This appendix provides additional material: §A contains further results on "in-the-wild" data (§A.1), Kinetics-600 (K600) [10] and Kinetics-700 (K700) [11] data (§A.2) and on the effect of key implementation details (§A.3).

§B contains additional implementation details for: Unsupervised pre-training (§B.1), and downstream evaluation in Kinetics (§B.2), AVA (§B.3), Charades (§B.4), Something-Something V2 (§B.5), UCF101 (§B.6), HMDB51 (§B.7).

A. Additional Results

A.1. Scaling "in-the-wild" data

As a follow-up experiment Table 12 compares training BYOL longer (200ep) to increasing its clips-size ρ but not training longer (50ep). For both (a) curated and (b) random data, this results in a significant gain of performance.

		BYOL	
ρ	ep	K400	UCF101
2	50	64.1	93.5
2	200	60.2	92.7
4	50	67.7	94.5

(a) **IG-Curated-1M**.

		BYOL	
ρ	ep	K400	UCF101
2	50	58.9	90.1
2	200	57.9	91.6
4	50	63.8	91.8

(b) **IG-Uncurated-1M**.

Table 12. **More epochs (ep) *vs*. more clips (ρ)**, Longer training degrades performance for BYOL, but increasing ρ does not.

We also explore an experiment for increasing the clip-size in MoCo and training longer (as MoCo works stable for more epochs). Table 13 shows the results. It can be observed that increasing the number of clips from ρ=2 to ρ=3 can increase the results by 1.6%/0.9% K400 and 0.4%/1% on UCF101 for 100/200ep training. Going to $\rho = 4$ brings further gain. In terms of efficiency, increasing ρ is both more accurate and faster than increasing the number of epochs, *e.g.* training MoCo (ρ=3, 100ep) takes only 63% of the duration that MoCo (ρ=2, 200ep) requires.

	MoCo (ρ=2)		**MoCo** (ρ=3)		**MoCo** (ρ=4)	
ep	K400	UCF101	K400	UCF101	K400	UCF101
100	67.5	93.3	69.1	93.7	69.8	94.9
200	69.0	93.4	69.9	94.4	69.9	94.9

Table 13. **More epochs (ep) *vs*. more clips (ρ)**: Dataset: **IG-Curated-1M**, ρ=2. Training longer is less effective than increasing the number of temporal clips per iteration (ρ).

Finally, we remark that the IG-Curated-1M is subsampled such that the hastags are uniformly distributed (roughly balanced). Therefore this dataset is matching K400 in terms of content and distribution. We revisit this point next by investigating the effect of scale, curation and balancing of the video data.

In this experiment, we increase the scale of the data from 128K to 1M distinct videos. We increase dataset size (number of videos) for IG-Curated [24], IG-Curated-Unbalanced [24] (which has random class distribution), and IG-Uncurated (which are random IG videos). The experiment with 200-epoch MoCo with ρ=2, linear protocol downstream evaluation on K400 is shown in Fig. 4 and reveals:

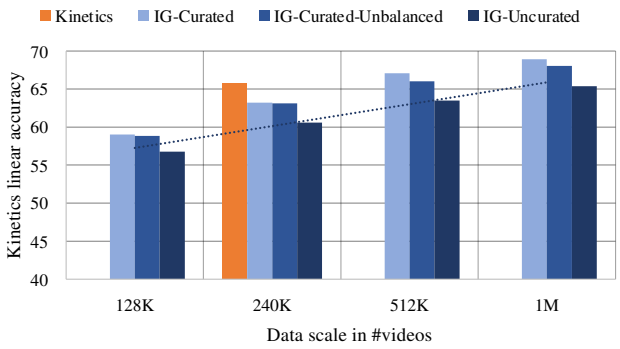

Figure 4. **Data scale and curation.** We increase dataset size (number of videos) for IG-Curated, IG-Curated-Unbalanced, and IG-Uncurated. By using 4× the number of videos, IG-Uncurated approaches the heavily curated Kinetics (K400) pre-training on K400 linear evaluation protocol. The dotted line represents a linear trend. Method: **MoCo**, 200 epochs, ρ=2.

(i) Comparing the curation axis: At 240K training samples, the four data sources provide 65.8%, 63.2%, 63.1%, 60.6% top-1 accuracy for K400, IG-Curated, IG-Curated-Unbalanced and IG-Uncurated, respectively. The decay from the heavily curated K400 to IG-Curated (2.6%) is similar to the one from IG-Curated to IG-Uncurated (2.5%), while the class balancing seems to have a minor effect on accuracy.

