Inducing Semi-Structured Sparsity by Masking for Efficient Model Inference in Convolutional Networks

David A. Danhofer
Department of Computer Science
ETH Zürich
ddanhofer@ethz.ch

Abstract

The crucial role of convolutional models, both as standalone vision models and backbones in foundation models, necessitates effective acceleration techniques. This paper proposes a novel method to learn semi-structured sparsity patterns for convolution kernels in the form of maskings enabling the utilization of readily available hardware accelerations. The approach accelerates convolutional models more than two-fold during inference without decreasing model performance. At the same time, the original model weights and structure remain unchanged keeping the model thus easily updatable. Beyond the immediate practical use, the effect of maskings on prediction is easily quantifiable. Therefore, guarantees on model predictions under maskings are derived showing stability bounds for learned maskings even after updating the original underlying model.[1]

1 Introduction

The increasing complexity of deep learning models [21], their deployment in applications [5], and the adoption of reflection incurring several inference passes per query, e.g., as in the O1 models from the GPT family [3], shifts the relative amounts of resources spent during the model lifetime from the training to the inference stage [7, 35]. It therefore becomes imperative to make models more efficient [46]. One way of achieving this is by spending a comparatively small, additional share of resources during training to learn a one-time modification of the model that lowers the model's inference and thus lifetime cost [30, 40]. First and foremost, such a modification is effective if it decreases the model's computational and time cost at a relatively low additional training overhead while not affecting the prediction performance of the model negatively [22]. Additionally, there are other desirable properties of such one-time modifications: From an application perspective the achievable gain in efficiency is only useful if it can be leveraged easily, a well-known challenge, e.g., with sparsifying models [8, 15]. Taking into consideration the increasing popularity of large, expensive to train, foundation models [16] or models employed in an online setting subject to continuous updates the proposed change should not affect the possibility to update the model, e.g., by changing the weights or architecture underlying the model. Ideally, if such a model is updated, the learned modification can even be reused under the constraint of the magnitude of change imposed by updating the model.

Semi-structured sparse maskings satisfy the above properties by replacing the dense matrix operations usually required during inference by cheaper and faster operations on semi-structured sparse matrices [4]. While many works have demonstrated that sparse (pruned) submodels can solve the same task at almost no loss of performance [2, 26] the sparsity of the models does not necessarily have to adhere to a specific pattern making it difficult to turn theoretically obtained computational speedups by saving on data loading and computational operations into practical efficiency gains [14]. Regular

[1] Code available at github.com/ddanhofer/Semi-Structured-Sparsity-CNNs

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

patterns are more "machine-friendly" inducing the desired efficiency *a priori* but limiting the choices for the sparse patterns, which thus need to be chosen carefully with the goal of minimizing the loss of inference performance in mind.

This paper proposes a novel method of learning regularly sparse masking patterns for convolutions, key building blocks for state-of-the art Computer Vision (CV) models [25] and foundation models building on CV models as their backbone [38]. The proposed method

- shows how to effectively use readily available hardware accelerations for semi-structured sparse matrices in convolution kernels to accelerate inference,
- outperforms available heuristics for semi-structured sparsity showing that semi-structured sparsity masks can be learned with a fraction of the original training resources while incurring a negligible performance loss in CV classification tasks,
- provides the additional advantage of not changing the original set of trained weights keeping models updatable and rendering the method especially attractive for use in large models, e.g., foundation models and in online settings,
- induces an easily quantifiable change to the model's prediction behavior and thus lends itself to settings where hard guarantees on model predictions are of interest.

In the following section the adoption of semi-structured sparsity and sparsity in convolutional models are addressed. Section 3 of the paper covers modeling semi-structured sparsity in general, in convolutional models, and the theoretical implications of such model alterations in inference. The results of empirically testing the method on widely used convolutional architectures are presented in Section 4 followed up by a discussion of the method presented and a conclusion.

2 Related Work

In the following the notion and adoption of semi-structured sparsity is introduced. Then, the implications on prediction performance and the computational challenges of sparsifying Convolutional Neural Networks (CNNs) are highlighted.

Semi-structured sparsity Semi-structured sparsity to accelerate network inference has been introduced in [37] as *N:M-sparsity* requiring N out of M elements of a contiguous block to be zero (s. a. 3.1). Beyond the general case of N:M sparsity, the practically interesting special case of 2:4 sparsity has been considered in more detail [17, 18] in which exactly half of the weights are pruned as illustrated in Figure 1. This setting enables hardware acceleration via NVIDIA sparse tensor cores available from the NVIDIA Ampere architecture on via the TensorRT v8.0 library [36]. Since half the elements are zeroed out and thus negligible, the amount of data to load from memory is almost halved with the number of Floating Point Operations (FLOPs) needed to conduct an operation on the sparse matrix also decreasing, e.g., linearly for addition and element-wise operations and super-linearly for multiplication, decomposition etc. [44]. This way 2:4 sparse matrix operations compute the same effective operation while reducing the time needed by a factor of two [36]. The difficulty in turning a dense matrix into a 2:4 sparse matrix, however, lies in selecting the most useful two of the four weights in each quadruple. To this end [37] propose a permutation regime that allows for preserving the weights based on magnitude and assess the found pattern via a scoring mechanism, the efficacy score. The functionality is available via NVIDIA's Apex library [33]. Notably, pruning via Apex requires finetuning the network again after pruning to achieve an inference performance comparable to that of the dense network in CV tasks, e.g., classification [36, 37], and therefore changes the original pretrained weights.

Sparsity in CNNs The state-of-the-art performance of CNNs in image-based and other tasks comes at the cost of a large memory footprint and computational cost. Pruning to obtain sparse networks is therefore a popular technique to decrease the computational and storage cost of deep neural networks [2]. Pruning techniques include magnitude-based pruning [12], iterative pruning [42], and dynamic pruning [24]. Although theoretically any sparsity reduces the computational costs of such networks, irregularity in the sparsity patterns makes it difficult to map the required computations to (parallel) processor operations [19]. Even extremely high levels of (irregular) sparsity, i.e., $> 97\%$, often yield no inference acceleration suffering from lack of library support [8, 41]. As visualized in Figure 2, in

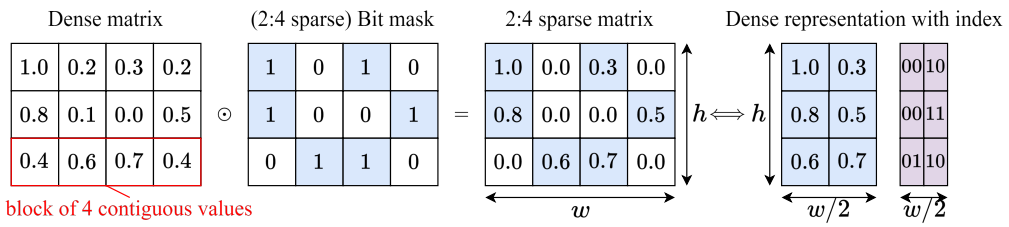

Figure 1: A 2:4 sparse matrix of floating point values – obtained from a dense matrix and a sparse bit mask – and its equivalent structured representation containing only the non-zero entries and a 2-bit index preserving the structure taking up roughly only half the space.

the case of CNNs different granularities of sparsity emerge naturally with more regular patterns being more "machine-friendly" effectively inducing smaller, still dense models [14]. Structured pruning approaches pruning filters or even entire channels at once [23, 31, 32], however, quickly deteriorate prediction performance [10, 28, 29]. This motivates semi-structured sparsity, a fine-grained yet structured sparsity pattern, to maintain a large degree of freedom in the selection of sparsity patterns to not impede performance while also observing some (machine-)usable regular structure.

