Formatting Instructions For NeurIPS 2022

Anonymous Author(s)
Affiliation
Address
email

Abstract

The abstract paragraph should be indented ½ inch (3 picas) on both the left- and right-hand margins. Use 10 point type, with a vertical spacing (leading) of 11 points. The word **Abstract** must be centered, bold, and in point size 12. Two line spaces precede the abstract. The abstract must be limited to one paragraph.

1 Introduction

Recent advancements in large-scale pretrained networks, such as CLIP [1] or MAE [2], have demonstrated remarkable and generalizable performance across a variety of computer vision tasks. Conventionally, these networks are fine-tuned for specific downstream tasks by adjusting their weights through gradient descent, either by training an additional layer on top of the pretrained network or by fine-tuning the entire network. One limitation of this approach is that it necessitates a full copy of the fine-tuned weights to be stored for each downstream task, leading to significant memory requirements.

In this work, we investigate the potential of identifying task-specific subnetworks that achieve good performance on downstream tasks, without modifying the original pretrained network weights. This approach can be implemented in a self-supervised or supervised manner and involves training a mask that selectively deactivates certain network weights. The mask is trained separately for each downstream task, enabling the network to dynamically adapt to different tasks.

Moreover, this method can provide practical advantages in some scenarios, such as reducing memory requirements compared to standard fine-tuning techniques, as masks are smaller in size than full copies of the network weights. In this work, we examine the feasibility and effectiveness of this approach by evaluating it across various vision tasks and comparing it with standard fine-tuning methods.

We show that the pass-through trick not only works on self-supervisedly trained models, but can also be used to perform self-supervised fine-tuning.

2 Related Work

2.1 Continual Learning

Continual learning seeks to enable a single neural network to learn and perform multiple tasks sequentially without forgetting previously learned tasks. Several works have done this by identifying task-specific subnetworks that are efficient in terms of memory storage.

Mallya, Davis, and Lazebnik were the first to use the pass-through trick, a method that learns subnetwork masks directly through gradient descent. They applied this method to pretrained models to achieve domain adaptation, their method reached performance very close to conventional fine-tuning. Ramanujan et al. found that as similar technique is also able to find well performing domain

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

specific subnetworks on random, untrained networks. However they found that the base network had to be significantly wider in order to perform as well as standard, weight-modifying gradient descent. In a follow up work, Wortsman et al. expand on this approach to teach a random, untrained network thousands of tasks.

2.2 Pruning and Neural Network Architectures

A similar technique, in combination with training the actual neural network weights has also been applied as a pruning technique for CNNs and vision transformersSanh, Wolf, and Rush[8].

Other fields where this technique has been applied include Neural Architecure Search[13], where Wortsman, Farhadi, and Rastegari use the pass-through trick to determine where to add or remove a connection, and the design of novel neural network architectures such as neural networks with binary weights (-1 or 1) and ternary weights (-1, 0 or 1)[3, 7]. These works have demonstrated the effectiveness of the pass through trick for learning certain kinds of discrete components through gradient descent.

2.3 Self-supervised Learning

Recently, self-supervised approaches have gained traction as a way to improve performance of computer vision models both in general and in label sparse situations[1, 2, 5]. These approaches enable a model backbone to be trained without the use of labels. The focus has mostly been on learning good foundation models, but some work has also shown that self supervised learning can also be used to improve performance on downstream tasks in computer vision [11] and natural language processing [4, 6]. Given the increasing importance of label-free machine learning, it is important to investigate whether the pass-through trick can be used in combination with self-supervised learning.