(ii) Comparing the scale axis: Doubling the data scale (number of videos) roughly linearly increases the accuracy across all datasets. With 1M uncurated videos the performance approaches 65.4% which is similar to the 65.8% produced by using K400 pre-training. The experiment indicates that it is possible to approach unsupervised Kinetics pre-training when using 4×more (1M *vs*. 240K in Kinetics), but *random*, videos when evaluating on Kinetics.

A.2. Scaling Kinetics data

As referenced in Sec. 4 of the main paper, Table 14 shows a series of extra results for pre-training on the larger-scale Kinetics-600 (K600) [10] and Kinetics-700 (K700) [11] datasets, and is analyzed next: The first row of the table shows supervised training on the respective datasets, where UCF101 has two entries, one for training-from-scratch and one for using K400 as pre-training.

For the experiments we focus on our temporally persistent MoCo algorithm and, as in the main paper, evaluate Kinetics with the linear classification protocol and UCF101 by finetuning all weights. The first unsupervised row in Table 14 shows our best **K400** pre-trained **MoCo** (ρ=4) model, achieving 69.0%, 70.0%, 54.2% and 93.6% on K400, K600, K700 and UCF101, respectively (this is the model with strong augmentations from Table 10 of the main paper).

method	pre-train data	pre-train #videos	K400	K600	K700	*finetune* UCF101
supervised	*scratch*		**74.7**	**78.1**	**65.2**	68.8
	K400	240k	*linear protocol*			**94.8**
MoCo (ρ=4)	K400	240k	69.0	70.0	54.2	93.6
MoCo (ρ=2)	K600	387k	69.6	70.7	55.1	92.7
MoCo (ρ=4)			71.5	72.8	57.7	94.5
MoCo (ρ=2)	K700	522k	70.0	71.4	56.2	92.8
MoCo (ρ=4)			71.7	73.2	58.1	**94.8**

Table 14. **Dataset scale**: Configuration: backbone: R-50, Slow 8 × 8, 200 epochs. Our approach, **MoCo** (ρ=4), is able to approach supervised pre-training on the popular UCF101 evaluation protocol, but there remains a gap for the linear protocol on K400, K600 and K700.

The next row shows MoCo trained on **K600** with a temporal persistency objective across two clips, ρ=2. This version is able to slightly outperform the K400 pre-trained variant on all datasets, except UCF101. Directly comparing this version with learning temporal persistency across ρ=4 clips can significantly increase accuracy on all datasets by ∼2%.

The final two rows of Table 14, show the same two models when pre-trained on **K700**. Here, we see that going from K400 to K700 increases accuracy by 2.7%, 3.2% and 3.9%, 1.2% on K400, K600, K700 and UCF101, respectively.

Overall the experiments suggest clear *benefits of using larger-scale datasets* for unsupervised pre-training and room for improvement under the linear classification protocol, especially when evaluated on larger datasets.

A.3. Key implementation specifics

While the full implementation details of all four meta-methodologies are provided in §B.1, we want to discuss the most impactful ones, which we found critical to achieve good performance in their realizations, throughout this section.

m_{base}	N/A	0.988	0.990	0.992	0.994	0.996
acc.	64.5	65.5	65.5	65.6	65.8	65.1

Table 15. **Momentum annealing for MoCo**. Dataset: **K400**, 200 epochs, ρ= 2. Using cosine-annealing of the momentum brings gains of ∼1% accuracy. We use 0.994 as default for MoCo.

Momentum annealing. BYOL is using an annealing of the rate at which parameters of the momentum encoder θ_m, that are a moving average, with momentum m, of the trained encoder θ. During training BYOL starts with a momentum of m_{base}=0.996 and increases it to 1 with a cosine annealing $m = 1 - (1 - m_{\text{base}}) \cdot (\cos(\pi k/K) + 1)/2$ with k the current iteration and K the maximum number of training iterations [32] (this is unrelated to the learning rate decay).

By default MoCo, is using a fixed momentum of $m = 0.999$ during training. In Table 15, we ablate the positive (or negative) effect of using momentum annealing with different starting rates m_{base} for MoCo. We observe that not using any annealing (N/A) produces 64.5% accuracy and using momentum annealing can boost this performance by ∼1%, while being relatively stable for different values of m_{base}. Consequently, we are using momentum annealing with $m_{\text{base}} = 0.994$ for all our MoCo experiments.