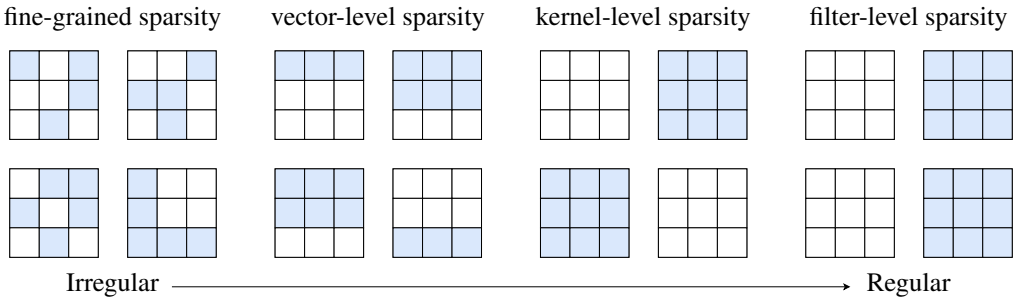

Figure 2: Different levels of granularity in a 4D-tensor as used in 2D-convolutions of multi-channel inputs with several filters; although the same number of weights is retained and pruned across all levels of structure the ease of processing increases as structure increases but limits the number of possible patterns at the same time.

3 Methods

In the following, the concept of modeling semi-structured sparsity in a network architecture and its effects on classification are introduced. Then, its application to convolutions is detailed out.

3.1 Modeling semi-structured sparsity

N:M sparsity divides a matrix into non-overlapping blocks of M contiguous elements requiring that N of these elements be zero as these elements can subsequently be ignored in many matrix operations, e.g., addition or multiplication. There are exactly $n = (M - N)!/N!$ ways of choosing N of the M elements without replacement and ignoring permutations, yielding n unique sparsity patterns. Selecting one of the n patterns can be modeled via a categorical variable z with class probabilities $\pi_1, ..., \pi_n$ s.t. each probability denotes the probability of selecting the corresponding N:M sparsity pattern. Sampling the choice vector z, a n-dimensional one-hot vector on the simplex Δ^{n-1}, from such a categorical distribution can be performed efficiently via the Gumbel-Max trick [11]

$$z = \text{onehot}(\arg\max_i [g_i + \log \pi_i]) \tag{1}$$

where $g_i \sim \text{Gumbel}(0, 1)$. Aggregating all n N:M sparsity patterns as column vectors in a pattern matrix $D \in \{0, 1\}^{N \times n}$ allows for constructing the (row major) semi-structured sparse mask M of

dimensions $h \times Nw$ with $h, w \in \mathbb{N}$ from one-dimensional block entries sampled from independent categorical distributions.

$$M = \begin{pmatrix} b^T & \cdots \\ \vdots & \ddots \end{pmatrix} \in \{0,1\}^{h \times Nw} \quad \text{where} \quad b = Dz \in \{0,1\}^N,\ z \sim \text{Categorical}(\pi_1, ..., \pi_n) \quad (2)$$

Since the $\arg\max$ operator employed to sample z according to (1) is discrete and thus not differentiable this method cannot be used directly in a neural network to optimize the parameters $(\pi_i)_{i \in [n]}$ via backpropagation. Instead a differentiable approximation is constructed by replacing the $\arg\max$ operator with a softmax. The choice vector z can now be drawn as follows

$$z_i = \frac{\exp((g_i + \log \pi_i)/\tau)}{\sum_k \exp((g_k + \log \pi_k)/\tau)} \quad (3)$$

yielding a Gumbel-Softmax (GS) distribution [20] over the n choices additionally parameterized by the temperature parameter τ. While not identical to a categorical distribution, the GS distribution approximates a categorical distribution over the choices for small temperature values, e.g., $\tau = 0.1$. This distribution allows for expressing the gradient as a deterministic function of the choice weights π and an independent source of random noise and thus gradient propagation through the stochastic node [20]. By updating the class probabilities in the distribution in respect to the classification objective the choice weights can be optimized for the classification task and the optimal choice is selected. After convergence the choice weights are frozen and the inference bit mask is obtained by sampling the blocks one final time from the GS distributions yielding the sparse bit mask. An element-wise multiplication of the bit mask with the dense weight matrix results in the desired semi-structured sparse matrix and, by extension, CNN.

3.2 Effects of maskings on classifier class predictions

Understanding how sparsity-inducing maskings affect a classifier's predictions is crucial to effectively trade off inference acceleration and potential performance inhibitions. The following Lemma contains the classifier definition and states a useful property. This definition is used in all subsequent theoretical results and models the architectures considered for experiments closely. All proofs in this section are deferred to Appendix A.

Lemma 3.1. *Let $f(x) = (\textit{softmax} \circ f_d \circ ... \circ f_1)(x)$ be a compositional classifier of depth d with $f_i(x) = \sigma(W_i x + b_i)$ predicting the class probabilities of an input sample $x \in X$ across c classes. Let σ be a non-linear element-wise activation function and L-Lipschitz. Then $f(x)$ is L_f-Lipschitz with $L_f \leq L^d \prod_i \|W_i\|$. Let such a classifier be a compositional (L_f-)Lipschitz classifier.*

Let the labels be one-hot elementary vectors $e_i \in \mathbb{R}^c$ and let $\lambda(\tilde{y}) = \min_i \|\tilde{y} - e_i\|_\infty$ be the function discretizing the classifier's probability prediction $\tilde{y} \in \mathbb{R}^c, \sum_i \tilde{y}_i = 1, 0 \leq \tilde{y}_i \leq 1\ \forall i$ into a class prediction. Given a prediction $\tilde{y} \in \mathbb{R}^c$ by a classifier $f(x)$ on a sample $x \in X$ let the *confidence* $0 \leq \gamma_f(x) \leq \frac{1}{2} + \epsilon, \epsilon > 0$ of a classifier be defined as the minimum change $\min_{\delta \in \mathbb{R}^c} \|\delta\|_\infty$ s.t. $\tilde{y} + \delta$ is a valid probability vector and $\lambda(\tilde{y}) \neq \lambda(\tilde{y} + \delta)$. Intuitively, this minimum change vector shifts the probability mass from the class with the highest probability to the class with the second highest probability to change the discretized class prediction $\lambda(\cdot)$ with a minimal shift. Let W be any weight matrix in a classifier and ΔW an additive perturbation of the weights yielding $W' = W + \Delta W$. Let a classifier be *stable* in respect to a perturbation and a given sample x if the perturbation doesn't affect the classifiers prediction on x, i.e., $\lambda(f_W(x)) = \lambda(f_{W'}(x))$.

