A. Implementation Details

A.1. Linear Classification.

For TKC on ResNet-50 [7], we freeze the ResNet-50 backbone and train a linear classifier after the frozen features from the global pooling layer. We use the student network as a pre-trained model. The classifier is trained for 100 epochs, with initialized learning rate $lr = 30$. We set momentum as 0.9, weight decay as 0, and decay the learning rate by 0.1 at the 60th epoch and 80th epoch. The batch size is set to 256 on 8 NVidia-1080ti GPUs.

A.2. Based on BYOL

Pretext Training. In the paper, we have implemented an experiment based on BYOL [4] to show that TKC can improve different methods (in Table 2). We use the BYOL baseline based on a pytorch implementation in Momentum^2 teacher [10]. Their code is publicly available at `https://github.com/zengarden/momentum2-teacher`.

We use momentum SGD with momentum 0.9 and weight decay 1e-4. We train both the BYOL baseline and our TKC for 100 epochs, the basic learning rate is 0.05. We use a warm-up stage at the beginning of training for 10 epochs, and then cosine decays the learning rate. The batch size is 256 on 8 NVidia-1080ti GPUs. The data augmentation and the architecture are the same as the original paper [4], except that we use batch normalization instead of SyncBN. The MLP projection head consists of two linear layers, with a batch norm layer and a ReLU layer between them. Our TKC+BYOL shares the same setting with the baseline, we set h as 3, and also use a symmetrized loss.

Linear Classification. The setup of linear classification is also following the reproduction in [10]. We fetch out the teacher encoder and freeze its backbone. Then we train a classifier consisting of a linear layer and a batch norm layer, following the global average pooling layer in the backbone. We train for 5 epochs. This reproduction is not strictly reimplemented the results in BYOL [4], thus we do not compare this result with other methods. The results in Table 2 show that TKC has good scalability, and can improve different methods.

A.3. Based on AlexNet

Pretext Training. We adopt the AlexNet [8] implementation in Deep Clustering [1] as the backbone, and additionally append a two-layers MLP behind it, following MoCo v2 [2]. We train the model on 4 NVidia-1080ti GPUs with 1024 batch size. The learning rate in initialized as 0.24, and cosine decayed for 200 epochs. We set τ as 0.2, α as 0.9, following MoCo v2 [2]. Differently, we set the number of negative samples K as 8192 to accelerate training.

Linear Classification. In this section, we freeze the backbone of the student model and train a classifier containing a linear layer with 1000 output dimensions after the backbone for 100 epochs. The initial learning rate is set as 0.01, and decayed by 0.1 at the 60 th epoch and 80 th epoch.

A.4. Linear Detection and Segmentation

Object Detection on PASCAL VOC. We have performed linear detection on PASCAL VOC in Table 5 to show that TKC learns better representations for object detection. We use Faster R-CNN [11] detector based on Detectron2 [12]. The backbone ends with conv5 stage and is set frozen. The training hyperparameters are kept consistent with finetuning, except the backbone is frozen. We train it for 48k iterations on VOC07+12 trainval. The initial learning rate is 0.02 and decayed at 36k and 44k iterations. The warmup stage lasts for 200 iterations.

Object Detection and Instance Segmentation on COCO. We have also implemented linear detection on COCO. We use Mask R-CNN [6] detector based on Detectron2 [12], and synchronously train the object detection head and the instance segmentation head following [5], meanwhile we freeze the backbone. We also use a 2x scheduler the same as finetuning, where we train it for 180k iterations. The size of the shorter side is in [640,800] pixels during training and is fixed as 800 at inference. We use 8 GPUs and 16 batchsize.

B. Architecture of History Bank

We use *history bank* as a more effective implementation of the *temporal teacher*. Fig 4 illustrates the architecture and mechanism of history bank. The history bank is a matrix with $size\ of\ \mathcal{D}$ rows and h columns. $\mathcal{D}$ is the train set, h is the number of *temporal teachers*. A row of *history bank* stores the features from the same image but different teachers, while a column of *history bank* stores the ones from the same teacher but different images. This matrix can be saved at CPU memory, with no need to allocate the GPU memory. We illustrate the *history bank* by the blue cube in Fig 4.

