Supplementary Material

The organization of the supplementary material is as follows:

A Implementation Details

A.1 Datasets

The details of the datasets are presented.

Kinetics400. Kinetics400(Kay et al. 2017) is a large-scale action recognition audio-visual dataset consisting of 400 action classes. It has a total of 240K video clips with an average duration of 10 seconds. Following (Sarkar and Etemad 2023; Alwassel et al. 2020; Morgado, Vasconcelos, and Misra 2021; Tong et al. 2022; Feichtenhofer et al. 2022), we use the Kinetics400 for both pretraining and downstream evaluation.

AudioSet. AudioSet (Gemmeke et al. 2017) is a very large-scale audio-visual dataset comprised of 1.8M videos and collected from YouTube. The samples from AudioSet are of average duration of 10 seconds and spread over 632 audio events. We use AudioSet to conduct experiments on large-scale pretraining. Please note that none of the labels are used in pretraining.

UCF101. UCF101(Soomro, Zamir, and Shah 2012) is a popular action recognition dataset used for downstream evaluation. It consists of a total of 13K clips of an average duration of 7 seconds and spread over 101 action classes. We use UCF101 for downstream evaluation on video action recognition.

HMDB51. HMDB51 (Kuehne et al. 2011) is one of earlier video action recognition datasets. It contains 7K video clips distributed over 51 action classes. HMDB51 is used for downstream evaluation on video action recognition.

Kinetics-Sound. Kinetics-Sound(Arandjelovic and Zisserman 2017) is originally a subset of Kinetics400, which has a total of 22K video clips consisting of 32 action classes. Following (Sarkar and Etemad 2023), we use Kinetics-Sound for multimodal action classification by late fusion. We particularly choose Kinetics-Sound for multimodal evaluation as it consists of hand-picked samples from Kinetics400 which are prominently manifested both audibly and visually. In addition to downstream evaluation, Kinetics-Sound is also used to conduct experiments on small-scale pretraining.

ESC50. ESC50 is a popular sound classification dataset that consists of a total of 2K samples of 50 environmental sound classes, with an average duration of 5 seconds. Following (Akbari et al. 2021; Sarkar and Etemad 2023; Morgado, Vasconcelos, and Misra 2021; Recasens et al. 2021), we use ESC50 for downstream evaluation of audio representations.

FSD50K. FSD50K is a popular audio dataset of human-labeled sound events, which consists of 50K audio samples distributed over 200 classes. Moreover, FSD50K is a multi-labeled dataset, where samples are between 0.3 to 30 seconds. Following (Niizumi et al. 2022a) We use FSD50K for downstream evaluation of audio representations.

A.2 Augmentation

We adopt standard augmentation strategies (Sarkar and Etemad 2023; Chen et al. 2020; Niizumi et al. 2022a) to create the global and local views from the raw audio and visual streams. Particularly, we apply Multi-scale crop, Color Jitter, and Random Horizontal Flip to create global visual views. To create local views, we apply Gray Scale and Gaussian Blur, in addition to the augmentations applied on the global views with minor changes in the augmentation parameters. For example, different from global visual views, we apply aggressive cropping to create local visual views. The details of the augmentation parameters are presented in Table S1. Next, Volume Jitter is applied to the audio stream to create the global audio views. We apply Random Crop in addition to the Volume Jitter to create the local audio views. The details of the audio augmentation parameters are presented in Table S2.

Augmentation	Param	Global View	Local View	Linear Eval.
Multi-Scale Crop	Crop Scale	[0.2, 1]	[0.08, 0.4]	[0.08, 1]
Horizontal Flip	Probability	0.5	0.5	0.5
Color Jitter	Brightness	0.4	0.4	0.4
	Contrast	0.4	0.4	0.4
	Saturation	0.4	0.4	0.4
	Hue	0.2	0.2	0.2
Gray Scale	Probability	n/a	0.2	0.2
Gaussian Blur	Probability	n/a	0.5	n/a

Table S1: **Visual augmentation** parameters for pretraining and linear evaluation are presented. n/a: not applied.

Augmentation	Param	Global View	Local View	Linear Eval.
Volume Jitter	Scale	$\pm$ 0.1	$\pm$ 0.2	$\pm$ 0.1
Random Crop	Range	n/a	[0.6, 1.5]	n/a
	Crop Scale	n/a	[1, 1.5]	n/a

Table S2: **Audio augmentation** parameters for pretraining and linear evaluation are presented. n/a: not applied.

A.3 Pretraining

We train XKD using an AdamW (Loshchilov and Hutter 2017; Reddi, Kale, and Kumar 2019) optimizer and warm-up cosine learning rate scheduler. During ablation studies, the models are trained for 400 epochs using Kinetics400. We use ViT-B (Dosovitskiy et al. 2020) as the backbone for both audio and visual modalities. Following (He et al. 2021), we use a shallow decoder for reconstruction and adopt a similar projector head to the one proposed in (Caron et al. 2021). Please see the additional architecture details in Table S3. We perform distributed training with a batch size of 256 using 8 NVIDIA V100 32 GB GPUs in parallel. As mentioned in Table S3, we use visual frames with resolutions of 96^2 and 112^2, such lower resolutions allow us to train the proposed XKD with a limited computation setup. Additionally, we perform mixed-precision (Micikevicius et al. 2018) training to save the computation overhead. In our setup, it takes approximately 10 days to train the XKD for 800 epochs with Kinetics400. We present the additional details for the training parameters in Table S4. Please note that the parameters are tuned based on Kinetics400 and the same parameters are used for both AudioSet and Kinetics-Sound. Due to resource constraints, we do not further tune the parameters for individual datasets. As the AudioSet is very large, we pretrain XKD for 400 epochs, whereas, the models are pretrained for 800 epochs for Kinetics400 and Kinetics-Sound.

A.4 Modality-agnostic setups

In Table S5, we summarize the detailed network setups of the modality-agnostic variants. The student encoders are shared in XKD-MAS, while both the teachers' and students' backbones are shared in XKD-MATS. Please note that in none of the setups, the input projection layer and the decoders are shared.

A.5 Linear evaluation

UCF101 and HMDB51. To perform linear evaluation, we feed 32 frames with a spatial resolution of 112^2 to the pretrained visual encoder. We extract the fixed features as the mean of all the patches from the last layer of the encoder. We randomly select 25 samples per clip during training, while during testing, we uniformly fetch 3 clips per sample to extract frozen features. Next, the features are used for action classification using a linear SVM kernel. We sweep a range of cost values $\{0.00001, 0.0001, 0.001, 0.01, 0.1, 1\}$ and report the best top-1 accuracy. We present the augmentation parameters applied on the training set in Table S1. Please note that none of the augmentations are applied to the test set. We then tune the models using split 1 of the datasets and report the top-1 accuracies averaged over all the splits.

ESC50. We feed 4 seconds of audio input to the pretrained audio encoder and extract approximately 10-epochs worth of fixed features. Similar to UCF101 and HMDB51 we extract the features

Name	Value
Video (global)	32×112^2
Video (local)	8×96^2
Video Patch	4×16^2
Audio (global)	80×448
Audio (local)	80×112
Audio Patch	4×16
Backbone	ViT-Base (87M) embed dim:768 depth:12 num heads:12
Decoder	Transformer embed dim:384 depth:4 num heads:12
Projector	MLP out dim: 8192 bottleneck dim: 256 hidden dim: 2048 num layers: 3

Table S3: **Architecture** details of XKD.