Lemma 3.2. *Let W be any weight matrix in a compositional (L_f-)Lipschitz classifier $f(x)$ and ΔW an additive perturbation of the weights yielding $W'_j = W_j + \Delta W$ and a perturbed compositional classifier $f'(x)$. Then $f'(x)$ is $L_{f'}$-Lipschitz with $L_{f'} \leq L_f + L \|\Delta W\| \prod_{i=1, i \neq j}^{d} \|W_i\|$.*

A masking of a matrix W, i.e., zeroing out some (or all) of the entries, can be modeled as an element-wise product of the matrix W with a bit mask B. However, any masking can always equivalently be described as an additive perturbation. Let $\mu(B, W)$ be the masking function yielding this additive perturbation.

Lemma 3.3. *For any bit mask B and a matrix W the additive perturbation $\Delta W = \mu(B, W)$ s.t. $W + \Delta W = B \odot W$ always exists and fulfills $\|\Delta W\|_\infty \leq \|W\|_\infty$. The bound is tight.*

The model of a compositional classifier from Lemma 3.1 yields the following result guaranteeing stability of the classifier as a function of the perturbation and prediction confidence.

Lemma 3.4. *Let W_j be the j-th weight matrix in a compositional Lipschitz classifier $f(x)$ and ΔW_j an additive perturbation of the weights $W'_j = W_j + \Delta W_j$. Then the classifier is guaranteed to be stable in respect to such a perturbation and a sample x predicted with a confidence of $\gamma_f(x) > L^d \, \|\Delta W\| \, \|x\| \prod_{i=1, i\neq j}^{d} \|W_i\|$.*

Combining the statements made in Lemma 3.4 above with the reformulation of maskings of weights as additive perturbations from Lemma 3.3 yields the following result left without proof.

Lemma 3.5. *Let W_j be the j-th weight matrix in a compositional Lipschitz classifier $f(x)$ and $\Delta W_j = \mu(B, W)$ an additive perturbation of the weights $W'_j = W_j + \Delta W_j$ induced by any masking B. Then the classifier is guaranteed to be stable in respect to such a perturbation and a sample x predicted with a confidence of $\gamma_f(x) > L^d \, \|x\| \prod_{i=1}^{d} \|W_i\|$*

The looser bound in Lemma 3.5 addresses the general case in which any masking is considered and thus a worst case assumption. For a specific masking or a constrained class of maskings a tighter bound as in Lemma 3.4 can be derived. Lastly, consider the case in which one such specific masking has been learned for a classifier, which has since been updated, e.g., due to new data available. Applying the same masking to the updated classifier yields the following bound.

Lemma 3.6. *Let W_j be the j-th weight matrix in a compositional Lipschitz classifier $f(x)$, $\Delta W_j = \mu(B, W)$ an additive perturbation of the weights $W'_j = W_j + \Delta W_j$ induced by any masking B yielding the masked classifier $f'(x)$. Let the update U_j be an additive perturbation of the weights $V_j = W_j + U_j$ yielding the updated classifier $f_U(x)$. Then the masked and updated classifier $f'_U(x)$ obtained from applying the mask B to the updated weight matrix V_j is guaranteed to be stable in respect to a sample x predicted with a confidence of $\gamma_f(x) > L^d \, (\|W_j\|_\infty + \|U_j\|_\infty) \, \|x\|_\infty \prod_{i=1, i\neq j}^{d} \|W_i\|_\infty$.*

Statements of the kind made above are always theoretical in nature and gauge worst case effects of alterations of a model on the considered outcome. The weak, yet tight, bound on the norm of additive perturbations obtained from masking weights propagates through subsequent statements and thus limit the obtainable guarantees. As the results in the following section of the paper show, however, applying the proposed method yields highly promising results in application.
Furthermore, the bounds obtained from above should be considered useful in two regards. Firstly, they yield an easily quantifiable estimate of what can still be guaranteed when applying the proposed method of introducing semi-structured sparsity to a model via maskings. In fact, commonly used regularization techniques such as weight decay or norms on weights in loss functions directly yield smaller bounds on the norms of the mask-induced perturbations. Secondly, understanding in what ways sparse masks affect model performance can guide a practitioner to develop heuristics that minimize the downside effect on (guaranteed) model performance, e.g., by specifically masking weights of low magnitude and by bounding the maximum magnitude of weights to be masked.

3.3 Regularly sparse convolutions

Since CV models commonly rely on two-dimensional convolutions to process the (image) inputs, the application of semi-structured sparsity on convolutions is illustrated in two dimensions. The method extends to other dimensions in an analogue fashion. A discrete two-dimensional convolution with a kernel $H \in \mathbb{R}^{c_{\text{in}} \times h \times w}$ convolves an input tensor $X \in \mathbb{R}^{c_{\text{in}} \times b \times d}$ into an output tensor $y \in \mathbb{R}^{b \times d}$ assuming zero padding. In the below formulation the functions f and g handle the padding for invalid combinations of input values, i.e., out of range values, else return the sum of the two input values:

$$y_{ij} = \sum_{c=1}^{c_{\text{in}}} \sum_{u=1}^{w} \sum_{s=1}^{h} H_{cus} X_{cf(i,u)g(j,s)} \tag{4}$$

Usually such a convolution is conducted with c_{out} kernels to obtain an output $Y \in \mathbb{R}^{c_{\text{out}} \times b \times d}$. Alternatively, this convolution can also be expressed as a matrix multiplication between the same input $X \in \mathbb{R}^{c_{\text{in}} \times b \times d}$ and a weight matrix $W \in \mathbb{R}^{c_{\text{out}} \times (c_{\text{in}} wh)}$ constructed from the c_{out} kernels in the convolutional layer:

$$\tilde{Y} = W\mathcal{U}(X) \tag{5}$$

The unfold operator $\mathcal{U}(\cdot)$ turns the matrix X into a flattened matrix $\tilde{X} \in \mathbb{R}^{c_{\text{in}}wh \times L}$ where $L = (b + 2p_1 - w - 1)(d + 2p_2 - h - 1)$ denotes the number of blocks in the input. In the case of zero padding, i.e., full padding, the padding sizes in the respective dimensions for uneven kernel dimensions are $p_1 = \lfloor \frac{w}{2} \rfloor$ and $p_2 = \lfloor \frac{h}{2} \rfloor$ and thus $L = bd$. Reshaping $\tilde{Y}$ recovers the exact same Y as in (4). Note, that the described reformulation of (4) as (5) does not change the mathematical operations conducted but rather makes the non-contiguous memory access of the convolution explicit. The cost of the convolution (neglecting the details of data loading) therefore does not change. To achieve a 2:4 sparse convolution compatible with the accelerated matrix multiplication for 2:4 sparse matrices a mask $M \in \{0, 1\}^{c_{\text{out}} \times c_{\text{in}}wh}$ whose (block) entries are sampled according to a GS distribution as described in (2) is multiplied entry-wise to the weight matrix.

$$\tilde{Y} = (M \odot W)\, \mathcal{U}(X) \tag{6}$$

The corresponding masking layer therefore learns as many GS distributions as there are blocks of four elements in the matrix. Note, that this assumes that the product $c_{\text{out}} \cdot c_{\text{in}}wh$ is a multiple of four, since M can only contain a multiple of four entries to account for the size of the sparsity pattern. If this is not the case the matrix needs to be augmented column- or row-wise to contain a multiple of 4 entries. A schematic illustration of the two views on convolutions is illustrated in Figure 3.