3 Method

3.1 Masking

Subnetworks are represented by a binary mask M which indicates what weights remain active and what weights are zeroed out. In order to learn an appropiate mask, each weight is given a corresponding score S which is initialized to a value higher than the threshold. If the score is above a threshold μ, the weight is active, otherwise it is zeroed out. The score is updated during gradient descent using the gradient of the loss with respect to the mask such that if the the mask should go up to decrease the loss, the score is increased. And conversely, if it should go down, the score is decreased. In this way, across multiple gradient descent iterations, the score goes down for weights that increase the loss, and eventally the score reaches the threshold and the weight is deactivated. Conversely, if the weight was already deactivated, but now it would decrease the loss if it was active, the score will go up, and eventually the weight will be reactivated.

$$\theta^t = \overline{\theta} \cdot M^t \tag{1}$$

$$M^t = \begin{cases} 1 & S^t > \mu \\ 0 & S^t \leq \mu \end{cases} \tag{2}$$

Scores are updated by:

$$S^t = S^{t-1} - \lambda \frac{d\mathcal{L}^{t-1}}{dM^{t-1}} \tag{3}$$

$$\tag{4}$$

(In reality, momentum is also used, see appendix for the full equation)

The trick to implement this is then simply to set the gradient of the score with respect to the loss to be the gradient of the mask with respect to the loss instead.

$$\frac{d\mathcal{L}^t}{dS^t} = \frac{d\mathcal{L}^t}{dM^t} \tag{5}$$

$$\tag{6}$$

In this way, even though the mask itself is never updated, we use it's gradient to update the weight's score.

[Explain submasking and passthrough trick]

When compared to full finetuning, this method would appear to introduce two additional parameters, the threshold μ and the initial value of the scores S^0. However, it turns out out that S^0 and μ can be set to arbitrary values with no loss of generality as long as $S^0 > \mu$ and SGD without weight decay is used. This can be derived from proofs A.1.1 and A.1.2. The first shows that if you shift the threshold and the score initialization by the same amount, the resulting mask after training will be exactly the same, thus we just set $\mu = 0$. The second shows that scaling the score initialization by a factor α is a eivalent to scaling the learning rate by a factor $\frac{1}{\alpha}$ thus we set $S^0 = 1$.

4 Experiments

5 Results

5.1 Submasking

Table 1: Accuracy

Model+Method	cifar10	cifar100	dtd	eurosat	flowers	oxfordpets	sun397	ucf101
rn18-timm+LP	0.87	0.59	0.62	0.92	0.93	0.89	0.49	0.66
rn18-timm+FS	0.95	0.76	0.66	0.97	0.96	0.85	0.48	0.66
rn18-timm+FFT	0.95	0.76	0.69	0.97	0.96	0.89	0.53	0.69
rn50-swav+LP	0.91	0.75	0.76		0.98	0.88	0.66	0.78
rn50-swav+FS	0.96	0.80	0.71	0.97	0.97	0.86	0.48	0.63
rn50-swav+FFT	0.96	0.82	0.74	0.98	0.99	0.89	0.62	0.67
vitb32-clip+FFT	0.96	0.82	0.72	0.98	0.97	0.89	0.64	0.81
vitb32-clip+FS	0.97	0.83	0.74	0.98	0.97	0.89	0.67	0.82
Zrn50-timm+LP	0.88	0.70	0.72	0.96	0.95	0.93	0.61	0.77
Zrn50-torch+LP	0.89	0.71	0.71	0.96	0.93	0.92	0.62	0.77
vitb32-clip+LP	0.95	0.80	0.75	0.95	0.97	0.89	0.75	0.83

Table 2: Sparsity

Also plot sparsity at each layer for each model.

Other ideas:

- Measure task similarity through masked weights (cmp with abs diff on FFT)
- On re-training with the same parameters using submasking, what weights are still submasked?
- Show experimental evidence for the threshold and lr/score invariance? Or is proof enough.

