Table 5: **Pretraining hyper-parameters for V-JEPA.**

Hyper-parameter	ViT-L/16_{224}	ViT-H/16_{224}	ViT-H/16_{384}
data			
datasets	VideoMix2M	VideoMix2M	VideoMix2M
resolution	224	224	384
num_frames	16	16	16
temporal_stride	4	4	4
horizontal_flip	true	true	true
random_resize_scale	(0.3, 1.0)	(0.3, 1.0)	(0.3, 1.0)
random_resize_aspect_ratio	(0.75, 1.33)	(0.75, 1.33)	(0.75, 1.33)
masking			
block_aspect_ratio	(0.75, 1.5)	(0.75, 1.5)	(0.75, 1.5)
shortrange_mask_num_blocks	8	8	8
shortrange_mask_spatial_scale	0.15	0.15	0.15
longrange_mask_num_blocks	2	2	2
longrange_mask_spatial_scale	0.7	0.7	0.7
optimization			
batch_size	3072	3072	2400
total_number_of_iterations	90000	90000	90000
warmup_iterations_fraction	0.15	0.15	0.15
lr	6.25e-4	6.25×10^{-4}	6.25×10^{-4}
start_lr	2×10^{-4}	2×10^{-4}	2×10^{-4}
final_lr	1×10^{-6}	1×10^{-6}	1×10^{-6}
start_momentum	0.998	0.998	0.998
final_momentum	1.0	1.0	1.0
start_weight_decay	0.04	0.04	0.04
final_weight_decay	0.4	0.4	0.4
scheduler_scale_factor	1.25	1.25	1.25
architecture			
patch_size	16	16	16
tubelet_size	2	2	2
pred_depth	12	12	12
pred_embed_dim	384	384	384
hardware			
dtype	bfloat16	bfloat16	bfloat16
accelerator	A100 80G	A100 80G	A100 80G

A PRETRAINING DETAILS

In section, we report V-JEPA pretraining details. Table 5 summarizes the main hyperparameters used during pretraining.

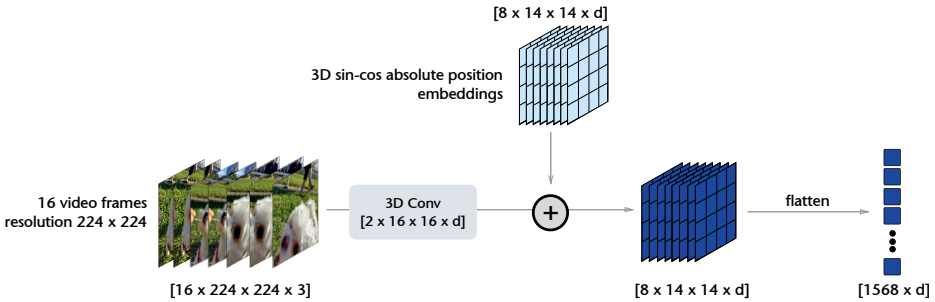


Figure 4: **V-JEPA** training operates on a video clip flattened into a sequence of tokens. To convert a video clip of size $16 \times 224 \times 224 \times 3$ into a 1D token sequence, we apply a 3D convolution comprising d filters of size $2 \times 16 \times 16$ with a temporal stride of 2 and a spatial stride of 16, resulting in a tensor of shape $8 \times 14 \times 14 \times d$. Next we add absolute 3D sin-cos positional embeddings to the spatio-temporal feature map and flatten it, resulting in a 1D token sequence of shape $1568 \times d$.

Table 6: **Frozen Evaluation hyper-parameters.**

Hyper-parameter	K400	SSv2	IN1K	Place205	iNat21
data					
num_clips	8	1	N.A.	N.A.	N.A.
num_frames	16	16	N.A.	N.A.	N.A.
temporal_stride	4	4	N.A.	N.A.	N.A.
horizontal_flip	true	true	true	true	true
random_resize_scale	(0.08, 1.0)	(0.08, 1.0)	(0.08, 1.0)	(0.08, 1.0)	(0.08, 1.0)
random_resize_aspect_ratio	(0.75, 1.33)	(0.75, 1.33)	(0.75, 1.33)	(0.75, 1.33)	(0.75, 1.33)
auto_augment	false	false	true	false	false
optimization					
batch_size	256	256	512	256	256
epochs	20	20	20	20	20
lr	1e-3	1e-3	1e-3	1e-3	1e-3
final_lr	0	0	0	0	0
weight_decay	0.01	0.01	0.01	0.01	0.01

