MULTIFLOW: Shifting Towards Task-Agnostic Vision-Language Pruning

Supplementary Material

In this Supplementary Material we complement the content of the main paper. The document is structured as follows: Sec. A reports comparisons with more baselines and an additional unimodal experiment; Sec. B contains all implementation details of Sec. 5 of the main paper; in Sec. C we analyze the pruning efficiency of each algorithm in terms of runtime and extend the discussions of Sec. 6 of the main paper; in Sec. D we discuss the limitations as well as the broader impacts of the proposed setting and method.

A. Additional Experiments

A.1. Additional baselines

Mask Training. Here we compare MULTIFLOW with a procedure requiring much more compute: Task-Agnostic Mask Training (TAMT) [43]. In a nutshell, TAMT works by optimizing the pruning masks as trainable parameters during the pretraining phase with straight-through gradient estimation. In its default setup, TAMT would require the entire pretraining dataset D_p and many optimization epochs, thus going against the principles of the lightweight pruning strategy proposed in the main paper. To meet the constraints of TA-VLP and to ensure a fair comparison, for this experiment we fix the computational budget of TAMT to that of gradient-based pruning algorithms in our competitor suite, meaning that we effectively train the pruning masks on $\sim 5\%$ of the 4M pretraining dataset for 1 epoch.

Layer-wise ℓ_2 normalization. Additionally, we examine a different way to unify global magnitude pruning and layer-wise calibration other than LAMP, *i.e.*, normalizing the score tensor of each layer by its Frobenius norm before global pruning (which is equivalent to layer-wise normalization by the ℓ_2 norm of the flattened score tensor). The scoring criterion before normalization is the weight magnitude, as in LAMP. This procedure was shown effective in the context of CNNs [47], so we aim to shed light on its performance with large-scale transformers.

Results are reported in Tab. 4 for XVLM at all compression levels, from which we can observe that TAMT is an effective procedure for moderated sparsity levels, but becomes comparable to even a data-free procedure such as LAMP at extreme compression rates, thus hardly justifying the additional compute it requires. A final observation is that a simple layer-wise ℓ_2 normalization is not sufficient for accurate pruning of large-scale VLMs.

A comparison between MULTIFLOW and TAMT for pruning BLIP is in Tab. 5. Consistently with the results of the main paper, also TAMT fails in pruning BLIP at great sparsities, so the table only reports 63% and 75% results. “ERR” indicates that the pruned model did not converge to meaningful results, with an average R@1 $\sim 0\%$. In general, the insights of this experiment do not diverge from the observations of the main paper, *i.e.*: ① pruning BLIP is more challenging than pruning XVLM also for TAMT; ② MULTIFLOW is more efficient and outperforms TAMT across all tasks; ③ the gap between MULTIFLOW and TAMT increases with the sparsity, regardless of the downstream task.

A.2. Pruning Vision-only models

While a core component of MULTIFLOW is the multimodality-aware injection of layer-wise pruning ratios, the method may also be applied to unimodal models with slight changes. Specifically, since no modalities to disambiguate are present in these scenarios, then the pruning ratios can be directly injected from global magnitude pruning. For any layer, this entails retaining as many parameters as one would get by applying global magnitude pruning, according to the scoring function of MULTIFLOW.

Setup. We test this straightforward variation on a Vision Transformer [17] pretrained with the DINO method [7], by comparing it to OMP and TAMT. Once the pruned models are obtained, we transfer them to 3 downstream datasets for image classification, namely CIFAR10, CIFAR100, and Flowers102 [32, 49]. Following Sec. 5 of the main paper, MULTIFLOW and TAMT use $\sim 5\%$ of the model's pretraining set for pruning (*i.e.*, 64k images from ImageNet-1k [15]). We also test MULTIFLOW under extreme data scarcity, using only 128 images for pruning, and denote the results with MF_{128}. All pruned models are trained on 224×224 images for 10 epochs with a batch size of $b = 32$, and test performance after training is reported. We use AdamW with $\beta = (0.9, 0.999)$, and cosine learning rate decay with a warmup to 1×10^{-5} during the first 10% of the training steps.

Results are in Tab. 6 for all sparsity levels. Notably, MULTIFLOW outperforms TAMT even with 500x less data and, on a general note, it outperforms the selected baselines for this additional experiment. This result suggests that the scoring function proposed in MULTIFLOW remains sound regardless of the target model to prune, and we hope this additional insight may be of use for future research on network pruning.