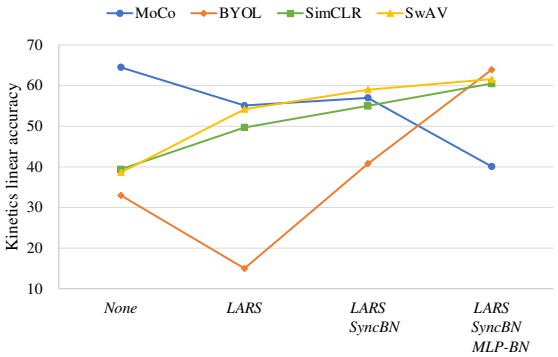

Figure 5. **Key implementation specifics.** BYOL, SimCLR, SwAV heavily rely on *LARS*, *SyncBN*, and BN in the MLP (*MLP-BN*), MoCo does not require these, but does not benefit of having them.

Normalization and optimization. Here, we present normalization specifics that we found critical to achieve good performance in the underlying implementation of the methods: SimCLR, BYOL and SwAV are using synchronized Batch-Normalization (BN) [42] statistics (*SyncBN*) across 8 GPUs during training, batch-normalization after every MLP layer (*MLP-BN*), and a large-batch optimizer (*LARS*) [95]. LARS adaptively scales the learning rate for each individual parameter by using the ratio between gradient and parameter magnitudes. MoCo is not using these components (*None*) by default. In Fig. 5 we illustrate the results. It shows accuracy on K400 linear readout, if step-by-step adding these specifics to the methods. We make the following observations:

(i) Using None of the augmentations provides best performance for MoCo (its default) but significantly degrades BYOL, SimCLR and SwAV. Here, it is worth noting that BYOL provides decent accuracy of 32.9% without *SyncBN*, *LARS* and any BN in the MLP.

(ii) Adding *LARS* optimizer reduces performance in MoCo and BYOL, while having a boost of around 10% for both SimCLR and SwAV. It is interesting, that solely using a more advanced optimizer, which adapts the learning rates of the weights according to their gradient magnitudes, decreases performance in methods using a momentum encoder (MoCo, BYOL), but boosts it without (SimCLR, SwAV).

(iii) further adding *SyncBN* and *MLP-BN* increases BYOL performance dramatically; this related to recent studies [74] which suggest that normalization is important to achieve good performance using BYOL.

(iv) While BYOL, SimCLR and SwAV do show further gains for adding *SyncBN* and *MLP-BN*, MoCo shows no significant change for using *SyncBN*, and degrades drastically in performance for using BN in the MLP-head.

Projection MLP. It has been shown that using a deeper projection MLP in pre-training can increase the accuracy of the resulting representations for image classification [12, 14, 13]. Here, we investigate the effect of more hidden layers for video classification, across all four meta architectures. The results are shown in Table 16 and discussed next.

(i) MoCo achieves a significant gain of 1.2% on K400 for using a 3-layer (2 hidden layers) MLP *vs*. using a 2-layer MLP and there is no gain for using a 4$^{\text{th}}$ layer. UCF performance appears stable to this modification. The gain is in line with results in image classification [14].

(ii) For BYOL, which has an additional *Predictor MLP*, with weights θ_p (see Fig. 2c), we ablate two dimensions: increasing the projection depth, and the prediction depth. Our results show that using 3-layer projection *vs*. 2-layer does not affect performance on K400, and has a decay of -0.7% on UCF101. Increasing also the depth of the predictor from our default value of 2 to 3 layers will lead to a significant decrease of -2.2% and -2.5% on both K400 and UCF101.

(iii) SimCLR, shows similar behavior as MoCo: A consistent gain for using 3 projection layers (+1.5% on K400, +0.5% on UCF101), and no further gain for a 4-layer MLP.

(iv) SwAV shows continuing gains on K400 for adding more MLP layers, +1.3% for going from 2 to 3 and another +0.4% for 4-layer MLP; however, its UCF-101 performance is decaying with more projection layers.

Overall, Table 16 suggests that K400 linear evaluation accuracy gernally benefits from deeper projection heads, while the performance for fine-tuned UCF101 downstream performance is relatively unchanged and rather shows a decaying effect for deeper MLPs. When studying the training complexity for pre-training, which we measure as floating point operations (FLOPs) and Parameters for the full training architecture (encoders + MLPs), Table 16 shows that FLOPs are mostly unchanged by deeper MLPs (as they operate on feature maps of size $1\times1\times1$), but parameters increase leading to large models especially for momentum encoder based approaches (MoCo and BYOL).

method	MLP layers	training		accuracy	
		FLOPs	Param	K400	UCF101
MoCo	2	41.74G	72.2M	64.6	**91.3**
	3	41.74G	80.6M	**65.8**	**91.0**
	4	41.75G	88.9M	65.7	91.0
BYOL	2, predictor: 2	41.75G	86.4M	**65.8**	**92.7**
	3, predictor: 2	41.77G	119.9M	65.8	92.0
	3, predictor: 3	41.78G	153.5M	63.6	90.2
SimCLR	2	41.74G	36.1M	59.0	88.4
	3	41.75G	40.3M	60.5	**88.9**
	4	41.75G	44.5M	**60.6**	88.5
SwAV	2	41.74G	36.2M	60.3	**88.1**
	3	41.75G	40.4M	61.6	87.3
	4	41.75G	44.6M	**62.0**	87.1

Table 16. **Varying depth of MLPs.** Dataset: **K400**, 200 epochs, ρ=2. Training complexity is measured in floating point operations (FLOPs) and Parameters. Accuracy is reported as linear evaluation (K400) and fine-tuning (UCF101) of the backbone without MLPs.