Architectures. We use Vision Transformer (Dosovitskiy et al., 2020) (ViT) architectures for the context-encoder and target-encoder. We train three V-JEPA encoders: a ViT-L/16_{224}, a ViT-H/16_{224} and a ViT-H/16_{384}. All three encoders take as input a short video clip of 16 frames with a temporal stride of 4 between consecutive frames. The subscripts, 224 and 384, indicate the spatial resolution of the video clip. V-JEPA flattens the video clip into a sequence of non-overlapping spatio-temporal patches of size $16 \times 16 \times 2$ (see Figure 4). For all three models, the predictor is designed as a narrow ViT architecture, consisting of 12 transformer blocks with an embedding dimension of 384. For simplicity, we keep the number of self-attention heads in the predictor equal to that of the backbone used for the context-encoder/target-encoder. V-JEPA is pretrained *without* using a `[cls]` token.

Optimization. We use AdamW (Loshchilov & Hutter, 2017) to optimize the context-encoder and predictor weights. The ViT-L/16_{224} and ViT-H/16_{224} models use a batch size of 3072 while the ViT-H/16_{384} uses a batch size of 2400. Models are trained for a total of 90,000 iterations. The learning rate is linearly increased from 2×10^{-4} to 6.25×10^{-4} during the first $12,000$ iterations of pre-training, and decayed to 10^{-6} following a cosine schedule. Weight-decay is also linearly increased from 0.04 to 0.4 throughout pretraining. The target-encoder weights are initialized identically to the context-encoder, and subsequently updated as an exponential moving average (EMA) (Tarvainen & Valpola, 2017) of the context-encoder weights using a momentum value which starts at 0.998 and is linearly increased to 1.0 during training (Caron et al., 2021; Assran et al., 2022b). We scale all hyper-parameter schedules 25% beyond the actual training schedule. Specifically, the learning rate schedule, weight-decay schedule, and EMA schedule are computed assuming a training length of 112,500 iterations, even though we only train our model for 90,000 iterations. We found the last 25% of the default scheduler period to update hyper-parameters too aggressively, and simply truncating the schedulers improved performance.

Masking. As described in Section 3, we propose a 3D Multi-Block masking strategy. We use two type of masks: short-range masks, where we take the union of 8 randomly sampled target blocks with a spatial scale of 0.15, and long-range masks, where we take the union of 2 randomly sampled target blocks with a spatial scale of 0.7. In both cases, the aspect ratio for all sampled blocks is randomly chosen in the range $(0.75, 1.5)$.

B Evaluation details

B.1 Frozen classification

Attentive Probing. Given an input video, $\boldsymbol{x}_L$, the V-JEPA target encoder $E_{\bar{\theta}}(\cdot)$ outputs a sequence of L tokens, $E_{\bar{\theta}}(\boldsymbol{x}_L) = (s_1, \ldots, s_L)$, where $s_i \in \mathbb{R}^d$. To pool this sequence of tokens into a single feature vector, we apply a lightweight cross-attention using a learnable query token (Chen et al.,

2022). Specifically, our attentive pooling performs the following computation:

$$\sum_{i=1}^{L} \frac{\exp(q^\top \mathbf{W_k} s_i)}{\sum_j \exp(q^\top \mathbf{W_k} s_j)} \mathbf{W_v} s_i,$$

where $\mathbf{W_k}, \mathbf{W_v} \in \mathbb{R}^{\mathbf{d}\times\mathbf{d}}$ are the key and value matrices, and $q \in \mathbb{R}^d$ is a learnable query token. The output of the attentive pooler is then fed into a standard linear classifier, and its parameters are jointly learned with that of the linear classifier for the downstream task, while the encoder parameters are kept frozen. Note that, in practice, we actually use an attentive pooler with 12 heads, each of dimension $d/12$. In Appendix C we show that both V-JEPA and baselines benefit from the attentive probing protocol.

Optimization. For all the tasks, we use AdamW optimizer with a cosine scheduler (no warmup) that decays the learning rate from 0.001 to 0. We use a fixed weight-decay of 0.01 and apply simple data augmentations (random resized crops and horizontal flips) during training of the attentive probe, except on ImageNet, where we apply AutoAugment (Dogus Cubuk et al., 2019). Table 6 reports the hyperparameters used for the different evaluation tasks.