We also observe that MULTIFLOW is robust to data scarcity: pruning with only 128 images is comparable to using much more data in most cases. Enlarging the size of D_g tends to be more useful when the task is “difficult”, *i.e.*, it is more useful on CIFAR100 than it is on CIFAR10, or

Method	Image-Text Retrieval		Image Captioning	
	Text R@1	Image R@1	BLEU@4	CIDEr
XVLM$_{\text{CLIP}}$	78.2	60.9	39.0	130.4
ℓ_2-NORM	74.5 | 67.9 | 14.8	57.6 | 51.1 | 10.8	37.9 | 35.7 | 18.6	125.7 | 117.6 | 48.0
LAMP	75.3 | 69.4 | 21.4	58.4 | 53.2 | 14.6	37.8 | 36.3 | 20.7	125.7 | 120.1 | 56.9
TAMT	76.8 | 72.2 | 21.6	59.9 | 55.6 | 14.6	37.8 | 36.7 | 20.2	126.2 | 122.1 | 54.1
MULTIFLOW	**77.4** | **73.9** | **46.9**	**60.2** | **56.9** | **34.8**	**38.5** | **37.4** | **30.8**	**128.6** | **124.6** | **94.3**

Table 4. Additional baselines on XVLM, formatted as 63 | 75 | 90 sparsities [%]. We also report LAMP to facilitate comparing with ℓ_2-NORM.

Method	Image-Text Retrieval		Image Captioning	
	Text R@1	Image R@1	BLEU@4	CIDEr
BLIP$_{\text{BASE}}$	80.7	95.1	39.1	131.1
TAMT	75.2 | ERR	57.9 | ERR	37.3 | 34.1	124.0 | 111.5
MULTIFLOW	**76.3** | **65.7**	**59.0** | **49.9**	**37.7** | **35.7**	**125.4** | **116.3**

Table 5. Comparison between TAMT and MULTIFLOW on BLIP, formatted as 63 | 75 sparsities [%].

when the sparsity is greater.

B. Implementation Details

Remark. It is not straightforward to test existing VLM pruning methods [58, 65] in TA-VLP. First, $|\mathcal{D}_g| << |\mathcal{D}_p|$, while [65] entails pretraining a compact VLM on the entire $\mathcal{D}_p$ with knowledge distillation from the dense VLM. Second, both [58] and [65] require task-specific knowledge since they prune using the actual target downstream datasets during training, while we design $\mathcal{D}_g$ to be explicitly task-agnostic and prune at pretrained initialization. Finally, [58] and [65] target structured pruning, while our benchmarking on TA-VLP is focused on unstructured pruning, since the latter category generally leads to better performance (see, *e.g.*, [11]). Thus, to avoid unfair comparisons, we acknowledge but do not compare with [58, 65].

B.1. Experimental methodology

In this section, we provide additional details on our pruning, training and evaluation methodologies.

Usage of calibration data. For one-shot, data-driven methods (*i.e.*, SNIP and MULTIFLOW) we use the $B = 3000$ batches from $\mathcal{D}_g$. For CHITA, execution is bound by hardware availability since it requires keeping an FP32 matrix of size $B \times |\Theta|$ into volatile memory. As a consequence, only for this algorithm, we use $B = 500$ batches from both datasets, accounting for a total of ~ 368GB and ~ 453GB of RAM usage to store such a matrix for XVLM and BLIP, respectively[1]. For all iterative methods (*i.e.*, ITERSNIP and CHITA++), we follow the experimental setup of [52] and fix the computational budget of their one-shot counterparts. Thus, we use $T = 60$ pruning iterations and $B_t = 50$ batches per iteration, such that $T \times B_t = 60 \times 50 = 3000 = B$. Both algorithms use an exponential sparsity schedule, as suggested by the corresponding authors [4, 52].

Peculiarities when pruning. BLIP [39] is composed of (i) a vision-encoder and (ii) a mixture of encoder-decoder transformers (MED) for text encoding and text decoding. The text-encoder and decoder of the MED share *all* the weights except for the self-attentions. In our work, we follow this design choice and require that these properties are preserved by the pruned models. As a consequence, we share the pruning masks according to the MED, *i.e.*, shared weights also share the pruning masks. In contrast, XVLM [71] comprises a vision-encoder, a text-encoder and a fusion encoder on top. No weights are tied, so no modules share their respective pruning masks.