Name	Value
Epochs	800
Batch Size	256
Optimizer	`AdamW`
Optimizer Betas	$[0.9, 0.95]$
LR Scheduler	`Cosine`
LR Warm-up Epochs	30
LR Warm-up	0
LR Base	0.0001
LR Final	0
Weight Decay	0.3
EMA Scheduler	`Cosine`
EMA Base	0.997
EMA Final	1
Video Mask Ratio	0.85%
Audio Mask Ratio	0.80%
Video Student Temp.	0.1
Audio Student Temp.	0.1
Video Teacher Temp.	0.1
Audio Teacher Temp.	0.07
Center Momentum	0.9

Table S4: XKD **pretraining** setups for Kinetics400.

Variant	Input Projection	Teacher	Student	Decoder
XKD-MAS	*NS*	*NS*	*S*	*NS*
XKD-MATS	*NS*	*S*	*S*	*NS*
XKD	*NS*	*NS*	*NS*	*NS*

Table S5: Summary of parameter sharing in **modality-agnostic** setups. Here, *S* and *NS* refer to shared and not-shared parameters between audio and visual modalities respectively.

as the mean of all the patches from the last layer of the encoder. During training, we apply Volume Jitter as augmentation and no augmentation is applied at test time. Similar to the setup of UCF101 and HMDB51, we sweep the same range of cost values to find the best model. We report the final top-1 accuracy averaged over all the splits.

FSD50K. To perform the downstream evaluation on FSD50K we follow the same setup as ESC50, with the exception of using a linear fully-connected layer instead of SVM. During training, we randomly extract 10 clips per sample and feed them to the pretrained audio encoder. Moreover, during test, we uniformly select 8 clips per sample and report the mean average precision (mAP) (Fonseca et al. 2022). The extracted features are used to train a fully-connected layer using an AdamW (Loshchilov and Hutter 2017; Reddi, Kale, and Kumar 2019) optimizer for 50 epochs at a fixed learning rate of 0.001. Moreover, we apply weight decay of $1e-5$ and dropout of 0.3 to prevent overfitting. We use a batch size of 256 and train on a single RTX6000 24 GB NIVIDIA GPU.

Kinetics-Sound and Kinetics400. We mainly use Kinetics-Sound for multimodal evaluation using late fusion. To perform late fusion, we simply concatenate the corresponding audio and visual features, and the joint representations are then directly used to perform action classification using a linear kernel SVM. Finally, we report the top-1 accuracy.

We follow similar setups to those used for UCF101 and HMDB51 to perform linear evaluation on Kinetics400, with the exception of randomly sampling 3 clips per video during training and testing to extract the features (we do not sample more clips during training due to memory constraints). Lastly, we report top-1 accuracy in action classification. In addition to the SVM (used in ablation study and analysis), we also perform FC-tuning (used to compare with the prior works in Table 7) on Kinetics400, considering Kinetics400 is sufficiently large compared to UCF101 and HMDB51. Similar to the setup mentioned earlier, we extract fixed features, followed by a linear fully-connected layer is trained to perform action recognition. We use an AdamW (Loshchilov and Hutter 2017; Reddi, Kale, and Kumar 2019) optimizer with a Cosine scheduler having a base learning rate of 0.01 and batch size of 512 to train for 20 epochs.

A.6 Finetuning

Kinetics400, UCF101, and HMDB51. We use the pretrained visual encoder and add a fully-connected layer to finetune on Kinetics400, UCF101, and HMDB51. Different from our pretraining setup, we use a spatial resolution of 224^2 in finetuning. During training, we apply Random Augmentation (Cubuk et al. 2020) in addition to Multi-scale Crop and Random Horizontal Flip on the visual data, similar to (Tong et al. 2022; Feichtenhofer et al. 2022; He et al. 2021). We use an AdamW (Loshchilov and Hutter 2017; Reddi, Kale, and Kumar 2019) optimizer to train the network for 100 epochs using a cosine learning rate scheduler. In addition to the cosine learning rate scheduler, we employ layer-wise-layer-decay (Bao, Dong, and Wei 2021) which further stabilizes the training. We perform distributed training with a batch size of 96 using 16 NVIDIA V100 32 GB GPUs in parallel. Additionally, we employ MixUp (Zhang et al. 2017), CutMix (Yun et al. 2019), and label smoothing (Szegedy et al. 2016), which boosts the finetuning performance. We present the details of the finetuning parameters in Table S6. During training, we randomly sample 1 clip per sample, while during testing, 3 clips are uniformly selected per sample. We report the sample level prediction (top-1 and top-5 accuracy) averaged over all the clips.

Name	Kinetics400	UCF101	HMDB51
Epochs		100	
Batch Size		96	
Optimizer		`AdamW`	
Optimizer Betas		$[0.9, 0.999]$	
LR Scheduler		`Cosine`	
LR Warm-up Epochs	30	5	20
LR Warm-up		0	
LR Base	0.0005	0.0005	0.0001
LR Final		$1.0e^{-06}$	
Weight Decay		0.05	
RandAug		(9, .05)	
Label Smoothing		0.1	
MixUp	0.6	0.6	1.0
CutMix	0.5	0.5	1.0
Drop Path		0	
Dropout		0.5	
Layer-wise LR Decay	0.45	0.45	0.55

Table S6: **Video finetuning** parameters.

ESC50 and FSD50K. We use the pretrained audio encoder and add a linear layer on top of it to finetune on ESC50 and FSD50K. During training, we randomly sample 1 audio segment of 4 seconds for both datasets. However, in the case of validation, we uniformly sample 3 and 8 clips per sample for ESC50 and FSD50K respectively. The number of clips per sample during testing is chosen based on the average length of the samples to ensure the full length of the audio signal is captured. Following (Sarkar and Etemad 2023), we apply strong augmentations including Volume Jitter, Time-frequency Mask, Timewarp, and Random Crop. We use an AdamW optimizer with a batch size of 64 to train the network for 100 epochs. We use standard categorical cross-entropy error for ESC50, and binary cross-entropy error for FSD50K, as FSD50K is a multi-label multi-class classification dataset. We report top-1 accuracy for ESC50 and mAP for FSD50K. The networks are trained on 4 NVIDIA V100 32 GB GPUs. We list the additional hyperparameters in Table S7.

B Additional Experiments and Results

Following, we present an in-depth study analysing the key concepts of our proposed framework. Our thorough analysis includes the effect of temperature and EMA schedules; and the impact of local views and masking ratios; design choices for domain alignment, feature refinement, and projector head; and the performance of different XKD variants; among others. The models are pretrained for 400 epochs on Kinetics400 and we report linear evaluation top-1 accuracy using the split-1 of UCF101, HMDB51, and ESC50 unless stated otherwise.

B.1 Effect of temperature schedules

We study the effect of a wide range of temperature settings (τ in Equation 10) in our proposed framework. The results presented

Name	ESC50	FSD50K
Epochs	100	30
Batch Size	64	
Optimizer	AdamW	
Optimizer Betas	$[0.9, 0.999]$	
LR Scheduler	cosine	fixed
LR Warm-up Epochs	10	-
LR Base	0.0001	0.0001
LR Final	0	-
Weight Decay	0.005	$1.0e-05$
Early stop	no	yes
Volume Jitter	0.2	
Time-mask	$[0, 20]$	
Frequency-mask	$[0, 10]$	
Num of masks	2	
Timewarp window	20	
Drop Path	0	
Dropout	0.5	0.3
Layer-wise LR Decay	0.65	0.65

Table S7: **Audio finetuning** parameters.