Extension to multiple clips. On Kinetics-400 frozen evaluation, our attentive probe takes 8 video clips as input to increase the temporal coverage of the video. Specifically, we first divide a video in 8 equal-length temporal segments and sample 1 clip at random per segment. The video encoder $E_{\bar{\theta}}$ processes each clip separately and produces a clip-level feature map. The feature maps for each clip are then concatenated together and fed to the attentive probe.

Application of video models to images. To evaluate the video models on image tasks, we simply duplicate input images to generate still video clips of 16 frames. We perform this duplication operation simply for convenience in evaluation of the video models, however we find this step to be unnecessary in general. Given a video tokenizer implemented as a 3D-conv with a temporal stride of 2, it is sufficient to simply duplicate the image into a 2 frame video clip. This would result in the same number of input tokens as that produced by a static image model with a 2D-conv tokenizer.

Application of image models to videos. To evaluate image models such as DINOv2 and OpenCLIP on video tasks, we simply process each frame independently with the image encoder to produce a frame-level feature map. The feature maps for each frame are then concatenated and fed to the attentive probe, just as we do with the clip-level feature maps when evaluating video models.

B.2 Frozen detection

We evaluate our model on the AVA (Gu et al., 2018) spatio-temporal localization of human actions dataset, containing 211k training and 57k validation video segments. We follow the experimental protocol of (Feichtenhofer et al., 2021), and use precomputed masks from a pretrained Faster-RCNN adapted to videos, which uses a ResNeXt-101-FPN backbone and is pretrained on ImageNet and COCO. We train a linear classifier on top of the *frozen* V-JEPA features to classify the extracted regions of interest and report mean Average Precision (mAP) on the 60 most common classes. Hyperparameters are provided in Table 7. Our frozen features are obtained by concatenating the last layer of the transformer encoder with three intermediate layers. We use a batch size of 64 and pretrain for 30 epochs with AdamW using a learning rate of 0.0001 with 2 epochs of warmup and a weight decay of 0.05.

B.3 Finetuning

We evaluate in Appendix 4.4 our V-JEPA ViT-H/16_{384} model on Kinetics-400 and Something-Something v2 in the finetuning setting. Following Tong et al. (2022), we finetune a linear layer on top of our model, using a layer decay schema and mixup as the data augmentation pipeline. We provide all hyper-parameters for both K400 and SSv2 in Table 8.

Table 7: **Frozen Detection hyper-parameters.**

Hyper-parameter	ViT-L/16	ViT-H/16
out_layers	[18, 20, 22, 24]	[26, 28, 30, 32]
batch_size	64	64
epochs	30	30
opt	AdamW	AdamW
opt_eps	0.00000001	0.00000001
momentum	0.9	0.9
weight_decay	0.05	0.05
lr	0.0001	0.0001
warmup_lr	0.000001	0.000001
min_lr	0.000001	0.000001
warmup_epochs	2	2
warmup_steps	1	1

Table 8: **Finetuning Evaluation hyper-parameters.**

Hyper-parameter	K400	SSv2
data		
num_segments	1	1
num_frames	16	32
sampling_rate	4	4
resolution	384	384
model		
model_name	ViT-H	ViT-H
tubelet_size	2	2
drop_path	0.2	0.2
head_drop_rate	0.5	0.5
optimization		
batch_size	64	64
epochs	35	20
opt	adamw	adamw
opt_eps	0.00000001	0.00000001
momentum	0.9	0.9
weight_decay	0.05	0.05
lr	0.0003	0.0001
layer_decay	0.8	0.8
warmup_lr	0.00000001	0.00000001
min_lr	0.000001	0.000001
warmup_epochs	5	5
warmup_steps	1	1
augmentations		
color_jitter	0.4	0.4
num_sample	2	2
aa	rand-m7-n4-mstd0.5-inc1	rand-m7-n4-mstd0.5-inc1
smoothing	0.1	0.1
train_interpolation	bicubic	bicubic
test_num_segment	5	2
test_num_crop	3	3
erase		
prob	0.25	0.25
mode	pixel	pixel
count	1	1
split	False	False
mixup		
mixup	0.8	0.8
cutmix	1.0	1.0
mixup_prob	1.0	1.0
mixup_switch_prob	0.5	0.5
mixup_mode	batch	batch

C EXTRA RESULTS

C.1 FROZEN EVALUATION.

