Appendix

In this Appendix, we provide further ablations for video (§A) and image (§B) classification. §C contains the implementation details, and §D provides more qualitative results.

A. Ablations on Video Classification

model	pre-train	top-1	top-5	FLOPs×views	Param
SlowFast 16×8 +NL [14]	-	81.8	95.1	234×3×10	60
X3D-XL [13]	-	81.9	95.5	48×3×10	11
MoViNet-A6 [22]	-	84.8	96.5	386×1×1	31
MViT-B-24, 32×3 [12]	-	84.1	96.5	236×1×5	53
Swin-B, 16×2 [26]	Sup., IN-21K	84.0	96.5	282×3×4	88
Swin-L↑384, 32×2 [26]	Sup., IN-21K	86.1	97.3	2107×5×10	200
ViViT-H [1]	Sup., JFT-300M	85.8	96.5	3981×3×4	654
Florence↑384 [37]	Text, FLD-900M	87.8	97.8	N/A×3×4	647
MViTv2-L, 16×4 [24]	Sup., IN-21K	85.8	97.1	377×1×10	218
MViTv2-L, 16×4 [24]	**MaskFeat**, K600	**86.4**	**97.4**	377×1×10	218
MViTv2-L↑312, 40×3 [24]	Sup., IN-21K	87.5	97.8	2828×3×4	218
MViTv2-L↑312, 40×3 [24]	**MaskFeat**, K600	**88.3**	**98.0**	2828×3×4	218

(a) **Kinetics-600**

model	pre-train	top-1	top-5	FLOPs×views	Param
SlowFast 16×8 +NL [14]	-	71.0	89.6	234×3×10	60
MoViNet-A6 [22]	-	72.3	N/A	386×1×1	31
MViTv2-L, 16×4 [24]	Sup., IN-21K	76.7	93.4	377×1×10	218
MViTv2-L, 16×4 [24]	**MaskFeat**, K700	**77.5**	**93.8**	377×1×10	218
MViTv2-L↑312, 40×3 [24]	Sup., IN-21K	79.4	94.9	2828×3×4	218
MViTv2-L↑312, 40×3 [24]	**MaskFeat**, K700	**80.4**	**95.7**	2828×3×4	218

(b) **Kinetics-700**

Table 9. **Comparison with previous work on K600 & K700**. We report the inference cost with a single "view" (temporal clip with spatial crop) × the number of views (FLOPs×view$_{\text{space}}$×view$_{\text{time}}$). Each "view" consists of T frames with τ temporal stride, $T \times \tau$. Magnitudes are Giga (10^9) for FLOPs and Mega (10^6) for Param. Accuracy of models trained with external data is de-emphasized.

Kinetics-600 and Kinetics-700. Table 9 compares with prior work on K600 [3] and K700 [4]. Both are larger versions of Kinetics. An MViTv2-L, 16×4 is pre-trained with MaskFeat for 300 epochs and fine-tune for 75 epochs on both datasets. The models achieve the top accuracy of 86.4% on K600 and 77.5% on K700, using no external image data amd over 10×fewer FLOPs compared to previous Transformer-based methods.

Finally, we fine-tune these MViTv2-L, 16×4 models at a larger input spatial resolution of 312 and a longer duration of 40×3 to achieve **88.3**% top-1 on K600 and **80.4**% top-1 on K700, setting a new state-of-the-art with a large margin over the previous best on each dataset, *without* any external supervised pre-training (*e.g.* on IN-21K or JFT-300M).

ratio	20%	40%	60%	80%
top-1	81.9 (-0.3)	**82.2**	**82.2**	82.0 (-0.2)

Table 10. **Masking ratio.** Varying the percentage of masked patches. MaskFeat is robust to masking ratio in video domain.

