Momentum Contrastive Pruning

Siyuan Pan[1(✉)], Yiming Qin[1], Tingyao Li[1], Xiaoshuang Li[1], Liang Hou[2]
[1]Shanghai Jiao Tong University, [2]Institute of Computing Technology, Chinese Academy of Sciences
{pansiyuan, yiming_qin, tingyao_lee, lixiaoshuang}@sjtu.edu.cn, houliang17z@ict.ac.cn

Abstract

Momentum contrast [16] (MoCo) for unsupervised visual representation learning has a close performance to supervised learning, but it sometimes possesses excess parameters. Extracting a subnetwork from an over-parameterized unsupervised network without sacrificing performance is of particular interest to accelerate inference speed. Typical pruning methods are not applicable for MoCo, because in the fine-tune stage after pruning, the slow update of the momentum encoder will undermine the pretrained encoder. In this paper, we propose a Momentum Contrastive Pruning (MCP) method, which prunes the momentum encoder instead to obtain a momentum subnet. It maintains an unpruned momentum encoder as a smooth transition scheme to alleviate the representation gap between the encoder and momentum subnet. To fulfill the sparsity requirements of the encoder, alternating direction method of multipliers [40] (ADMM) is adopted. Experiments prove that our MCP method can obtain a momentum subnet that has almost equal performance as the over-parameterized MoCo when transferred to downstream tasks, meanwhile has much less parameters and float operations per second (FLOPs).

1. Introduction

Learning effective visual representations without human supervision is a long-standing problem. Several recent studies [1,4,5,16,19,20,28,32,36,42] achieve satisfying results on self-supervised visual representation learning using contrastive loss and its variants. In industrial scenarios, most models need to be lightweight to reduce the size and relieve the computing burden before being deployed to embedded or Internet of things systems [14]. Self-supervised learning combined with pruning is a common and effective way to achieve this, during which the pretrained model is first transferred to downstream tasks then pruned. However, the downstream tasks are diverse and usually require multiple iterations with the update of the dataset. Minor changes of the task may lead to duplicated work for re-transferring and re-pruning. Thus, we aim to prune the pretrained model only once in a self-supervised manner. When transferred to downstream applications, it needs to be robust to iterations without re-pruning and maintain the maximum transfer performance under such circumstances.

Currently, weight pruning techniques [7, 11, 14, 14, 26, 27, 33, 37] have achieve high weight reduction ratio on supervised learning models with little accuracy loss. However, few studies have been published in pruning for self-supervised learning models. The main difference between pruning for supervised and self-supervised learning models is the task. The pruning for supervised learning models goes with specific tasks defined by the labels, while a pretext task constrains that for self-supervised learning models. [3] proved that if the performance of the pruned subnetwork is maintained on the pretext task, the performance can also be kept after being transferred to downstream tasks. MoCo is outstanding among the unsupervised learning algorithms and sometimes can outperform its supervised learning counterparts by large margins. Currently, many studies [4,6] based on MoCo have surged and brought about improved self-supervised learning methods. As a result, we choose MoCo as the pruning object to provide some insights and experience for self-supervised pruning.

If we interpret pruning from another perspective, it aims to obtain a subnetwork with comparable performance and representation with the over-parameterized network. Fortunately, MoCo's mechanism drives its encoder and momentum encoder to have consistent representation, naturally benefiting the pruning process. However, both the encoder and momentum encoder participate in the updating strategy of pretraining MoCo so that hard pruning any one of them will generate a huge gradient or degrade the convergence. As a result, the proof in [3] is not feasible for MoCo. Sec. 3.1 gives a detailed description of the MoCo mechanism and the reasons why the typical pruning algorithms collapse in this case. Thus, the main problem is properly pruning MoCo's encoder or momentum encoder without harming the updating strategy and maintaining maximum transfer performance simultaneously.

To address this issue, we propose the Momentum Contrastive Pruning (MCP) method, which is a self-supervised

