63x	2.66x
35.58	5	90.75	2.78x	2.78x
14.42	6	90.39	3.50x	3.60x

Table 6: Comparison of ResNet-50 pruning for CIFAR10 and CIFAR100. ST = *Synthetic Training*, i.e. training using synthetic samples.

Dataset	Algorithm	Accuracy	FLOP Reduction	Param Reduction
CIFAR10	Unpruned	94.99	1x	1x
	DFPC	90.25	1.46x	2.07x
	L_2	15.91	4.07x	4.71x
	L_2 w/ ST	90.12	4.07x	4.71x
	Ours	**91.02**	**4.07x**	**5.36x**
CIFAR100	Unpruned	78.85	1x	1x
	DFPC	70.31	1.27x	1.22x
	L_2	16.77	1.93x	1.40x
	L_2 w/ ST	**73.83**	1.93x	**1.40x**
	Ours	70.93	**1.93x**	1.38x

with respect to the CIFAR100 training set. We randomly sample 1,000 samples from each of the 100 classes to create our dataset. This dataset is obtained from here.

C.3.3 Effect of Data Quality

To study the effect of data quality on the performance of our algorithm in vision tasks, we apply the pruning algorithm using synthetic datasets based on CIFAR10 generated with different FIDs. We use a diffusion model [37] to generate 3 datasets of differing quality by changing the number of diffusion steps (4,5, and 6). We report the results of our pruning algorithm with different quality datasets in Table 5. We observe that higher quality data leads to an improved sparsity - accuracy tradeoff.

C.4 Additional Pruning Experiments

We present additional pruning experiments in Tables 6 and 7.

C.4.1 Ablation of weight compensation and BatchNorm correction

In this section, we perform ablations for each component of our pruning algorithm, simple pruning, correcting batch norm statistics and weight compensation. We report our results for pruning ResNet-

Table 7: Comparison of ResNet-101/VGG-19 pruning on CIFAR10 and CIFAR100. ST = *Synthetic Training*, i.e. training using synthetic samples.

Dataset	Model	Algorithm	Accuracy	FLOP Reduction	Param Reduction
CIFAR-100	VGG19	Unpruned	72.02	1x	1x
		DFPC	70.10	1.26x	1.50x
		L_2	56.46	1.50x	2.40x
		L_2 w/ ST	**72.42**	1.50x	**2.40x**
		Ours	70.26	**1.51x**	2.31x
CIFAR10	ResNet-101	Unpruned	95.09	1x	1x
		DFPC	89.80	1.53x	1.84x
		L_2 w/ ST	90.49	4.20	**5.29x**
		Ours	**91.20**	**4.21x**	4.79x
	VGG19	Unpruned	93.50	1x	1x
		DFPC	90.25	1.46x	2.07x
		L_2 w/ ST	89.23	2.39x	**9.19x**
		Ours	**91.80**	**2.39x**	5.52x