Masking ratio. We study the effect of the masking ratio in Table 10. Interestingly, a *wide* range of masking ratios from 40% to the extreme 80% can produce similar fine-tuning accuracy, and only a small ratio of 20% leads to a slight drop of -0.3%. This is different from the observation on images, where ratios larger than 40% lead to degraded accuracy (see discussions in Appendix B). This indicates that in the video domain visual patterns are indeed more *redundant* than in images, and thus MaskFeat enjoys a larger masking ratio to create a properly difficult task.

type	center patch	cube
top-1	**82.2**	82.0 (-0.2)

Table 11. **Target design.** Predicting *center patch* HOG or all HOG in a *cube* gives similar results. Default in gray .

Target design. On video, each output token corresponds to a space-time cube. Our default setting is to simply predict the feature of the 2-D spatial patch temporally centered in each masked space-time cube. In Table 11 we consider another straightforward way of predicting the entire cube, *i.e.*, HOG features of each 2-D patch in the 3-D cube. Results are similar and we use center patch prediction for simplicity.

epoch	param. (M)	300	800
MViTv2-S, 16×4	36	82.2	82.0 (-0.2)
MViTv2-L, 16×4	218	83.1	84.3 (+1.2)

Table 12. **Pre-training schedule.** Large model benefits more from longer pre-training schedule.

Pre-training schedule. We show different pre-training schedule lengths on K400 in Table 12. Each result is fine-tuned from a fully trained model instead of an intermediate checkpoint. For MViTv2-S with 36M parameters, extending pre-training from 300 epochs to 800 epochs results in a small performance degradation of 0.2% accuracy. In contrast, for MViTv2-L longer pre-training provides a significant gain of +1.2% accuracy. This suggests that MaskFeat is a scalable pre-training task that can be better utilized by models with larger capacity and longer schedule.

B. Ablations on Image Classification

epoch	300	800	1600
ViT-B	83.6	83.9 (+0.3)	**84.0** (+0.4)
ViT-L	84.4	85.4 (+1.0)	**85.7** (+1.3)

Table 13. **Pre-training schedule.** Gains with longer schedules are observed. The large model benefits more from longer schedules.

Pre-training schedule. We show different lengths of pre-training in Table 13. Each result is fine-tuned from a fully trained model instead of an intermediate checkpoint.

For both base and large size models, improvements are observed with longer pre-training schedules. Interestingly, the large size model benefits more from longer pre-training with +1% gain from 300 epochs to 800 epochs, while the base-size model is only improved by +0.3%. This suggests that MaskFeat is a sufficiently difficult task such that (i) excessive long pre-training does not cause over-fitting of large models, and (ii) MaskFeat is sufficiently difficult for high capacity models. Training for 1600 epochs only gives another +0.1% improvement for ViT-B.

ratio	20%	40%	60%	80%
top-1	83.5 **(-0.1)**	**83.6**	83.1 **(-0.5)**	82.5 **(-1.1)**

Table 14. **Masking ratio (image).** Varying the percentage of masked patches. A smaller percentage of masking is preferred.

Masking ratio. We vary the percentage of masked patches in Table 14 with block-wise masking following BEiT [2]. We observe that masking out 20%~40% patches works well and that stronger masking degrades accuracy. MaskFeat requires enough visible patches to set up a meaningful objective. Note that 20%~40% masking is more than 15% masking used in masked language modeling (BERT [10]), reflecting redundancy in raw visual signals.

aug.	RRC	RRC + color jit.	RRC + Rand Aug.
top-1	**83.6**	**83.6**	83.2 **(-0.4)**

(a) **Augmentation.** Our MaskFeat works best with only Random Resized Crop (RRC) as augmentation.

scale	[0.08, 1.0]	[0.2, 1.0]	[0.5, 1.0]	[0.8, 1.0]
top-1	83.4 **(-0.2)**	83.4 **(-0.2)**	**83.6**	83.4 **(-0.2)**

(b) **Random resized crop scale.** A relatively large scale of random crops provides a small gain.

Table 15. **Data augmentation** in MaskFeat. Defaults are gray .