B. Additional Implementation Details

B.1. Unsupervised pre-training

Training details. We use the initialization outlined in [38]. The projection and prediction MLP weights are initialized with [27]. We optimize with synchronized SGD training on 64 GPUs with a mini-batch size of 8 clips per GPU; therefore, the total mini-batch size is 512. We train with Batch Normalization (BN) [42], and the BN statistics are computed within each 8 clips for MoCo and 64 clips by synchronizing across 8 GPUs (*SyncBN*) for BYOL, SimCLR and SwAV. We adopt a half-period cosine schedule [57] of learning rate decaying: the learning rate at the n-th iteration is $\eta{\cdot}0.5[\cos(\frac{n}{n_{\text{max}}}\pi)+1]$, where n_{max} is the maximum training iterations and the base learning rate η is set for each method to $\eta_{\text{MoCo}} = 0.4$, and $\eta_{\text{SimCLR}} = \eta_{\text{BYOL}} = \eta_{\text{SwAV}} = 4.8$. We apply (*LARS*) [95] (except for bias and BN parameters [32]), with trust coefficient of 0.001, for BYOL, SimCLR, and SwAV training. The SGD weight decay is 10^{-4} for MoCo and 10^{-6} for for BYOL, SimCLR and SwAV. The temperature parameter $\alpha = 0.1$ for MoCo, SimCLR and SwAV. The projection MLP output dimensions are $d_{\text{MoCo}} = d_{\text{SimCLR}} = \eta_{\text{SwAV}} = 128$, and $d_{\text{BYOL}} = 256$, as in their original publications [36, 12, 9, 32].

MoCo details. We use a queue storing 65536 negatives and shuffling BN to avoid intra-batch communication among samples [36]. We use a 3-layer (2 hidden layers, ablation in Table 6 of the main paper) projection MLP with hidden dimension 2048, ReLU activation [64] and no BN. Other hyperparameters are as in [36, 14]. The momentum encoder weights θ_m are updated with an annealed momentum $m = 1 - (1 - m_{\text{base}}) \cdot (\cos(\pi k/K) + 1)/2$ with k the current iteration and K the maximum number of training iterations [32], starting with $m_{\text{base}} = 0.994$. The corresponding ablation is in Table 3 of the main paper.

BYOL details. Our BYOL implementation uses a momentum annealing starting from $m_{\text{base}} = 0.996$. We minimize the negative cosine similarity in equation (2) of the main paper multiplied by 2 which is equivalent to BYOL's MSE of ℓ_2-normalized vectors [32]. The projection and prediction MLPs have 2 layers (one hidden layer with dimension 4096) and use BN following the original publication [32].

SimCLR details. We follow the default implementation [12]. We use a 3-layer projection MLP with a hidden dimension of 2048, ReLU and BN. The loss in equation (1) of the main paper is computed synchronized over the full batch size.

SwAV details. We follow the default implementation [9], using 3 Sinkhorn-Knopp iterations [15] and freezing the prototypes for the first epoch. The Sinkhorn regularization parameter is set to 0.05. As in the default implementation [9], the matrix normalization statistics of the Sinkhorn-Knopp algorithm are computed synchronized over the full training batch. The projection MLP uses ReLU and BN and is identical to the one used in [9], only that we use a 3-layer MLP instead of 2 (ablations are in Table 6 of the main paper).

Encoder details. Our default encoder, f_θ, is a R-50 Slow model [20], *i.e.* a ResNet-50 [39] with a temporal dimension of size T and sample rate τ. We perform all ablations with default $T\times\tau$ of 8×8. We show the architecture in Table 17.

Augmentation details. We perform video decoding and data augmentation using PyTorch's torchvision package.

We obtain different clips from a video by the following procedure. For the temporal dimension, we randomly sample a clip (of $T\times\tau$ frames) from the full-length video, and the input to the ResNet encoder are T frames subsampled from the raw clip with a stride of τ; for the spatial dimension, we randomly crop 224×224 pixels from a video, or its horizontal flip, with a shorter side randomly sampled in [256, 320] pixels [20] (VGG-style [77, 39] spatial cropping, a comparison to Inception-style [81] cropping, which we use for results in §4.5, is given in Table 9 of the main paper).

