Self-Supervised Audio-Visual Representation Learning with Relaxed Cross-Modal Synchronicity (Supplementary Material)

Pritam Sarkar[1, 2] **Ali Etemad**[1]
[1] Queen's University, Canada [2] Vector Institute
{pritam.sarkar, ali.etemad}@queensu.ca
https://pritamqu.github.io/CrissCross

The organization of the supplementary material is as follows:

- Appendix A: Pseudocode;
- Appendix B: Qualitative Analysis;
- Appendix C: Datasets;
- Appendix D: Data Augmentations;
- Appendix E: Evaluation Protocols;
- Appendix G: Hyperparameters;
- Appendix F: Architectures;
- Appendix H: Limitations;
- Appendix I: Broader Impact.

A Pseudocode

We present the pseudocode of our proposed CrissCross framework in Algorithm 1.

Algorithm 1: CrissCross pseudocode (PyTorch style).

```
# fv: visual encoder (backbone+projection mlp)
# fa: audio encoder (backbone+projection mlp)
# hv: visual predictor head (prediction mlp)
# ha: audio predictor head (prediction mlp)
# D: loss function, following Eqn. 1

def forward(v1, v2, a1, a2):
   """
   v1,V2: minibatch of augmented visual samples
   a1,a2: minibatch of augmented audio samples
   """

   # visual
   zv1, zv2 = fv(v1), fv(v2) # visual embeddings
   pv1, pv2 = hv(zv1), hv(zv2) # predictor output

   # audio
   za1, za2 = fa(a1), fa(a2) # audio embeddings
   pa1, pa2 = ha(za1), ha(za2) # predictor output

   # loss calculation

   # intra-modal loss, following Eqn. 2
   L_intra = D(pv1, zv2)/2 + D(pv2, zv1)/2 + \
          D(pa1, za2)/2 + D(pa2, za1)/2

   # synchronous cross-modal loss, following Eqn. 3
   L_sync = (D(pv1, za1)/2 + D(pa1, zv1)/2 + Lv2a2 +\
          D(pv2, za2)/2 + D(pa2, zv2)/2)/2

   # asynchronous cross-modal loss, following Eqn. 4
   L_async = (D(pv1, za2)/2 + D(pa2, zv1)/2 +\
           D(pa1, zv2)/2 + D(pv2, za1)/2)/2

   # total loss, following Eqn. 5
   L_CrissCross = (L_async + L_sync + L_intra)/3

   return L_CrissCross
```

B Qualitative Analysis

To perform a qualitative analysis of the learned representations in an unsupervised setup, we present the nearest neighborhoods of video-to-video and audio-to-audio retrieval in Figures S1 and S2. In this experiment, we use Kinetics400 (Kay et al. 2017) to pretrain CrissCross. Next, we use the features extracted from randomly selected samples of the validation split to query the training features. We find that in most of the cases CrissCross performs fairly well, we notice very few instances of wrong retrieval, which generally occur when the visual scenes or sound events are very similar. For instance, 'playing piano' and 'playing organ' for video-to-video retrieval and 'playing keyboard' and 'playing xylophone' for audio-to-audio retrieval.

C Datasets

C.1 Pretraining Datasets

We use 3 datasets of different sizes for pretraining, namely, Kinetics-Sound (Arandjelovic and Zisserman 2017), Kinetics400 (Kay et al. 2017), and AudioSet (Gemmeke et al. 2017). Kinetics-Sound is a small-scale action recognition dataset, which has a total of 22K video clips, distributed over 32 action classes. Kinetics400 is a medium-scale human action recognition dataset, originally collected from YouTube. It has a total of 240K training samples and 400 action classes. Please note that Kinetics-Sound is a subset of Kinetics400, and consists of action classes which are prominently manifested audibly and visually (Arandjelovic and Zisserman 2017). Lastly, AudioSet (Gemmeke et al. 2017) is a large-scale video dataset of audio events consisting of a total of 1.8M audio-video segments originally obtained from YouTube spread over 632 audio classes. Please note that none of the provided labels are used in self-supervised pretraining.