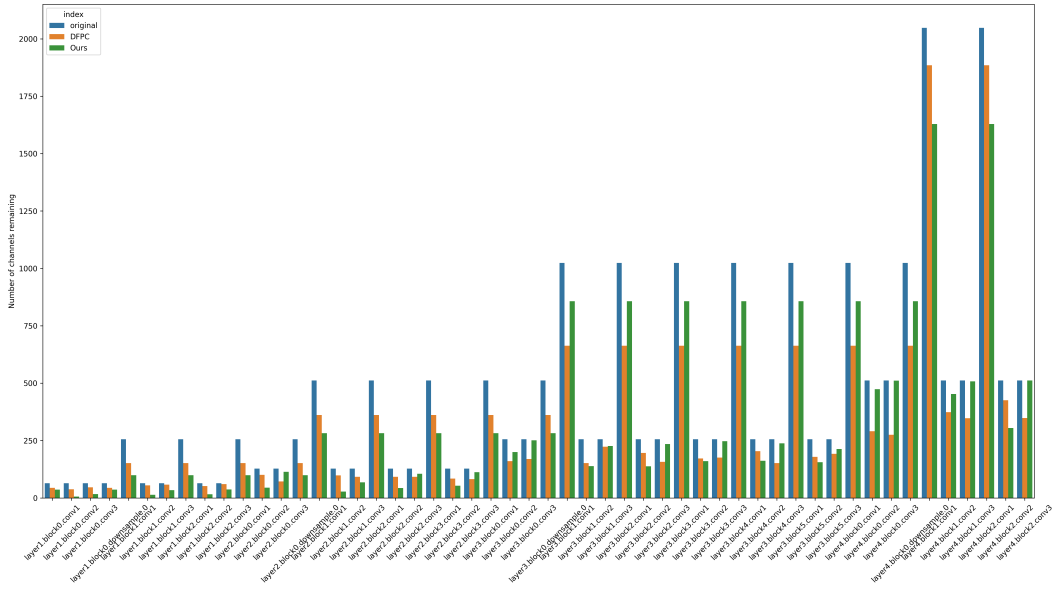


Figure 14: Number of remaining channels of pruned ImageNet model compared with DFPC (30)

50 on CIFAR 10 in Table 8. We observe that each component allows for a better accuracy sparsity trade-off.

C.4.2 Final ImageNet Pruned Model

In Figs. 14 and 15 we compare the final pruned models for ResNet-50 on ImageNet with DFPC [61]. We observe that our pruning algorithm removes more channels in later coupled channels than DFPC leading to higher gains in sparsity.

C.4.3 Baseline selection for LLM Pruning

We choose ShortGPT [52] and SliceGPT [2] as baselines against which we compare ModHiFi. We do so for two broad reasons: all three methods together represent three different granularities for conducting structured pruning for LLMs, and both ShortGPT and SliceGPT are the state-of-the-art within their respective lanes.

The three different granularities are

1. Layer pruning: Entire layers (i.e. transformer decoder blocks) are removed from the network. This is viable since transformers are constant width networks, i.e., there are no architectural restrictions to the ordering or number of layers. ShortGPT falls within this granularity.
2. Embedding pruning: The width of the network (i.e. the embedding dimension) is pruned at a uniform rate across the entire network. This entails a form of feature selection: along with weight matrix pruning, one also has to prune the corresponding dimensions from the feature matrix being fed into every layer. SliceGPT falls within this granularity.

Table 8: Ablation of different components

BatchNorm	Compensation	Accuracy	FLOP Reduction	Param Reduction
No	No	93.37	1.61x	1.53x
No	Yes	93.49	2.21x	2.17x
Yes	No	93.17	2.53x	2.39x
Yes	Yes	93.76	3.22x	3.30x