Data augmentation. We study the effect of data augmentation during MaskFeat pre-training in Table 15. All three entries in Table 15a use random horizontal flipping. Our approach works best with only random resized crop (RRC), while color jittering has no influence on the result and stronger augmentation (RandAugment [9]) degrades the performance slightly by 0.4%. This suggests that strong augmentations might lead to artificial patterns that in turn lead to a gap in pre-training and finetuning and MaskFeat works nearly augmentation-free. Conversely, contrastive-based methods are arguably dependent on "augmentation engineering" to provide prior knowledge (*e.g.*, [6, 18]), which could lead to conflicting clues [29] and over-fitting to a specific combination of augmentations [36].

We further study the effect of the RRC $[\min, \max]$ scales in Table 15b. Our approach is robust to this hyper-parameter. MaskFeat works best with low strength of RRC, $[0.5, 1.0]$, which covers a large fraction of each sample.

targets	pixel	HOG	pixel + HOG
top-1	82.5 **(-1.1)**	**83.6**	82.3 **(-1.3)**

Table 16. **Multi-tasking.** Simply combining two targets with two separate linear prediction heads results in a drop, suggesting conflict in the objectives. The default entry is marked as gray .

Multi-tasking. Finally, we investigate if combining different targets in a multi-task loss helps. Specifically, we combine pixel and HOG, two single-stage target features, by predicting each target with a separate linear layer. The two prediction losses are simply averaged with equal weighting. The results are summarized in Table 16. We see that multi-tasking of pixel and HOG provides a small gain over the scratch baseline (82.3% *vs.* 81.8%), but the accuracy is lower than pixel or HOG only. Though further tuning the loss weighting might improve this result, it signals that the two objectives can not benefit each other. This is reasonable, as HOG targets are locally normalized while pixel colors are strongly influenced by local brightness changes.

block	8th	16th	24th
top-1	**67.7**	66.0	55.9

Table 17. **Linear probing.** We perform linear probing after the 8th, 16th, 24th (last) block of MaskFeat pre-trained ViT-L. Lower layers obtain better linear accuracy.

Linear probing. Besides the fine-tuning protocol, we consider linear probing in Table 17 which is commonly used to evaluate contrastive methods [5, 20]. We train randomly initialized linear classifiers right at transformer block outputs. Specifically, we consider the average pooled outputs of the 8th, 16th and 24th (last) transformer blocks of a ViT-L pre-trained with 1600 epochs of MaskFeat on IN-1K. We observe that lower layers (*e.g.*, the 8th) tend to have higher linear accuracy. This is different from contrastive based methods whose higher layers tend to obtain better linear accuracy [34, 35]. All layers lag behind contrastive methods by a large margin. For instance, MoCo v3 [7] has 77.6% at the last block of ViT-L. This suggests that contrastive-based and masked visual prediction methods have very different features. MaskFeat learns good visual knowledge revealed by fine-tuning protocol but not linearly separable features.

Our hypothesis here is that instance discrimination losses in contrastive learning create different embeddings (classes) for different images which can be largely reduced to class-level information (a subset of classes) with a linear layer.

C. Implementation Details

C.1. ImageNet and Kinetics Experiments

Architecture. For *ImageNet* experiments, we use the standard ViT architecture [11] in base and large sizes. We use a single linear layer to transform the output of the last block to form the target predictions. We do not use relative positional bias or layer scaling.

For *Kinetics* experiments, we use MViTv2 [24], the improved version of MViT [12]. There are two main modifications. First, instead of using absolute positional embeddings as in MViT, relative positional embeddings [31] are incorporated, which are *decomposed* in height, width,

config	ImageNet	Kinetics
optimizer	AdamW [28]	
optimizer momentum	$\beta_1, \beta_2{=}0.9, 0.999$	
weight decay	0.05	
learning rate schedule	cosine decay [27]	
warmup epochs [16]	30	
augmentation	hflip, RandomResizedCrop	
gradient clipping	0.02	
drop path [23]	✗	
base learning rate[†]	2e-4	8e-4
batch size	2048	512