C.2 Downstream Datasets

Following the standard practices of prior works (Morgado, Vasconcelos, and Misra 2021; Morgado, Misra, and Vasconcelos 2021; Alayrac et al. 2020; Alwassel et al. 2020;

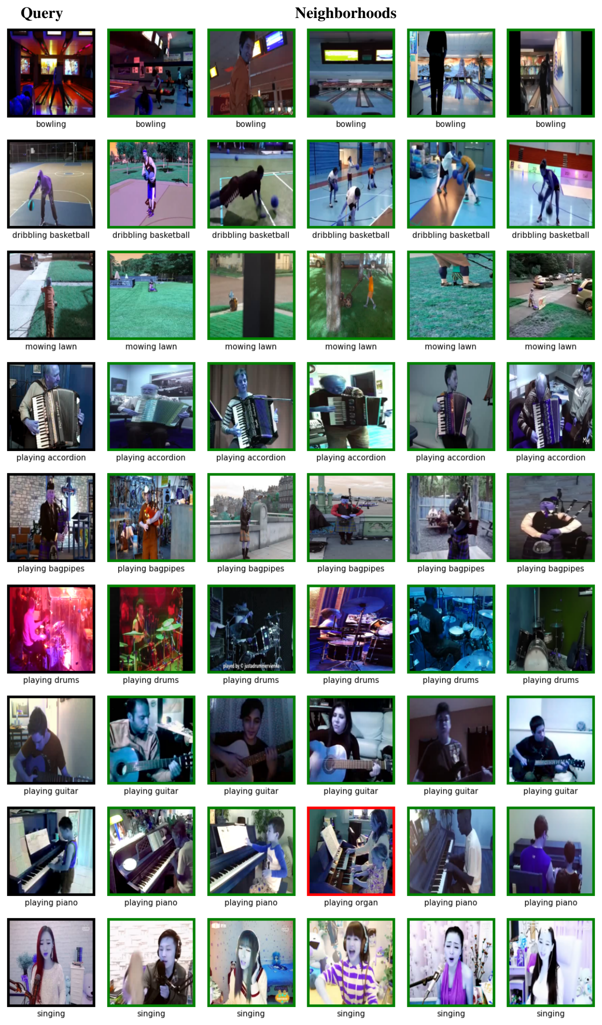

Figure S1: We present a few randomly selected samples of **video-to-video** retrieval. Here, the frames with black borders represent the query, and the next 5 frames represent the top-5 neighborhoods. The correct retrievals are marked with green, while the wrong ones are marked with red.