To each clip, we apply a random horizontal flip, color distortion and Gaussian blur following the SimCLR and MoCo v2 implementation [12, 14]. For color augmentation we use the `ColorJitter` (probability 0.8) and `RandomGrayscale` (probability 0.2) method from `torchvision.transforms` module of PyTorch with the color strength parameter s: {brightness, contrast, saturation, hue} = $\{0.4s, 0.4s, 0.4s, 0.1s\}$ By default s=0.5. Ablations are given in Table 8 of the main paper. For Gaussian blur we use a spatial kernel with standard-deviation $\in [0.1, 2.0]$ applied with probability of 0.5.

B.2. Details: Kinetics Action Classification

Datasets. Kinetics-400 [47] consists of ~240k training videos and 20k validation videos in 400 human action categories. Kinetics-600 [10] has ~392k training videos and 30k validation videos in 600 classes. Kinetics-700 [11] has ~523k training videos and 35k validation videos in 600 classes.

Linear classification protocol. We validate the methods by linear classification on frozen features, following the common protocol in image classification [36]. After unsupervised pre-training on Kinetics, we freeze the features of the encoder and train a linear classifier on top of the last layer features (*e.g.* pool_5 in Table 17). For all ablations in the paper the classifier is trained for 60 epochs (using 100 epochs will increase accuracy by ~0.2%) using the same cosine schedule as for pre-training (Sec. B.1) with a base learning rate of $\eta = 4.0$ (10×higher than in pre-training), linear warm-up in the first 8 epochs, and weight decay of 0.

stage	kernels	output sizes $T\times S^2$
raw clip	-	$T\tau\times 224^2$
data layer	stride τ, 1^2	$T\times 224^2$
conv_1	1×7^2, 64 stride 1, 2^2	$T\times 112^2$
pool_1	1×3^2 max stride 1, 2^2	$T\times 56^2$
res_2	$\begin{bmatrix} 1\times 1^2, 64 \\ 1\times 3^2, 64 \\ 1\times 1^2, 256 \end{bmatrix}\times 3$	$T\times 56^2$
res_3	$\begin{bmatrix} 1\times 1^2, 128 \\ 1\times 3^2, 128 \\ 1\times 1^2, 512 \end{bmatrix}\times 4$	$T\times 28^2$
res_4	$\begin{bmatrix} \underline{3\times 1^2}, 256 \\ 1\times 3^2, 256 \\ 1\times 1^2, 1024 \end{bmatrix}\times 6$	$T\times 14^2$
res_5	$\begin{bmatrix} \underline{3\times 1^2}, 512 \\ 1\times 3^2, 512 \\ 1\times 1^2, 2048 \end{bmatrix}\times 3$	$T\times 7^2$
pool_5	global average pool	1×1^2

Table 17. **R-50, Slow pathway** [20]. The dimensions of kernels are denoted by $\{T\times S^2, C\}$ for temporal, spatial, and channel sizes. Strides are denoted as {temporal stride, spatial stride^2}. Non-degenerate temporal filters are underlined. Residual blocks are in brackets. Temporal pooling is only performed at the last layer, collapsing spacetime dimensions. By default $T\times\tau$ = 8×8.

Training augmentation. We use the default training augmentation [20]. We randomly sample a clip (of $T\times\tau$ frames) from the full-length video and randomly crop 224×224 pixels from a video, or its horizontal flip, with a shorter side randomly sampled in [256, 320] pixels.

Inference. Following common practice, in video classification [20], we report 30-view, top-1 classification accuracy on the Kinetics validation set. We uniformly sample 10 clips from a video along its temporal axis. For each clip, we scale the shorter spatial side to 256 pixels and take 3 crops of 256×256 to cover the spatial dimensions. We average the softmax scores for prediction.

B.3. Details: AVA Action Detection

Dataset. The AVA dataset [33] has bounding box annotations for spatiotemporal localization of (possibly multiple) human actions. It has 211k training and 57k validation video segments. We follow the standard protocol reporting mean Average Precision (mAP) on 60 classes [33] on AVA v2.2.