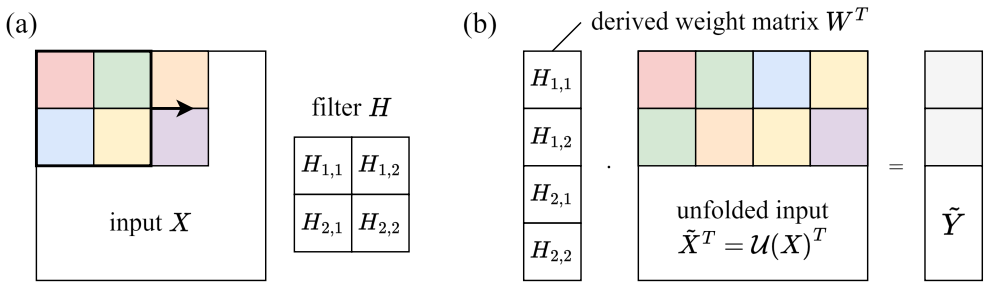

Figure 3: Simplified visualizations of convolutions on single channel input X of unspecified width and height and a single filter H as (a) "standard" convolution with a moving filter and (b) as a matrix product between an unfolded input $\tilde{X}$ and a weight matrix W derived from the filter

4 Results

In the following the architectures considered for experimental evaluation of the proposed method and the empirical results are reported. The figures were obtained by evaluating the models on the **ImageNet-1K** multi-class classification challenge [39]. The details of the dataset and comparisons between reported metrics on validation and test set performance are deferred to Appendix A.6. The details of the training procedure employed to train the masking layers of the modified architectures are reported in Appendix A.7.

Architectures To empirically evaluate whether the proposed performance gain via semi-structured sparsity can in fact be achieved without any significant loss in inference performance the following architectures were considered: **ResNet** [13], and **ConvNeXt** [25]. The models were obtained from PyTorch's Torchvision module [1]. In each variant of the models considered, the convolutional layers, unless grouped, were reformulated as in (5) yielding a matrix product in which the weights of k contiguous blocks of entries could be masked to yield a 2:4 sparse matrix as described in (6). To obtain the mask a trainable masking layer learning k GS distributions modeling the masking choices for the k blocks per layer was added from which the mask can be drawn. Since grouped convolutions constitute sparse operations themselves, they induce highly sparse large weight matrices in a high-level reformulation as described in (5) and would have slowed down model training and inference beyond feasibility on the available hardware while only promising negligible efficiency gains at the same time. They were thus not altered. Likewise, linear dense layers were not considered for a modification of the above kind as they do not contribute to the compute cost of the models significantly. E.g., in ResNet-50, linear layers account for only 0.3% of total FLOPs [8] despite accounting for roughly 8% of the parameters of the model [13]. This further shows, that the number

of parameters itself is not a reliable measure of the computational cost incurred by a layer. The pretrained weights of the original architectures were copied into the modified models initializing the choice weights of the masking layers randomly using Glorot initialization [9] and a normal distribution with $\sigma = 10^{-6}$ for weights and biases respectively. Only the weights associated with the masking layers were configured to be trainable leaving the set of pretrained weights unchanged.

Inference performance Table 1 summarizes the results obtained for the ResNet and ConvNeXt architectures indicating the training effort needed to converge to a top-1 accuracy comparable to or better than the originally reported figures. The results show that both the ResNet-based as well as the ConvNeXt-based architectures converged to the non-sparse performance levels with negligible to no loss in performance, both in top-1 and top-5 accuracy. Convergence was reached for all architectures after training periods of a fraction of the length of the original training periods showing empirically that only a comparatively small share of additional resources is needed to learn the proposed efficiency modification. This neglects the fact that modern training recipes [25] make heavy use of augmentation diminishing the additionally spent share even further in comparison. Further training beyond convergence to the reported performance, reported in Table 2, showed further improvement both in the top-1 and top-5 accuracy commonly beyond the reported performance for the unmodified networks.

Table 1: Validation classification performance of the 2:4 sparse networks measured as the top-k accuracy on ImageNet-1K [39] and the number of epochs needed to converge to a comparable or better top-1 accuracy than reported in contrast to the original number of training epochs

Architecture	**reported**			**2:4 sparse**		
	top-1	**top-5**	**epochs**	**top-1**	**top-5**	**epochs**
ResNet-18	69.76	89.08	90	70.01	88.17	1
ResNet-34	73.31	91.42	90	75.33	91.19	1
ResNet-50	76.13	92.86	90	78.54	92.86	1
ConvNeXt-T	82.52	96.15	300	82.51	94.67	10
ConvNeXt-S	83.62	96.65	300	83.76	95.00	9

Table 2: Validation classification performance of the 2:4 sparse networks measured as the top-k accuracy on ImageNet [39] after spending 10% of the resources used to initially train the network measured by the number of epochs without data augmentation

Architecture	**reported**			**2:4 sparse**		
	top-1	**top-5**	**epochs**	**top-1**	**top-5**	**epochs**
ResNet-18	69.76	89.08	90	70.22	88.27	9
ResNet-34	73.31	91.42	90	75.45	91.25	9
ResNet-50	76.13	92.86	90	78.78	92.96	9
ConvNeXt-T	82.52	96.15	300	85.63	96.09	30
ConvNeXt-S	83.62	96.65	300	87.53	96.71	30

To compare the results of the proposed method to a state of the art method of computing a 2:4 sparse subnetwork two variants of the heuristic proposed in [37], the so-called efficacy score to evaluate pruning patterns, available via NVIDIA's Apex library [33] were used in the same regime. To conform to the idea of not altering the original weights the networks were not retrained[2] after pruning as originally proposed in [37] and the results are aggregated in Table 3. As such, no significant compute resources needed to be spent. It can be observed that for all variants of ResNet and ConvNeXt the loss in performance is significant even in the better performing variant allowing for channel permutations before selecting the 2:4 sparse subnetwork. The method proposed in this paper always manages to learn a significantly better sparse pattern while spending less than a tenth of the resources.

[2]Performance metrics are reported for select models in [37] indicating no performance loss after retraining the pruned models for an additional 100 epochs.

Table 3: Validation classification performance of the 2:4 sparse networks obtained via the apex library [37] measured as the top-k accuracy on ImageNet [39]. The networks are compared in two settings disallowing and allowing permutations of the channels before pruning.

Architecture	dense		2:4 sparse (Apex)			
	reported		not permuted		permuted	
	top-1	top-5	top-1	top-5	top-1	top-5
ResNet-18	69.76	89.08	17.20	36.13	21.48	41.76
ResNet-34	73.31	91.42	43.75	68.27	49.27	73.71
ResNet-50	76.13	92.86	30.09	52.52	48.37	72.47
ConvNeXt-T	82.52	96.15	72.61	90.85	75.90	92.70
ConvNeXt-S	83.62	96.65	75.07	92.49	76.44	93.22

5 Discussion

Spending only a fraction of the resources invested to pretrain the network the results show that it is possible to learn semi-structured 2:4 sparsity patterns that can accelerate CNN inference while not impeding or even improving classification performance. This shows that extending the native support of 2:4 sparse matrix operations to 2:4 sparse convolutional kernels is a highly promising avenue towards more efficiency that is achievable today.