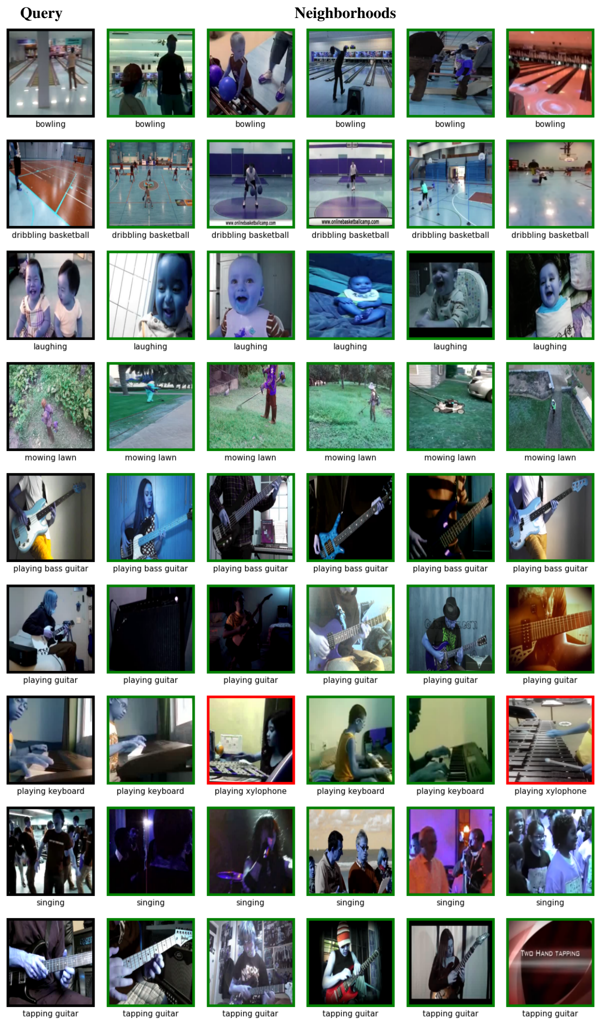

Figure S2: We present a few randomly selected samples of **audio-to-audio** retrieval. Here, the frames with black borders represent the query, and the next 5 frames represent the top-5 neighborhoods. The correct retrievals are marked with green, while the wrong ones are marked with red.

Asano et al. 2020; Korbar, Tran, and Torresani 2018), we evaluate our self-supervised methods on two types of downstream tasks: (*i*) action recognition based on visual representations and (*ii*) sound classification based on audio representations. To perform action recognition, we use two popular benchmarks, i.e., UCF101 (Soomro, Zamir, and Shah 2012) and HMDB51 (Kuehne et al. 2011). UCF101 consists of a total of 13K clips distributed among 101 action classes, while HMDB contains nearly 7K video clips distributed over 51 action categories. To perform sound classification, we use two popular benchmarks ESC50 (Piczak 2015) and DCASE2014 (Stowell et al. 2015). ESC50 is a collection of 2K audio events comprised of 50 classes and DCASE2014 is an audio event dataset of 100 recordings spread over 10 categories.

D Data Augmentation

Here we present the details of the augmentation parameters for both visual and audio modalities.

D.1 Visual Augmentations

The parameters for visual augmentations are presented in Table S1. Some of the parameters are chosen from the literature, while the rest are found through empirical search. We set the parameters of Multi-Scale Crop, Gaussian Blur, and Gray Scale as suggested in (Chen et al. 2020), and the parameters for Color Jitter are taken from (Morgado, Vasconcelos, and Misra 2021). We use TorchVision (Paszke et al. 2019) for all the implementations of visual augmentations, except Cutout where we use the implementation available

Augmentation	**Parameters**
Multi Scale Crop	`min area = 0.08`
Horizontal Flip	`p = 0.5`
Color Jitter	`brightness = 0.4` `contrast = 0.4` `saturation = 0.4` `hue = 0.2`
Gray Scale	`p = 0.2`
Gaussian Blur	`p = 0.5`
Cutout	`max size = 20` `num = 1`

Table S1: Visual augmentation parameters.

Augmentation	**Parameters**
Volume Jitter	`range = ±0.2`
Time Mask	`max size = 20` `num = 2`
Frequency Mask	`max size = 10` `num = 2`
Timewarp	`wrap window = 20`
Random Crop	`range = [0.6,1.5]` `crop scale = [1.0,1.5]`

Table S2: Audio augmentation parameters.

here[1]. Please note that for the Cutout transformation, the mask is created with the mean value of the first frame in the sequence.

D.2 Audio Augmentations

We present the parameters used for audio augmentations in Table S2. We use the Librosa(McFee et al. 2015) library to generate mel-spectrograms. We use the techniques proposed in (Park et al. 2019) to perform Time Mask, Frequency Mask, and Time Warp transformations[2]. The parameters for the audio augmentations are set empirically, except for Random Crop which we adopt from (Niizumi et al. 2021).

E Evaluation Protocol

To evaluate the representations learned with self-supervised pretraining, we test the proposed framework in different setups, namely linear evaluation, full finetuning, and retrieval. The details of the evaluation protocols are mentioned below.

E.1 Linear Evaluation

To perform linear evaluations of the learned representations on downstream tasks, we extract fixed features (also called frozen features) using the pretrained backbones. We train a linear classifier using the fixed feature representations. The details are presented below.