For a sample x, we first get the feature r_0^S from the student model, and the feature r_n^T from the EMA teacher model. Then we fetch out all the features from the same image x from the *history bank* as $z_{n-h}^T \sim z_{n-1}^T$, as shown in the middle in the Fig 4. For the implementation based on MoCo, we also fetch out the negative samples from the *history bank*. For each $z_j^T j \in [n-h, n-1]$, the corresponding negative features are randomly selected from the same column in the *history bank* as r_j^-. For the implementation based on BYOL, there is no need of the negative samples. *History bank* is an effective implementation of *temporal teacher*, the training procedure is the same as in the paper.

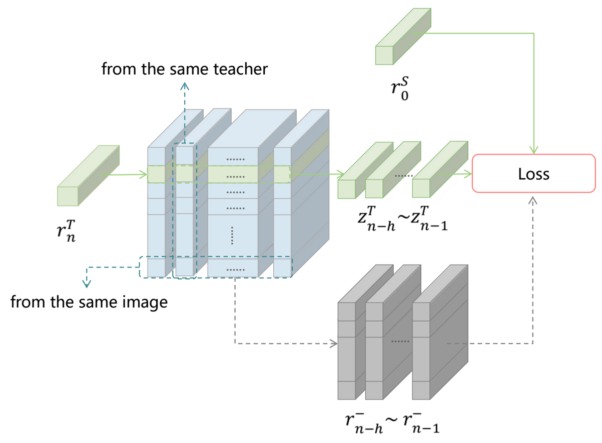

Figure 4. **The architecture of history bank.** History bank is an effective implementation of the temporal teacher. A row in the history bank stores the features from the same image, a column in the history bank stores the features from the same teacher. The green cubes indicate positive features, the grey cubes indicate negative features.

C. Computational Cost

We compare the computational cost of the two methods to show the efficiency of TKC. TKC use the *history bank* to approximate the *temporal teachers*. *History bank* stores the features of the recent epochs in CPU memory, and only part of them corresponding to the current batch will be dumped to GPU memory. This optimization can avoid duplicate forwarding, and the features from *history bank* are the same as forwarding the image into the *temporal teachers*.

Method	GPU	batchsize	GPU·Time/Epoch	memory/GPU
MoCo v2	8× 2080ti	256	3.4h	4.9G
TKC	8× 2080ti	256	5.0h	5.0G

Table 10. Computational cost. We report the time the GPU memory cost of our method and MoCo v2 baseline.

TKC has not too many additional costs thanks to *history bank*. As shown in Table 10, TKC has similar memory allocation with MoCo v2, which indicates that TKC has no special requirements for the capacity of machines. The time cost is higher than MoCo v2 for 47 %, which is mainly from the matrix multiplications between temporal features $r_j^T, j \in [n-h, n-1]$ and r_0^S, the knowledge transformer, and the data transport between memory and GPU memory.

D. Further analysis

D.1. More Experiment on AlexNet Backbone

In this section, we implement MoCo v2 based on AlexNet [8] backbone to show that the temporal knowledge introduced by TKC can improve instance discrimination methods on the different backbone. In Table 3, we only compare TKC with SOTA methods, here we supplement the result of MoCo v2. The MoCo v2 baseline follows the same setup with TKC. As shown in Table 11, TKC outperforms MoCo v2 baseline for all conv1 to conv5. The results from the bottom layers have more improvements, the results on conv4 especially surpass MoCo v2 for 7.2 %. This may because the temporal knowledge brought by the previous teachers can introduce the consistency between different epochs. The consistency can especially mitigate the dramatically changes and accelerates the convergence of the bottom layers.

Method	conv1	conv2	conv3	conv4	conv5
MoCo v2 [2]	17.2	26.6	36.5	39.0	**42.8**
TKC	**20.3**(+3.1)	**34.2**(+7.6)	**42.6**(+6.1)	**46.2**(+7.2)	**44.0**(+1.2)

Table 11. Comparison with MoCo v2 baseline on AlexNet.