(a) **Pre-training setting.**

config	ImageNet		Kinetics	
	ViT-B	ViT-L	MViTv2-S	MViTv2-L
optimizer	AdamW [28]			
optimizer momentum	$\beta_1, \beta_2{=}0.9, 0.999$			
weight decay	0.05			
learning rate schedule	cosine decay [27]			
warmup epochs [16]	5			
augmentation	RandAug (9, 0.5) [9]			
mixup [39]	0.8			
cutmix [38]	1.0			
label smoothing [33]	0.1			
drop out [32]	✗			
base learning rate[†]	2e-3	1e-3	4.8e-3	9.6e-3
layer-wise decay [8]	0.65	0.75	0.75	0.875
batch size	2048	1024	512	256
training epochs	100	50	200	75
drop path [23]	0.1	0.1	0.1	0.2

(b) **Fine-tuning setting.**

Table 18. **Configurations for ImageNet and Kinetics.** [†]We use the linear *lr* scaling rule [16]: *lr* = *base_lr*×*batch_size* / 256.

and temporal axes. Second, a new residual pooling connection is introduced inside the attention blocks. Specifically, the pooled query tensor is added to the output sequence of self-attention. These two modifications improve the training-from-scratch and supervised-pre-trained baselines. We do not use channel dimension expansion within attention blocks [24] but at MLP outputs [12] which has similar accuracy. Our approach which focuses on pre-training techniques is orthogonal to these architectural modifications and provides further gains over the improved baselines.

Unlike ViT models sharing the spatial size of 14^2 for all blocks, the MViTv2 architecture is multi-scale and has four scale stages. *Stage 1* output is of spatial size 56^2 and *stage 4* output is of spatial size 7^2. To share hyper-parameters with ViT models which are of spatial size 14^2, we remove MViTv2s' query pooling before the last MViTv2 stage for MaskFeat pre-training only, resulting in a 14^2 final output size, the same as ViT models. This modification introduces little extra computation as *stage 4* is small and has only two Transformer blocks. For the fine-tuning stage, the MViTv2 models are unchanged, with 7^2 output to fairly compare with the MViTv2 baselines. Relative positional embeddings are linearly interpolated when the shape is not matched.

When sampling masked tokens for MViTv2 models on the pre-training stage, we first sample a map of the final output size, 14^2. This masking map is then nearest-neighbor resized to the *stage 1* size or input size, 56^2. In this way the set of input tokens corresponding to the same output token are masked out together, avoiding trivial predictions.

Pre-training. Table 18a summarizes the pre-training configurations. Most of the configurations are *shared* by ImageNet and Kinetics, without specific tuning. This shows that MaskFeat is *general* across tasks. The gradient clipping value is set after monitoring training loss over short runs. It is 0.02 for HOG targets and 0.3 for pixel color prediction and deep feature targets.

Fine-tuning. Table 18b summarizes the fine-tuning configurations. Most of the configurations are *shared* across models, except that deeper models use larger layer-wise learning rate decay and larger drop path rates.

For extra-large, long-term video models with 312 and 352 spatial resolutions as well as 32×3 and 40×3 temporal durations, we initialize from their 224 resolution, 16×4 duration counterparts, disable mixup, and fine-tune for 30 epochs with a learning rate of 1.6e-5 at batch size 128, a weight decay of 1e-8, a drop path [23] rate of 0.75 and a drop out rate of 0.5 for the final linear projection. Other parameters are shared with Table 18b.

C.2. AVA Experiments

The AVA action detection dataset [19] assesses the spatiotemporal localization of human actions in videos. It has 211K training and 57K validation video segments. We evaluate methods on AVA v2.2 and use mean Average Precision (mAP) on 60 classes as is standard in prior work [14].