The results presented were obtained using a simple, generic training procedure. However, recent works indicate the high relevance of the training recipe, which could go as far as being the sole reason why (vision) transformers outperformed CNNs in image classification tasks in recent years [25]. The effect of more sophisticated training procedures including, e.g., data augmentation [6, 43, 45, 47], needs to be studied offering potential to accelerate convergence and reaching even higher levels of classification performance. Furthermore, the proposed method does not yet make use of available heuristics with patterns still being randomly initialized. In the case of, e.g., ConvNeXt, in which more than one epoch was needed to converge, a meaningful initialization cheaply obtainable, e.g., [37], could serve as an improved starting point and reduce lifetime resource spending even further. Lastly, while working exceptionally well, the work thus far only explores image classification. However, CV models are also frequently employed for object detection and segmentation tasks. Future experiments could be aimed at surveying all CV tasks relevant for CV models employed as backbones in foundation models. From a theoretical viewpoint more assumptions, e.g., on the distribution of weights could lead to more constrained yet tighter bounds. While losing some generality, this could lead to results that are even more interesting from a practical viewpoint to guide effective trades off between inference acceleration and model performance.

Beyond the aforementioned proposals for future work several additional avenues come into consideration: Firstly, the proposed architectural change introduces the temperature parameter τ of the GS distribution to the reformulated convolutional models, but the effects on performance and convergence are yet to be studied. Secondly, the work can be extended to cover more models, both convolutional and non-convolutional, as well as other frequently used, costly layer types. Lastly, more detailed insights into what information the network loses when modified as proposed could prove valuable.

6 Conclusion

In this paper, a novel method for accelerating inference in CV architectures has been presented. By expressing convolutions as semi-structured sparse matrix operations existing hardware accelerations for semi-structured sparsity can be used to directly translate model sparsity into significant savings in regards to data loading and FLOPs spent during inference. The proposed use of semi-structured sparsity patterns bridges the gap between practical requirements induced by compute hardware and the theoretical desire to not limit the choice of sparse models to not affect model performance negatively.

To obtain the sparsity patterns, a semi-structured sparse masking of the pretrained model's weights is learned from the training data optimizing for the same goal as the unchanged model. The resources spent on learning the maskings constitute a fraction of the resources spent during the original training of the model effectively reducing the resources spent during the model's lifetime. At the same time, despite dropping out half of the weights in each convolutional layer, the performance of the model is not affected negatively, even increasing in many instances as the classification experiments conducted show. From a theoretical perspective, the effects of masking the weights of a classifier are quantified in the paper in the form of guarantees on class predictions under maskings. Combining these results with model changes induced by updates of the pretrained weights guarantees for reusing learned sparsity patterns can be derived.

In conclusion, the proposed method demonstrates that extending the support of readily available acceleration techniques to natively support convolutional kernels is a promising avenue to accelerate convolutional models more than two-fold while retaining the pretrained performance.

Acknowledgments

I would like to thank Dr. Peter Belcák for the insightful discussions on the presented work, and Prof. Dr. Roger Wattenhofer and the Distributed Computing (DISCO) Lab at ETH Zürich for providing the necessary resources without which this work would not have been possible.

References

[1] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. K. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, S. Zhang, M. Suo, P. Tillet, X. Zhao, E. Wang, K. Zhou, R. Zou, X. Wang, A. Mathews, W. Wen, G. Chanan, P. Wu, and S. Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 929–947, La Jolla CA USA, Apr. 2024. ACM.

[2] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165*, July 2020.

[4] A. Buluç and J. R. Gilbert. Parallel Sparse Matrix-Matrix Multiplication and Indexing: Implementation and Experiments. *SIAM Journal on Scientific Computing*, 34(4):C170–C191, Jan. 2012.

[5] J. Chai, H. Zeng, A. Li, and E. W. Ngai. Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6:100134, Dec. 2021.

[6] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[7] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo. Compute and Energy Consumption Trends in Deep Learning Inference. *Sustainable Computing: Informatics and Systems*, 38:100857, Apr. 2023.

[8] T. Gale, E. Elsen, and S. Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[10] M. Grimaldi, D. C. Ganji, I. Lazarevich, and S. S. Deeplite. Accelerating Deep Neural Networks via Semi-Structured Activation Sparsity. In *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1171–1180, Paris, France, Oct. 2023. IEEE.

[11] E. J. Gumbel. *Statistical Theory of Extreme Values and Some Practical Applications: A Series of Lectures*, volume 33. US Government Printing Office, 1954.

[12] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[14] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. 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.

[15] S. Hooker. The hardware lottery. *Communications of the ACM*, 64(12):58–65, Dec. 2021.

[16] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*, Oct. 2021.

[17] W. Huang, Y. Hu, G. Jian, J. Zhu, and J. Chen. Pruning Large Language Models with Semi-Structural Adaptive Sparse Training. *arXiv preprint arXiv:2407.20584*, Aug. 2024.

[18] I. Hubara, B. Chmiel, M. Island, R. Banner, J. Naor, and D. Soudry. Accelerated sparse neural training: A provable and efficient method to find n: M transposable masks. *Advances in neural information processing systems*, 34:21099–21111, 2021.

[19] E. Iofinova, A. Peste, M. Kurtz, and D. Alistarh. How Well Do Sparse ImageNet Models Transfer? In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12256–12266, New Orleans, LA, USA, June 2022. IEEE.

[20] E. Jang, S. Gu, and B. Poole. Categorical Reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144*, Aug. 2017.

[21] D. Justus, J. Brennan, S. Bonner, and A. S. McGough. Predicting the Computational Cost of Deep Learning Models. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3873–3882, Seattle, WA, USA, Dec. 2018. IEEE.

[22] D. Li, X. Chen, M. Becchi, and Z. Zong. Evaluating the Energy Efficiency of Deep Convolutional Neural Networks on CPUs and GPUs. In *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, pages 477–484, Atlanta, GA, USA, Oct. 2016. IEEE.

[23] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning Filters for Efficient ConvNets. *arXiv preprint arXiv:1608.08710*, Mar. 2017.

[24] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi. Dynamic Model Pruning with Feedback. *arXiv preprint arXiv:2006.07253*, June 2020.

[25] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A ConvNet for the 2020s. *arXiv preprint arXiv:2201.03545*, Mar. 2022.

[26] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the Value of Network Pruning. *arXiv preprint arXiv:1810.05270*, Mar. 2019.

[27] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*, Jan. 2019.

[28] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally. Exploring the Granularity of Sparsity in Convolutional Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1927–1934, Honolulu, HI, USA, July 2017. IEEE.

[29] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally. Exploring the Regularity of Sparse Structure in Convolutional Neural Networks. *arXiv preprint arXiv:1705.08922*, June 2017.

[30] G. Menghani. Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better. *ACM Computing Surveys*, 55(12):1–37, Dec. 2023.

[31] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning Convolutional Neural Networks for Resource Efficient Inference. *arXiv preprint arXiv:1611.06440*, June 2017.

[32] A. Nova, H. Dai, and D. Schuurmans. Gradient-free structured pruning with unlabeled data. In *International Conference on Machine Learning*, pages 26326–26341. PMLR, 2023.