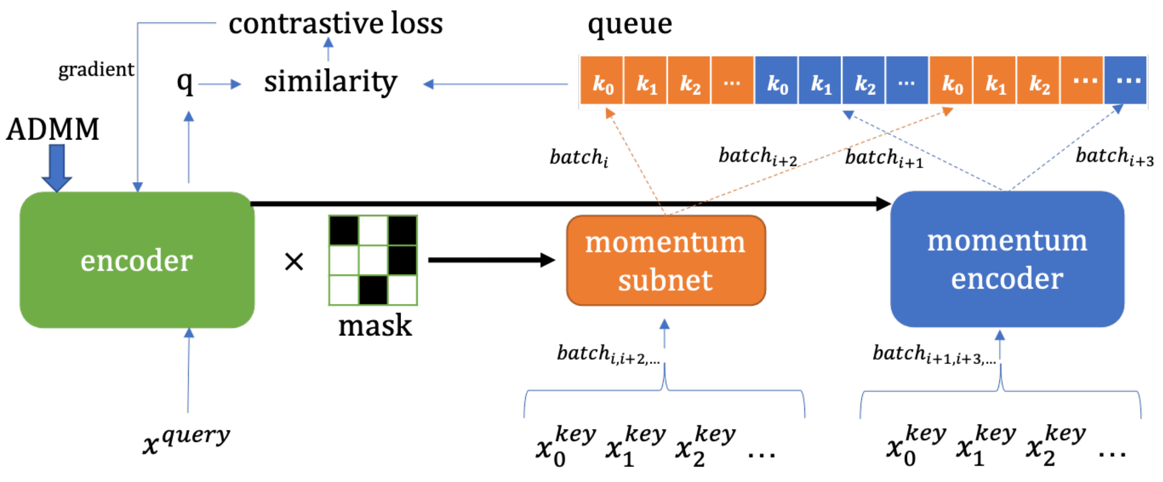

Figure 1. A framework of MCP algorithm. The momentum encoder is pruned while training MoCo with InfoNCE loss function. ADMM is employed on the encoder to constrain its sparsity and generate a dynamic mask. The momentum subnet is obtained by calculating the dot product of the mask and the momentum encoder. We maintain another momentum encoder that is not pruned and let it stand in for momentum subnet with a decayed frequency to alleviate the inconsistency between the representation of encoder and momentum subnet at the beginning of training. After training, the momentum subnet can be transferred to downstream tasks and then deployed directly.

pruning algorithm for MoCo with the following characteristics (Fig. 1):

- MCP takes the momentum encoder as the pruning target instead of the encoder. The reason is that removing the parameters of the encoder will destroy the pretrained model.
- The MCP is a pruning method with no fine-tuning [1]. Instead, the ADMM is used to gradually reduce the encoder weights of a preset amount to zero, and the subnet is obtained directly by the dynamic mask determined by the largest weights.
- An un-pruned momentum encoder is maintained during training to provide a smooth transition scheme at the beginning to solve the problem of representation inconsistency between the encoder and momentum subnet.

When using the MobileNetV2 as the encoder and transferred to CIFAR-10 [21] classification task, our momentum subnet achieves almost equal performance as the un-pruned MoCo when 92.7% of the parameters are pruned. Meanwhile, its inference speed is 13x faster. To the best of our knowledge, [3] is the only work on unsupervised pruning at present. As shown in Fig. 5, our performance outperforms theirs significantly.

[1]Self-supervised learning has two stages: pretrain (train a feature extractor on pretext tasks) and fine-tune (freeze the feature extractor and train the sequential layers for downstream tasks). Meanwhile, pruning algorithms also have two stages: pretrain (train an over-parameterized network) and fine-tune (prune and retrain it to restore accuracy). To avoid confusion, the 'fine-tune' in this paper refers to the retraining of pruning algorithms.

2. Related work

2.1. Self-supervised learning

Self-supervised learning is a subset of unsupervised learning methods. It refers to learning methods in which convolution networks are explicitly trained with pretext tasks. Pretext tasks are pre-designed tasks for networks to solve, and visual features are learned by learning objective functions of pretext tasks. A wide range of pretext tasks has been proposed. Examples include recovering the input under some corruption, *i.e.*, denoising auto-encoders [34], context auto-encoders [30], or cross-channel auto-encoders (colorization) [39]. Contrastive learning [12] refers to a kind of pretext task that attracts the positive sample pairs and repulses the negative sample pairs. Simple and effective instantiations of contrastive learning have been developed using siamese networks [1, 4, 5, 16, 38].

MoCo. MoCo [5, 16] is a kind of contrastive learning algorithm using siamese networks which maintains a queue of negative samples and turns one branch into a momentum encoder to improve the consistency of the queue. MoCo can outperform its supervised pretraining counterpart in seven detection/segmentation tasks on PASCAL VOC [9], COCO [24], and other datasets, sometimes surpassing them by large margins.

2.2. Network pruning

Pruning is a model compression approach [27] in which weights or nodes/filters are removed, typically by clamping them to zero. The regular process is training the original network, removing parameters, and then fine-tuning [14].

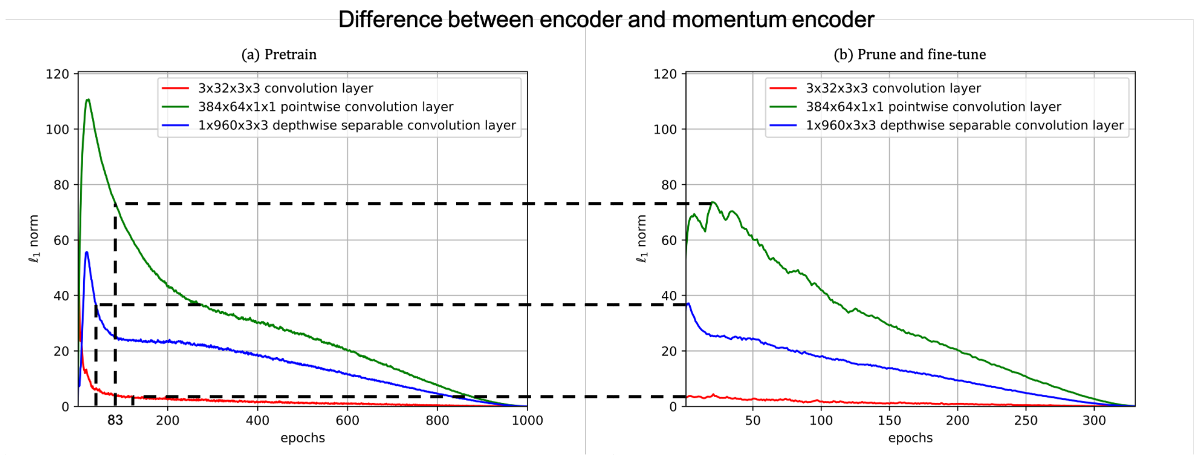

Figure 2. The weight difference of encoder and momentum encoder on three types of convolution layers. MobileNetV2 [31] is used as the structure of MoCo encoder and momentum encoder. (a) MoCo is pretrained for 1000 epochs. (b) Prune the encoder and momentum encoder in (a) and fine-tune them for 300 epochs. For a fair comparison, the difference values of the un-pruned layer are multiplied by the remaining proportions of the pruned layer.