We use MViTv2-L$\uparrow$312, 40×3 as the backbone and follow the same detection architecture in [12, 14, 24] that adapts Faster R-CNN [30] for video action detection. Specifically, we extract region-of-interest (RoI) features [15] by frame-wise RoIAlign [21] on the spatiotemporal feature maps from the last MViTv2 layer. The RoI features are then max-pooled and fed to a per-class sigmoid classifier for action prediction. The training recipe is identical to [12, 24] and summarized next. The region proposals are identical to the ones used in [12, 14, 24]. We use proposals that have overlaps with ground-truth boxes by IoU $>$ 0.9 for training. The models are trained with synchronized SGD training with a batch size of 64. The base learning rate is 0.6 per 64 batch size with cosine decay [27]. We train for 30 epochs with linear warm-up [16] for the first five epochs and use a weight decay of 1e-8, a drop path of 0.4 and a head dropout of 0.5.

C.3. SSv2 Experiments

The SSv2 dataset [17] contains 169K training, and 25K validation videos with 174 human-object interaction classes. We fine-tune the pre-trained MViTv2-L↑312, 40×3 Kinetics models and take the same recipe as in [12, 24]. Specifically, we train for 40 epochs with a batch size of 128. The base learning rate is 0.02 per 128 batch size with cosine decay [27]. We adopt synchronized SGD and use weight decay of 1e-4 and drop path rate of 0.75. The training augmentation is the same as Kinetics in Table 18b, except we disable random flipping in training. We use the segment-based input frame sampling [12, 25] (split each video into segments, and sample one frame from each segment to form a clip). During inference, we take a single temporal clip and three spatial crops over a single video.

D. Qualitative Experiments

We provide more qualitative results of image HOG predictions in Fig. 4 using ImageNet-1K validation images and for video HOG predictions in Fig. 5 using Kinetics-400 validation videos.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A video vision transformer. In *ICCV*, 2021. 1

[2] Hangbo Bao, Li Dong, and Furu Wei. BEiT: BERT pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 2

[3] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018. 1

[4] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019. 1

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 2

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

[7] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021. 2

[8] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020. 3

[9] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. RandAugment: Practical automated data augmentation with a reduced search space. In *CVPR*, 2020. 2, 3

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019. 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2

[12] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021. 1, 2, 3, 4

[13] Christoph Feichtenhofer. X3D: Expanding architectures for efficient video recognition. In *CVPR*, 2020. 1

[14] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019. 1, 3

[15] Ross Girshick. Fast R-CNN. In *ICCV*, 2015. 3

[16] 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 preprint arXiv:1706.02677*, 2017. 3

[17] 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. 4

[18] 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 *NeruIPS*, 2020. 2

[19] Chunhui Gu, Chen Sun, Sudheendra Vijayanarasimhan, Caroline Pantofaru, David A. Ross, George Toderici, Yeqing Li, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, 2018. 3

[20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 3

[22] Dan Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew Brown, and Boqing Gong. MoviNets: Mobile video networks for efficient video recognition. In *CVPR*, 2021. 1

[23] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. FractalNet: Ultra-deep neural networks without residuals. In *ICLR*, 2017. 3

[24] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Improved multiscale vision transformers for classification and detection. *arXiv preprint arXiv:2112.01526*, 2021. 1, 2, 3, 4

[25] Ji Lin, Chuang Gan, and Song Han. TSM: Temporal shift module for efficient video understanding. In *ICCV*, 2019. 4