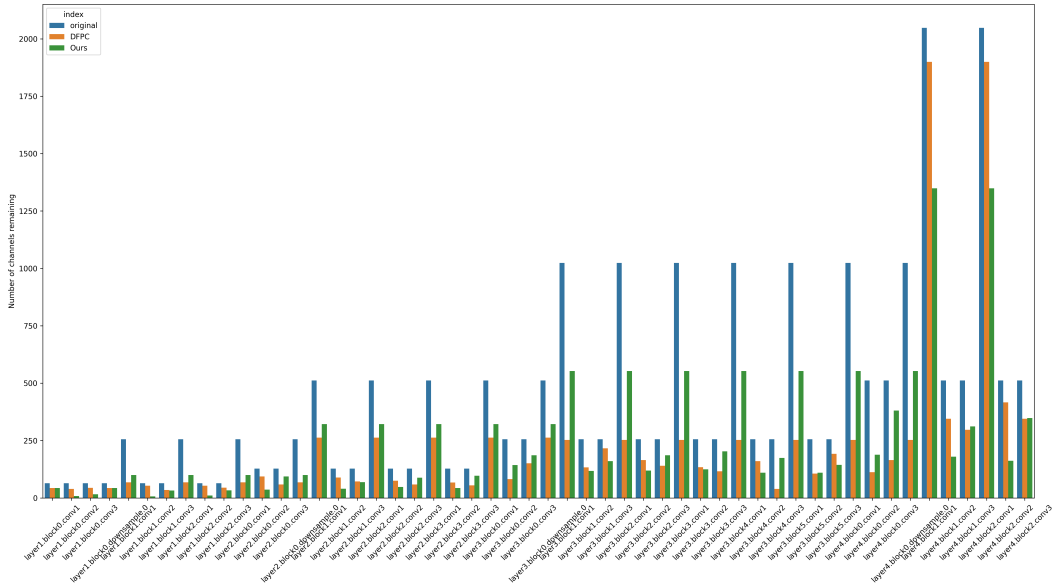


Figure 15: Number of remaining channels of pruned ImageNet model compared with DFPC (54)

3. Hidden dimension pruning: Here, the number of layers and the width of the embedding are left unchanged. Instead, one prunes the hidden dimensions within the modules that constitute a transformer decoder block. ModHiFi falls within this granularity.

We would like to emphasize that both SliceGPT and ShortGPT are designed to operate on Transformer models, and as such are able to leverage specifics of the architecture to their advantage. In return for this specificity, however, they trade off the ability to generalize to CNNs, something that ModHiFi does with ease due to its architecture-agnostic nature; the only assumption made by the Fidelity Score is that the components being scored belong to linear layers.

C.5 Additional Unlearning Experiments

We report additional experiments in Table 9 on class unlearning on different architectures. For VGG-19 networks, we remove the HiFi channels for the forget class of the last 12 convolution layers. We also compare our work with DisCEdit-U from [58] wherein we remove discriminative components from the last 8 convolutional layers. We use a custom implementation of the algorithm for our VGG19 and ResNet50 models for CIFAR10, as those models are unavailable in the codebase of [58].

We also compare our work with DisCEdit-U on ResNet50 trained on CIFAR10 as well, which we present in Table 10

We show that our unlearning method achieves similar or superior performance to that of [58] without fine-tuning. Moreover, unlike [58], our approach uses only *synthetic samples*, showing the efficacy of our work in classwise unlearning, even in the absence of training data.

Unlearning with finetuning Here we compare our method with 3 additional epochs of finetuning on synthetic samples of the remaining class data. Although this setup does not fall into the setup of

Table 9: Class unlearning on CIFAR10 for VGG19

Model	Algorithm	Forget Accuracy	Remain Accuracy
VGG19	-	93.50	93.50
	DisCEdit-U	2.39	**84.2**
	Ours	**0.86**	77.85

Table 10: Class unlearning on CIFAR10 for ResNet50

Model	Algorithm	Forget Accuracy	Remain Accuracy
ResNet50	-	94.99	94.99
	DisCEdit-U	3.2	91.6
	Ours	**0.2**	**92.98**

Table 11: Class unlearning with 3 epochs of finetuning on synthetic samples

Model	Remain Accuracy	Forget Accuracy
ResNet-50	93.1	0
Swin-T	83.6	0.1

the work since we do not assume access to the loss function, we provide these results to indicate that even using very few synthetic samples we can perform perfect unlearning. We present these results in Table 11, where we observe almost perfect unlearning for both ResNet-50 and Swin-Transformers.

Unlearning with baseline budgets In this section, we compare our method when allowing for the same amount of finetuning as [35], with both synthetic data and training data access. While this violates our assumptions about loss function and training data access, we present these results to provide a fair comparison of our algorithm when run within the same constraints as our baselines. Our results can be found in Table 12 for the Swin Transformer.

C.6 Compute Platform

Implementation Details We implement our proposed methods in PyTorch [64] and use Huggingface's `transformers` [91] for LLM implementations.