In the early stage, researchers proposed fine-grained pruning [14, 15] by cutting off the connections within the weight matrix. However, such methods are not friendly to CPUs or GPUs and require dedicated hardware [29, 41] to support sparse matrix multiplication, which is highly demanding to design [35]. Later, some researchers proposed channel level pruning [18, 23, 25] by pruning the entire convolution channel based on some importance criteria (*i.e.* ℓ_1-norm) to enable acceleration on general-purpose hardware. However, these studies are mainly iterative heuristic pruning methods, therefore lacking guarantees on the weight reduction ratio and convergence time. To mitigate these limitations, [40] presented a systematic weight pruning framework of deep networks using ADMM. In this paper, we adopt filter pruning [22] which is a structured algorithm in all the experiments since unstructured pruning will lead to sparse weight matrices, which cannot achieve compression and speedup without dedicated hardware/libraries [13].

Self-supervised pruning. To the best of our knowledge, [3] is the only work on unsupervised pruning at present. They investigated the use of standard pruning methods developed primarily for supervised learning on unsupervised learning networks. They showed that pruned models obtained with or without labels reached comparable performance when retrained on labels. This suggests that pruning operates similarly for self-supervised and supervised learning. In addition, they found that pruning preserved the transfer performance of self-supervised subnetwork representations. They adapted rotation classification [10] and the "Exemplar" approach [8] as the self-supervised algorithm and pruned the network based on the lottery tickets hypothesis [8]. MoCo is outstanding among the unsupervised learning algorithms and sometimes can even outperform its supervised learning counterparts by large margins. Currently, many studies [4, 6] based on MoCo have surged and brought about improved self-supervised learning methods. As a result, we choose MoCo as the pruning object. However, typical pruning methods are improper for MoCo because the momentum encoder updates too slow to support the rapid precision recovery in fine-tuning after pruning.

3. Method

Our method is listed in five sections. Sec. 3.1 explains why traditional pruning algorithms are inappropriate for MoCo. Sec. 3.2 verifies that the momentum encoder is capable enough to act as an alternative to pruning object. Sec. 3.3 verifies that MoCo's mechanism can be used as a pruning constraint but has limitations. Sec. 3.4 describes that ADMM is employed to break such limitations by generating a mask. Sec. 3.5 proposes a smooth transition strategy to optimize the pruned effect of the momentum subnet.

3.1. Pruning problem caused by momentum

In MoCo, a query and a key are considered a positive sample pair if they originate from the same image and otherwise as a negative sample pair. During pretraining, the query of an image is encoded by the encoder, while the momentum encoder encodes the corresponding key. Its pretext task can be described as driving the query and key of a positive sample pair to be similar and that of a negative pair to be different. The hypothesis of MoCo is that good features can be learned by a large dictionary that covers a rich set of negative samples. So a queue is built to make the dictionary large, but it also makes it intractable to update the momentum encoder by back-propagation (the gradient should propagate to all samples in the queue). The momentum update strategy is proposed by He *et al.* [16] to address this issue. The parameters of encoder f_{k} is denoted as θ_{k} and those of momentum encoder f_{q} as θ_{q}. θ_{k} is updated by

$$\theta_{\mathrm{k}} \leftarrow m\theta_{\mathrm{k}} + (1-m)\theta_{\mathrm{q}}, \tag{1}$$

where $m \in [0,1)$ is a momentum coefficient normally set to 0.999 since a relatively large momentum works better than a smaller one. During pretraining, with the decrease of learning rate (stages learning rate schedule in MoCoV1 [16] and cosine learning rate schedule in MoCoV2 [5]) and the convergence of encoder, the fluctuation of θ_{q} gradually slows down. A large momentum makes θ_{k} slowly approach θ_{q}, and after sufficient pretrain iterations, they should be consistent with each other. This process is depicted in Fig. 2(a), which shows the weight difference between the encoder and momentum encoder. Three types of convolution layers at the head, middle, and tail of the network are selected. The weight difference is calculated by the ℓ_1-norm: $\|\theta_{\mathrm{q}_i} - \theta_{\mathrm{k}_i}\|_1$, where q_i and k_i denote the i-th layer of the encoder and momentum encoder.

However, this consistency is fragile and sensitive to gradient changes. The pruning operation removes parameters and reduces accuracy. Thus the subnetwork produces a considerable gradient in the following fine-tuning. For the typical pruning methods, fine-tuning can restore accuracy in a few iterations. But for MoCo, a weight fluctuation on the encoder will destroy the consistency to a great extent, almost as much as starting from scratch. It needs many iterations to reduce the difference due to the large momentum. Fig. 2(b) shows the weight difference between encoder and momentum encoder during fine-tuning the pruned model of the last checkpoint in Fig. 2(a). The difference is pulled up and takes 300 epochs to converge again.