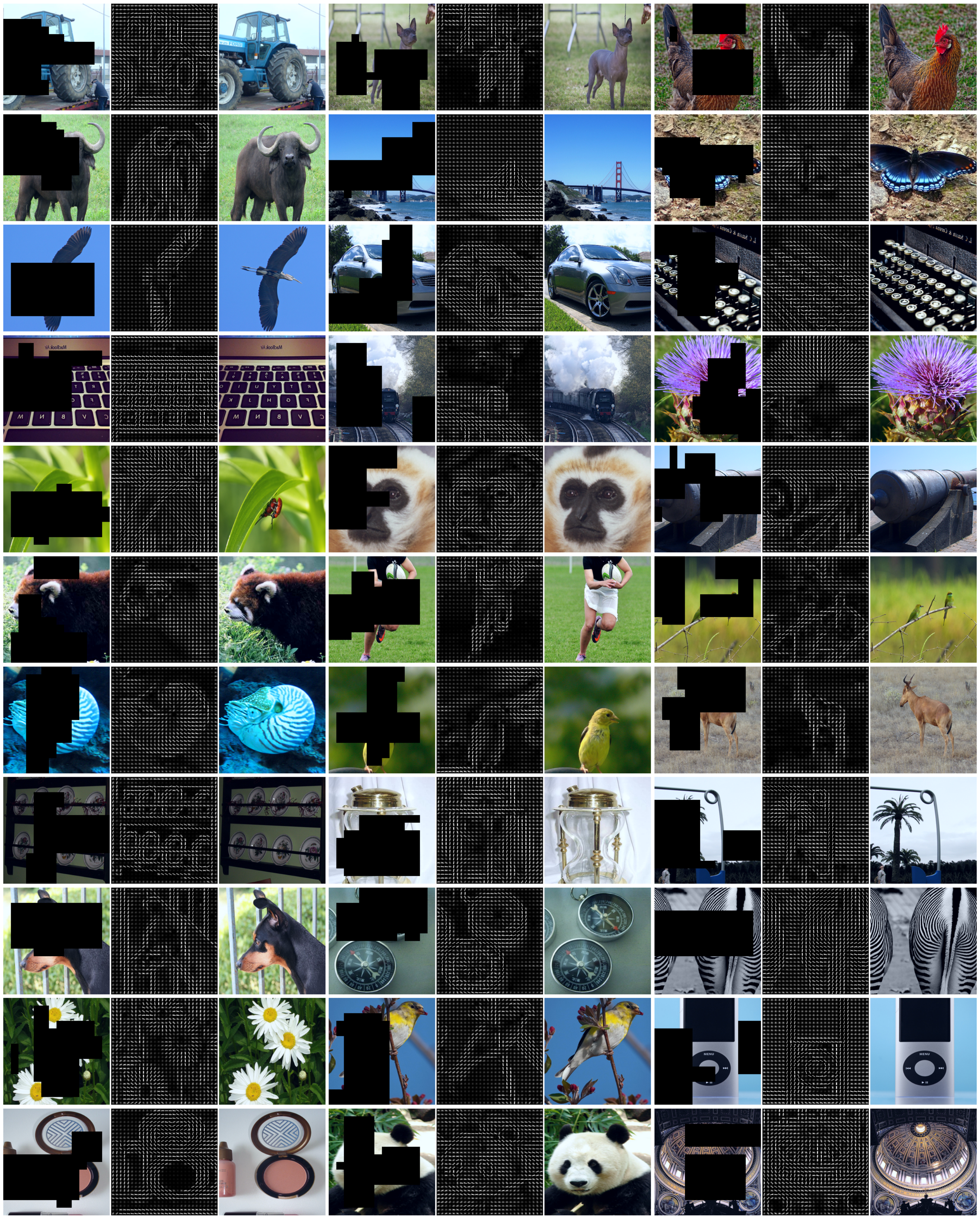

Figure 4. More visualizations of HOG predictions. The images are from IN-1K *validation* set. For each column, we show masked input (*left*), HOG predictions (*middle*) and original images (*right*). Original images are not used for prediction. Best viewed in color with zoom.