Inference time measurements We follow the inference time measurement setting of [61, 72]. Inference time is the time taken for a model to compute the forward pass for an input and does not account for loading data into memory. We compute the inference time for a batch of 640 random tensors for GPU and 64 for CPU. 100 iterations are used for warm up, after which the inference time is averaged over the next 1000 forward passes. We compute CPU and GPU measurements on a machine whose specifications can be found in Appendix C.6.

JIT Compilation We present inference time numbers with JIT compilation on Pytorch [63].

Hardware Table 13 details the hardware we use to conduct our experiments. Values in (*) indicate reported values obtained from `https://www.amd.com/en/products/accelerators/instinct/mi200/mi210.html`. This machine runs Ubuntu 22.04.3 LTS with kernel 6.8.0-40-generic with the hardware in Table 13. Our software stack comprises of Python 3.12.8, PyTorch 2.5.1 built for ROCm 6.2, and torchvision version 0.20.1 built for ROCm 6.2.

Inference times are measured on a machine running Ubuntu 20.04.1 LTS with kernel 5.15.0-91-generic on the hardware specified in Table 14. The software stack used for inference consists of Python 3.12.8, PyTorch 2.5.1, and Torchvision 0.20.1 for CUDA 12.3.

Table 12: Class unlearning with 10 epochs of finetuning

Approach	Dataset	Forget Accuracy	Remain Accuracy
Jia et al. [35]	CIFAR10 Train	1.20	90.69
Ours	CIFAR10 Synthetic	0.37	84.63
Ours	CIFAR10 Train	**0.00**	**91.1**

Table 13: Specifications of GPU hardware used for computation

CPU Model Name	AMD EPYC 9654 96-Core Processor
CPU(s)	192
Thread(s) per core	1
Core(s) per socket	96
Socket(s)	2
NUMA node(s)	2
CPU MHz(Max)	3707.8120
L1d & L1i cache	6 MiB
L2 cache	192 MiB
L3 cache	768 MiB
RAM	1.48 TiB (DDR5, 4800 MT/s)
GPU Model name	Instinct MI210
GPU(s)	4
GPU Architecture	AMD Aldebaran
Dedicated Memory Size(per GPU)	64 GB
ROCm Version	6.0.2
Peak FP32 Performance*	22.6 TFLOPs
Peak FP64 Performance*	22.6 TFLOPs
Memory Clock*	1.6 GHz
Peak Memory Bandwidth*	1.6 TB/s

Table 14: Specifications of GPU and CPU hardware used for computing inference time

CPU Model Name	Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz
CPU(s)	64
Thread(s) per core	2
Core(s) per socket	16
Socket(s)	2
NUMA node(s)	2
CPU MHz(Max)	3200
L1d & L1i cache	1 MiB
L2 cache	32 MiB
L3 cache	44 MiB
RAM	62.53 GiB (DDR4, 2666 MT/s)
GPU Model name	NVIDIA GeForce RTX 2080 Ti
CUDA version	12.3
GPU(s)	8
GPU Architecture	NVIDIA Turing
Dedicated Memory Size(per GPU)	11.81 GB

C.6.1 Module-level Time Consumption

In this section, we break down the time each component of our algorithm takes. For 2000 samples batched into batches of size 64, when running the algorithm on a ResNet-50:

- Computation of fidelity scores takes between 32GB to 51GB of VRAM, and between 2 minutes to 5 minutes, on 1 GPU of machine 13, across data from CIFAR10, CIFAR100, and ImageNet.

- Computing $\delta_c^\star$ across 4 GPUs using an average of 60GB per GPU takes 60 minutes for CIFAR10/100, and 90 minutes for ImageNet, averaging to roughly 1 minute per layer.

C.7 Hyperparameters and Training Procedure

C.7.1 Hyperparameters for Experiments

We typically set the percentile of removed components to be between 0.01 to 0.2. We randomly select 2% of our synthetic samples to select data for vision tasks and select 128 samples for NLP tasks.