Table. 1 shows the comparison between typical pruning methods on MoCo with training a lightweight network from scratch. The first experiment trains a MoCo for 1000 epochs. The second experiment prunes the pretrained MoCo's encoder and fine-tunes it for 300 epochs. The third experiment trains a lightweight MoCo with the same architecture as the pruned encoder from scratch for 300 epochs. The first and second experiments show that the pruned MoCo achieves 82.870% top-1 accuracy when transferring to CIFAR-10 classification, which is 5.44% lower than the pretrained MoCo. Meanwhile, there is little accuracy difference between the pruned model and the model trained from scratch. This means the pretrained model is barely functional.

3.2. An alternative for pruning object

Sec. 3.1 concludes that traditional pruning algorithms cannot be applied to MoCo's encoder. As a result, We circumvent this by finding an alternative for pruning object.

Model	#FLOPs	#Params	Epoch	Accuracy (%)
Un-pruned	312.86M	2.24M	1000	88.310
Pruned	92.30M	0.56M	300	82.870
Trained from scratch	92.30M	0.56M	300	81.620

Table 1. Comparison of typical pruning methods on MoCo with training a lightweight network from scratch. All models are trained on CIFAR-10 [21] use the MobileNetV2 structure as the encoder. The implementation details are in Sec. 4.1.

MoCoV1 [16] and MoCoV2 [5] show that the representations of the pretrained encoder can be well transferred to the downstream tasks. What they did not mention is that after pretraining, the momentum encoder also has sufficient capacity. Fig. 2(a) shows that θ_{q} and θ_{k} gradually assimilate after enough iterations. The goal of pruning is to obtain a subnetwork with comparable performance and representation with the original network. So the consistency between momentum encoder and encoder provides the premise for momentum encoder as a pruning object. However, in practice, there will not be sufficient pretraining to ensure such consistency. So a simple verification experiment is designed to show that despite scant pretraining iterations and large pretraining loss, the momentum encoder still has equal transferability as the encoder, as shown in Table. 2. Thus, if there is a way to reduce the parameters of the momentum encoder while maintaining its transferring performance, the momentum encoder can be pruned instead of the encoder.

Pretrain Epoch	Pretrain Loss	Accuracy (%)	
		Encoder	Momentum Encoder
50	8.6492	62.670	61.510
200	7.4439	79.330	79.660
800	6.8876	88.150	88.230
1000	6.8912	88.310	88.420

Table 2. Performance of the encoder and momentum encoder transferred to CIFAR-10 classification after pretrained for 50, 200, 800, and 1000 epochs, respectively. All models used MobileNetV2 as the encoder.

3.3. The intrinsic pruning constraint of MoCo

Two properties of MoCo contribute to the consistency between encoder and momentum encoder. One is the updating strategy (Sec. 3.1), the other is the contrastive loss function, InfoNCE [28] formulated as

$$\mathcal{L}_{q,k^+,\{k^-\}} = -\log \frac{\mathrm{e}^{q\cdot k^+/\tau}}{\mathrm{e}^{q\cdot k^+/\tau} + \sum\limits_{k^-} \mathrm{e}^{q\cdot k^-/\tau}}. \tag{2}$$

Here q denotes a query, k^+ denotes a positive (similar) key sample, k^- denotes a negative (dissimilar) key sample, and τ is a temperature hyper-parameter. InfoNCE enhances the attraction of positive pairs and repulses the negative pairs to benefit the consistency between the encoder and momentum encoder and ensure convergence. This is also a shared target with the updating strategy and pruning. Based on this, we suppose that MoCo's mechanism can be used as a pruning constraint and design a simple verification experiment. The updating strategy is modified as

$$\theta_\mathrm{k} \leftarrow m\theta_\mathrm{k} + \phi \odot (1-m)\theta_\mathrm{q}. \tag{3}$$

Here $\phi \in \{0,1\}^{|\theta_q|}$ is a mask on encoder parameters such that the initialization on momentum encoder is $\phi \odot \theta_\mathrm{q}$, where $\odot$ is the element-wise product. Under such updating strategy, we obtain a momentum subnet that only possesses part of the encoder parameters. Besides, two other training schemes are conducted for comparison. The original size MoCo trained from scratch shows the due ability of the un-pruned network. The lightweight MoCo (with the same structure as momentum subnet) trained from scratch provides a lower limit for an effective pruned model. For each model, we evaluate the transfer performance on both random initialization and after 500 epochs training.

The results are shown in Table. 3. From the third row of Table. 3, we can find that such a simple partially updating strategy improves the momentum subnet from complete chaos (18.32%) to 48.57%. Despite huge accuracy reduction, it proves that MoCo's mechanism is able to act as a pruning constraint. Due to two limitations, it loses 34.36% accuracy compared with the lightweight model in the second row. First, the mask is random. It cannot control the part of elements to be deleted, which is normally the smallest ones. Second, the masked elements in the encoder are large, which makes it challenging to represent consistency with the momentum subnet.

