Broaden Your Views for Self-Supervised Video Learning

Adrià Recasens[1] Pauline Luc[1] Jean-Baptiste Alayrac[1] Luyu Wang[1]
Florian Strub[1] Corentin Tallec[1] Mateusz Malinowski[1] Viorica Pătrăucean[1] Florent Altché[1]
Michal Valko[1] Jean-Bastien Grill[1] Aäron van den Oord[1] Andrew Zisserman[1,2]
[1]DeepMind [2]VGG, Dept. of Engineering Science, University of Oxford

Abstract

Most successful self-supervised learning methods are trained to align the representations of two independent views from the data. State-of-the-art methods in video are inspired by image techniques, where these two views are similarly extracted by cropping and augmenting the resulting crop. However, these methods miss a crucial element in the video domain: time. We introduce BraVe, a self-supervised learning framework for video. In BraVe, one of the views has access to a narrow temporal window of the video while the other view has a broad access to the video content. Our models learn to generalise from the narrow view to the general content of the video. Furthermore, BraVe processes the views with different backbones, enabling the use of alternative augmentations or modalities into the broad view such as optical flow, randomly convolved RGB frames, audio or their combinations. We demonstrate that BraVe achieves state-of-the-art results in self-supervised representation learning on standard video and audio classification benchmarks including UCF101, HMDB51, Kinetics, ESC-50 and AudioSet.

1. Introduction

Over the past few years, self-supervised methods have revolutionized the field of representation learning [18, 37, 69]. These methods directly learn from data without the need for manually defined labels that are hard to get at scale. Doing so, one can successfully leverage large amounts of *uncurated* data to improve representations. Even more importantly, self-supervised learning enables richer training tasks to be defined, compared to the standard approach of trying to categorize diverse visual inputs into a fixed set of categories. This has led to self-supervised representations outperforming supervised ones on downstream tasks [34]. Video is a natural domain for self-supervised learning since data is rich and abundant but hard to annotate at scale due to the additional temporal complexity. However, most methods in the video domain take direct inspiration from methods developed for images without fully taking advantage of its distinctly different dimension: time.

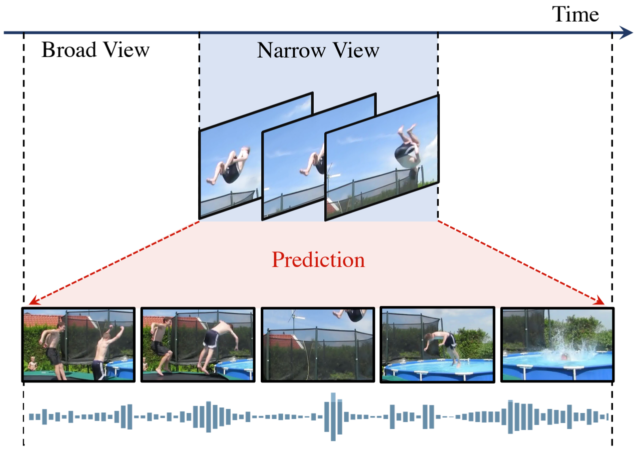

Figure 1. Given a *narrow view* corresponding to a video clip of a few seconds, **BraVe** is tasked with predicting a *broad view* that spans a longer temporal context of the video in different modalities (here visual and audio). Solving that task requires the representation to extrapolate what happened before, during and after the *narrow view*, and results in state-of-the-art video representations.

In particular, one common aspect of self-supervised methods for images is to extract two views from a given instance using the same general augmentation procedure, feed them into a shared backbone, and extract a supervisory signal from the fact that these two views originate from the same source. This is true for most recent approaches irrespective of their underlying learning principle: contrastive approaches [18], clustering-based method [14], or regression algorithms [69]. The same principle has been followed in the video domain [3, 68]. Specifically, most video methods extract the different views from a source video clip in a *symmetric* fashion with respect to time: all extracted views have the same temporal extent in the video [3, 45, 68]. However, doing so does not benefit from learning from information contained at different time scales.