[33] NVIDIA Corporation. Apex (A PyTorch Extension) — Apex 0.1.0 documentation. https://nvidia.github.io/apex/, 2018.

[34] NVIDIA Corporation. DALI (nvidia-dali-cudaX.X), 2024.

[35] J. Park, M. Naumov, P. Basu, S. Deng, A. Kalaiah, D. Khudia, J. Law, P. Malani, A. Malevich, S. Nadathur, J. Pino, M. Schatz, A. Sidorov, V. Sivakumar, A. Tulloch, X. Wang, Y. Wu, H. Yuen, U. Diril, D. Dzhulgakov, K. Hazelwood, B. Jia, Y. Jia, L. Qiao, V. Rao, N. Rotem, S. Yoo, and M. Smelyanskiy. Deep Learning Inference in Facebook Data Centers: Characterization, Performance Optimizations and Hardware Implications. *arXiv preprint arXiv:1811.09886*, Nov. 2018.

[36] J. Pool, A. Sawarkar, and J. Rodge. Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT. https://developer.nvidia.com/blog/accelerating-inference-with-sparsity-using-ampere-and-tensorrt/, July 2021.

[37] J. Pool and C. Yu. Channel Permutations for N:M Sparsity. *Advances in neural information processing systems*, 34:13316–13327, 2021.

[38] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *arXiv preprint arXiv:2103.00020*, Feb. 2021.

[39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec. 2015.

[40] N. Thompson, K. Greenewald, K. Lee, and G. F. Manso. The Computational Limits of Deep Learning. In *Ninth Computing within Limits 2023*, Virtual, June 2023. LIMITS.

[41] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[42] H. You, C. Li, P. Xu, Y. Fu, Y. Wang, X. Chen, R. G. Baraniuk, Z. Wang, and Y. Lin. Drawing Early-Bird Tickets: Towards More Efficient Training of Deep Networks. *arXiv preprint arXiv:1909.11957*, Feb. 2022.

[43] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

[44] R. Yuster and U. Zwick. Fast sparse matrix multiplication. *ACM Transactions on Algorithms*, 1(1):2–13, July 2005.

[45] H. Zhang. Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[46] Y. Zhang, Y. Yao, P. Ram, P. Zhao, T. Chen, M. Hong, Y. Wang, and S. Liu. Advancing model pruning via bi-level optimization. *Advances in Neural Information Processing Systems*, 35:18309–18326, 2022.

[47] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020.

A Appendix / Supplemental Material

A.1 Proof of Lemma 3.1

Proof. The claim follows from the fact that the Lipschitz constant propagates through each layer of the compositional function. Consider a single layer $f_i(x) = \sigma(W_i x + b_i)$. Since σ is L-Lipschitz it follows that f_i is L_i-Lipschitz with $L_i := L\|W_i\|$ as indicated below where the first inequality uses the Lipschitz-continuity of σ and the last inequality uses the sub-multiplicative property of norms.

$$\begin{aligned}
\|\sigma(W_i x_1 + b_i) - \sigma(W_i x_2 + b_i)\| &\leq L\|W_i x_1 + b_i - (W_i x_2 + b_i)\| && (7)\\
&= L\|W_i(x_1 - x_2)\| && (8)\\
&\leq L\|W_i\|\|x_1 - x_2\| && (9)\\
&=: L_i\|x_1 - x_2\| && (10)
\end{aligned}$$

Now, consider the composed function $f_k \circ f_{k-1} \circ ... \circ f_1$; then:

$$\begin{aligned}
&\|f_k(f_{k-1} \circ ... \circ f_1(x_1)) - f_k(f_{k-1} \circ ... \circ f_1(x_2))\| && (11)\\
&\leq L_k\|f_{k-1} \circ ... \circ f_1(x_1) - f_{k-1} \circ ... \circ f_1(x_2)\| && (12)\\
&\leq L_k L_{k-1} ... L_1\|x_1 - x_2\| && (13)
\end{aligned}$$

Thus, the function $f_k \circ ... \circ f_1$ is $L_k L_{k-1} ... L_1$-Lipschitz. Finally, since the softmax function softmax(z) for a vector z is a normalized exponential function it is known to be 1-Lipschitz. If $f_d \circ ... \circ f_1$ is L'-Lipschitz, where $L' = L_1 L_2 ... L_d$, then the overall classifier $f(x)$ is consequently L_f-Lipschitz with $L_f = L' \cdot 1 = L'$. □

A.2 Proof of Lemma 3.2

Proof. The claim follows from the fact that the masking changes the Lipschitz constant of the masked layer which then propagates through each layer of the compositional function. Let the i-th layer be the masked layer, i.e., $W_i' = W_i + \Delta W$ and $f_i'(x) = \sigma(W_i' x + b_i)$. Since σ is L-Lipschitz it follows that f_i' is $(L_i + L\|\Delta W\|)$-Lipschitz with $L_i := L\|W_i\|$ as indicated below where the first inequality uses the Lipschitz-continuity of σ and the latter two inequalities use the sub-multiplicative property of norms and the triangle inequality respectively.

$$\begin{aligned}
\|\sigma(W_i' x_1 + b_i) - \sigma(W_i' x_2 + b_i)\| &\leq L\|W_i' x_1 + b_i - (W_i' x_2 + b_i)\| && (14)\\
&= L\|W_i'(x_1 - x_2)\| && (15)\\
&\leq L\|W_i + \Delta W\|\|x_1 - x_2\| && (16)\\
&\leq (L\|W_i\| + \|\Delta W\|)\|x_1 - x_2\| && (17)\\
&=: (L_i + L\|\Delta W\|) =: L_i' && (18)
\end{aligned}$$

Reusing the results from the proof of Lemma 3.1 on the Lipschitz constant of a compositional function composed with the softmax function it follows: If $f_d \circ ... \circ f_i \circ ... \circ f_1$ is L''-Lipschitz, where $L'' = L_1 ... L_i' ... L_d = L_1 ... (L_i + L\|W\|) ... L_d = L' + L_1 ... \|\Delta W\| L_d = L' + L\|\Delta W\| \prod_{i=1, i\neq j}^{d} \|W_i\|$ (cf. proof of Lemma 3.1), then the overall classifier $f(x)$ is consequently L_f-Lipschitz with $L_f = L'' \cdot 1 = L''$. □

A.3 Proof of Lemma 3.3

Proof. Given the matrix W and the bit mask B the additive perturbation can be obtained via the following construction

$$(\Delta W)_{ij} := (B_{ij} - 1)W_{ij} \tag{19}$$

yielding

$$W_{ij} + (\Delta W)_{ij} = W_{ij} + (B_{ij} - 1)W_{ij} = \begin{cases} W_{ij} - W_{ij} & \text{if} \quad B_{ij} = 0 \\ W_{ij} - 0 & \text{if} \quad B_{ij} = 1 \end{cases} \tag{20}$$

which is equivalent to the element-wise product of the matrix with the bit mask:

$$B_{ij} W_{ij} = \begin{cases} 0 & \text{if} \quad B_{ij} = 0 \\ W_{ij} & \text{if} \quad B_{ij} = 1 \end{cases} \tag{21}$$