Table 9: **Linear vs. Attentive Probe Evaluation for V-JEPA and VideoMAE.** We evaluate the effect of linear (Lin.) and attentive (Att.) probing when adapting V-JEPA to the K400 and SSv2 tasks. V-JEPA and VideoMAE benefit from using a non-linear attentive probe. Specifically, using an attentive probe with V-JEPA leads to an improvement of $+22$ points on K400 and $+17$ points on SSv2.

		K400		SSv2	
Method	**Arch.**	Lin.	Att.	Lin.	Att.
VideoMAE	ViT-L/16	52.5	77.6	41.3	61.2
V-JEPA	ViT-L/16	56.7	**79.1**	50.1	**67.1**

Table 10: **Linear vs. Attentive Probe Evaluation for DINOv2 and OpenCLIP.** We evaluate the effect of linear (Lin.) and attentive probing (Att.) when adapting DINOv2 and OpenCLIP. Image-baselines benefit from using an attentive probing strategy. Results shown in gray are reported from the linear probe evaluation in Oquab et al. (2023).

		K400		SSv2		IN1K		Place205		iNat21	
Method	**Arch.**	Lin.	Att.	Lin.	Att.	Lin.	Att.	Lin.	Att.	Lin.	Att.
DINOv2	ViT-g/14	78.4	84.4	38.3	50.0	86.5	86.2	67.5	68.4	85.7	88.8
OpenCLIP	ViT-G/14	78.3	83.3	35.8	39.0	86.2	85.3	69.8	70.2	76.0	83.6

Linear vs. Attentive probe We compare the effect of using an attentive versus a linear probe when adapting a pretrained model to various downstream tasks. Table 9 shows that V-JEPA and VideoMAE benefit from using a non-linear attentive probe on the K400 and SSv2 downstream tasks. In particular, using an attentive probe with V-JEPA leads to an improvement of $+22$ points on K400 and $+17$ points on SSv2. Additionally, Table 10 shows that attentive probing leads to better performance on average for DINOv2 and OpenCLIP models. Since attentive probing improves the performance of all models, we use it as our default evaluation protocol.

Table 11: **Temporal Coverage on Kinetics-400.** We evaluate the effect of temporal coverage on K400. We train an attentive probe on K400 using either 1 clip ($\approx$ 2 seconds of a video) or 8 clips ($\approx$ 16 seconds of a video). To sample N clips, we first divide a video in N equal-length temporal segments and sample one clip at random per segment. The video encoder processes each clip in parallel and all the encoder output tokens are concatenated at the input of the attentive probe. Increasing the temporal coverage from 1 clip per video to 8 clips significantly improves the performance for both our VideoMAE baseline and V-JEPA. Specifically, using 8 clips leads to an improvement of $+6.2$ points on K400 with a V-JEPA ViT-H/16_{384}.

Method	**Arch.**	1 Clip	8 Clips
VideoMAE	ViT-L/16	69.4	77.6
V-JEPA	ViT-L/16	72.3	79.1
	ViT-H/16_{384}	**75.8**	**82.0**

Temporal coverage on Kinetics-400. We examine the impact of changing the temporal coverage of a model during downstream evaluation on K400 action classification. In Table 11, we evaluate VideoMAE and V-JEPA models using an attentive probe with access to either the feature map of 1 clip randomly sampled from the video, or the concatenated feature map of 8 clips randomly sampled from the video. To sample 8 clips from a video, we first divide the video into 8 equal length temporal segments, and sample 1 clip at random from each segment. A single clip corresponds to ≈ 2 seconds of a video on average, while 8 clips correspond to ≈ 16 seconds. The video encoders processes each clip separately to produce a clip-level feature map, which are then concatenated at the input to the attentive probe.

Increasing the temporal coverage from 1 clip per video to 8 clips improves the performance of both V-JEPA and VideoMAE on K400 action classification. Specifically, using 8 clips leads to an improvement of $+6.2$ points on K400 with a V-JEPA ViT-H/16_{384}. We therefore use the 8 clip attentive probing setup as our default evaluation pipeline on K400 for all video and image

models. While we would expect multi-clip evaluation to be helpful for other downstream video action classification tasks, we still only sample one clip when training an attentive probe on SSv2, as videos from that dataset are only 2 to 4 seconds long on average.