Action Recognition.
To perform linear evaluations on action recognition, we follow standard evaluation protocols laid out in prior works (Alayrac et al. 2020; Recasens et al. 2021; Patrick et al. 2021; Morgado, Vasconcelos, and Misra 2021). The details are presented below.

HMDB51 and UCF101. We perform linear evaluations in 2 setups, i.e., 8-frame and 32-frame inputs. We evaluate on 8-frame inputs for the design explorations and 32-frame inputs for large-scale experiments.

Following the protocols mentioned in (Alayrac et al. 2020; Recasens et al. 2021), we feed 8-frame inputs to the video backbone, with a spatial resolution of 224^2. During training, we randomly pick 25 clips per sample to extract augmented representations, while during testing, we uniformly select 10 clips per sample and report top-1 accuracy at sample-level prediction by averaging clip-level predictions. The augmentation techniques are mentioned in Section D. We don't apply the Gaussian Blur while extracting the training features since it deteriorates the performance. Moreover, to perform a deterministic evaluation, we don't apply any augmentations during validation. The visual features are extracted from the final convolution layer and passed to a max-pool layer with a kernel size of $(1, 4, 4)$ (Morgado, Vasconcelos, and Misra 2021). Finally, we use the learned visual representations to train a linear SVM classifier, we sweep the cost values between {0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 1} and report the best accuracy.

When validating on 32-frame inputs, we could not perform SVM as the feature vector is too large to hold in the memory. Hence, we use a linear fully-connected layer at the end of the video backbone. Note that during training the backbone is kept frozen and only the linear layer is trained. we keep the rest of the setup the same as described earlier, with the exception of training where we randomly select 10 clips per sample.

Kinetics400. As Kinetics400 (Kay et al. 2017) is a large-scale dataset, the feature vector is too large to save in memory. Following (Morgado, Vasconcelos, and Misra 2021), we use a fully connected layer at the end of the frozen back-

	MSC	HF	CJ	GS	GB	C
Pretraining	✓	✓	✓	✓	✓	✓
Full-finetune	✓	✓	✓	✓	✗	✓
Linear evaluation	✓	✓	✓	✓	✗	✓

Table S3: Audio augmentation summary.

	VJ	Mask	RC	TW
Pretraining	✓	✓	✓	✗
Linear evaluation	✓	✓	✓	✓

Table S4: Visual augmentation summary.

[1]https://github.com/uoguelph-mlrg/Cutout
[2]https://github.com/s3prl/s3prl

bone and feed 8×224^2 frame inputs. During training, we randomly pick 1 clip per sample, while during validation, we uniformly select 10 clips per sample. Note that the rest of the setups remain the same, as described for HMDB51 and UCF101. Finally, we obtain the sample-level prediction by averaging the clip-level predictions and report the top-1 accuracy.

Sound Classification.

In case of evaluating audio representations, we follow the evaluation protocol laid out in prior works (Morgado, Vasconcelos, and Misra 2021; Alwassel et al. 2020; Alayrac et al. 2020; Recasens et al. 2021) for respective datasets. The details are mentioned below.

ESC50. We perform linear evaluations on ESC50 in 2 setups, we use 2-second audio input for design exploration and 5-second audio input for large-scale experiments. Following (Patrick et al. 2021), we extract 10 epochs worth of augmented feature vectors from the training clips. During testing, when using 2-second inputs, we extract 10 equally spaced audio segments (Morgado, Vasconcelos, and Misra 2021; Patrick et al. 2021; Alwassel et al. 2020), and when using 5-second inputs, we extract 1 segment (Alayrac et al. 2020; Recasens et al. 2021) from each sample. We perform the augmentations mentioned in Section D to extract the training features. We notice that unlike self-supervised pretraining, time warping improves the model performance in the linear evaluation. We do not apply any augmentations during validation. We extract the representations from the final convolution layer and pass it through a max-pool layer with a kernel size of $(1, 3)$ and a stride of $(1, 2)$ (Patrick et al. 2021). Similar to action recognition, we perform classification using a one-vs-all linear SVM classifier, we sweep the cost values between $\{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 1\}$ and report the best accuracy.