Ideas to explore once all data is collected:

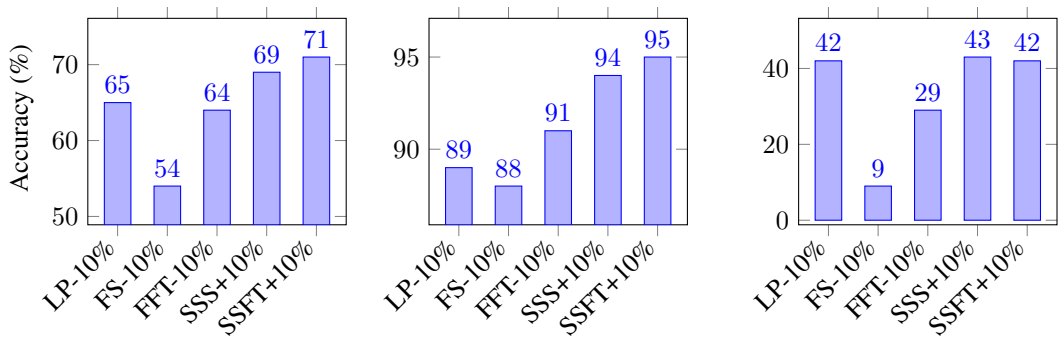
- Correlation between sparsity and change in accuracy (zero-shot vs trained)
- Correlation between dataset properties and accuracy under submasking.

Things to mention:

- Submasking may require less parameter tuning (so far, same parameters for rn18timm as rn50swav, but FFT needs different parameters)

5.2 Label Sparsity

Figure 1: ResNet-50 (SWAV); Top@1 accuracy when comparing different ways to fine-tune the model backbone. From left to right: Linear Probe on 10% of labeled data, Full Submasking on 10% of labeled data, Full Fine-Tuning on 10% of labeled data, Self-Supervised Submasking on 100% of unlabeled data plus a linear probe on 10% of labeled data, Self-Supervised Fine-Tuning on 100% of unlabeled data plus a linear probe on 10% of labeled data.

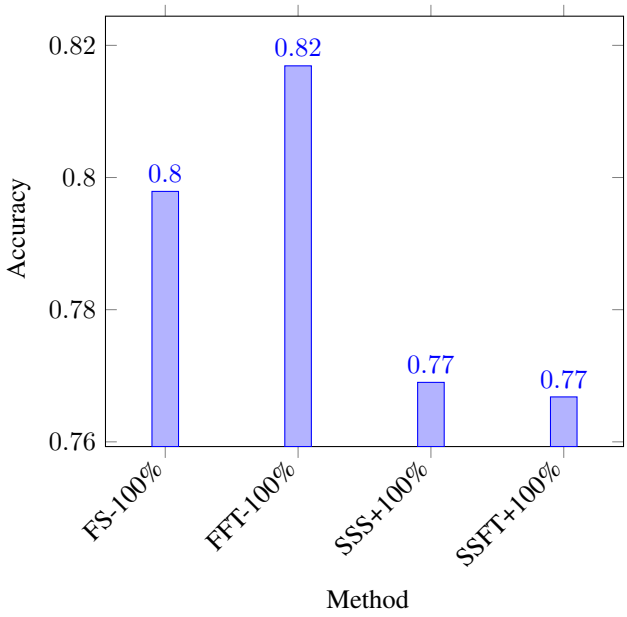


(a) CIFAR-100 (b) CIFAR-10 (c) SUN-397

Note on fig 1c, KNN accs are better, 52% and 50% respectively,.

6 Conclusion

Figure 2: ResNet-50 (SWAV), in the 100% of data case, SSL underperforms by quite a bit on C100. Probably best to just mention this and and put the plot in the appendix, as it is not what we care about.)



A Appendix

For each of the models RN18-TiMM3a3b, RN50-PYTORCH, RN50-SWAV4a4b, RN50-CLIP, T-CLIP.