C.2 Sample Efficiency of Pretraining

We compare the sample efficiency of pretraining various state-of-the-art image and video models. Specifically, we look at the number of samples (image or video clips) processed by the network during pretraining, which is larger than the size of the pretraining dataset for multi-epoch training. Notably, our results with V-JEPA are obtained while processing an order of magnitude fewer samples than previous methods, and notably two orders of magnitude fewer samples than OpenCLIP. We believe that further investment towards improving the video pretraining data distribution could lead to substantial gains in downstream image and video tasks.

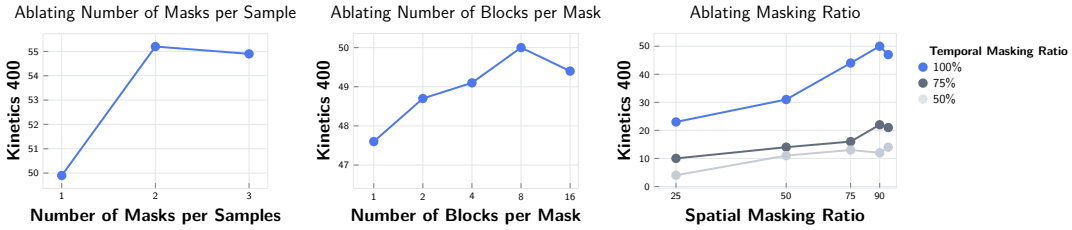
Table 12: **Sample efficiency.** We compare the sample efficiency of pretraining various state-of-the-art image and video models. The **#Samples Seen** entry corresponds to the number of samples (image or video clips) processed by the network during pretraining, which is larger than the size of the pretraining dataset for multi-epoch training. The V-JEPA results in this paper are obtained while processing an order of magnitude fewer samples than previous methods.

Method	Arch.	Data	#Samples Seen
OpenCLIP	ViT-G/14	LAION-2B	39000M
DINOv2	ViT-g/14	LVD 142M	1900M
VideoMAEv2	ViT-g/14	UnlabeledHybrid	1600M
V-JEPA	ViT-H/16_{384}	VideoMix2M	210M

C.3 Masking Strategy

An important component of the V-JEPA pretraining strategy is the 3D clip masking strategy. In this section, we detail 26 ablation experiments exploring different masks. For all the experiments, we pretrain a ViT-B/16 pretrained on K400. Figure 5 presents a summary of those results.

Recall that each video mask is constructed by sampling several (possibly overlapping) blocks and taking their union. These spatial multi-block masks are then repeated along the temporal dimension to create a 3D Multi-Block mask. Figure 5c shows the effect of changing the spatial and temporal masking ratio. Figure 5b ablates the number of sampled blocks used to construct the masks given a fixed effective masking ratio of 90%. Finally, in Figure 5a we examine our multi-masking strategy and find that sampling two masks for each clip (long-range and short-range) to be more effective than sampling just a single mask for each clip. By default we sample masks that remove roughly 90% of the frame and extend along the entire temporal dimension of the clip.



(a) (b) (c)

Figure 5: **Masking Strategy Ablation.** Evaluating a linear probe on a ViT-B/16 pretrained with V-JEPA on K400 under various 3D Multi-Block masking settings. We examine the impact of **(a)** sampling several masks per video, **(b)** varying the number of blocks in a mask, and **(c)** varying the average spatial and temporal masking ratio. A temporal masking ratio of 100% extends the spatial mask across all the frames in the clip. We find it important to maintain a high spatial and temporal masking ratio during pretraining.

In Table 13, we explore different average spatial and temporal masking ratio, i.e. the spatial/temporal ratio of the area that is covered by a mask on average for a clip. Recall that each mask is constructed by sampling several (possibly overlapping) blocks and taking their union. We change the average

Table 13: **Masking Ratio.** We explore the impact of the spatial and temporal ratio masking. Low spatial or temporal coverage results in a trivial prediction task, which degrades downstream performance.