DCASE. To validate on DCASE, we follow the protocol mentioned in (Morgado, Vasconcelos, and Misra 2021). We extract 60 clips per sample and train a linear classifier on the extracted representations. Note that the augmentation and feature extraction schemes remain the same as mentioned for ESC50. We report the top-1 sample level accuracies by averaging the clip level predictions.

Multi-modal Fusion. To perform a multi-modal linear evaluation with late fusion, we extract features from Kinetics-Sound. During training, we randomly pick 10 audio-visual clips per sample, each 2 seconds long. Next, we extract feature vectors of dimension 2048 from the last convolution layer by using max-pooling with kernel sizes of $(1, 2, 2)$ and $(1, 4)$ for visual and audio respectively. Following, the feature vectors are concatenated to train a linear SVM classifier. Finally, we report the top-1 sample level accuracy for action classification.

E.2 Full Finetuning

Following earlier works (Alwassel et al. 2020; Morgado, Vasconcelos, and Misra 2021; Morgado, Misra, and Vasconcelos 2021; Asano et al. 2020), we use the pretrained visual backbone along with a newly added fully-connected layer for full finetuning on UCF101 (Soomro, Zamir, and Shah 2012) and HMDB51 (Kuehne et al. 2011). We adopt two setups for full finetuning, 8-frame inputs and 32-frame inputs. In both cases, we use a spatial resolution of 224^2. Lastly, we replace the final adaptive average-pooling layer with an adaptive max-pooling layer. We find that applying strong augmentations improves the model performance in full-finetuning. Please see the augmentation details in Section D. During testing, we extract 10 equally spaced clips from each sample and do not apply any augmentations. We report the top-1 accuracy at sample-level prediction by averaging the clip-level predictions. We use an SGD optimizer with a multi-step learning rate scheduler to finetune the model. We present the hyperparameters of full-finetuning in Table S11.

E.3 Retrieval

We follow the protocol laid out in (Patrick et al. 2021; Xu et al. 2019). We uniformly select 10 clips per sample from both training and test splits. We fit 2-second inputs to the backbone to extract representations. We empirically test additional steps such as l2-normalization and applying batch-normalization on the extracted features, and notice that they do not help the performance. Hence, we simply average the features extracted from the test split to query the features of the training split. We compute the cosine distance between the feature vectors of the test clips (query) and the representations of all the training clips (neighbors). We consider a correct prediction if k neighboring clips of a query clip belong to the same class. We calculate accuracies for $k = 1, 5, 20$. We use the NearestNeighbors[3] API provided in SciKit-Learn in this experiment.

F Architecture Details

In this study, we use a slightly modified version of R(2+1)D-18 (Tran et al. 2018) as the video backbone as proposed in (Morgado, Vasconcelos, and Misra 2021), and ResNet-18 (He et al. 2016) as the audio backbone. For the sake of completeness, we present the architecture details in Tables S5 and S6, respectively. The predictor and projector heads are made of fully-connected layers following (Chen and He 2021), and their architecture details are presented in Table S7.

G Hyperparameters and Training Details

In this section, we present the details of the hyperparameters, computation requirements, as well as additional training details of self-supervised pretraining and full finetuning.

G.1 Pretraining Details

We present the pretraining hyperparameters of CrissCross in Table S10. Most of the parameters remain the same across all 3 datasets, with the exception of a few hyperparameters such as learning rates and epoch size which are set depending on the size of the datasets. We train on Kinetics-Sound with a batch size of 512, on a single node with 4 Nvidia RTX-6000 GPUs. Next, when training on Kinetics400 and AudioSet, we use 2 nodes and set the batch size to 2048.