Model	#FLOPs	#Params	Epoch	Accuracy (%)
Train from scratch	312.86M	2.24M	0 500	18.460 86.180
Train from scratch	92.30M	0.56M	0 500	18.400 82.930
Momentum Subnet	92.30M	0.56M	0 500	18.320 48.570

Table 3. Effectiveness verification of using the MoCo's mechanism as the pruning constraint. All models are trained on CIFAR-10 and use the MobileNetV2 structure as the encoder.

3.4. ADMM on the encoder

To break through the limitations mentioned above and strengthen the constraint, an ADMM variant is applied to the encoder. [40] presented a systematic weight pruning framework for deep neural networks (DNNs) using ADMM. The training process gradually reduces the weights of a preset amount to zero in the encoder. Then this part of weights can be deleted or marked in a mask. By employing ADMM on MoCo, the encoder can provide a dynamic mask ϕ, which can be applied to the momentum encoder.

Here we introduce the derivation of ADMM for pruning. Considering an N-layer encoder, we denote the collection of weights in the i-th (convolution or fully-connected) layer of the encoder as θ_{q_i}. Our goal is to minimize the InfoNCE and constrain the weight's cardinality in each layer, then prune the weights of the momentum encoder according to the mask. The problem can be formulated as

$$\begin{aligned} \min_{\{\theta_{q_i}\}} \quad & f\left(\{\theta_{q_i}\}\right) \\ \text{s.t.} \quad & \operatorname{card}\left(\theta_{q_i}\right) \le l_i, i = 1, \ldots, N, \end{aligned} \tag{4}$$

where card$(\cdot)$ returns the number of nonzero elements of its matrix argument and l_i is the desired number of weights in the i-th layer of the momentum encoder. The above problem can be rewritten in the ADMM form as

$$\begin{aligned} \min_{\{\theta_{q_i}\}} \quad & f\left(\{\theta_{q_i}\}\right) + \textstyle\sum_{i=1}^N g_i\left(\mathbf{Z}_i\right) \\ \text{s.t.} \quad & \theta_{q_i} = \mathbf{Z}_i, i = 1, \ldots, N, \end{aligned} \tag{5}$$

in which Z_i can be regarded as a restriction as a sparsity constraint on θ_{q_i}. $g_i(\cdot)$ is an indicator function, *i.e.*,

$$g_i\left(\theta_{q_i}\right) = \begin{cases} 0 & \text{if } \theta_{q_i} \ne \mathbf{Z}_i, \\ +\infty & \text{otherwise} . \end{cases} \tag{6}$$

The augmented Lagrangian of the above optimization problem is given by

$$\begin{aligned} L_\rho\left(\{\theta_{q_i}\}, \{\mathbf{Z}_i\}, \{\mathbf{U}_i\}\right) = f\left(\{\theta_{q_i}\}\right) + \sum_{i=1}^N g_i\left(\mathbf{Z}_i\right) \\ + \sum_{i=1}^N \frac{\rho_i}{2} \left\|\theta_{q_i} - \mathbf{Z}_i + \mathbf{U}_i\right\|_\mathrm{F}^2 - \sum_{i=1}^N \frac{\rho_i}{2} \left\|\mathbf{U}_i\right\|_\mathrm{F}^2, \end{aligned} \tag{7}$$

where $\mathbf{U}_i$ has the same dimension as θ_{q_i} and is calculated by dividing Lagrange multiplier by ρ_i, corresponding to the constraint $\theta_{q_i} = \mathbf{Z}_i$. The positive scalars $\{\rho_1, \ldots, \rho_N\}$ are penalty parameters and $\|\cdot\|_\mathrm{F}^2$ denotes the Frobenius norm. Then we solve the above function in the k_th iteration as

$$\left\{\theta_{q_i}^{k+1}\right\} := \underset{\{\theta_{q_i}\}}{\arg\min} \;\; L_\rho\left(\{\theta_{q_i}\}, \left\{\mathbf{Z}_i^k\right\}, \left\{\mathbf{U}_i^k\right\}\right), \tag{8}$$

Algorithm 1 Pseudocode of MCP in a PyTorch-like style

```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# prune_rate: preset prune
# frq: stand-in frequency of momentum encoder
# dr: decay rate for stand-in frequency
# p: indicator

f_k.params = f_q.params # init momentum encoder
p = frq # init indicator

for x in loader: # load a minibatch x with N samples
    x_q = aug(x) # a randomly augmented version
    x_k = aug(x) # another randomly augmented version

    # select momentum encoder or subnet in the iteration
    use_momentum_encoder = True if p>0 else False
    # reassign p and attenuate frq
    if p == 0: frq -= dr; p = frq if frq > 0
    # get mask from encoder
    threshold = f_q.sort()[f_q.size*prune_rate]
    phi = where(f_q<threshold, f_q, 0)

    q = f_q.forward(x_q) # queries: NxC
    if use_momentum_encoder:
        k = f_k.forward(x_k) # keys: NxC
    else: # use momentum sbunet
        f_ks = f_k.params.dot(phi)
        k = f_ks.forward(x_k) # keys: NxC
    k = k.detach() # no gradient to keys
    # positive logits: Nx1; negative logits: NxK
    l_pos = bmm(q.view(N, 1, C), k.view(N, C, 1))
    l_neg = mm(q.view(N, C), queue.view(C, K))
    # contrastive loss and ADMM loss, Eq. (7)
    loss = InfoNCELoss(l_pos, l_neg)+ ADMMLoss(f_q, phi)

    # SGD update: query network
    loss.backward(); update(f_q.params)
    # momentum update: key network
    f_k.params = m*f_k.params+(1-m)*f_q.params
    # update dictionary
    enqueue(queue,k) # enqueue the current minibatch
    dequeue(queue) # dequeue the earliest minibatch
    p -= 1 # p decreases after each iteration
```