Audio	0.04	0.06	0.07	0.08	0.07	0.07	0.07	0.07	$[0.04, 0.06]$	$[0.04, 0.06]$
Video	0.04	0.06	0.07	0.08	0.09	0.10	0.11	0.12	$[0.04, 0.06]$	$[0.09, 0.11]$
HMDB51	55.3	57.5	57.5	55.6	54.4	55.9	54.5	53.8	**58.5**	52.7
UCF101	80.0	81.0	82.0	81.6	81.0	81.9	79.5	78.5	**82.6**	79.0
ESC50	85.8	85.3	84.3	85.8	87.0	**89.0**	**89.0**	87.3	88.0	86.5

Table S8: **Effect of temperature schedules.** We highlight the best with **bold** and the second best with underline. We find that a temperature schedule of $[0.4, 0.6]$ (increasing 0.4 to 0.6 using a Cosine scheduler) achieves an overall stable performance. Interestingly, excessively increasing the video temperature to 0.10 or 0.11 (i.e., sharpening the video distribution) improves the performance of audio downstream tasks, however, it hurts the performance of visual tasks. On the other hand, a fixed temperature of 0.6 or 0.7 shows stable performance in video tasks, but it hurts audio performance. By default, we use $[0.4, 0.6]$ in all the experiments.

in Table S8 show that a relatively lower temperature works well on the audio teacher (0.04 to 0.07), whereas a high temperature (0.1 to 0.11) is required for the video teacher. Please note that the students' temperatures are kept fixed at 0.1. To find the best performance, we sweep a range of combinations and notice that for both audio and video teachers, increasing the temperatures from 0.04 to 0.06 works fairly well for both modalities. Additionally, we notice that setting fixed temperatures of 0.07 and 0.1 for audio and video teachers respectively, works slightly better on ESC50; however, a performance drop is noticed for video, specifically on HMDB51. We conjecture that carrying more information and thus the wider distribution of the video stream (see Figure 2 in the main paper) requires a higher temperature in order to further sharpen the distribution. Based on the results in Table S8, we use a scheduler to increase the temperature from 0.04 to 0.06 for both audio and video teachers throughout our experiments, unless mentioned otherwise.

B.2 Effect of different EMA schedules

We update the weights of the teachers using EMA as described in Equation 13. In particular, we set the values of EMA coefficients

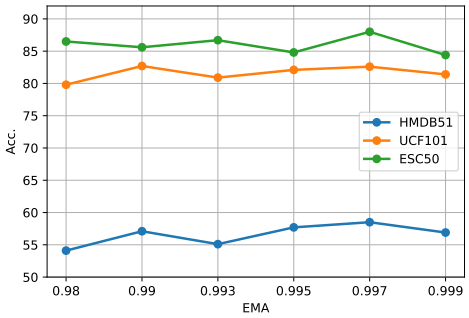

Figure S1: **Effect of EMA schedules.** EMA of 0.997 achieves the most stable performance in both audio (ESC50) and visual (HMDB51 and UCF101) tasks.

using a cosine scheduler with a final value of 1. This means that we update the teachers more frequently at the beginning of training, and slow down their weight update towards the end of the training. This strategy results in a very smooth update of the networks' parameters and results in stable performance (Tarvainen and Valpola 2017). To find the optimal setup, we vary a wide range of EMA coefficients between 0.98 to 0.999. The results presented in Figure S1 show that the EMA coefficient of 0.997 works well on both audio and video-related tasks.

B.3 Effect of local views

Temporal window size. We study the effect of the size of temporal windows used to create the local views for training the student networks. For the video stream, we create the local views by taking spatial crops of 96×96 pixels and vary the temporal windows in the range of $\{1, 2, 4\}$ seconds. Similarly for the audio stream, we explore temporal windows in the range of $\{1, 2\}$ seconds. Our results presented in Table S9 reveal some interesting findings. First, local views comprised of audio and visual sequences of just 1 second work the best on all three benchmarks. Interestingly, we notice that the downstream audio classification performance drops significantly when XKD uses long visual local views (see column $4-1$ in Table S9), while the downstream video classification performance remains more or less unchanged. On the other hand, we notice considerable performance drops on downstream video classification when using large audio segments to create the local views (see columns $2-2$ and $1-2$ in Table S9). These observations indicate that local views comprised of relatively shorter temporal windows are most effective when learning from the cross-modal teachers, which in turn results in improved downstream classification performance.

Vid-Aud (sec.)	**1−1**	**2−2**	**2−1**	**1−2**	**4−1**
HMDB51	**58.5**	55.2	57.1	55.6	58.4
UCF101	**82.6**	81.1	82.3	81.0	**82.6**
ESC50	**88.0**	86.8	87.3	86.8	82.3

Table S9: **Effect of different temporal window sizes for local views.** Here **bold** shows the best and underline shows the worst results. Our results indicate that local views comprised of relatively shorter temporal windows are most effective to learn from the cross-modal teachers.

Number of local views. We further investigate the effect of varying the number of local views. This experiment shows that single local

views for both audio and visual modalities work fairly well. We also find that adding more visual local views (e.g., 2 or 3) when using a single audio local view improves the performance on HMDB51 and UCF101, as shown in columns $2-1$ and $3-1$ in Table S10. Additionally, we notice that adding more audio local views worsens the model's performance in video classification (see columns $1-2$ and $2-2$ in Table S10). Lastly, we notice considerable improvements in sound classification when using 2 local views for both audio and video.

Vid-Aud (no.)	$1-1$	$1-2$	$2-1$	$2-2$	$3-1$
HMDB51	58.5	56.3	**58.8**	56.1	57.5
UCF101	82.6	79.5	82.2	79.0	**82.9**
ESC50	88.0	85.8	87.0	**90.8**	88.8

Table S10: **Effect of additional local views.** We highlight the best results in **bold** and the worst with underline. We find that XKD learns very strong representations just by using a single local view. Additionally, we encounter a few setups which show superior results on different downstream benchmarks. However, we do not see one single variant that performs the best in all setups.

B.4 Effect of different mask ratio

We study the effect of different amounts of masking ratios in our proposed framework. Specifically, we test different combinations of audio-masking ratio vs. video-masking ratio to test if there is an internal dependency that might result in optimal performance for both modalities. We sweep a wide ranges of mask ratios, particularly $\{0.80, 0.85, 0.90\}$ for video masking and $\{0.75, 0.80, 0.85\}$ for audio masking. The results presented in Figure S2 show that a video mask ratio of 85% and an audio mask ratio of 80% work well on both UCF101 and HMDB51. However, we notice that a slightly lower video mask ratio (e.g., 80%) improves audio performance. We conjecture that this setup benefits from the availability of more visual patches which provides better supervision through cross-modal knowledge distillation. Please note that unless stated otherwise, we use a visual mask ratio of 85% and an audio mask ratio of 80%.

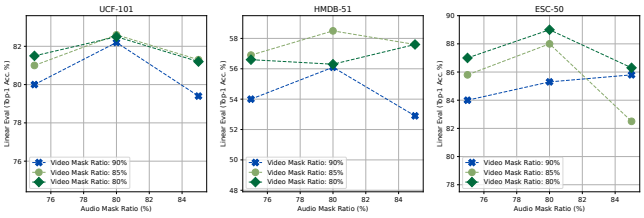

Figure S2: **Effect of different masking ratios.** We find that a video mask ratio of 85% works best on both UCF101 and HMDB51. Additionally, an audio mask ratio of 80% achieves superior results on ESC50.