Table 3: ResNet-18 (TIMM)

rn18-timm	Full FT	Submasked	Linear Probe
cifar10	$0.95 \pm 0.00_1$	$0.95 \pm 0.00_1$	$0.87 \pm 0.00_5$
cifar100	$0.76 \pm 0.00_1$	$0.76 \pm 0.00_1$	$0.59 \pm 0.00_1$
dtd	$0.69 \pm 0.00_1$	$0.66 \pm 0.00_1$	$0.62 \pm 0.00_1$
eurosat	$0.97 \pm 0.00_1$	$0.97 \pm 0.00_1$	$0.92 \pm 0.00_1$
flowers	$0.96 \pm 0.00_1$	$0.96 \pm 0.00_1$	$0.93 \pm 0.00_1$
oxfordpets	$0.89 \pm 0.00_1$	$0.85 \pm 0.00_1$	$0.89 \pm 0.00_1$
sun397	$0.53 \pm 0.00_1$	$0.48 \pm 0.00_1$	$0.49 \pm 0.00_1$
ucf101	$0.69 \pm 0.00_1$	$0.66 \pm 0.00_1$	$0.66 \pm 0.00_1$

(a) Accuracy

rn18-timm	Submasked
cifar10	$0.15 \pm 0.00_1$
cifar100	$0.36 \pm 0.00_1$
dtd	$0.07 \pm 0.00_1$
eurosat	$0.07 \pm 0.00_1$
flowers	$0.07 \pm 0.00_1$
oxfordpets	$0.04 \pm 0.00_1$
sun397	$0.32 \pm 0.00_1$
ucf101	$0.11 \pm 0.00_1$

(b) Sparsity

Table 4: ResNet-50 (SWAV)

rn50-swav	Full FT	Submasked	Linear Probe
cifar10	$0.96 \pm 0.00_1$	$0.96 \pm 0.00_1$	$0.91 \pm 0.00_1$
cifar100	$0.82 \pm 0.00_1$	$0.80 \pm 0.00_1$	$0.75 \pm 0.00_1$
dtd	$0.74 \pm 0.00_1$	$0.71 \pm 0.00_1$	$0.76 \pm 0.00_1$
eurosat	$0.98 \pm 0.00_1$	$0.97 \pm 0.00_1$	
flowers	$0.99 \pm 0.00_1$	$0.97 \pm 0.00_1$	$0.98 \pm 0.00_1$
oxfordpets	$0.89 \pm 0.00_1$	$0.86 \pm 0.00_1$	$0.88 \pm 0.00_1$
sun397	$0.62 \pm 0.00_1$	$0.48 \pm 0.00_1$	$0.66 \pm 0.00_1$
ucf101	$0.67 \pm 0.00_1$	$0.63 \pm 0.00_1$	$0.78 \pm 0.00_1$

(a) Accuracy

rn50-swav	Submasked
cifar10	$0.10 \pm 0.00_1$
cifar100	$0.23 \pm 0.00_1$
dtd	$0.05 \pm 0.00_1$
eurosat	$0.04 \pm 0.00_1$
flowers	$0.05 \pm 0.00_1$
oxfordpets	$0.04 \pm 0.00_1$
sun397	$0.19 \pm 0.00_1$
ucf101	$0.08 \pm 0.00_1$

(b) Sparsity

Table 5: Accuracy KNN

rn18-timm	Full FT	Submasked	Zero Shot
cifar10	$0.95 \pm 0.00_1$	$0.95 \pm 0.00_1$	$\text{nan} \pm \text{nan}_5$
cifar100	$0.76 \pm 0.00_1$	$0.76 \pm 0.00_1$	$0.59 \pm 0.00_1$
dtd	$0.67 \pm 0.00_1$	$0.66 \pm 0.00_1$	$0.59 \pm 0.00_1$
eurosat	$0.97 \pm 0.00_1$	$0.97 \pm 0.00_1$	$0.90 \pm 0.00_1$
flowers	$0.94 \pm 0.00_1$	$0.95 \pm 0.00_1$	$0.68 \pm 0.00_1$
oxfordpets	$0.89 \pm 0.00_1$	$0.85 \pm 0.00_1$	$0.88 \pm 0.00_1$
sun397	$0.51 \pm 0.00_1$	$0.48 \pm 0.00_1$	$\text{nan} \pm \text{nan}_1$
ucf101	$0.69 \pm 0.00_1$	$0.66 \pm 0.00_1$	$0.60 \pm 0.00_1$