C.7.2 Training procedure

Pretraining procedure: For CIFAR10 and CIFAR100, we train models using SGD with a momentum factor of 0.9 and weight decay of 5×10^{-4}, for 200 epochs using Cosine Annealing step sizes with an initial learning rate of 0.1.

ImageNet post training: For ImageNet, we use off-the-shelf pretrained models from Torchvision [63]. We train the model for 3 epochs after each iteration of pruning with learning rates of 0.1, 0.01, 0.001. After the pruning ends, we finally train the network for 160 epochs with a batch size of 512. We use the SGD Optimizer with a momentum factor of 0.9 and weight decay of 1×10^{-4} and start with an LR warm-up for 10 epochs, followed by Cosine Annealed step sizes with an initial learning rate of 0.1 with Cutmix and Mixup augmentations.

L_2 Post training procedure: For the synthetic training experiments mentioned in Section 5, we first prune the model using L_2 norm as the grouped saliency to a similar sparsity as our algorithm. We then train the model using 50000 samples from the synthetic dataset for 100 epochs with a batch size of 128 using SGD optimizer with momentum factor of 0.9 with initial learning rate of 0.01 and a MultiStepLR learning rate scheduler with milestones at 60 and 80 epochs.

D Additional Algorithm details

In this section, we discuss algorithmic nuances not discussed in the main body of the paper.

D.1 Clarification on Lipschitz Bounds and Their Role

In Section 3, we introduced a local-to-global error bound (Theorem 3.6) that connects intermediate-layer deviations to changes in the final-layer output, assuming the model is composed of Lipschitz-continuous layers with constants C_l. This result serves to theoretically motivate the use of local reconstruction error – what we formalize as Subset Fidelity – as a proxy for reconstruction error at the output.

Importantly, we do not compute or estimate Lipschitz constants in any part of our algorithm. Our pruning and unlearning algorithms do not depend on knowledge of the values of C_l. The bound in Theorem 3.6 is used qualitatively to support the intuition that preserving high-fidelity intermediate representations leads to stability in the *final* model predictions.

Empirically, we find that Subset Fidelity correlates strongly with the effect of component removal on prediction quality (see Fig. 1 and Appendix C), even in the absence of explicit Lipschitz bound estimation. This supports our design choice to treat Theorem 3.6 as a motivating principle, not an operational tool.

We believe this distinction is important to clarify: while our framework draws conceptual inspiration from Lipschitz continuity, it remains loss-free, and hyperparameter-driven in practice, with no reliance on any difficult-to-estimate constants.

D.2 Additional Algorithmic details on Fidelity Estimation

To efficiently estimate the fidelity of each component at a given layer, we use a saliency measure to approximate the fidelity score. This is based on the component's contribution to reconstructing the layer's output, computed via the inner product between the layer output and the component-specific activation contribution. This can be written as

$$\tilde{R}^l_{ci} = \mathbb{E}\left[\langle \boldsymbol{Y}_c(\mathrm{X}), \boldsymbol{A}_{ci}(\mathrm{X})\rangle\right] = \langle \boldsymbol{Q}^c_i, \mathbf{1}\rangle = \alpha^l_{ci}\mathbb{E}[||\boldsymbol{A}_{ci}||^2]$$

In networks that include **BatchNorm** (e.g., ResNets [27], VGG [76]), we refine this reconstruction by centering the activations using the BatchNorm's stored running mean. This leads to a modified formulation of the component similarity matrix:

$$\tilde{Q}^c_{ij} = \mathbb{E}\left[\langle \boldsymbol{A}_{ci}(\mathrm{X}), \boldsymbol{A}_{cj}(\mathrm{X})\rangle\right] - \langle \mathbb{E}[\boldsymbol{A}_{ci}(\mathrm{X})], \mathbb{E}[\boldsymbol{A}_{cj}(\mathrm{X})]\rangle$$

These quantities are computed efficiently using modern GPU architectures. The forward activations from a calibration set are batched and evaluated across multiple GPUs in parallel. In sequence-based architectures such as Transformers [49], we compute the expectation over all elements in the sequence dimension.