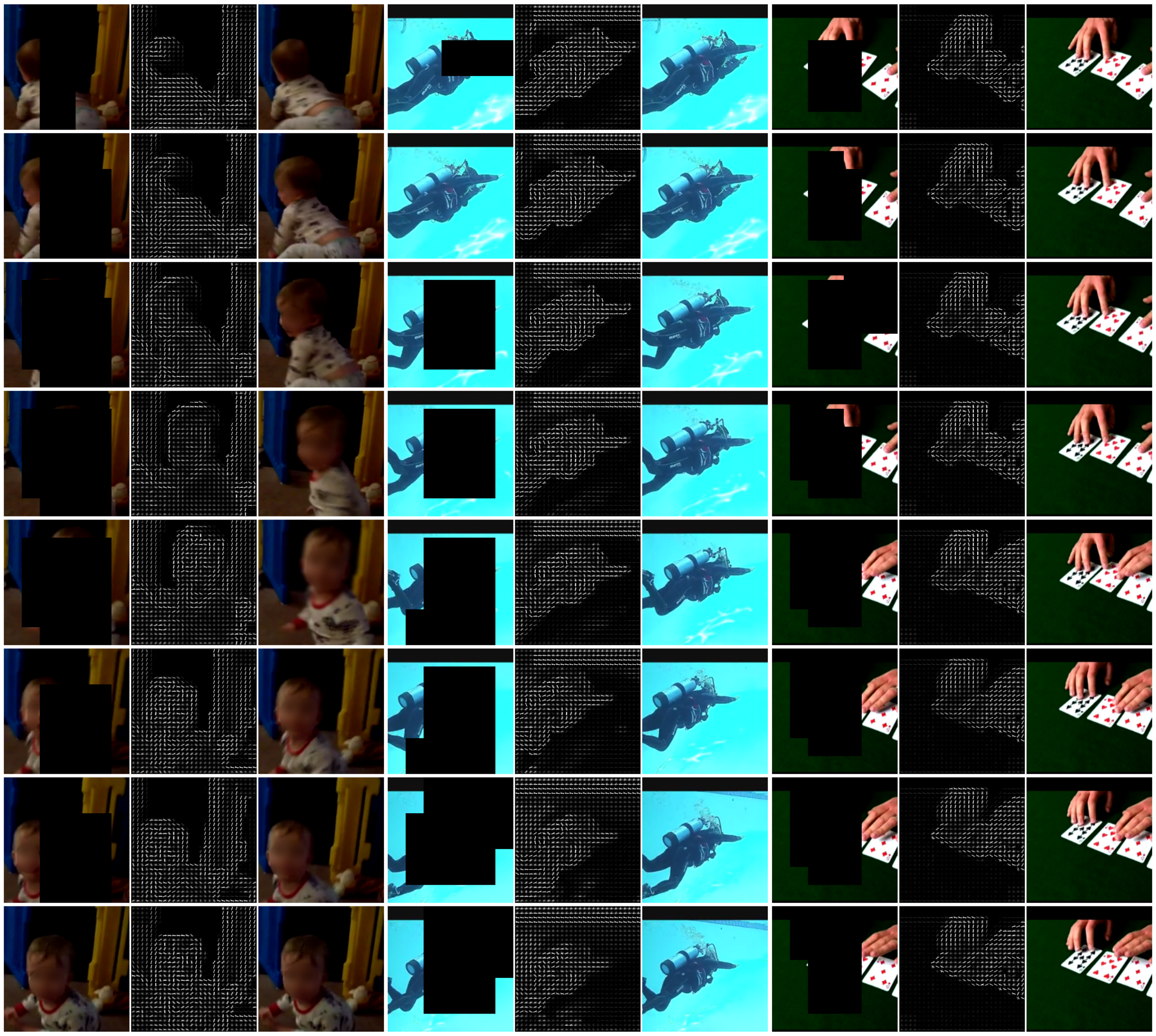

Figure 5. More visualizations of HOG predictions (video). The video clips are from K400 *validation* set. For each column, we show masked input (*left*), HOG predictions (*middle*) and original video frames (*right*), and we show eight frames from top to bottom. Original video clips are not used for prediction. Best viewed in color with zoom.

[26] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 1

[27] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 3, 4

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 3

[29] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. In *NeurIPS*, 2020. 2

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 3

[31] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018. 2

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014. 3

[33] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 3

[34] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Con-

trastive multiview coding. In *ECCV*, 2020. 2

[35] Zhirong Wu, Yuanjun Xiong, X Yu Stella, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018. 2

[36] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. In *ICLR*, 2020. 2

[37] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luowei Zhou, and Pengchuan Zhang. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021. 1

[38] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 3

[39] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 3