A.1 Proofs

A.1.1 Translation invariance of threshold and score initialization

$$m_i^t = [\theta_i^t > k] \tag{7}$$

where k is the threshold

Table 6: Accuracy KNN

rn50-swav	Full FT	Submasked	Zero Shot
cifar10	$0.96 \pm 0.00_1$	$0.96 \pm 0.00_1$	$0.83 \pm 0.00_1$
cifar100	$0.80 \pm 0.00_1$	$0.80 \pm 0.00_1$	$0.59 \pm 0.00_1$
dtd	$0.71 \pm 0.00_1$	$0.70 \pm 0.00_1$	$0.69 \pm 0.00_1$
eurosat	$0.98 \pm 0.00_1$	$0.97 \pm 0.00_1$	
flowers	$0.97 \pm 0.00_1$	$0.96 \pm 0.00_1$	$0.73 \pm 0.00_1$
oxfordpets	$0.89 \pm 0.00_1$	$0.86 \pm 0.00_1$	$0.73 \pm 0.00_1$
sun397	$0.59 \pm 0.00_1$	$0.47 \pm 0.00_1$	$0.54 \pm 0.00_1$
ucf101	$0.67 \pm 0.00_1$	$0.65 \pm 0.00_1$	$0.60 \pm 0.00_1$

$$\theta_i^t = \theta_i^0 - \sum_{\hat{t}=0}^{t-1} \gamma^{\hat{t}} g_i(\mathcal{M}^{\hat{t}}, \mathcal{B}^{\hat{t}}) \tag{8}$$

where $g_i(\mathcal{M}^{\hat{t}}, \mathcal{B}^{\hat{t}})$ is the gradient of the i'th weight given the mask $\mathcal{M}^{\hat{t}}$ and batch $\mathcal{B}^{\hat{t}}$ (including momentum if enabled)

$$m_i^0 = [\theta_i^0 > k] \qquad = [\theta_i^0 + a > k + a] \tag{9}$$

$$m_i^t = [\theta_i^0 - \sum_{\hat{t}=0}^{t-1} \gamma^{\hat{t}} g_i(\mathcal{M}^{\hat{t}}, \mathcal{B}^{\hat{t}}) > k] \qquad = [\theta_i^0 + a - \sum_{\hat{t}=0}^{t-1} \gamma^{\hat{t}} g_i(\mathcal{M}^{\hat{t}}, \mathcal{B}^{\hat{t}}) > k + a] \tag{10}$$

For the base case we then have that replacing θ_i^0 with $\theta_i^0 + a$ and k with $k + a$ will not change m_i^0 for any i, which means the network mask $\mathcal{M}^0$ is also invariant to this change. Consequently, m_i^1 is also invariant to this change because of eq. 10 and because the gradient $g_i(\mathcal{M}^0, \mathcal{B}^0)$ does not change. The same reasoning can be applied recursively to m_i^2 and so on. Thus, by induction, translating the initial score and threshold by the same amount will not change any of the network masks during training (under simple sgd without weight decay).

A.1.2 Scale invariance of learning rate and score initialization

Equation for sgd with weight decay:

$$\theta_i^t = \theta_i^{t-1} - \gamma^t \left(g_i(\mathcal{M}^{t-1}, \mathcal{B}^{t-1}) + \lambda \theta_i^{t-1} \right) \tag{11}$$

Say we replace γ^t with $\gamma^t \alpha$, θ_i^{t-1} with $\theta_i^{t-1} \alpha$ and λ with $\frac{\lambda}{\alpha}$ for some $\alpha \in \mathbb{R}^+$, then we get:

$$\alpha \theta_i^{t-1} - \alpha \gamma^t \left(g_i(\mathcal{M}^{t-1}, \mathcal{B}^{t-1}) + \frac{\lambda}{\alpha} \alpha \theta_i^{t-1} \right) = \alpha \left(\theta_i^{t-1} - \gamma^t \left(g_i(\mathcal{M}^{t-1}, \mathcal{B}^{t-1}) + \lambda \theta_i^{t-1} \right) \right) \tag{12}$$