B.5 Avoiding collapse and training instability

Here we summarize the setups that cause a collapse and instability in cross-modal knowledge distillation. We identify three such instances, (*i*) without normalization layer in the projectors, (*ii*) without $\mathcal{L}_{da}$, and (*iii*) without normalizing the teacher's representations based on the current batch statistics, i.e., Centering (Caron et al. 2021). To identify collapse, we mainly track (not optimize) the Kullback–Leibler divergence between teacher and students ($\mathcal{L}_{kld}$) and Knowledge Distillation ($\mathcal{L}_{kd}$) losses, as shown in Figure S3. A collapse can be identified if either $\mathcal{L}_{kd}$ or $\mathcal{L}_{kld}$ is zero. First, when no normalization layer is added in the projector head, the $\mathcal{L}_{kld}$ becomes zero, which indicates that the models output a constant vector. Second, without $\mathcal{L}_{da}$, the $\mathcal{L}_{kld}$ also reaches zero (with minor spikes), which indicates training collapse. Third, we notice that without the Centering, $\mathcal{L}_{kld}$ does not become zero, while the $\mathcal{L}_{kd}$ reaches zero, which also indicates training collapse. In addition to the ablation studies presented in Tables 1 and 2 (in the main paper), we attempt to train XKD without the masked reconstruction loss (i.e., setting λ_{ae} to 0 in Equation 12), and quite expectedly we face training collapse. This is due to the fact that in order to perform effective cross-modal knowledge distillation, the model first needs to learn meaningful modality-specific representations. To provide more insights into the training process, we present the default training curves in Figure S4.

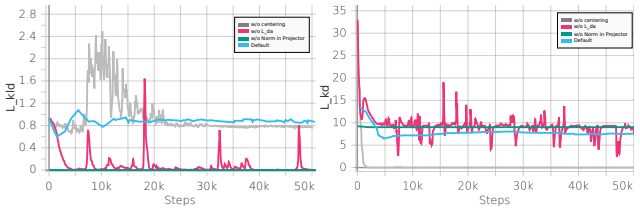

Figure S3: Visualizing **pretraining instability** and **collapse** in cross-modal knowledge distillation.

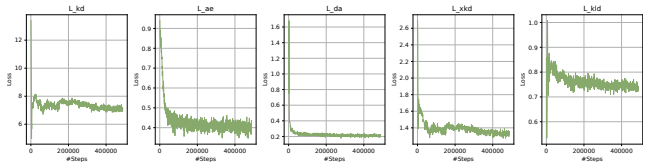

Figure S4: **Stable** training curves are presented.

B.6 Design choices for projector head

The configuration of the projector heads plays an important role in the performance of our method. We experiment with a variety of different setups including normalization, output dimension, and the number of layers.

Effect of normalization. We investigate three setups, (*a*) no normalization, (*b*) normalization on the last layer, and (*c*) normalization on every layer. Our study shows that the cross-modal knowledge distillation fails in the absence of the normalization layer (please see additional discussions in B.5). We then observe that adding normalization on the last layer prevents the model from collapsing while adding normalization on every layer shows significant improvements (2.9% to 4.5%) as presented in Figure S5.

Effect of the number of layers and output dimension. We further explore different setups for the projector head. First, we present the effect of varying the number of layers and output dimensions in Tables S11 and S12. The results show that a projector head consisting of 3 layers performs most steadily on all the datasets. We also find that an output dimension of 8192 shows stable performance on all the downstream benchmarks.

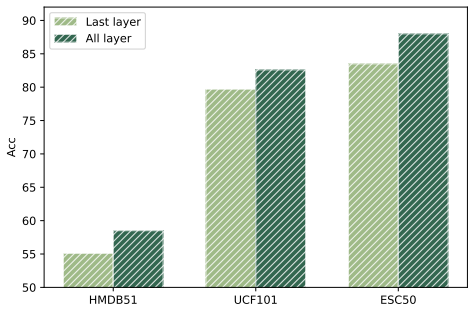

Figure S5: **Effect of normalization layers in the projector head.** We find that applying normalization to all the layers of the projector head improves the performance across all the downstream tasks.

No. of layers	2	3	4
HMDB51	55.4	**58.5**	55.7
UCF101	81.0	**82.6**	81.0
ESC50	83.4	**88.0**	84.6

Table S11: **Effect of different number of projector layers.** We notice an MLP projector head with 3 layers achieves the best performance.

Output dim.	4096	8192	16384	32768
HMDB51	53.2	**58.5**	57.0	54.4
UCF101	78.1	**82.6**	81.5	81.7
ESC50	87.5	**88.0**	84.1	83.8

Table S12: **Effect of different projector output dimensions.** We notice that the output dimensions of 8192 achieve the best performance. Please note that in all the setup the hidden layers consist of fixed dimensions of 2048.

B.7 Design choices for domain alignment

We conduct a thorough study exploring the optimal setup for domain alignment ($\mathcal{L}_{\text{da}}$). Recall, $\mathcal{L}_{\text{da}}$ is defined as:

$$\mathcal{L}_{\text{da}} = \mathcal{L}_{\text{mmd}}(f_s^v, f_s^a) + \mathcal{L}_{\text{mmd}}(f_t^v, f_t^a)$$

We explore the following setups. First, instead of minimizing the MMD loss between teachers (f_t^a, f_t^v) and between students (f_s^a, f_s^v) as done in $\mathcal{L}_{\text{da}}$, we minimize the distance between the teachers and students as:

$$\mathcal{L}_{\text{da}}^1 = \mathcal{L}_{\text{mmd}}(f_t^a, f_s^v) + \mathcal{L}_{\text{mmd}}(f_t^v, f_s^a). \tag{S1}$$

Next, we study the effect of optimizing $\mathcal{L}_{\text{da}}^1$ in addition to $\mathcal{L}_{\text{da}}$, expressed as:

$$\mathcal{L}_{\text{da}}^2 = \mathcal{L}_{\text{da}} + \mathcal{L}_{\text{da}}^1. \tag{S2}$$

The results presented in Table S13 indicate drops in performance when using $\mathcal{L}_{\text{da}}^1$ and $\mathcal{L}_{\text{da}}^2$. In other words, we find that just minimizing the MMD loss between representations of a similar nature is more effective for domain alignment and minimizing domain discrepancy. To further clarify, teachers are slowly updated than the students and teachers provide global representations while students focus on local representations. We use $\mathcal{L}_{\text{da}}$ by default in all our experiments.

	$\mathcal{L}_{\text{da}}^1$	$\mathcal{L}_{\text{da}}^2$	$\mathcal{L}_{\text{da}}$ **(default)**
HMDB51	56.1	55.4	**58.5**
UCF101	81.1	82.2	**82.6**
ESC50	87.3	87.5	**88.0**

Table S13: **Exploring the design choices for domain alignment.** We find that just minimizing the MMD loss between the teachers and between the students compared to the other variants works best for effective cross-modal knowledge distillation.

B.8 Design choices for feature refinement

Here, we perform additional studies to evaluate our design choices for the refine step.