[3]sklearn.neighbors.NearestNeighbors

Layer	X_s	X_t	C	K_s	K_t	S_s	S_t
frames	112	8	3	-	-	-	-
conv1	56	8	64	7	3	2	1
maxpool	28	8	64	3	1	2	1
block2.1.1	28	8	64	3	3	1	1
block2.1.2	28	8	64	3	3	1	1
block2.2.1	28	8	64	3	3	1	1
block2.2.2	28	8	64	3	3	1	1
block3.1.1	14	4	128	3	3	2	2
block3.1.2	14	4	128	3	3	1	1
block3.2.1	14	4	128	3	3	1	1
block3.2.2	14	4	128	3	3	1	1
block4.1.1	7	2	256	3	3	2	2
block4.1.2	7	2	256	3	3	1	1
block4.2.1	7	2	256	3	3	1	1
block4.2.2	7	2	256	3	3	1	1
block5.1.1	4	1	512	3	3	2	2
block5.1.2	4	1	512	3	3	1	1
block5.2.1	4	1	512	3	3	1	1
block5.2.2	4	1	512	3	3	1	1
avg-pool	-	-	512	-	-	-	-

Table S5: Architecture of the video backbone: R(2+1)D-18.

Adam (Kingma and Ba 2015) optimizer is used to train our proposed framework. We use LARC[4](You, Gitman, and Ginsburg 2017) as a wrapper to the Adam optimizer to clip the gradients while pretraining with a batch size of 2048. In this work, we stick to batch sizes of 512 and 2048, because (*i*) as they show stable performance based on the findings of (Chen and He 2021); (*ii*) they fit well with our available GPU setups. Additionally, we perform mixed-precision training (Micikevicius et al. 2018) using PyTorch AMP (Paszke et al. 2019) to reduce the computation overhead.

Ablation Parameters. In the ablation study, we keep the training setup exactly identical across all the variants, with the exception of the learning rates, which we tune to find the best performance for that particular variant. For example, we set the base learning rate for $\mathcal{L}_{v1v2}$ and $\mathcal{L}_{a1a2}$models as 0.0001 and 0.00001 respectively. Next, the predictor learning rates are set to 0.001 and 0.0001 for the $\mathcal{L}_{v1v2}$ and $\mathcal{L}_{a1a2}$ variants.

G.2 Full Finetuning Details

The full fine-tuning hyperparameters for both benchmarks are presented in Table S11. We use a batch size of 32 for the 32-frame input and 64 for the 8-frame input. We use an SGD optimizer with a multi-step learning rate scheduler to finetune the video backbones. Please note that we perform the full finetuning on a single Nvidia RTX-6000 GPU.

H Limitations.

The notion of asynchronous cross-modal optimization has not been explored beyond audio-visual modalities. For example, our model can be expanded to consider more than 2 modalities (e.g., audio, visual, and text), which are yet to be studied. Additionally, we notice a considerable performance gap between full-supervision and self-supervision when both methods are pretrained with the same large-scale dataset (Kinetics400 or AudioSet), showing room for further improvement.

[4]https://github.com/NVIDIA/apex/blob/master/apex/parallel/LARC.py

Layer	X_f	X_t	C	K_s	K_t	S_f	S_t
spectrogram	80	200	1	-	-	-	-
conv1	40	100	64	7	7	2	2
maxpool	20	50	64	3	3	2	2
block2.1.1	20	50	64	3	3	2	2
block2.1.2	20	50	64	3	3	2	2
block2.2.1	20	50	64	3	3	2	2
block2.2.2	20	50	64	3	3	2	2
block3.1.1	10	25	128	3	3	2	2
block3.1.2	10	25	128	3	3	2	2
block3.2.1	10	25	128	3	3	2	2
block3.2.2	10	25	128	3	3	2	2
block4.1.1	5	13	256	3	3	2	2
block4.1.2	5	13	256	3	3	2	2
block4.2.1	5	13	256	3	3	2	2
block4.2.2	5	13	256	3	3	2	2
block5.1.1	3	7	512	3	3	2	2
block5.1.2	3	7	512	3	3	2	2
block5.2.1	3	7	512	3	3	2	2
block5.2.2	3	7	512	3	3	2	2
avg-pool	-	-	512	-	-	-	-

Table S6: Architecture of the audio backbone: ResNet-18.