D.2. Relation to No EMA Methods

Some recent works [3] claim that the EMA encoder is not necessary to prevent model collapse. However, their works have no conflict with our work. SimSiam has shown that the stop gradient but not the EMA encoder is the key to prevent model collapse, but it also admits that the EMA encoder can improve accuracy (in the last paragraph in Section 2). Table 4 in [3] reveal that SimSiam with EMA encoder (BYOL) surpasses it by 3.0% for 800 epochs training, which shows that EMA encoder is important to learn good representations. However, the EMA encoder is not good enough, for it can not learn the temporal consistency between different training stages, as shown in our works.

We note that all the reproductions in [3] applies the symmetrized loss. Section 4.6 in [3] shows that the symmetrized loss can boost the accuracy for 3%, and the computational cost has also doubled. So comparing TKC which is asymmetric to the symmetric methods [4, 3] is unfair. Our improvement based on BYOL shows that the temporal knowledge is orthogonal to the symmetric loss. We consider providing the result of symmetric TKC in the next version and compare it with the symmetric methods.

D.3. Does TKC improve the consistency?

In this section, we visualize the inconsistency during training and indicate that TKC can improve the stability during pretext training. We randomly select some images and compare the stability of each sample between TKC and MoCo v2 baseline.

Firstly, We define stability of a sample as the cosine similarity between current teacher output r_n^T and the counterpart in the last epoch z_{n-1}^T. The formulation is:

$$stable(x) = r_n^T \cdot z_{n-1}^T \tag{8}$$

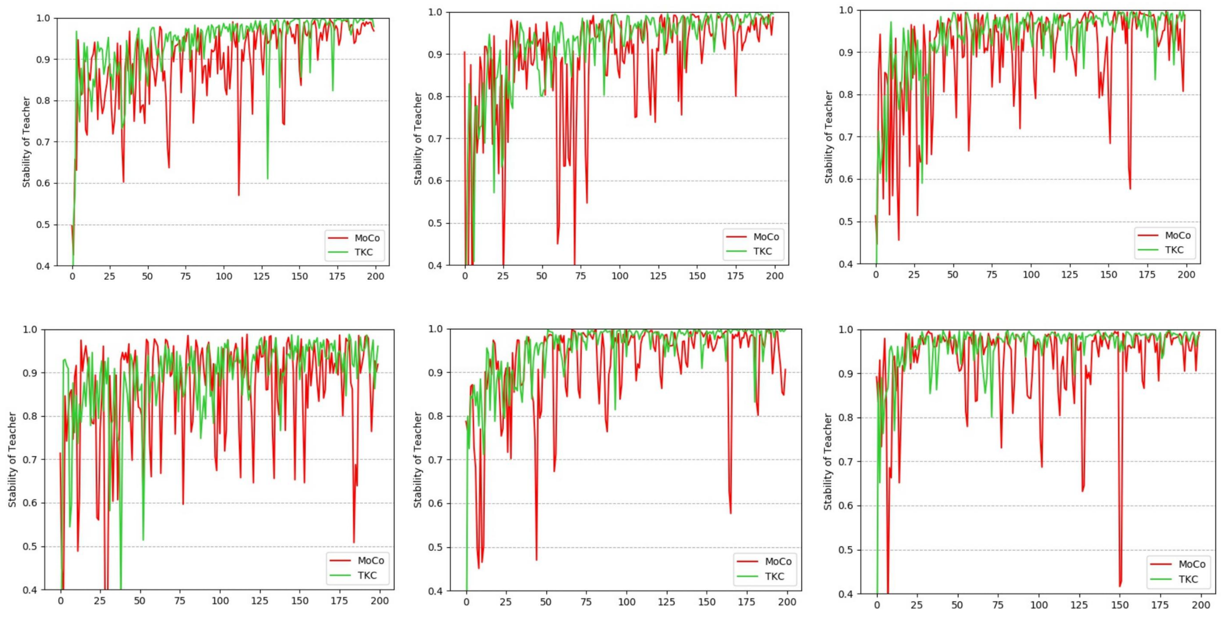

Figure 5. Comparison of stability between two methods in some randomly selected samples. The red curve represents MoCo v2, and the green curve represents TKC. The curves demonstrate that TKC can lead a consistent training and yield better representations.