Cross-modal attention map. To calculate cross-modal feature relevance, we use the cross-modal attention maps as mentioned in Equation 4. We find that a potential alternative to our default Equation 4 can be Equation S3. In particular, the scaling operation in Equation 4 can be simply replaced by a softmax function.

$$\begin{aligned} A_v^{'\times} &= \text{softmax}(\text{MeanPool}(A_v \cdot A_a^T)); \\ A_a^{'\times} &= \text{softmax}(\text{MeanPool}(A_a \cdot A_v^T)) \end{aligned} \tag{S3}$$

We notice that this technique works slightly better on audio representations in comparison to our default setup; however, a considerable performance drop is noticed on visual representations. Please see the results in Table S14. Please note that unless mentioned otherwise, we use the default Equation 4 in all the setups.

	$\mathbf{A}^{'\times}$	w/o `CLS`	default
HMDB51	54.6	54.5	**58.5**
UCF101	79.8	79.3	**82.6**
ESC50	**88.3**	84.5	88.0

Table S14: **Exploring design choices for feature refinement.** We find that an alternative strategy to obtain the cross-modal attention maps ($\mathbf{A}^{'\times}$) may work marginally better on audio, but it significantly degrades the video performance. Additionally, we notice that the models benefit from the availability of `CLS` token in both audio and visual tasks. These results confirm the superiority of our proposed design choices for effective cross-modal knowledge distillation.

CLS token. As mentioned in Equation 3, by default we use the `CLS` tokens to obtain A_v and A_a, which are then used to generate $A_v^{'\times}$ and $A_a^{'\times}$. Here, we explicitly study if `CLS` tokens are necessary in our framework. To test this, we modify Equation 3 and calculate the intra-modal attention map as the correlation of the mean of the query embeddings and the key embeddings of all other patches. Please note that the rest of the setup remains the same. Table S14 shows that removing the `CLS` token significantly degrades the model performance on both audio and visual representations.

B.9 Comparing different variants of XKD

We conduct a thorough comparison between XKD and its different variants. In Table S15, we present a detailed comparison between the modality-specific and modality-agnostic variants using both teacher and student encoders. As the teachers and students may

	XKD-MATS		XKD-MAS		XKD	
	T (MA)	S (MA)	T (MS)	S (MA)	T (MS)	S (MS)
HMDB51	53.5	52.4	56.4	55.2(↓4.1)	**59.3**	57.8
UCF101	80.3	80.0	81.9	81.5(↓3.2)	**84.7**	84.4
ESC50	90.3(↓0.7)	89.8	91.0	89.5	**91.0**	90.3

Table S15: **Comparison between teachers (T) and students (S) for both MA and MS variants.** We highlight the performance drop in the best MA variants with respect to the **best** MS variant. Moreover, teachers always perform better with respect to the students.

converge at different rates due to their difference in the weight update schedule, these variants are trained for full training schedules of 800 epochs. We report top-1 accuracy of linear evaluations using the split-1 of UCF101, HMDB51, and ESC50. As shown in Table S15, modality-agnostic variants show performance drops (e.g., 3.2% on UCF101 and 0.7% on ESC50) in comparison to the modality-specific one. Moreover, as presented in the main paper, the performance gap between the modality-agnostic and modality-specific variants further reduces when finetuned (e.g., 0.7% on UCF101, 1.7% on Kinetics400, and 0.3% on FSD50K). Amongst the modality-agnostic backbones, XKD-MATS works slightly better on ESC50, whereas, XKD-MAS shows better performance on both UCF101 and HMDB51. Lastly, when comparing the performance between the teachers and students, teachers show slightly better performance on all the benchmarks and in all the setups. Following (Caron et al. 2021), we adopt the terminology of 'teacher-student' to simply refer to the stronger network as a teacher and the slightly weaker one as a student.

B.10 Training schedule

In Figure S6, we present the effect of longer pretraining schedules on video action classification for both teacher and student networks. We pretrain the XKD for up to 800 epochs and find that pretraining for around 600 epochs shows improvement, while beyond that point, a slight performance drop is noticed on HMDB51 and no change is noticed on UCF101. Moreover, we find that the teacher always outperforms the student in downstream evaluation.

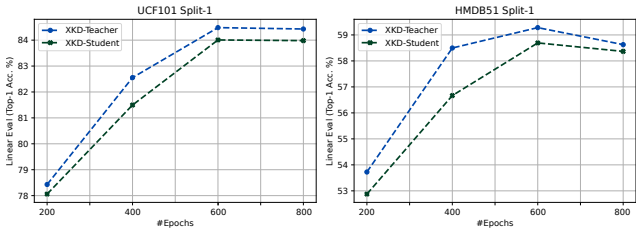

Figure S6: **Effect of longer pretraining schedules** with respect to the downstream task performance. We notice that the teacher always outperforms the students throughout the training. Additionally, little or no improvements are noticed beyond 600 epochs.

B.11 Detailed comparison with VideoMAE

In addition to the earlier results presented in the main paper, here we perform a thorough comparison of our method to video masked reconstruction, i.e., VideoMAE (Tong et al. 2022), on video action recognition, and present the results in Table S16. In this experiment, we strictly follow the standard linear evaluation protocol (Morgado, Vasconcelos, and Misra 2021; Alwassel et al. 2020; Sarkar and Etemad 2023; Xiao et al. 2020) since it is a more appropriate method for evaluating self-supervised representation learning as pointed out in (Xiao et al. 2020; Qian et al. 2021). Our comparisons are mainly 3-fold.

First, we compare XKD to VideoMAE (Tong et al. 2022) and notice that XKD outperforms VideoMAE[2] by 3.8% and 3.1% on UCF101 and HMDB51, respectively. The improvements in XKD's results clearly show the benefits of cross-modal knowledge distillation in learning better and more discriminative representations. However, it should be noted that VideoMAE (Tong et al. 2022) is pretrained using industry-level computation (64 32GB V100 GPUs) with a higher frame resolution of 224^2, whereas, we use a lower resolution of 112^2 (using just 8 V100 GPUs). We use a frame resolution of 112^2 to reduce the computation cost, i.e., while VideoMAE (Tong et al. 2022) is pretrained with 1568 visual tokens, XKD uses 392 tokens, reducing the computation cost by almost $4\times$.

Therefore, to have a fair comparison between these two methods, we pretrain VideoMAE with a frame resolution of 112^2 using the publicly available VideoMAE (Tong et al. 2022) codes. We refer to this variant as 'VideoMAE (112)' as presented in Table S16. XKD outperforms VideoMAE (112) by 6.7% and 5.8% on UCF101 and HMDB51. Lastly, we find another minor difference in the input setup between XKD and VideoMAE. VideoMAE uses 16 frames as input with a temporal patch size of 2, whereas we use 32 frames and a temporal patch size of 4. Therefore, for a more fair comparison, we pretrain VideoMAE in an identical setup to ours, i.e, with visual inputs of 32×112^2 and patch size of 4×16, and denote it as VideoMAE* in Table S16. As presented in Table S16, XKD outperforms VideoMAE* by 4.9% and 5.9% on UCF101 and HMDB51, respectively. Lastly, XKD outperforms VideoMAE* by a very large margin of 13.9 and 15.7 on Kinetics-Sound and Kinetics400. Such large performance improvement demonstrates that XKD learns better features compared to video-masked reconstruction frameworks.

C Qualitative analysis

C.1 Effect of feature refinement

In Figure S7, we present several examples showing the impact of our proposed refine strategy in identifying the key visual features.