Layer	Dimensions
input	512
fc-bn-relu	2048
fc-bn-relu	2048
fc-bn	2048

Table S7: Architecture of projector heads.

Layer	Dimensions
input	2048
fc-bn-relu	512
fc	2048

Table S8: Architecture of predictor heads.

I Broader Impact.

Better self-supervised audio-visual learning can be used for detection of harmful contents on the Internet. Additionally, such methods can be used to develop better multimedia systems. Lastly, the notion that relaxed cross-modal temporal synchronicity is useful, can challenge our existing/standard approaches in learning multi-modal representations and result in new directions of inquiry. The authors don't foresee any major negative impacts.

Abbreviations	**Name**	**Description**
bs	batch size	The size of a mini-batch.
es	epoch size	The total number of samples per epoch.
ep	toal epochs	The total number of epochs.
lr $\mathrm{lr}_{\mathrm{ab}}$ $\mathrm{lr}_{\mathrm{vb}}$ $\mathrm{lr}_{\mathrm{ap}}$ $\mathrm{lr}_{\mathrm{vp}}$	learning rate audio backbone lr video backbone lr audio predictor lr video predictor lr	The learning rates to train the networks.
lrs	learning rate scheduler	The learning rate scheduler to train the network.
ms	milestones	At every ms epoch the learning rate is decayed.
γ	lr decay rate	The learning rate is decayed by a factor of γ.
wd	weight decay	The weight decay used in the SGD optimizer.
mtm	momentum	The momentum used in the SGD optimizer.
drp	dropout	The dropout rate.

Table S9: Abbreviations and descriptions of the hyperparameters.

dataset	**bs**	**es**	**ep**	**optim**	**lrs**	$\mathbf{lr_{vb}}$**(start/end)**	$\mathbf{lr_{ab}}$**(start/end)**	$\mathbf{lr_{vp}}$	$\mathbf{lr_{ap}}$	**wd**	**betas**
KS	512	220K	100	Adam	Cosine	0.0002/0	0.0002/0	0.002	0.002	0.0001	0.9, 0.999
K400	2048	1M	100	Adam*	Cosine	0.0002/0.0001	0.0002/0.0001	0.002	0.002	0.0001	0.9, 0.999
AS	2048	3.5M	100	Adam*	Cosine	0.0001/0	0.0001/0	0.001	0.001	0.0001	0.9, 0.999

Table S10: Pretext training parameters. Note the abbreviations used below, KS: Kinetics-Sound, K400: Kinetics400, AS: AudioSet, Adam*: Adam with LARC

dataset	**input**	**es**	**bs**	**ep**	**ms**	**optim**	**lrs**	**lr**	γ	**wd**	**mtm**	**drp**
UCF101	8×224^2	95K	64	20	6/10/14	SGD	multi-step	0.0005	0.3	0.0	0.9	0.0
UCF101	32×224^2	95K	32	20	8/12/16	SGD	multi-step	0.00007	0.3	0.0	0.9	0.0
HMDB51	8×224^2	35K	64	20	6/10/14	SGD	multi-step	0.0005	0.1	0.0	0.9	0.0
HMDB51	32×224^2	35K	32	20	8/12/16	SGD	multi-step	0.0001	0.3	0.0	0.9	0.0

Table S11: Full-finetuning hyperparameters for action recognition when pretrained on Kinetics400.

Acknowledgments We are grateful to the Bank of Montreal and Mitacs for funding this research. We are thankful to SciNet HPC Consortium for helping with the computation resources.

References

Alayrac, J.-B.; Recasens, A.; Schneider, R.; Arandjelovic, R.; Ramapuram, J.; De Fauw, J.; Smaira, L.; Dieleman, S.; and Zisserman, A. 2020. Self-Supervised MultiModal Versatile Networks. *NeurIPS*, 2(6): 7.