Numerical Stability and Regularization. To compute the optimal linear compensation for modifying components, we solve a least-squares system involving the component similarity matrix $\tilde{Q}^c$. However, this matrix may be ill-conditioned or rank-deficient in practice. To ensure numerical stability and avoid inversion errors, we add a small ℓ_2 regularization term ($\lambda = 10^{-4}$) to the diagonal before solving.

Behavior of HiFi Components During Editing. The role of HiFi components depends on the editing task:

- **Structured Pruning:** We retain HiFi components and discard the rest. While we cannot guarantee a fixed sparsity level in the output model (since HiFi components may span all inputs), we observe in practice that reasonable sparsity emerges naturally. For more aggressive pruning, the algorithm is applied iteratively.
- **Class Unlearning:** Simply discarding low-fidelity components is insufficient. Instead, we aim to remove or disrupt the influence of HiFi components that are specific to the forget class. The editing strategy depends on the network type:
  - In BatchNorm networks, we zero out the weights of HiFi components computed as per the forget class samples.
  - In LayerNorm-based networks with residual connections (e.g., Swin-T), we negate the weights of HiFi components. This rotates the forget-class representation in the opposite direction due to the residual path.

The unlearning strategy for Transformer-based architectures is captured in the following procedure:

Algorithm 2 ViT-Edit-X: Structured Editing for Transformers

Require: Model parameters θ, HiFi components H, coupled channels CC
Ensure: Edited model parameters $\hat{\theta}$
if $X = \texttt{Prune}$ **then**
  for $i \in [c^l_{\text{in}}] \setminus \{i \mid (c,i) \in \bigcup_l H_l\}$ **do**
    $\boldsymbol{W}^l_{c,i} \leftarrow 0$ $\quad \forall c \in [c^l_{\text{out}}],\ l \in CC$
else if $X = \texttt{Unlearn}$ **then**
  for each layer $l \in CC$ **do**
    $\hat{W}^l_{c,i} \leftarrow -W^l_{c,i}$ $\quad \forall (c,i) \in H_l$
Return: $\hat{\theta}$

Fidelity Estimation in LLMs Due to the large scale of LLMs and the range of floating point values, estimation of scores becomes more challenging. We estimate the the fidelity scores by computing the row norms of the regularized Cholesky decomposition of $\boldsymbol{Q}$. The scores are estimated as

$$\mathrm{FS}(\{i\}) \approx ||\boldsymbol{L}_i||^2 \ \text{ where } \ \boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^\top \ \text{ is the Cholesky decomposition of } \boldsymbol{Q}$$

We use the Cholesky decomposition since it is efficient to compute.

D.3 Computational cost

Let N be the number of data points used to estimate the saliency and M^l be the complexity of computing the input contribution at layer l for a single sample in a set of coupled channels with m layers. The complexity to compute the set of retained channels for an output channel of a layer is, $t^l_{sal} = O(NM^l C^l_{in} d^l)$. To select the components for the coupled channels, the top p elements for each layer and output channel in them are collected, this costs $O(\sum_{l=1}^{m} C^l_{out}(C^l_{in} \log C^l_{in} + t^l_{sal}))$. The algorithm shows a linear dependence on the number of layers in the network, compared with the BGSC algorithm [61] which has a quadratic dependence.

E Full LLM Disclosure

In this work, we use LLMs to clean up the grammar and make improvements to the language to enhance clarity. We use LLMs to generate code that generates some of the plots presented in this work. We do not use LLMs to generate code for any of the algorithms presented in this work. We also use LLMs to simplify the proof presented in Lemma B.3.