Mask Statistics		3D Multi-Block Mask Details			
Avg. Depth	Avg. Spatial Size	Spatial Size of Block	Frames per Block	Blocks per Mask	**K400 Acc.**
100%	25 %	112 × 112	16	1	0.23
	50 %	160 × 160	16	1	0.31
	75 %	192 × 192	16	1	0.44
	90 %	176 × 176	16	2	0.50
	95 %	192 × 192	16	2	0.47
75%	25 %	112 × 112	12	1	0.10
	50 %	160 × 160	12	1	0.14
	75 %	192 × 192	12	1	0.16
	90 %	176 × 176	12	2	0.22
	95 %	192 × 192	12	2	0.21
50%	25 %	112 × 112	8	1	0.04
	50 %	160 × 160	8	1	0.11
	75 %	192 × 192	8	1	0.13
	90 %	176 × 176	8	2	0.12
	95 %	192 × 192	8	2	0.14

spatial or temporal masking ratio by changing block spatial or temporal size, as well as the overall number of blocks. We found that low spatial or temporal coverage results in a trivial prediction task, which degrades downstream performance. Based on those results, we sample masks that remove roughly 90% of the frame and extend along the entire temporal dimension of the clip by default.

Table 14: **Block Spatial Size.** We investigate the impact of blocks spatial size given an effective masking ratio of 75%. We find that sampling several small blocks to perform better than sampling a single large block.

Mask Statistics		3D Multi-Block Mask Details			
Avg. Depth	Avg. Spatial Size	Spatial Size of Block	Frames per Block	Blocks per Mask	**K400 Acc.**
100%	75%	64 × 64	16	16	0.49
		96 × 96	16	8	0.50
		128 × 128	16	6	0.49
		160 × 160	16	2	0.48
		192 × 192	16	1	0.47

In Table 14, we explore different block size given an effective spatial masking ratio of 75% and temporal ratio of 100%. We keep the masking ratio approximately constant by changing the block size and the number of block at the same time. We find that sampling several blocks to perform better than sampling a single large block. Figure 6 visually illustrates the effect of sampling several smaller blocks to construct a mask.

Table 15: **Number of Masks Per Sample.** We explore the effect of sampling several mask for each video clip in the batch. Sampling two masks for each clip, with different spatial block sizes for each, is more effective than sampling just a single mask.

	3D Multi-Block Mask Details			
Masks per Sample	Spatial Size of Block	Frames per Block	Blocks per Mask	**K400 Acc.**
1	160 × 160	16	2	0.50
2	160 × 160	16	2	0.55
3	160 × 160	16	2	0.55

In Table 15, we explore the effect of sampling various number of masks per samples. We find that sampling two masks for each clip, with different spatial block sizes for each, to be more effective than sampling just a single mask. We hypothesize that this masking strategy induces complementary tasks. In our experiment, we use this as our default masks sampling.

D Theoretical Motivation of EMA for L1 Loss

Consider just the first loss term in the summation of equation 1. To condense the notation, denote the output of the context encoder with parameters θ by $z_N(\theta)$, and denote the first token output

(a) Num. Blocks: 8, Spatial Block Size: 32×32

(b) Num. Blocks: 4, Spatial Block Size: 80×80

(c) Num. Blocks: 2, Spatial Block Size: 160×160

Figure 6: Illustration of mask with number of blocks and block size. Each mask is constructed by sampling several (possibly overlapping) blocks and taking their union.

by the predictor as $p(\boldsymbol{z}_N(\theta)) = [P_\phi(\boldsymbol{z}_N, \boldsymbol{m}_M)]_{i_1}$. Finally, define the corresponding target token (output by the target encoder) as a random vector $X \in \mathbb{R}^d$. Now, if we were to compute the optimal predictor under our loss function, we would obtain the following functional expression,

$$p^\star(\boldsymbol{z}_N(\theta)) = \text{argmin}_p \|p(\boldsymbol{z}_N(\theta)) - X\|_1 = \text{median}(X|\boldsymbol{z}_N(\theta)).$$

Substituting this expression for the optimal predictor into the loss function and evaluating the expected gradient of the context encoder gives

$$\nabla_\theta \mathbb{E}\|p^\star(\boldsymbol{z}_N(\theta)) - X\|_1 = \nabla_\theta \sum_{l=1}^{d} \text{MAD}(X_l|\boldsymbol{z}_N(\theta)),$$

where X_l is the l^{th} entry of the random vector X, and $\text{MAD}(\cdot\,|\boldsymbol{z}_N(\theta))$ is the median absolute deviation of a random variable conditioned on $\boldsymbol{z}_N(\theta)$. Thus, in the case where the predictor is optimal, the context encoder must learn to capture as much information about the masked clip as possible to minimize the deviation of the target. The hypothesis is that by updating the target encoder weights via an exponential moving average of the context encoder weights, we ensure that the predictor evolves much faster than the target encoder and remains close to optimal, thereby preventing collapse.