Extracting gradient information from $\mathcal{D}_g$. Among the algorithms benchmarked in Sec. 5, SNIP, ITERSNIP, CHITA and CHITA++ require gradient information. However, using a task-specific loss function to compute gradients is not possible in TA-VLP. Therefore, for all the aforementioned algorithms, we use the general purpose objective encoded by the pretraining loss of each model ($\mathcal{L}_p$). For BLIP, $\mathcal{L}_p = \mathcal{L}_{\text{ITC}} + \mathcal{L}_{\text{ITM}} + \mathcal{L}_{\text{LM}}$, where (i) $\mathcal{L}_{\text{ITC}}$ is the image-text contrastive loss introduced by [53]; (ii) $\mathcal{L}_{\text{ITM}}$ is the image-text matching loss, computed with binary cross-entropy between the outputs of the matching head and the groundtruth given by in-batch hard negative mining and (iii) $\mathcal{L}_{\text{LM}}$ is the

[1]We experiment on compute nodes with ∼490GB RAM, and exceeding the limit of $B = 500$ batches can lead to OOM errors with BLIP.

Method	Cifar10	Cifar100	Flowers102
DINO$_{B/16}$	99.1	91.7	98.8
OMP	98.0 | 94.7 | 58.1	83.5 | 66.7 | 23.9	92.9 | 78.0 | 22.3
TAMT	97.8 | 95.8 | 57.9	77.2 | 69.2 | 22.8	92.3 | 84.4 | 21.8
MULTIFLOW$_{\vert D_g \vert = 128}$	**98.1** | **96.4** | <u>66.4</u>	<u>84.9</u> | <u>74.0</u> | <u>28.7</u>	**96.3** | <u>87.1</u> | <u>30.4</u>
MULTIFLOW	**98.1** | <u>96.3</u> | **67.4**	**85.6** | **74.6** | **29.2**	<u>96.2</u> | **87.5** | **30.5**

Table 6. Image Classification with pruned DINO models. Formatted as 63 | 75 | 90 sparsities [%].

Method	Runtime [s]	
	BLIP	XVLM
SNIP	2770.61 (×2.82)	1847.04 (×2.67)
ITERSNIP	2963.97 (×3.02)	1966.03 (×2.85)
CHITA	8351.92 (×8.51)	3879.94 (×5.62)
CHITA++	40538.03 (×41.30)	28368.73 (×41.08)
MULTIFLOW	981.45	690.64

Table 7. Mean runtime [s] of data-driven pruning algorithms, computed over the sparsity levels of Secs. 5 and 6 of the main paper. In brackets the speedup of MULTIFLOW.

language modeling loss, computed with the outputs of the image-grounded text decoder of the MED. For XVLM, we use $\mathcal{L}_p = \mathcal{L}_{\text{ITC}} + \mathcal{L}_{\text{ITM}} + \mathcal{L}_{\text{MLM}}$, where $\mathcal{L}_{\text{MLM}}$ is the masked language modeling loss on the outputs of the fusion encoder. For additional details, please refer to the original papers [39, 71].

Tuning CHITA and CHITA++ hyperparameters. Both CHITA and CHITA++ rely on a ridge penalty, denoted by λ, to penalize deviations from dense weights. In line with the authors of these methods, we find both algorithms to be very sensitive to this hyperparameter. Hence, to ensure a fair comparison we run a grid search on λ using ITR as a proxy task. Specifically, we train all pruned models at 63% sparsity with λ ranging from 10^{-5} to 10^{3} for one epoch, then pick the best value for the benchmark of the main paper. We report full results of this tuning in Tab. 10.

Downstream task fine-tuning. We train all the pruned models with 4 NVIDIA A100 GPUs and mixed-precision, then average the results over three different runs. All fine-tuning configurations are reported in Tab. 11, for each task. We also re-execute all the dense baselines under the same setup, to ensure that hardware differences do not play any role when measuring the performance drops after pruning.

C. Additional analyses

C.1. Pruning efficiency.

We report in Tab. 7 the total runtime (s) of all data-driven methods (data-free methods can be executed in negligible time by most modern hardware). When pruning, we fix hardware requirements to only one NVIDIA A100 GPU for all algorithms, and use the FP32 data type. As the table shows, MULTIFLOW is much faster than data-driven methods requiring gradient information and/or combinatorial optimization, being approximately 3× faster than SNIP (second fastest) and 41× faster than CHITA++ (the slowest in our benchmark).