The bound on the infinity norm on ΔW follows from the definition of the infinity norm $\|W\|_\infty = \max_{1\leq j\leq n}\sum_{j=1}^{m}|W_{ij}|$ and the increasing property of sums of non-negative values justifying the inequality below:

$$\|\Delta W\|_\infty = \max_{1\leq j\leq n}\sum_{j=1}^{m}|(B_{ij}-1)W_{ij}| \leq \max_{1\leq j\leq n}\sum_{j=1}^{m}|W_{ij}| = \|W\|_\infty \tag{22}$$

□

A.4 Proof of Lemma 3.4

Proof. Let $f(x) = (\text{softmax} \circ f_d \circ ... \circ f_j \circ ... \circ f_1)(x)$ be a compositional classifier of depth d with $f_i(x) = \sigma(W_i x + b_i)$. Let $W'_j = W_j + \Delta W$ be the additive perturbation yielding the perturbed layer $f'_j(x) = \sigma(W'_j x + b_j)$ and classifier $f'(x) = (\text{softmax} \circ f_d \circ ... \circ f'_j \circ ... \circ f_1)(x)$. Let x_{j-1} be the input to the j-th layer, i.e., $x_{j-1} = (f_{j-1} \circ ... \circ f_1)(x) = f_{j-1,1}(x)$. Then the the following can be observed about the variance in the j-th layer, where the first inequality uses the Lipschitz-continuity of σ and the last inequalities use the triangle inequality and the sub-multiplicative property of norms respectively.

$$\begin{aligned}
\|f_j(x_{j-1}) - f'_j(x_{j-1})\| &= \|\sigma(W_j x_{j-1} + b_j) - \sigma((W_j + \Delta W)x_{j-1} + b_j)\| && (23)\\
&\leq L\|W_j x_{j-1} + b_j - ((W_j + \Delta W)x_{j-1} + b_j)\| && (24)\\
&= L\|W_j x_{j-1} + b_j - (W_j x_{j-1} + b_j) - \Delta W x_{j-1}\| && (25)\\
&\leq L(\|W_j x_{j-1} + b_j - (W_j x_{j-1} + b_j)\| + \|-\Delta W x_{j-1}\|) && (26)\\
&\leq L\|\Delta W\|\|x_{j-1}\| && (27)
\end{aligned}$$

This can be extended to a statement on the entire classifier as follows:

$$\begin{aligned}
\|f(x) - f'(x)\| &= \|(f_{d,j+1} \circ f_j \circ f_{j-1,1})(x) - (f_{d,j+1} \circ f'_j \circ f_{j-1,1})(x)\| && (28)\\
&\leq L^{d-(j+1)+1}\left[\prod_{i=j+1}^{d}\|W_i\|\right]\|f_j(x_{j-1}) - f'_j(x_{j-1})\| && (29)\\
&\leq L^{d-j}\left[\prod_{i=j+1}^{d}\|W_i\|\right]\|\Delta W\|\|x_{j-1}\| && (30)\\
&\leq L^{d-j+1}\left[\prod_{i=j+1}^{d}\|W_i\|\right]\|\Delta W\|\, L^{j-1}\left[\prod_{i=1}^{j-1}\|W_i\|\right]\|x\| && (31)\\
&\leq L^{d}\,\|\Delta W\|\,\|x\|\prod_{i=1,i\neq j}^{d}\|W_i\| && (32)
\end{aligned}$$

□

A.5 Proof of Lemma 3.6

Proof. The setting of the lemma is illustrated below: the classifier $f(x)$ for which a masking B has been learned is updated yielding the question what bounds can be obtained for the updated classifier $f_U(x)$ if masked with the same mask B.

$$\begin{tikzcd}
f(x) \arrow[r, "U_j"] \arrow[d, "\Delta W=\mu(B,W_j)"'] & f_U(x) \arrow[d, "\Delta V=\mu(B,V_j)"] \\
f'(x) \arrow[r, "U_j+\mu(B,U_j)"'] & f'_U(x)
\end{tikzcd}$$

Updating the classifier and then applying the mask yields the following:

$$\begin{aligned}
(V_j)_{ik} + (\mu(B,V_j))_{ik} &= (W_j)_{ik} + (U_j)_{ik} + (\mu(B, W_j + U_j))_{ik} && (33)\\
&= \begin{cases}(W_j)_{ik} + (U_j)_{ik} - (W_j)_{ik} - (U_j)_{ik} & \text{if} \quad B_{ik} = 0\\ (W_j)_{ik} + (U_j)_{ik} - 0 & \text{if} \quad B_{ik} = 1\end{cases} && (34)
\end{aligned}$$

This can alternately be expressed as masking the weights of the classifier first and applying a sparse update $U_j + \mu(B, U_j)$ instead corresponding to the lower path in the above diagram. The two orders of updates are equivalent:

$$(W_j)_{ik} + (\mu(B, W_j))_{ik} + ((U_j)_{ik} + (\mu(B, U_j))_{ik}) \tag{35}$$

$$= \begin{cases} (W_j)_{ik} - (W_j)_{ik} + (U_j)_{ik} - (U_j)_{ik} & \text{if} \quad B_{ik} = 0 \\ (W_j)_{ik} - 0 + (U_j)_{ik} - 0 & \text{if} \quad B_{ik} = 1 \end{cases} \tag{36}$$

All perturbations shown in the diagram are additive perturbations. From the associative property of matrix addition it follows that for any two or more perturbations applied to a weight matrix the induced change can be expressed via a single perturbation. This yields $W_j + U_j + \Delta V = W_j + (U_j + \Delta V) = W_j + (U_j + \mu(B, W_j + U_j))$. From Lemma 3.4 it follows directly that f'_U is stable respect to a sample x predicted with a confidence of $\gamma_f(x) > L^d \, \|U_j + \mu(B, W_j + U_j)\|_\infty \, \|x\|_\infty \prod_{i=1, i\neq j}^{d} \|W_j\|_\infty$. Using the result from Lemma 3.3 the following bound can be obtained:

$$\begin{aligned} \|U_j + \mu(B, W_j + U_j)\|_\infty &\leq \|U_j\|_\infty + \|\mu(B, W_j + U_j)\|_\infty \\ &\leq \|U_j\|_\infty + \|W_j + U_j\|_\infty \leq \|W_j\|_\infty + 2\|U_j\|_\infty \end{aligned} \tag{37}$$

Consider therefore the second, lower path in the above diagram to obtain f'_U in which $f(x)$ is masked first and then updated. Observe that

$$[U_j + \mu(B, U_j)]_{ik} = \begin{cases} (U_j)_{ik} + 0 & \text{if} \quad B_{ik} = 0 \\ (U_j)_{ik} - (U_i)_i k & \text{if} \quad B_{ik} = 1 \end{cases} \tag{38}$$

and thus $\|U_j + \mu(B, W_j + U_j)\|_\infty \leq \|U_j\|_\infty$ (cf. argument in the proof of Lemma 3.3). This yields $W_j + \Delta W + (U_j + \mu(B, W_j + U_j)) = W_j + (\Delta W + U_j + \mu(B, W_j + U_j))$ implying for the stability of f'_U is stable respect to a sample x predicted with a confidence of $\gamma_f(x) > L^d \, \|\Delta W + U_j + \mu(B, W_j + U_j)\|_\infty \, \|x\|_\infty \prod_{i=1, i\neq j}^{d} \|W_i\|_\infty$. The following bound obtained from the auxiliary result above yields

$$\|\Delta W + U_j + \mu(B, W_j + U_j)\|_\infty \leq \|\Delta W\|_\infty + \|U_j + \mu(B, W_j + U_j)\|_\infty \leq \|W_j\|_\infty + \|U_j\|_\infty \tag{39}$$

which is a strict improvement over the naive bound from above for any non-trivial update U_j. It can be concluded that f'_U is stable respect to a sample x predicted with a confidence of

$$\gamma_f(x) > L^d \left(\|W_j\|_\infty + \|U_j\|_\infty\right) \|x\|_\infty \prod_{i=1, i\neq j}^{d} \|W_i\|_\infty \tag{40}$$