Detection architecture. We exactly follow the detection architecture in [20] to allow direct comparison of the pre-trained models used as a backbone for the AVA task [33]. The detector is similar to Faster R-CNN [73] with minimal modifications adapted for video. Region-of-interest (RoI) features [26] are extracted at the last feature map of res_5

(*cf.* Table 17) by extending a 2D proposal at a frame into a 3D RoI by replicating it along the temporal axis, followed by application of frame-wise RoIAlign [37] and temporal global average pooling. We set the spatial stride of res_5 to 1 (instead of 2), and use a dilation of 2 for its filters [20]. This increases the spatial resolution of res_5 by $2\times$. The RoI features are then max-pooled and fed to a per-class sigmoid classifier for prediction.

Training. For direct comparison, the training procedure and hyper-parameters for AVA follow [20] without modification. The network weights are initialized from the Kinetics models and we use step-wise learning rate decay, that is reduced by $10\times$ after 16, 24 and 28 epochs. We train for 32 epochs on ~211k data, with linear warm-up [30] for the first 5 epochs and use a weight decay of 10^{-7}, as in [20]. For 8 GPU training, we use a batch-size of 64, a learning rate of 0.05 for the supervised pre-trained Kinetics models and 0.3 for the unsupervised ones, as this gives the best result for each of them.

The region proposal extraction also follows [20] and is summarized here for completeness. Our region proposals are computed by an off-the-shelf person detector, *i.e.*, that is not jointly trained with the action detection models. We adopt a person-detection model trained with *Detectron* [25]. It is a Faster R-CNN with a ResNeXt-101-FPN [92, 55] backbone. It is pre-trained on ImageNet and the COCO human keypoint images [56]. We fine-tune this detector on AVA for person (actor) detection. The person detector produces 93.9 AP@50 on the AVA validation set. Then, the region proposals for action detection are detected person boxes with a confidence of > 0.8, which has a recall of 91.1% and a precision of 90.7% for the person class.

Inference. We perform inference on a single clip with 8 frames sampled with stride 8 centered at the frame that is to be evaluated.

B.4. Details: Charades Action Classification

Dataset. Charades [76] has ~9.8k training videos and 1.8k validation videos in 157 classes in a multi-label classification setting of longer activities spanning ~30 seconds on average. Performance is measured in mean Average Precision (mAP).

Training. For Charades, we fine-tune the Kinetics models, but extend their duration by $2\times$ ($T\times\tau = 16\times 8$) to account for the long-term nature of the dataset. This increase accuracy of all models by ~3 mAP. Our training augmentation is the same as as in §B.2. A per-class sigmoid output is used for mutli-class prediction. We train for 60 epochs using a batch size of 64 and a base learning rate of 0.2 (for 8 GPUs) with $10\times$ step-wise decay at epoch 40 and 50, after warm-up in the first 5 epochs. We use weight decay of 10^{-4} and dropout of 0.5. Other training details are analogous to Kinetics.

Inference. This is as for Kinetics (§B.2), but to infer the actions over a single video, we spatiotemporally max-pool prediction scores in testing [20].

B.5. Details: Something-Something V2 (SSv2)

Dataset. The Something-Something V2 dataset [31] contains 169k training, and 25k validation videos. The videos show human-object interactions to be classified into 174 classes. We report top-1 accuracy on the validation set.

Training. We fine-tune the pre-trained Kinetics models. We train for 22 epochs using a batch size of 64 and a base learning rate of 0.12 (for 8 GPUs) with $10\times$ step-wise decay at epoch 14 and 18. Weight decay is set to 10^{-6} and dropout 0.5. Our training augmentation is the same as in §B.2, but as Something-Something V2 requires distinguishing between directions, we disable random flipping during training. We use segment-based input frame sampling [54] that splits each video into segments, and from each of them, we sample one frame to form a clip.

Inference. We perform single center clip testing to form predictions over a single video.

B.6. Details: UCF-101 Action Classification

Dataset. UCF101 [78] has 13320 human action videos in 101 categories. Our ablations are performed on the first train/val split, and for the comparison to prior work we report the mean average accuracy over the three splits.

Training. We fine-tune the pre-trained Kinetics models and use the same augmentation as for Kinetics. We train for 200 epochs using a batch size of 64 and a base learning rate of 0.025 (for 8 GPUs) with $10\times$ step-wise decay at epoch 60, 120 and 180. Weight decay is set to 0 and dropout to 0.8.

Inference. We use the same procedure as in Kinetics (§B.2).

B.7. Details: HMDB-51 Action Classification

Dataset. HMDB51 [50] contains 6766 videos that have been annotated for 51 actions. Our evaluation follows the protocol for UCF101.

Training and Inference. Our settings are *identical* to the ones used for UCF101 and we expect further tuning of hyper-parameters to increase its downstream performance.