Then we randomly select some samples from the training set and compute the stability of these samples respectively during the whole training procedure. We compute the stability for both MoCo v2 [2] baseline and TKC. Each figure in Fig 5 represents the stability of the same sample in different methods, the red curve represents MoCo v2, the green curve represents TKC.

As shown in Fig 5: (1) The output of the teacher model can dramatically vary even in the later stage during training. The stability of the samples usually gets down below 0.8 shows that the training target is inconsistent. Also, the stability is changed a lot in different epochs. These phenomena have confirmed our hypothesis that the targets from the teacher are noisy and inconsistent. (2) The stability of TKC is totally better than MoCo, where the green curve is higher than the red curve as a whole, which shows that the temporal knowledge from our method can lead to a more consistent training procedure, and improve the quality of the teacher's output.

D.4. Ablation study about knowledge transformer

structure of KT	Top-1	Top-5
2-layer	65.91	87.07
4-layer	66.21	87.04
2-layer bottleneck	66.31	87.11

Table 12. Ablation study about knowledge transformer. All experiments are run on ResNet-50 for 100 epochs.

In this section, we conduct ablation studies on different structures of the *knowledge transformer*, as shown in the table 12. All models are trained on ResNet50 for 100 epochs. In our work, we use an MLP to implement the knowledge transformer. This MLP consists of a linear layer with output dimension 256 followed by a ReLU nonlinearity and a final linear layer with output size 256. This structure can reach an accuracy of 65.91. We observe that increasing the layer of MLP can better extract the importance of different teachers to achieve better performance. The 4-layer MLP can further improve the top-1 accuracy to 66.31. And a design of bottleneck MLP can also boost the performance, where we change the hidden size from 256 to 4096 to obtain a bottleneck structure. This structure can boost the result to 66.21 with less additional computational cost. These results show that the *temporal teacher* depends on the *knowledge transformer* to leverage the importance of different teachers. Using more complex structures as attention may further improve the performance. We will explore it in future work.

E. Difference with Related Works

Some previous works also involve information from previous periods. MoCo [5] and Temporal Ensembling [9] both use the samples from previous training. In this section, we will clarify the difference between TKC and their works in both motivation and methodology.

Difference with MoCo v2. MoCo [5] believe that a large and consistent group of negative samples is critical for contrastive learning, and use the EMA encoder to construct a large and consistent negative bank. They think that the negative samples from previous training stages are harmful, and only use the negative samples which are near in time.

In our work, we notice that the outputs of the teacher can vary dramatically on the same sample during different training stages, which can introduce unexpected noise and lead to catastrophic forgetting caused by inconsistent objectives. We believe that the knowledge from previous stages is essential to learn the instance temporal consistency and stable the position of the teacher's outputs in the latent space. Empirically results show that the output of *temporal teachers* can provide the temporal knowledge and gain the performance. Note that our negative bank is all consistent [5], for we use the negative samples from the same teacher to compute the temporal loss in Eq 4.

Difference with Temporal Ensembling. Temporal Ensembling [9] is a semi-supervised learning method that ensemble the output of the same sample from previous epochs as the predicting target. Their work is different from ours in this aspect: (1) Temporal Ensembling relies heavily on dropout regularization to obtain various outputs in different epochs to yield a more accurate target. TKC also works well with networks without dropout layer [7], for TKC can restrict consistency between different epochs. (2) Temporal Ensembling also uses an exponential moving average to ensemble the output from different epochs, which can't leverage the importance of different outputs. On the contrary, TKC preserves the *temporal teacher* independently and uses *knowledge transformer* to dynamically learn their importance.

[1] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018. 1

[2] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1, 2, 3

[3] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020. 2

[4] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 2

[5] 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 (CVPR)*, June 2020. 1, 3, 4

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[7] 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 (CVPR)*, June 2016. 1, 4

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1, 2

[9] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *International Conference on Machine Learning*, 2017. 3, 4

[10] Zeming Li, Songtao Liu, and Jian Sun. Momentumˆ 2 teacher: Momentum teacher with momentum statistics for self-supervised learning. *arXiv preprint arXiv:2101.07525*, 2021. 1

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[12] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 1