$$= \alpha \theta_i^t \tag{13}$$

Similarly to the previous proof, the initial masks $m_i^0 = [\theta_i^0 > 0]$ are invariant to the scale change $m_i^0 = [\alpha \theta_i^0 > 0]$, so the replacement of θ_i^0 with $\alpha \theta_i^0$, combined with the other replacements, does not change the gradient $g_i(\mathcal{M}^0, \mathcal{B}^0)$. Combined with eq 13, this means that the updated parameter after the first SGD step is only different in scale when compared to what it would have been without the scale change ($= \alpha \theta_i^t$). Apply this reasoning recursively and it can be seen through induction that the network masks will be the same during training as for the original learning rate, score initialization and weight decay.

Note that the conclusion still holds when momentum is enabled

References

[1] Mathilde Caron et al. *Emerging Properties in Self-Supervised Vision Transformers*. May 24, 2021. DOI: 10.48550/arXiv.2104.14294. arXiv: arXiv:2104.14294. URL: http://arxiv.org/abs/2104.14294 (visited on 11/22/2022). preprint.

[2] Mathilde Caron et al. *Unsupervised Learning of Visual Features by Contrasting Cluster Assignments*. Jan. 8, 2021. arXiv: arXiv:2006.09882. URL: http://arxiv.org/abs/2006.09882 (visited on 12/03/2022). preprint.

[3] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. *BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations*. Apr. 18, 2016. DOI: 10.48550/arXiv.1511.00363. arXiv: arXiv:1511.00363. URL: http://arxiv.org/abs/1511.00363 (visited on 11/22/2022). preprint.

[4] Suchin Gururangan et al. "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. ACL 2020. Online: Association for Computational Linguistics, July 2020, pp. 8342–8360. DOI: 10.18653/v1/2020.acl-main.740. URL: https://aclanthology.org/2020.acl-main.740 (visited on 12/14/2022).

[5] Kaiming He et al. *Momentum Contrast for Unsupervised Visual Representation Learning*. Mar. 23, 2020. arXiv: arXiv:1911.05722. URL: http://arxiv.org/abs/1911.05722 (visited on 11/22/2022). preprint.

[6] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. ACL 2018. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 328–339. DOI: 10.18653/v1/P18-1031. URL: https://aclanthology.org/P18-1031 (visited on 12/14/2022).

[7] Itay Hubara et al. "Binarized Neural Networks". In: (), p. 9.

[8] François Lagunas et al. *Block Pruning For Faster Transformers*. Sept. 10, 2021. arXiv: arXiv:2109.04838. URL: http://arxiv.org/abs/2109.04838 (visited on 11/22/2022). preprint.

[9] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. *Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights*. Mar. 16, 2018. DOI: 10.48550/arXiv.1801.06519. arXiv: arXiv:1801.06519. URL: http://arxiv.org/abs/1801.06519 (visited on 11/22/2022). preprint.

[10] Vivek Ramanujan et al. "What's Hidden in a Randomly Weighted Neural Network?" Mar. 30, 2020. arXiv: 1911.13299 [cs]. URL: http://arxiv.org/abs/1911.13299 (visited on 07/27/2022).

[11] Colorado J. Reed et al. "Self-Supervised Pretraining Improves Self-Supervised Pretraining". In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2022, pp. 2584–2594. URL: https://openaccess.thecvf.com/content/WACV2022/html/Reed_Self-Supervised_Pretraining_Improves_Self-Supervised_Pretraining_WACV_2022_paper.html (visited on 04/07/2023).

[12] Victor Sanh, Thomas Wolf, and Alexander M Rush. "Movement Pruning: Adaptive Sparsity by Fine-Tuning". In: (), p. 12.

[13] Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. *Discovering Neural Wirings*. Nov. 16, 2019. arXiv: arXiv:1906.00586. URL: http://arxiv.org/abs/1906.00586 (visited on 11/22/2022). preprint.

[14] Mitchell Wortsman et al. "Supermasks in Superposition". In: (), p. 12.