References

[1] Pulkit Agrawal, João Carreira, and Jitendra Malik. Learning to see by moving. In *Proc. ICCV*, pages 37–45. IEEE, 2015. 2

[2] Jean-Baptiste Alayrac, Adrià Recasens, Rosalia Schneider, Relja Arandjelović, Jason Ramapuram, Jeffrey De Fauw, Lucas Smaira, Sander Dieleman, and Andrew Zisserman. Self-supervised multimodal versatile networks. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 8

[3] Humam Alwassel, Dhruv Mahajan, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. *NeurIPS*, 2020. 2, 8

[4] Relja Arandjelović and Andrew Zisserman. Look, listen and learn. In *Proc. ICCV*, 2017. 2

[5] Relja Arandjelović and Andrew Zisserman. Objects that sound. In *Proc. ECCV*, 2018. 2

[6] Suzanna Becker. Learning temporally persistent hierarchical representations. In *NeurIPS*, 1997. 2

[7] Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T. Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. SpeedNet: Learning the Speediness in Videos. In *Proc. CVPR*, 2020. 8

[8] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proc. ECCV*, 2018. 2, 8

[9] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020. 1, 2, 3, 4, 7, 14, 15

[10] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018. 12, 15

[11] João Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019. 12, 15

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 1, 2, 3, 4, 6, 7, 14, 15

[13] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems*, 33, 2020. 14

[14] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 6, 14, 15

[15] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, 2013. 15

[16] Ali Diba, Vivek Sharma, Luc Van Gool, and Rainer Stiefelhagen. DynamoNet: Dynamic Action and Motion Network. In *Proc. ICCV*, 2019. 2

[17] Carl Doersch, Abhinav Gupta, and Alexei Efros. Unsupervised visual representation learning by context prediction. In *Proc. ICCV*, 2015. 2

[18] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE PAMI*, 38(9):1734–1747, Sept 2016. 1, 2

[19] Haoqi Fan, Yanghao Li, Bo Xiong, Wan-Yen Lo, and Christoph Feichtenhofer. PySlowFast. https://github.com/facebookresearch/slowfast, 2020. 5

[20] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast Networks for Video Recognition. In *Proc. ICCV*, 2019. 2, 4, 5, 7, 8, 15, 16

[21] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *Proc. ICCV*, 2017. 2

[22] Chuang Gan, Boqing Gong, Kun Liu, Hao Su, and Leonidas J Guibas. Geometry guided convolutional neural networks for self-supervised video representation learning. In *Proc. CVPR*, 2018. 2

[23] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017. 8

[24] Deepti Ghadiyaram, Matt Feiszli, Du Tran, Xueting Yan, Heng Wang, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Proc. CVPR*, 2019. 4, 5, 8, 12

[25] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018. 16

[26] R. B. Girshick. Fast R-CNN. In *Proc. ICCV*, 2015. 15

[27] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011. 14

[28] Daniel Gordon, Kiana Ehsani, Dieter Fox, and Ali Farhadi. Watching the world go by: Representation learning from unlabeled videos. *arXiv preprint arXiv:2003.07990*, 2020. 2

[29] Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *Proc. ICCV*, 2015. 2

[30] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training ImageNet in 1 hour. *arXiv:1706.02677*, 2017. 16

[31] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "Something Something" video database for learning and evaluating visual common sense. In *ICCV*, 2017. 1, 4, 7, 16

[32] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020. 1, 2, 3, 4, 7, 13, 14

[33] Chunhui Gu, Chen Sun, David A. Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *Proc. CVPR*, 2018. 1, 4, 7, 15

[34] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Workshop on Large Scale Holistic Video Understanding, ICCV*, 2019. 2

[35] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. In *NeurIPS*, 2020. 2, 8

[36] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proc. CVPR*, 2020. 1, 2, 3, 4, 7, 14, 15

[37] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proc. ICCV*, 2017. 16

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 14

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 2, 4, 5, 7, 15

[40] Olivier J. Hénaff, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aäron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019. 2

[41] 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. In *Proc. ICLR*, 2019. 2

[42] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015. 13, 14

[43] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. Learning visual groups from co-occurrences in space and time. In *Proc. ICLR*, 2015. 2

[44] Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In *Proc. ICCV*, 2015. 2

[45] Simon Jenni, Givi Meishvili, and Paolo Favaro. Video representation learning by recognizing temporal transformations. *arXiv preprint arXiv:2007.10730*, 2020. 2

[46] Xu Ji, João F. Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proc. ICCV*, pages 9865–9874, 2019. 2

[47] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 1, 2, 3, 4, 15

[48] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *AAAI*, 2019. 2

[49] Bruno Korbar, Du Tran, and Lorenzo Torresani. Cooperative learning of audio and video models from self-supervised synchronization. In *NeurIPS*, 2018. 2