Alwassel, H.; Mahajan, D.; Korbar, B.; Torresani, L.; Ghanem, B.; and Tran, D. 2020. Self-Supervised Learning by Cross-Modal Audio-Video Clustering. *NeruIPS*, 33.

Arandjelovic, R.; and Zisserman, A. 2017. Look, listen and learn. In *ICCV*, 609–617.

Asano, Y. M.; Patrick, M.; Rupprecht, C.; and Vedaldi, A. 2020. Labelling unlabelled videos from scratch with multi-modal self-supervision. In *NeurIPS*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *ICML*, 1597–1607.

Chen, X.; and He, K. 2021. Exploring simple siamese representation learning. In *CVPR*, 15750–15758.

Gemmeke, J. F.; Ellis, D. P.; Freedman, D.; Jansen, A.; Lawrence, W.; Moore, R. C.; Plakal, M.; and Ritter, M. 2017. Audio set: An ontology and human-labeled dataset for audio events. In *ICASSP*, 776–780.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Korbar, B.; Tran, D.; and Torresani, L. 2018. Cooperative learning of audio and video models from self-supervised synchronization. In *NeruIPS*, 7774–7785.

Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; and Serre, T. 2011. HMDB: a large video database for human motion recognition. In *ICCV*, 2556–2563.

McFee, B.; Raffel, C.; Liang, D.; Ellis, D. P.; McVicar, M.; Battenberg, E.; and Nieto, O. 2015. librosa: Audio and music signal analysis in python. In *Python in Science Conference*, volume 8, 18–25.

Micikevicius, P.; Narang, S.; Alben, J.; Diamos, G.; Elsen, E.; Garcia, D.; Ginsburg, B.; Houston, M.; Kuchaiev, O.; Venkatesh, G.; et al. 2018. Mixed Precision Training. In *ICLR*.

Morgado, P.; Misra, I.; and Vasconcelos, N. 2021. Robust Audio-Visual Instance Discrimination. In *CVPR*, 12934–12945.

Morgado, P.; Vasconcelos, N.; and Misra, I. 2021. Audio-visual instance discrimination with cross-modal agreement. In *CVPR*, 12475–12486.

Niizumi, D.; Takeuchi, D.; Ohishi, Y.; Harada, N.; and Kashino, K. 2021. BYOL for Audio: Self-Supervised Learning for General-Purpose Audio Representation. *arXiv preprint arXiv:2103.06695*.

Park, D. S.; Chan, W.; Zhang, Y.; Chiu, C.-C.; Zoph, B.; Cubuk, E. D.; and Le, Q. V. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32: 8026–8037.

Patrick, M.; Asano, Y. M.; Kuznetsova, P.; Fong, R.; Henriques, J. F.; Zweig, G.; and Vedaldi, A. 2021. On compositions of transformations in contrastive self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9577–9587.

Piczak, K. J. 2015. ESC: Dataset for Environmental Sound Classification. In *ACM Conference on Multimedia*, 1015–1018. .

Recasens, A.; Luc, P.; Alayrac, J.-B.; Wang, L.; Strub, F.; Tallec, C.; Malinowski, M.; Patraucean, V.; Altché, F.; Valko, M.; et al. 2021. Broaden Your Views for Self-Supervised Video Learning. *arXiv preprint arXiv:2103.16559*.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Stowell, D.; Giannoulis, D.; Benetos, E.; Lagrange, M.; and Plumbley, M. D. 2015. Detection and classification of acoustic scenes and events. *IEEE Transactions on Multimedia*, 17(10): 1733–1746.

Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; and Paluri, M. 2018. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 6450–6459.

Xu, D.; Xiao, J.; Zhao, Z.; Shao, J.; Xie, D.; and Zhuang, Y. 2019. Self-supervised spatiotemporal learning via video clip order prediction. In *CVPR*, 10334–10343.

You, Y.; Gitman, I.; and Ginsburg, B. 2017. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*.