C.2 XKD reconstruction results

In Figures S8, S9 and S10, we present several examples showing the reconstruction capability of XKD with respect to varying masked inputs. Additional examples of reconstruction of video frames and audio spectrograms from highly masked inputs are presented in Figures S11 and S12. XKD demonstrates high reconstruction performance even when a high mask ratio is applied.

[2]We perform linear evaluations using the official checkpoints released at: https://github.com/MCG-NJU/VideoMAE/tree/main

Method	Backbone	Input	Patch Size	# Tokens	UCF101	HMDB51	Kinetics-Sound	Kinetics400†
VideoMAE** (224) (Tong et al. 2022)	ViT-B	16×224^2	2×16^2	1568	80.0	54.3	-	-
VideoMAE (112)	ViT-B	16×112^2	2×16^2	392	77.1	51.6	-	-
VideoMAE*	ViT-B	32×112^2	4×16^2	392	78.9	51.5	56.8	30.7/38.4
XKD	ViT-B	32×112^2	4×16^2	**392**	**83.8**	**57.4**	**70.7**	**46.4/51.4**

Table S16: **Comparison between VideoMAE and XKD** in linear evaluation. We report top-1 accuracy, averaged over all the splits for UCF101 and HMDB51. As Kinetics400 is sufficiently large, we perform linear evaluations in both setups, SVM and FC-tuning. XKD outperforms VideoMAE* by 4.9% on UCF101, 5.9% on HMDB51, 13.9% on Kinetics-Sound, and by 15.7% on Kinetics400. VideoMAE** (224) uses 1568 visual tokens, whereas, XKD uses 392 tokens, reducing the pretraining computational cost for the video encoder by 4 times. Note: * identical setup to XKD, ** industry level computation, †SVM/FC.

Figure S7: Visualizing the **effect of refine** in identifying the key visual features. (Best viewed in color.)

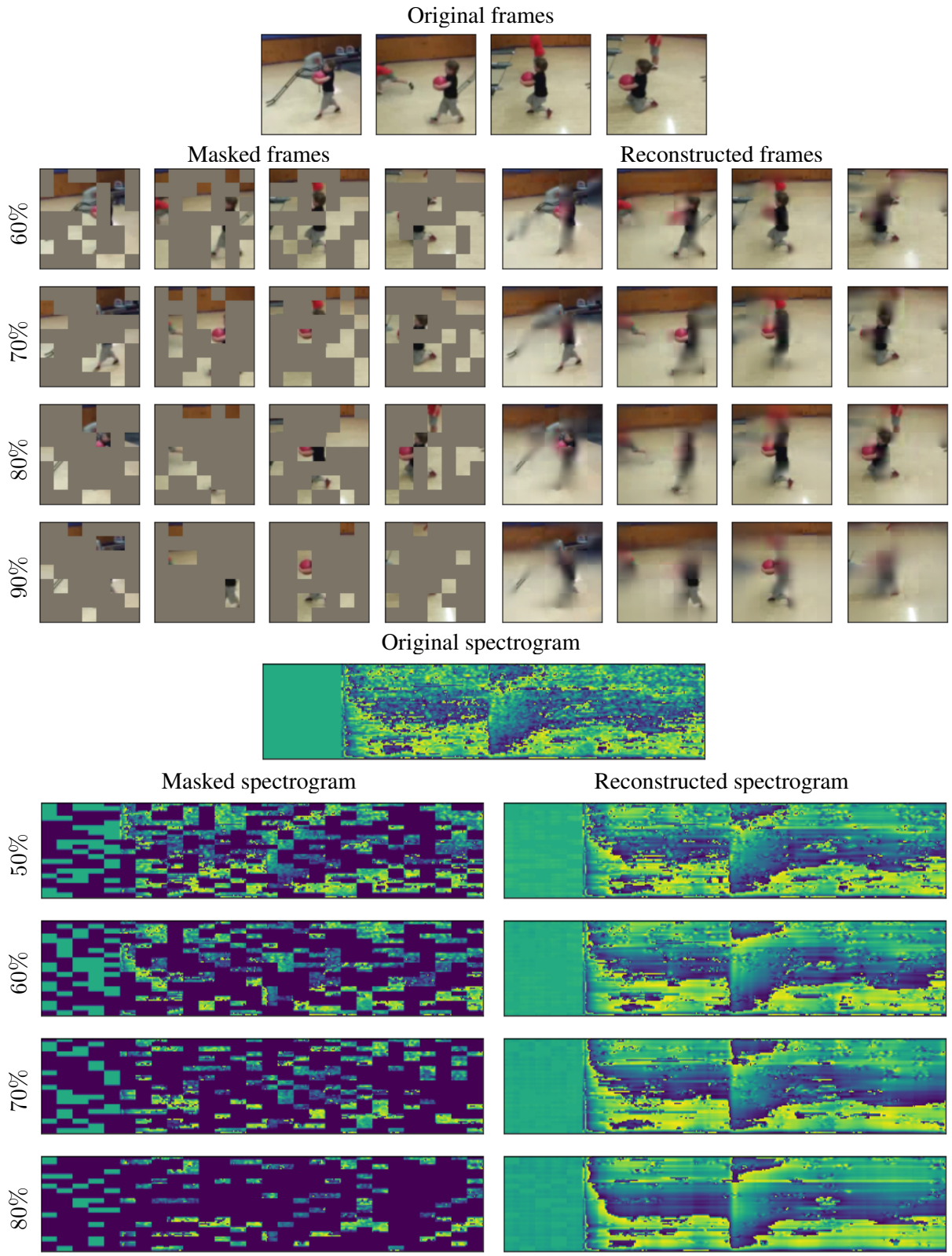

Figure S8: Examples of reconstruction with **varying masked inputs**. We use the pretrained XKD and during inference, the video mask ratio is varied from 60% to 90% and the audio mask ratio is varied from 50% to 80%. XKD demonstrates high reconstruction performance even when a very high mask ratio is applied for both audio and visual modalities.