$$\{\mathbf{Z}_i^{k+1}\} := \underset{\{\mathbf{Z}_i\}}{\arg\min}\ L_\rho\left(\{\theta_{q_i}^{k+1}\}, \{\mathbf{Z}_i\}, \{\mathbf{U}_i^k\}\right), \tag{9}$$

$$\mathbf{U}_i^{k+1} := \mathbf{U}_i^k + \theta_{q_i}^{k+1} - \mathbf{Z}_i^{k+1}. \tag{10}$$

Following [2], we solve the Eq. (9) by keeping the l_i elements of $\theta_{q_i}^{k+1} + \mathbf{U}_i^k$ with the largest magnitudes and setting the rest to 0. Note that the dynamic mask is $\phi \leftarrow (\mathbf{Z}^k > 0)$ after the k_{th} iteration.

3.5. Smooth transition

ADMM breaks through the first limitation by providing a dynamic mask. However, the second limitation remains. ADMM needs some iterations to reduce the parameters to zero, so the encoder parameters are not small enough at the beginning of training. At this moment, the representation of momentum subnet is inconsistent with the encoder, which brings about the problem mentioned in Sec.3.1.

We propose a smooth transition technique to address this issue. We maintain another momentum encoder that is not pruned and let it stand in for momentum subnet with a decayed frequency. With the impact of ADMM on the momentum encoder, the inconsistency gradually weakens, and the momentum subnet becomes the main strength for key generating. Algorithm 1 provides the pseudocode of MCP. The hyper-parameter *frq* denotes the stand-in operation frequency of the momentum encoder after each operation of the momentum subnet. *dr* denotes the decay rate for the stand-in frequency. A larger *frq* and smaller *dr* represent a smoother transition to momentum subnet. When *frq* decreases to 0, all keys are encoded by momentum subnet. In Fig. 1, the momentum encoder and subnet operate in turn, at this moment $frq = 1$ and $dr = 0$. Note that in Algorithm 1, the momentum subnet is obtained by $\phi \odot \theta_q$ each iteration when it needs to be used. This smooth transition scheme significantly alleviates the inconsistency at the beginning of training. Together with ADMM, it brings up a mature pruning method for MoCo.

4. Experiments

4.1. Implementation details

MoCo. We follow closely the setup of MoCoV2 [5]. The temperature hyper-parameter τ in Eq. (2) is set to 0.2. The momentum value m in Eqs. (1) and (3) is set to 0.999. An SGD optimizer with 0.0001 weight decay and 0.9 momentum is used. The input image size is 224×224, and the mini-batch size is 256 trained on 8 GPUs. The learning rate is initialized to 0.03 and subjects to cosine learning rate schedule. Most of the experiments are conducted on the CIFAR-10 dataset, though we also report some results on CIFAR-100. When the pruned model is transferred to the downstream task, a supervised linear classifier (usually a fully-connected layer) is trained with the pruned model fixed. Then the retrained model is evaluated with top-1 classification accuracy. For all experiments, the MoCo is pretrained for 1000 epochs, and then the linear classifier is retrained for 300 epochs to get the final results.

Pruning. The last checkpoint of the pretrained model is used for pruning. We set the same pruning ratio to each layer and prune the filters [27] according to the preset ratio. The final pruning proportion of the network is less than the preset ratio because the change of output channel number in one layer will affect the input channel number in the next layer. We train MoCo with ADMM for 450 epochs, during which every 15 epochs is followed by an ADMM epoch. The learning rate is set to 0.03 in each ADMM epoch multiplied by 0.1 for every 5 epochs. After pruning, the momentum subnet in the last checkpoint is used for linear classification training.

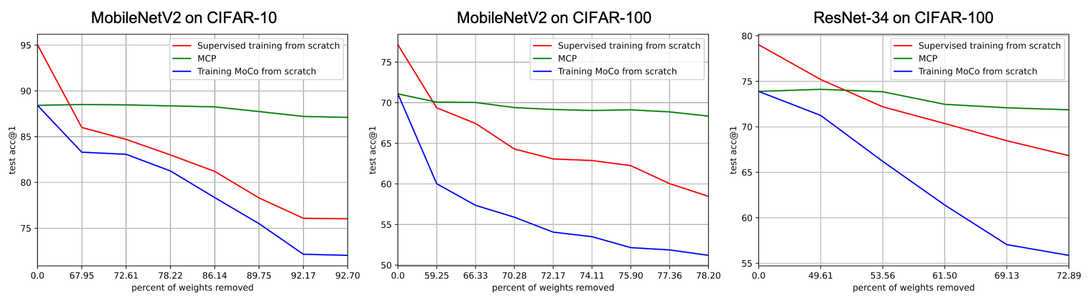

Figure 3. The performance comparisons between MCP under various pruning rates and the un-pruned schemes. It includes three combinations of training MobileNetV2 and ResNet34 [17] on CIFAR-10 and CIFAR-100. The x-axis indicates the amount of parameters removed. The red line denotes training a lighter model in a supervised manner from scratch. The blue line denotes training a lighter model with MoCo's manner from scratch.

