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.

1.1 Related Work

The identification of subnetworks has been explored as a way to achieve continual learning. Continual learning aims to teach the same neural network to perform multiple tasks, and finding subnetworks that are cheap to store can help achieve this objective.

1.1.1 Continual Learning

Continual learning seeks to enable a single neural network to learn and perform multiple tasks sequentially without forgetting previously learned tasks. This can be accomplished by identifying task-specific subnetworks that are efficient in terms of memory storage.

? 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. **?** focused on finding subnetworks within untrained models instead, using a different technique to

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

identify the subnetworks. These works have demonstrated the feasibility and potential benefits of identifying task-specific subnetworks within larger neural networks

1.1.2 Pruning

»

1.1.3 Novel Neural Network Architectures

» **?**

1.1.4 Self-supervised Learning

2 Method

2.1 Mask learning

[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 $S^0 = 1$.

2.2 Experiments

3 Results

3.1 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		0.99			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		0.97	0.89	0.67	0.82
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)

3.2 Label Sparsity

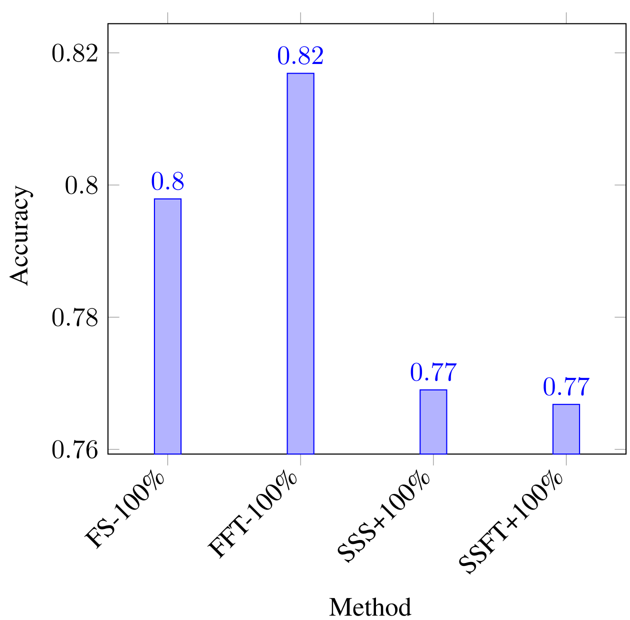
Figure 1: ResNet-50 (SWAV); NOTE: Merge both into one, and add another plot with 4k labels (or keep separate and put LP as a horizontal line?). Linear Probe, Full Submasking, Full Fine-Tuning, Self-Supervised Submasking, Self-Supervised Fine-Tuning.

(a) CIFAR-100

(b) CIFAR-10

4 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	$0.95 \pm 0.00_1$	$0.95 \pm 0.00_1$	$0.87 \pm 0.00_4$
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		$0.97 \pm 0.00_1$	
flowers	$0.99 \pm 0.00_1$	$0.97 \pm 0.00_1$	$0.98 \pm 0.00_1$
oxfordpets		$0.86 \pm 0.00_1$	$0.88 \pm 0.00_1$
sun397		$0.48 \pm 0.00_1$	$0.66 \pm 0.00_1$
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	$0.95 \pm 0.00_1$	$0.95 \pm 0.00_1$	nan ± nan_4
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	$0.51 \pm 0.00_1$	$0.48 \pm 0.00_1$	nan ± 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

$$m_i^t = [\theta_i^t > k] \tag{1}$$

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		$0.97 \pm 0.00_1$	
flowers	$0.97 \pm 0.00_1$	$0.96 \pm 0.00_1$	$0.73 \pm 0.00_1$
oxfordpets		$0.86 \pm 0.00_1$	$0.73 \pm 0.00_1$
sun397		$0.47 \pm 0.00_1$	$0.54 \pm 0.00_1$
ucf101	$0.67 \pm 0.00_1$	$0.65 \pm 0.00_1$	$0.60 \pm 0.00_1$

$$\theta_i^t = \theta_i^0 - \sum_{\hat{t}=0}^{t-1} \gamma^{\hat{t}} g_i(\mathcal{M}^{\hat{t}}, \mathcal{B}^{\hat{t}}) \tag{2}$$

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)

$$\begin{aligned}
m_i^0 &= [\theta_i^0 > k] & &= [\theta_i^0 + a > k + a] \qquad (3)\\
m_i^t &= [\theta_i^0 - \sum_{\hat{t}=0}^{t-1} \gamma^{\hat{t}} g_i(\mathcal{M}^{\hat{t}}, \mathcal{B}^{\hat{t}}) > k] & &= [\theta_i^0 + a - \sum_{\hat{t}=0}^{t-1} \gamma^{\hat{t}} g_i(\mathcal{M}^{\hat{t}}, \mathcal{B}^{\hat{t}}) > k + a] \qquad (4)
\end{aligned}$$

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. 4 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:

$$\theta_i^t = \theta_i^{t-1} - \gamma^t \left(g_i(\mathcal{M}^{t-1}, \mathcal{B}^{t-1}) + \lambda \theta_i^{t-1} \right) \tag{5}$$

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:

$$\begin{aligned}
\alpha \theta_i^{t-1} - \alpha \gamma^t \left(g_i(\mathcal{M}^{t-1}, \mathcal{B}^{t-1}) + \frac{\lambda}{\alpha} \alpha \theta_i^{t-1} \right) &= \alpha \left(\theta_i^{t-1} - \gamma^t \left(g_i(\mathcal{M}^{t-1}, \mathcal{B}^{t-1}) + \lambda \theta_i^{t-1} \right) \right) \qquad (6)\\
&= \alpha \theta_i^t \qquad (7)
\end{aligned}$$

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 7, 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