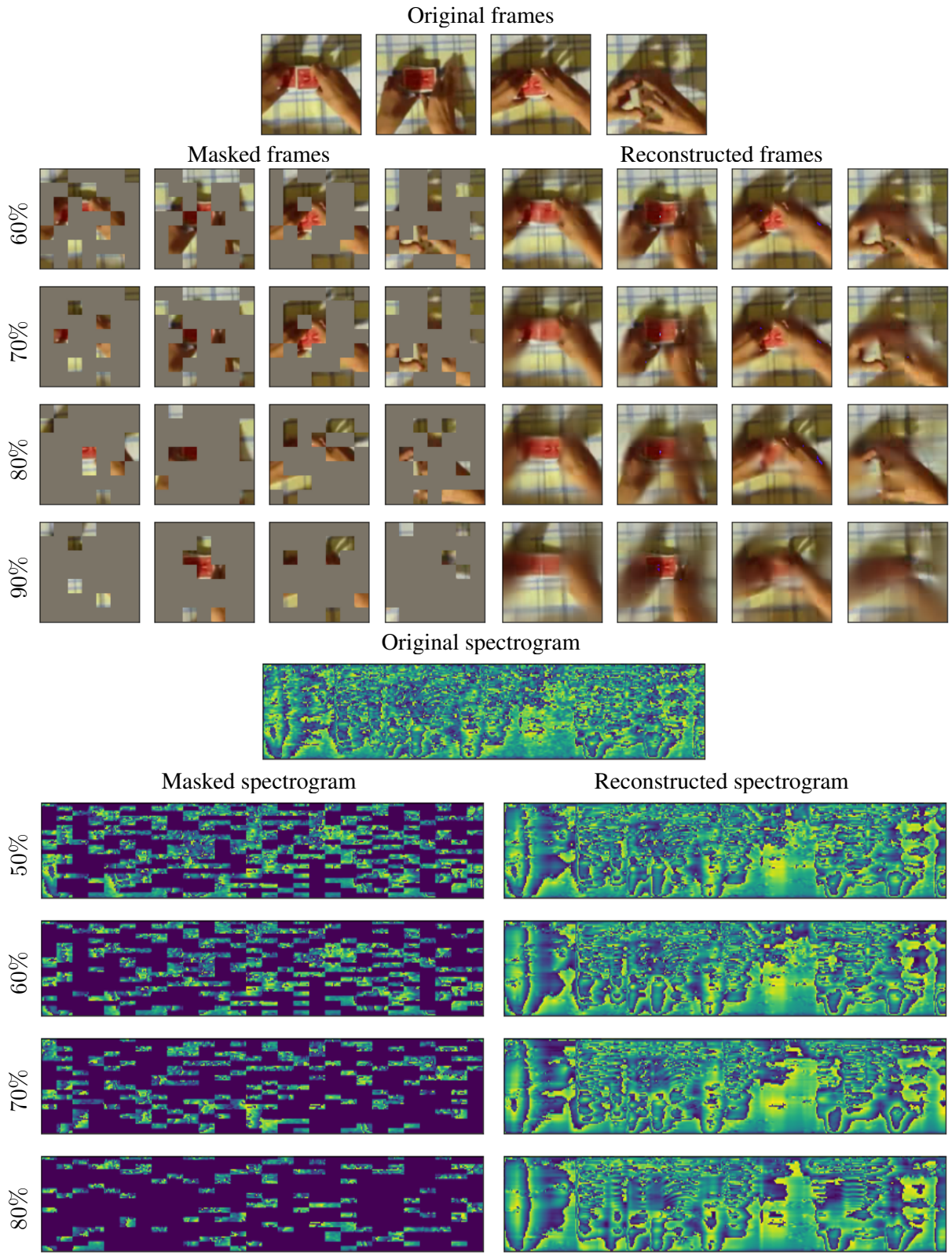

Figure S9: Examples of reconstruction with **varying masked inputs**. We use the pretrained XKD and during inference, the video mask ratio is varied from 60% to 90% and the audio mask ratio is varied from 50% to 80%. XKD demonstrates high reconstruction performance even when a very high mask ratio is applied for both audio and visual modalities.

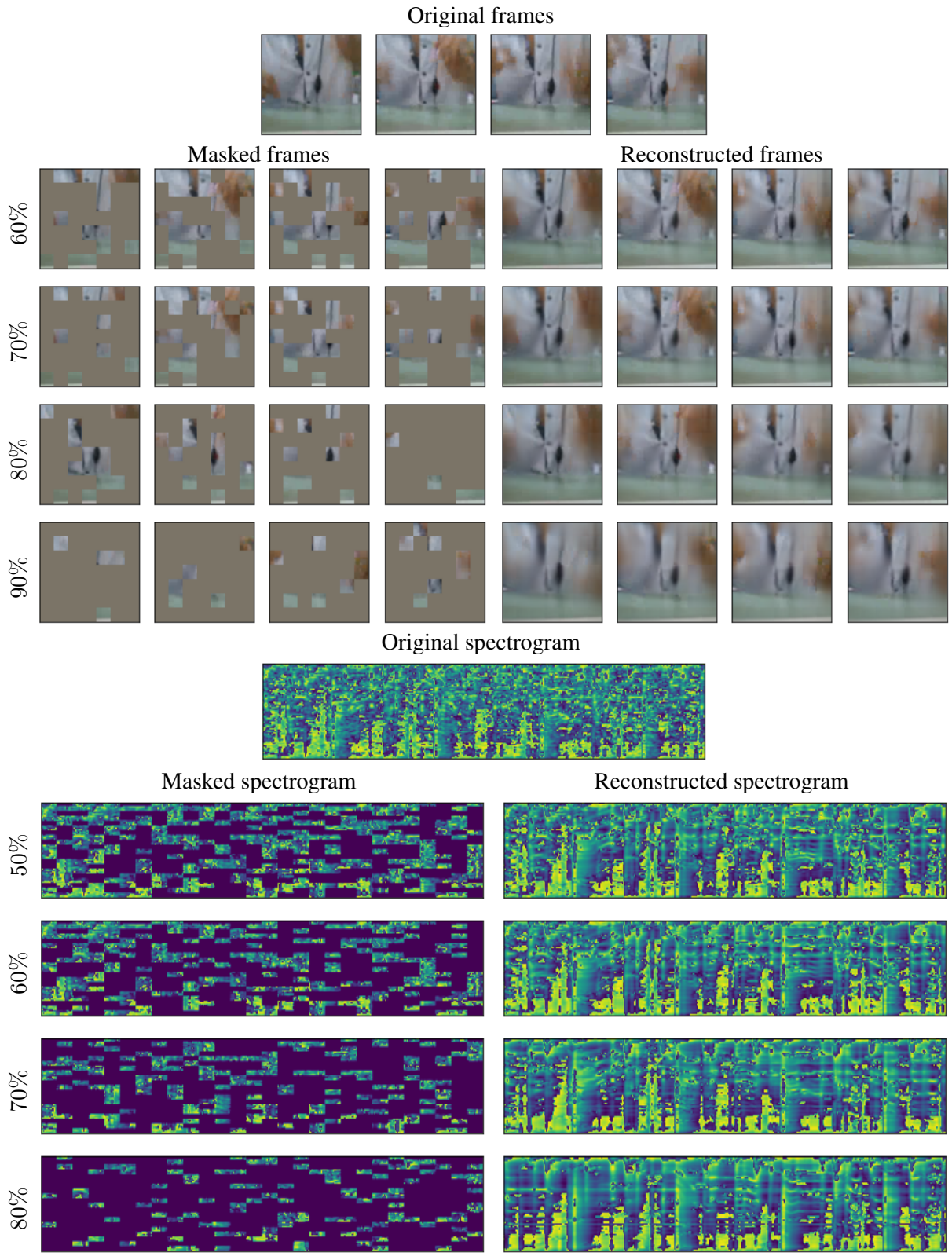

Figure S10: Examples of reconstruction with **varying masked inputs**. We use the pretrained XKD and during inference, the video mask ratio is varied from 60% to 90% and the audio mask ratio is varied from 50% to 80%. XKD demonstrates high reconstruction performance even when a very high mask ratio is applied for both audio and visual modalities.