4.2. Results

Performance of momentum subnet. In this experiment, we are interested in the features transferability as we vary the amount of pruned weights in the representation function. In particular, we train linear classifiers for CIFAR-10 and CIFAR-100 classification tasks with final representations given by a pretrained momentum subnet. The results are given in Fig. 3. A specific x value means $x\%$ parameters are pruned in MCP, and the un-pruned schemes use the same architecture as the pruned model. The left figure in Fig. 3 shows that pruning up to 90% weights of MobileNetV2 does not significantly deteriorate the resulting features quality (MobileNetV2 is a lightweight network with 312.86M FLOPs and 2.23M params. After 92.70% params being pruned, only 25.43M FLOPs and 161.16K params are left, and reduce the inference time by 91.88%.). In addition, we observe that sometimes, pruning even improves the transfer performance, such as when Resnet-34 pruned for 53.56% on CIFAR-100 (in the right figure of Fig. 3). One possible reason is that pruning removes the task-specific information features and leads to better transferability. The comparison in all three figures shows that when we train the networks with the same structure, the performance of MoCo and supervised training from scratch decreases as the structure becomes smaller, but MCP can maintain its performance.

Compare with transfer pruning. One contribution of this paper is a scheme of pruning before transferring (pretrain pruning), while the common practice is transferring before pruning (transfer pruning). In this experiment, we focus on the comparison between them. In Fig. 4, we evaluate the performance of transfer pruning and pretrain pruning on the CIFAR-100 classification task with MobileNetV2. When the pruning ratio is small, transfer pruning outperforms pretrain pruning slightly. However, pretrain pruning gets better performance with a large pruning ratio. In industrial scenarios, this enables our MCP to prune the model only once without regard to the iterations and types of the downstream task. In addition, the ability of self-supervised pruning is not fully reflected in this section. In order to make a quantitative comparison and prove the effectiveness of our method, the same training dataset is used for self-supervised pretrain and transferring. Actually, with a dataset that is larger and more diverse, we assume our MCP method can achieve more powerful and robust performance on downstream tasks.

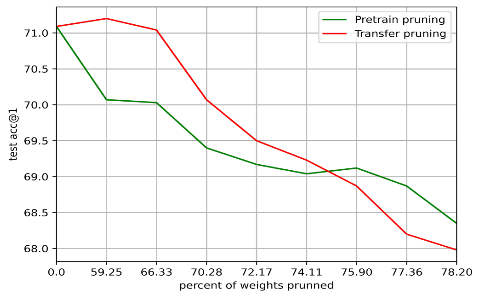

Figure 4. The performance comparison between transfer before pruning (transfer pruning) and prune before transferring (pretrain pruning).

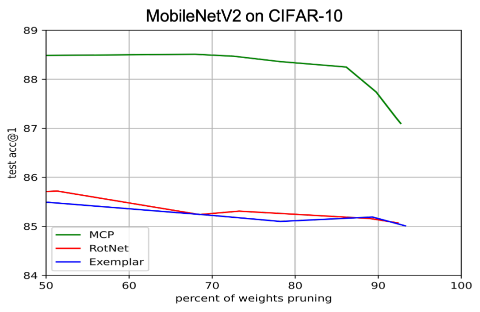

Figure 5. The performance comparison between MCP and other unsupervised pruning method. All models are trained on CIFAR-10 and use the MobileNetV2 structure as the encoder.

Compare with other unsupervised pruning. To the best of our knowledge, [3] is the only work on unsupervised pruning at present. They adapted rotation classification (RotNet) [10] and "Exemplar" approach [8] as the self-supervised algorithm and pruned the network based on the lottery tickets hypothesis [8]. Although they reported results on CIFAR-10, unstructured pruning is used on ResNet-50, which requires dedicated hardware to compress and accelerate. For fair comparisons, we implement the RotNet and the Exemplar with MobileNetV2 and apply structured pruning to them. The results are shown in Fig. 5. All three methods considerably retain their transferability with the increase of prune ratio. However, the overall performance of MCP surpasses the other two algorithms by a large margin. Our method provides experience and insight into the pruning for unsupervised learning methods developed on MoCo.

4.3. Ablation study

frq	*dr*	ρ	ADMM loss	MoCo loss	acc@1(%)
		0.1	0.127	5.314	88.13
100	1	0.01	0.315	5.308	**88.45**
		0.001	2.807	5.394	87.93
100	5	0.01	0.304	5.310	88.32
1	0	0.01	0.367	5.409	87.47
0	0	0.01	0.319	5.541	84.05

Table 4. The ablation studies for ρ, *frq* and *dr*. All models are trained on CIFAR-10 and use a MobileNetV2 structure with a fixed pruning ratio of 90%. It should be noticed that the MoCo losses are less than those in Table. 2 because a shorter queue for negative samples is adopted.