Edge	Nodes	Sparsity	Text R@1	Image R@1
✓	✗		76.97±0.32	59.51±0.12
✗	✓	63%	56.76±0.49	41.91±0.13
✓	✓		77.35±0.51	60.21±0.16
✓	✗		72.34±0.37	55.73±0.08
✗	✓	75%	39.63±0.52	28.49±0.10
✓	✓		73.87±0.13	56.94±0.10
✓	✗		14.37±0.53	10.59±0.17
✗	✓	90%	10.47±0.26	7.11±0.14
✓	✓		46.87±0.11	34.77±0.10

Table 8. Ablation of each component of the score (XVLM), computed considering only edges, only nodes, or both (as in the original MULTIFLOW algorithm).

C.2. On the importance of nodes and edges

In Sec. 4 we introduce MULTIFLOW and its scoring criterion, which incorporates an explicit formulation for neuron-level importance as well as for the importance of individual connections. While we demonstrate, in Sec. 6, that inverting the pruning mask in MULTIFLOW leads to even worse than random performance, thus verifying the soundness of our scoring function, here we assess the importance of each component of the score using ITR as a proxy task. We report results in Tab. 8. We observe that using only the importance of edges (*i.e.*, using the weight magnitude as the scoring criterion), performs better than using only node-level saliency. Despite this, we also observe that removing either component leads to large performance drops, confirming that both elements are beneficial for pruning.

C.3. Image Captioning with METEOR and SPICE.

For completeness, in Tab. 9 we report the METEOR [3] and SPICE [2] scores of the experiments shown in Tab. 2 of the main paper. Notably, these metrics confirm the findings emerged with the BLEU [51] and the CIDEr [64] scores: *i.e.*, MULTIFLOW outperforms all baselines for all models at 63% sparsity, and becomes slightly worse than only CHITA++ when pruning BLIP at 75% sparsity. Importantly, we observe that algorithms rank differently also with these metrics when the target VLM changes. For example, while CHITA++ outperforms LAMP when pruning BLIP, the latter performs better or comparably when pruning XVLM.

D. Limitations, Discussion and Future Works

Here, we provide final considerations on Task-Agnostic Vision-Language Pruning and MULTIFLOW.

Structured vs Unstructured sparsity. The entire benchmarking of the main paper, as well as MULTIFLOW, are designed and evaluated with unstructured sparsity. While this particular kind of sparsity already brings advantages in terms of memory reduction (*e.g.*, via the standard Compressed Sparse Row format) and quantitative performance, it struggles in bringing actual advantages in terms of FLOPs/runtime reductions, given the current state of neural network training and GPU devices. Nevertheless, advances in a parallel research line focusing on fast sparse operations have the potential to close this gap in upcoming years. Notable examples of active research in this direction are sparse GPU kernels [22] and NVIDIA's N:M sparsity acceleration. Moreover, research in unstructured sparsity has proven beneficial as a starting point to design structured pruning algorithms, such as in [48], where the importance of multiple connections is aggregated to obtain a neuron-level criterion, or [46], where a similar idea is applied at the attention-head level in language transformers. Consequently, since MULTIFLOW already integrates an explicit formulation of neuron-level importance, we believe extending it to structured sparsity is a promising direction for future research.

Broader impact. In our benchmark, we extensively study BLIP and XVLM in the context of TA-VLP, since they represent two significantly different ways to process and exploit multimodal information from visual and linguistic sources. We believe our work can pave the way to explore pruning also Vision-Language systems employing different ways to cope with multimodal sources (*e.g.*, [14] or [40]), or to explore whether our findings also apply when other modalities, such as audio, are involved [63]. Finally, we believe that good algorithms for Task-Agnostic Vision-Language Pruning can help reduce costs and energy requirements for both pruning and training Vision-Language systems in upcoming years, and do not see direct risks associated with our work.