□

A.6 Classification task and reported and validation classification performance in comparison

The pretrained networks were retrained and evaluated on the multi-class classification challenge provided by the **ImageNet-1K** dataset [39]. The data set contains approximately 1.2 million training images, 50.000 validation and 100.000 test images of 1000 object classes. To ensure consistency between the performance on the validation set and the reported accuracies on the test set the unmodified architectures were evaluated showing no significant differences (cf. Table 4). To assess whether reformulating the layers induced changes due to alteration of the floating point arithmetic, likewise comparisons for the unmodified architectures and the modified architectures without masking were conducted showing no change in predictions and performance as expected and are thus not reported.

The below experimental results can be reproduced with the provided code and were obtained on a NVIDIA GeForce RTX™ 3090 with 24 GB of RAM spending less about 3 to 3:30 minutes per variant of ResNet or ConvNeXt respectively.

A.7 Training Procedure

The pretrained modified architectures were trained according to a generic training procedure using the AdamW optimizer [27] for optimization with an initial learning rate $\eta = 1.0$, a momentum of $\beta = 0.9$ and weight decay with a factor of $\lambda = 10^{-4}$. Training images were random-cropped to a

Table 4: Reported and validation classification performance of the unmodified architectures measured as the top-k accuracy on ImageNet [39]

Architecture	**Parameters**	**reported**		**validation**	
		top-1	**top-5**	**top-1**	**top-5**
ResNet-18	11.7M	69.76	89.08	69.69	89.06
ResNet-34	21.8M	73.31	91.42	73.24	91.42
ResNet-50	25.6M	76.13	92.86	75.99	92.92
ConvNeXt-T	28.6M	82.52	96.15	82.18	95.96
ConvNeXt-S	50.2M	83.62	96.65	83.26	96.94

default resolution of 224^2, validation images to 256^2. The pixel values were normalized with a mean of $\mu = (0.485, 0.456, 0.406)$ and a standard deviation of $\sigma = (0.229, 0.224, 0.225)$. Image loading and processing was Graphics Processing Unit (GPU)-accelerated using the DALI library [34]. The learning rate was adjusted by a step scheduler with no warm-up period adjusting the learning rate every 3 epochs by a factor of $\gamma = 0.1$. The training did not make use of augmentation of the training data showing each sample in every epoch exactly once. All experiments were run on a compute node equipped with 8 processor cores of 8 GB RAM each and a single NVIDIA GeForce RTX™ 3090 with 24 GB of RAM spending roughly from 1-2 hours per epoch for all (row) entries in Table 1 and 2.

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: The introduction and the abstract explicitly formulate the contribution in theory and application of the paper limiting the scope to convolutional models and semi-structured sparsity empirically evaluated on image classification.

   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: The key limitations of the paper from a practical perspective, i.e., the number of experiments conducted in terms of models tested, the task variety, as well as training procedures and initialization, and from a theoretical perspective, i.e., few assumptions on the distribution of weight values, are detailed out in Section 5.

   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: All theoretical results are summarized in Section 3.2. The statement of all lemmata contains all assumptions made. Lemma 3.1, Lemma 3.2, Lemma 3.3, Lemma 3.4, and Lemma 3.6 are accompanied by a complete proof in Appendix A. The proof of Lemma 3.5 is so similar in structure and argument to the proof of 3.4 that it has been omitted for reasons of conciseness and reader comfort.

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: To reproduce the experiments three artifacts are needed: the code, the execution environment, and the data. Firstly, to provide the required code a public repository with a permissive license containing all required code files, i.e., the proposed architectural modification from Section 3.3, the modified models from Section 4 accompanied by any auxiliary files, e.g., to load data, and the execution script are provided. Beyond third party library code no other code files are needed to reproduce the experimental results. The execution script contains the commands to reproduce all numbers reported that have not been drawn from a cited paper, i.e., the numbers from Table 1, Table 2, and Table 3 in the main body of the paper, and Table 4 in Appendix A.6. Secondly, the environment used to conduct the experiments can be recreated by installing all required versioned third party libraries listed in a dedicated execution environment. This requirements file is also provided via the publicly accessible repository. To install the Apex library the installation instructions from [33] should be observed. Thirdly, the data, the Imagenet-1K dataset can be obtained as per the instructions and conditions provided by the authors [39].

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: All code needed to conduct the experiments presented in the main body and the appendix of the paper, i.e., the code containing modified models, training procedures and execution commands, is made publicly accessible. The data used for experiments already is publicly accessible. As such all components needed to reproduce the main experiments results are available.

   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: Section 4 and Appendix A.7 of the paper describe the training and test details in a complete and comprehensive manner. For the classification task used to train and evaluate the models the training and test data used, the data preprocessing steps conducted

before feeding the data into the models, etc. are reported. For the training procedure itself the specific models considered, the modification of the models, and the training procedure, including the optimizer, the learning rate schedule, weight decay, etc. were reported. The provided code also contains these details.

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: [No]

Justification: In line with related work conducted in this field and owing to the limited available computing resources, all experiments were ran once only and the figures reported in the paper. Statistical significance is therefore obtained via the number of different experiments/architectures considered and by testing the hypothesis for each model only once after random initialization rather than by rerunning the same setting a multitude of times.

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: The paper specifically reports the computing infrastructure in terms of compute capability and memory used and the amount of compute hours spent to obtain the figures reported in the paper throughout the main body and the appendix.

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: The work fully conforms with the NeurIPS Code of Ethics. Deviation was not necessary to achieve the theoretical results included in the paper nor to conduct the experiments presented.

   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: [Yes]

    Justification: The work with CV models impacts society mainly in two ways. Firstly, as classification performance increases with model size and thus compute resources spent, training, testing and deploying these models incurs a negative environmental impact due to the energy consumed. As the work is aimed to improve the resource efficiency of models based on convolutions, this impact is addressed in the paper throughout being mentioned explicitly in Section 1 and 5. Secondly, CV models should be scrutinized in terms of biases, robustness and fairness. This inherited limitation is not a consequence of the work presented and is addressed explicitly only in the checklist for the sake of readability and conciseness.

    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: The paper builds solely on (standard) models and (standard) data openly and readily available and does not change the capabilities of the models in such a way that models formerly not considered at risk for misuse are now prone to misuse.

    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: For all assets used, i.e., libraries, models and data, the original authors are credited in the main body of the paper. The licenses of the models and libraries, if applicable, are also included in the provided code.

    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: [Yes]

    Justification: The work includes the release of the training script and the modified models as described in Section 3 and Section 4. The code is accompanied by documentation explaining structure, function and execution.

    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.