In this section, the ablation studies for ρ in Eq. (7) and *frq* and *dr* in smooth transition are carried out. ρ is a hyper-parameter in ADMM to constrain the parameter sparsity. A large ρ increases such sparsity in the network that brings the parameter closer to zero, causing the network to converge worse on the loss function (InfoNCE in MoCo). In the upper part of Table. 4, different values of ρ are tested. The results indicate that a compromised ρ can balance the ADMM loss and MoCo loss, thus facilitating optimal transfer performance. We can also conclude from the MoCo loss column that a smaller ρ enlarges the representation difference between encoder and momentum subnet. The setting of this hyper-parameter depends on the datasets and network structure. When training on CIFAR-10 with MobileNetV2 structure, it's recommended to be set to 0.01. For other datasets and network structures, it can be determined by binary search [40]. $\rho = 0.01$ is used for the rest of the ablation study.

The hyper-parameters *frq* and *dr* together control the length and decay rate of the smooth transition. A larger *frq* and smaller *dr* indicate a smoother transition of operation from the momentum encoder to the momentum subnet. We experiment with several representative combinations, and the results are shown in the lower part of Table. 4. $frq = 1$ and $dr = 0$ means that momentum encoder and momentum subnet always alternate to produce keys. $frq = 0$ and $dr = 0$ denotes all keys are encoded by momentum subnet with no smooth transition. From the second and the fourth row, we can find that a smoother transition ($frq = 100$ and $dr = 1$) can obtain a better performance.

4.4. Limitations and discussion

This work lacks the following experiments because of the limitation of computing power and time. 1) Trials on the ImageNet dataset. 2) Transferring the pretrained model to other downstream tasks such as segmentation and detection. 3) Transferring the pretrained subnet from a large unsupervised dataset to a small supervised dataset to highlight the advantages of contrastive learning.

In this paper, we mainly study unsupervised pruning based on MoCo. It shows that for the pruned subnetwork, the performance on the pretext task guarantees the performance after being transferred to downstream tasks. Therefore, the lack of the above experiments will not affect the effectiveness of our method.

In addition, we pursue pruning in an unsupervised manner because it allows the backbone to be pruned only once. The pruned subnet can be directly deployed when transferred to the downstream tasks. However, many downstream tasks need extra components. For example, an additional decoder is required in segmentation, and a feature pyramid network and some heads are required in detection. These components account for a large part of the network. For such tasks, it may be better to prune the backbone with these components together after transferring.

5. Conclusion

In this paper, we propose a self-supervised pruning algorithm MCP for solving the problem that the traditional pruning algorithm is not applicable for MoCo. Our MCP can prune the momentum encoder properly by applying ADMM on the encoder and the smooth transition. Experiments show that MCP's performance outperforms other unsupervised pruning algorithms significantly. In industrial scenarios, MCP is able to prune the model only once without regard to the iterations and types of the downstream task. Then the model can be directly deployed when transferred to downstream tasks. At present, many works based on MoCo have surged and brought about improved self-supervised learning methods. We hope MCP can provide some insights and experience for these algorithms.

References

[1] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019. 1, 2

[2] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011. 6

[3] Mathilde Caron, Ari Morcos, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Pruning convolutional neural networks with self-supervision. *arXiv preprint arXiv:2001.03554*, 2020. 1, 2, 3, 8

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 1, 2, 3

[5] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1, 2, 4, 6

[6] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 1, 3

[7] Xiaoliang Dai, Hongxu Yin, and Niraj K Jha. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. *IEEE Transactions on Computers*, 68(10):1487–1497, 2019. 1

[8] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1734–1747, 2015. 3, 8

[9] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 2

[10] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018. 3, 8

[11] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *arXiv preprint arXiv:1608.04493*, 2016. 1

[12] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006. 2

[13] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016. 3

[14] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1, 2, 3

[15] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015. 3

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 1, 2, 4

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[18] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017. 3

[19] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, pages 4182–4192. PMLR, 2020. 1

[20] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018. 1

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Tech. Rep.*, 2009. 2, 4

[22] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 3

[23] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2178–2188, 2017. 3

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2

[25] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017. 3

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

[27] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pages 2498–2507. PMLR, 2017. 1, 2, 6

[28] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 1, 5

[29] Subhankar Pal, Jonathan Beaumont, Dong-Hyeon Park, Aporva Amarnath, Siying Feng, Chaitali Chakrabarti, Hun-Seok Kim, David Blaauw, Trevor Mudge, and Ronald Dres-

linski. Outerspace: An outer product based sparse matrix multiplication accelerator. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 724–736. IEEE, 2018. 3

[30] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016. 2

[31] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 3

[32] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer, 2020. 1

[33] Frederick Tung, Srikanth Muralidharan, and Greg Mori. Fine-pruning: Joint fine-tuning and compression of a convolutional network with bayesian optimization. *arXiv preprint arXiv:1707.09102*, 2017. 1

[34] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008. 2

[35] Hanrui Wang, Jiacheng Yang, Hae-Seung Lee, and Song Han. Learning to design circuits. *arXiv preprint arXiv:1812.02734*, 2018. 3

[36] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018. 1

[37] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5687–5695, 2017. 1

[38] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6210–6219, 2019. 2

[39] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 2

[40] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018. 1, 3, 5, 8

[41] Zhekai Zhang, Hanrui Wang, Song Han, and William J Dally. Sparch: Efficient architecture for sparse matrix multiplication. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 261–274. IEEE, 2020. 3

[42] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6002–6012, 2019. 1