Method	Sparsity	BLIP		XVLM	
		METEOR	SPICE	METEOR	SPICE
DENSE	0%	30.63	23.49	30.50	23.41
RANDOM	63%	16.35	9.04	21.35	14.14
SNIP		28.34±0.05	21.47±0.02	29.00±0.12	22.17±0.10
ITERSNIP		22.42±0.04	15.17±0.02	28.58±0.06	21.76±0.05
OMP		29.69±0.01	22.70±0.01	29.76±0.04	22.74±0.04
LAMP		28.82±0.02	21.97±0.01	29.78±0.06	22.79±0.01
CHITA		29.62±0.06	22.61±0.02	29.77±0.05	22.79±0.06
CHITA++		29.74±0.02	22.74±0.02	29.65±0.07	22.71±0.08
MULTIFLOW		**29.75±0.04**	**22.82±0.07**	**29.96±0.03**	**23.07±0.05**
RANDOM	75%	14.58	7.19	19.11	11.97
SNIP		25.98±0.11	18.91±0.07	27.02±0.22	20.24 ±0.28
ITERSNIP		14.19±0.06	6.92±0.05	25.51±0.25	18.52±0.27
OMP		27.79±0.01	20.96±0.03	28.52±0.05	21.70±0.03
LAMP		24.26±0.00	17.18±0.01	28.73±0.01	21.76±0.02
CHITA		27.86±0.03	21.03±0.03	28.67±0.19	21.78±0.17
CHITA++		**28.18±0.06**	**21.38±0.01**	28.64±0.04	21.79±0.06
MULTIFLOW		28.12±0.04	21.27±0.02	**29.21±0.03**	**22.40±0.01**

Table 9. Image Captioning results with METEOR [3] and SPICE [2] scores. Naming and coloring follow Tab. 2 of the main paper.

Method	λ	BLIP			XVLM		
		Text R@1	Image R@1	Avg. R@1	Text R@1	Image R@1	Avg. R@1
CHITA	10^{-5}	0.00	0.02	0.01	0.50	0.34	0.42
	10^{-4}	0.00	0.02	0.01	0.80	0.71	0.76
	10^{-3}	0.02	0.02	0.02	11.48	9.23	10.36
	10^{-2}	62.10	47.63	54.87	64.98	50.71	57.85
	10^{-1}	66.42	50.85	**58.64**	72.54	57.45	65.00
	10^{0}	65.48	50.20	57.84	72.54	57.40	64.97
	10^{1}	66.44	50.77	58.61	72.78	57.30	65.04
	10^{2}	65.10	50.42	57.76	72.66	57.23	64.94
	10^{3}	65.42	50.04	57.73	72.84	57.33	**65.09**
CHITA++	10^{-5}	0.06	0.02	0.04	0.02	0.02	0.02
	10^{-4}	0.00	0.02	0.01	0.00	0.02	0.01
	10^{-3}	0.02	0.02	0.02	0.06	0.02	0.04
	10^{-2}	0.00	0.02	0.01	61.64	48.44	55.04
	10^{-1}	68.18	52.01	**60.10**	73.80	58.77	**66.29**
	10^{0}	67.26	51.85	59.56	72.96	58.22	65.59
	10^{1}	66.00	50.75	58.38	73.44	57.71	65.58
	10^{2}	65.64	50.32	57.98	73.12	57.58	65.35
	10^{3}	66.52	50.87	58.70	72.76	57.33	65.05

Table 10. Hyperparameter Grid Search on λ (ridge penalty) for CHITA (top) and CHITA++ (bottom) on both BLIP (left) and XVLM (right). The "Avg. R@1" column reports the average between "Text R@1" and "Image R@1", and is used for the final choice of λ.

Model	Task	Batch Size	Epochs	Optimizer	LR	WD	Scheduler	Warmup
BLIP	ITR	128	6	AdamW, $\beta = (0.9, 0.999)$	10^{-5}	0.05	cosine	✗
	IC	256	5	AdamW, $\beta = (0.9, 0.999)$	10^{-5}	0.05	cosine	✗
	VQA	256	10	AdamW, $\beta = (0.9, 0.999)$	2×10^{-5}	0.05	cosine	✗
XVLM	ITR	128	10	AdamW, $\beta = (0.9, 0.98)$	3×10^{-5}	0.01	linear	✓, 10% steps
	IC	256	5	AdamW, $\beta = (0.9, 0.98)$	10^{-5}	0.01	linear	✓, 10% steps
	VQA	256	10	AdamW, $\beta = (0.9, 0.98)$	5×10^{-5}	0.01	linear	✓, 10% steps

Table 11. Fine-tuning hyperparameter configurations for different tasks and VLMs.