[50] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *Proc. ICCV*, pages 2556–2563, 2011. 2, 4, 16

[51] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequence. In *Proc. ICCV*, 2017. 2

[52] Ang Li, Meghana Thotakuri, David A Ross, João Carreira, Alexander Vostrikov, and Andrew Zisserman. The ava-kinetics localized human actions video dataset. *arXiv preprint arXiv:2005.00214*, 2020. 7

[53] Tianhao Li and Limin Wang. Learning spatiotemporal features via video and text pair discrimination. *arXiv preprint arXiv:2001.05691*, 2020. 2

[54] Ji Lin, Chuang Gan, and Song Han. Temporal shift module for efficient video understanding. In *ICCV*, 2019. 16

[55] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. CVPR*, 2017. 16

[56] 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 *Proc. ECCV*, 2014. 16

[57] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv:1608.03983*, 2016. 14

[58] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *Proc. ICLR*, 2017. 2

[59] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016. 2

[60] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proc. CVPR*, 2019. 8

[61] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *Proc. ECCV*, 2016. 2

[62] Hossein Mobahi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In *Proc. ICML*, pages 737–744, 2009. 2

[63] Pedro Morgado, Nuno Vasconcelos, and Ishan Misra. Audio-visual instance discrimination with cross-modal agreement. *arXiv preprint arXiv:2004.12943*, 2020. 2

[64] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. ICML*, 2010. 14

[65] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proc. ECCV*, pages 69–84. Springer, 2016. 2

[66] Andrew Owens, Phillip Isola, Josh H. McDermott, Antonio Torralba, Edward H. Adelson, and William T. Freeman. Visually indicated sounds. In *Proc. CVPR*, pages 2405–2413, 2016. 2

[67] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *Proc. CVPR*, 2017. 2

[68] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Proc. CVPR*, 2016. 2

[69] Mandela Patrick, Yuki M. Asano, Ruth Fong, João F. Henriques, Geoffrey Zweig, and Andrea Vedaldi. Multi-modal self-supervision from generalized data transformations. *arXiv preprint arXiv:2003.04298*, 2020. 2, 8

[70] AJ Piergiovanni, Anelia Angelova, and Michael S. Ryoo. Evolving losses for unsupervised video representation learning. In *Proc. CVPR*, 2020. 2

[71] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv preprint arXiv:2007.13916*, 2020. 2

[72] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. *arXiv preprint arXiv:2008.03800*, 2020. 2, 8

[73] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2016. 15

[74] Pierre H Richemond, Jean-Bastien Grill, Florent Altché, Corentin Tallec, Florian Strub, Andrew Brock, Samuel Smith, Soham De, Razvan Pascanu, Bilal Piot, et al. Byol works even without batch statistics. *arXiv preprint arXiv:2010.10241*, 2020. 13

[75] Pierre Sermanet et al. Time-contrastive networks: Self-supervised learning from video. In *Proc. Intl. Conf. on Robotics and Automation*, 2018. 2

[76] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016. 1, 4, 7, 16

[77] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 7, 15

[78] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 2, 4, 16

[79] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *Proc. ICML*, 2015. 2

[80] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Contrastive bidirectional transformer for temporal representation learning. *arXiv preprint arXiv:1906.05743*, 2019. 2

[81] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, 2015. 7, 15

[82] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Proc. ECCV*, 2020. 2

[83] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proc. CVPR*, 2018. 5

[84] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2, 3

[85] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabelled video. In *Proc. CVPR*, 2016. 2

[86] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proc. ICCV*, 2015. 2

[87] Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for self-supervised visual representation learning. In *Proc. ICCV*, 2017. 2

[88] Xiaolong Wang, Allan Jabri, and Alexei A. Efros. Learning correspondence from the cycle-consistency of time. In *Proc. CVPR*, 2019. 2

[89] Laurenz Wiskott and Terrence Sejnowski. Slow feature analysis: Unsupervised learning of invariances. In *Neural Computation*, 2002. 2, 5

[90] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. In *Proc. CVPR*, volume abs/1805.01978, 2018. 1, 2

[91] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. In *Proc. ECCV*, 2018. 5

[92] Weidi Xie. *Deep Neural Networks in Computer Vision and Biomedical Image Analysis*. PhD thesis, University of Oxford, 2017. 16

[93] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *Proc. CVPR*, 2019. 2

[94] Ceyuan Yang, Yinghao Xu, Bo Dai, and Bolei Zhou. Video representation learning with visual tempo consistency. *arXiv preprint arXiv:2006.15489*, 2020. 2, 8

[95] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 13, 14

[96] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *Proc. ECCV*, 2016. 2

[97] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. In *Proc. ICCV*, 2019. 2