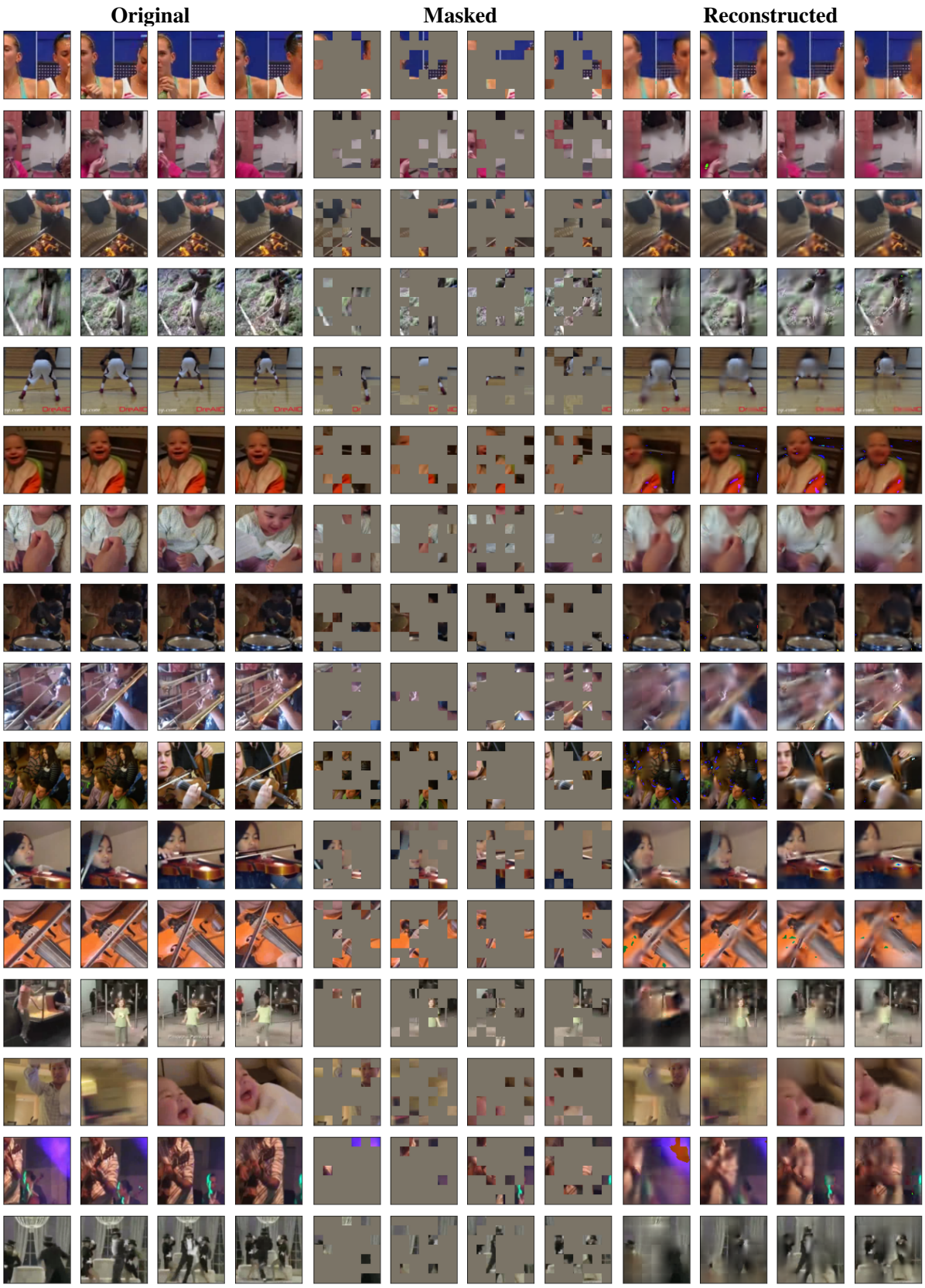

Figure S11: Examples of **video frame** reconstruction ability of XKD from highly masked (80%) inputs.

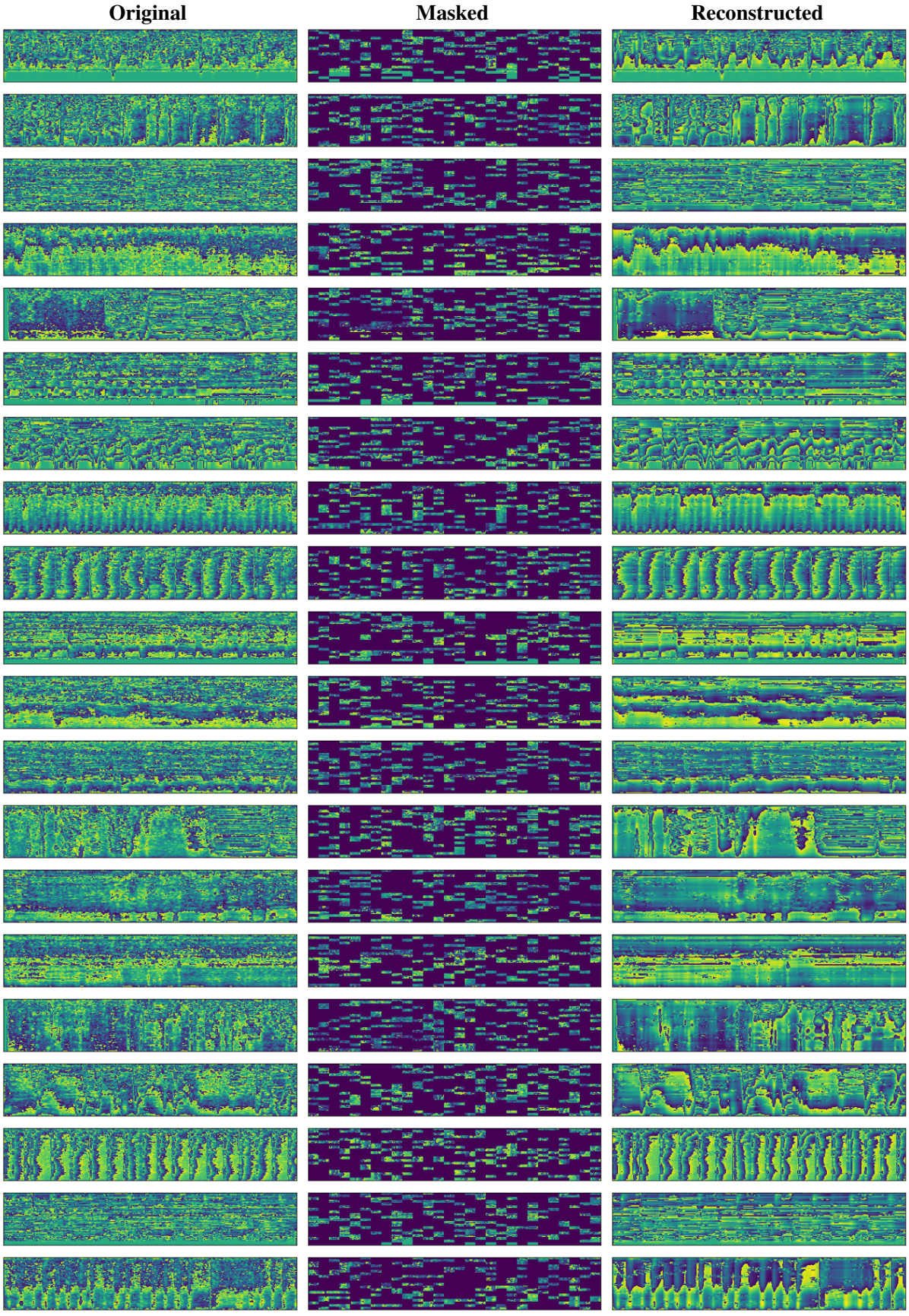

Figure S12: Examples of **audio spectrogram** reconstruction ability of XKD from highly masked (70%) inputs.