In this paper, we introduce an algorithm dubbed

[†] Correspondence to: Adrià Recasens (arecasens@google.com)

"**Bro**a**d**en your **Views**" (**BraVe**), that breaks this symmetry in order to improve representation learning from videos. In detail, given a *narrow view* corresponding to a video clip of a few seconds, **BraVe** learns a representation by predicting a *broad view* that spans the longer temporal context of the full video clip as illustrated in Figure 1. Solving such a task requires extrapolating to the general context in which a given event occurs. In the example of Figure 1, one has to predict what happened before the person is in the sky (they probably jumped with the help of some device, given the height), as well as what is going to happen next (they will probably fall down somewhere soft) in order to solve the task. This task arguably requires a good understanding of the structure of events and is therefore a promising task for learning representations. While related local-to-global proxy tasks have been studied in the image domain via network architectural designs [8, 38] or multi-size cropping [18], applying these techniques to videos is not straightforward, because of the increased computational complexity incurred by the time dimension and the artifacts introduced when doing similar resize operations in spatio-temporal volumes. To address this challenge, we propose to process broad views with a dedicated model. We demonstrate that under a fixed computational budget, learning from the supervision provided by our broad views performs better than alternatives relying on symmetric augmentation procedures. Our algorithm is simple and does not require a cumbersome creation of explicit negatives as in contrastive methods. Instead we use a direct regression-based approach inspired by BYOL [29], where the views are processed by dedicated backbones and regress each other. Breaking the symmetry enables the use of stronger augmentations and different modalities for the broad view, which improves the quality of the final representations.

Contributions. We make the following contributions: **(i)** We propose a novel framework for representation learning, called **BraVe**, which generates views at different time scales and learns representations via simple regression across views, **(ii)** We explore using different augmentations and modalities in the broad view such as audio, flow or randomly convolved RGB frames. **(iii)** We evaluate this framework in the video domain, both with and without audio as an auxiliary supervisory signal, where we obtain *state-of-the-art* results on video and audio classification benchmarks UCF101, HMDB51, Kinetics, ESC-50 and AudioSet.

2. Related work

Image-based self-supervised learning. Most successful self-supervised methods learn a representation by defining a *pretext task*, whose resolution typically entails learning useful representations [14, 15, 20, 21, 28, 59, 62, 90]. In particular, contrastive methods have provided spectacular performance [9, 18, 22, 34, 37, 40, 47, 56, 79, 80]. Contrastive methods learn by pulling representations of different transformations of the same image (positive instances) closer, and pushing representations of different images (negatives) apart [9, 60]. The main drawbacks of contrastive approaches are that they require a careful choice of positive and negative pairs [80] and that they often rely on large number of such negatives, inducing a high computational cost [18]. Alternatives to the contrastive approach, such as clustering and regression, avoid the need and cost of multiple negatives. Clustering-based methods [4, 7, 10, 14, 15, 39, 78, 86] alternate between learning representations using clusters as targets, and clustering using the current representations (either online or offline). Most related to our work are regression-based methods that instead try to directly regress a representation extracted from a different view of the image [27, 69]. **BraVe** is directly inspired from [29] but the views come from different modalities and augmentations, are processed by dedicated backbones and regress each other.

Video-based self-supervised learning. In the video domain, the pretext tasks for self-supervision have included predicting the future in pixel space by minimising an MSE loss [63, 75, 83] or adversarial losses [53, 82]. However, the predictions of these models are usually blurred and cannot go beyond predicting short clips into the future. To avoid these difficulties, other works focus on learning representations in a more abstract space, by using pretext tasks that predict the temporal order of video frames [57] or the arrow of time [85]. In this direction also, video contrastive methods have been very successful [19, 31, 32, 68]. In addition to data augmentations used for images, these works use temporal cues to build *positive pairs*. Yet the costs of training such systems are significant and complex hard-negative mining strategies are needed to improve the training efficiency [23]. Our method circumvents the use of negatives, considerably alleviating the training complexity while obtaining state-of-the-art performance on popular video benchmarks. Furthermore, our approach may leverage predictive tasks, such as predicting other crops in the video or optical flow, reminiscent of earlier predictive work [75, 84]; but predicting in a learned feature space by building on a more recent self-supervised approach [29].

Audio-video self-supervised learning. Video and audio have been used as a rich source of self-supervision [4, 5, 6, 45, 58, 61, 64, 71]. A simple but effective approach to train representations consists in classifying whether a video clip and an audio sample correspond to each other [5, 6, 45, 61, 71]. Some works propose to use language obtained from speech recognition as an additional supervisory signal [2, 3, 52, 54, 55, 70, 72, 77]. Related to ours, recent work finds that distilling flow and audio into a RGB encoder leads to strong representations [67], using an evolutionary search algorithm on the loss function. In contrast with this approach,

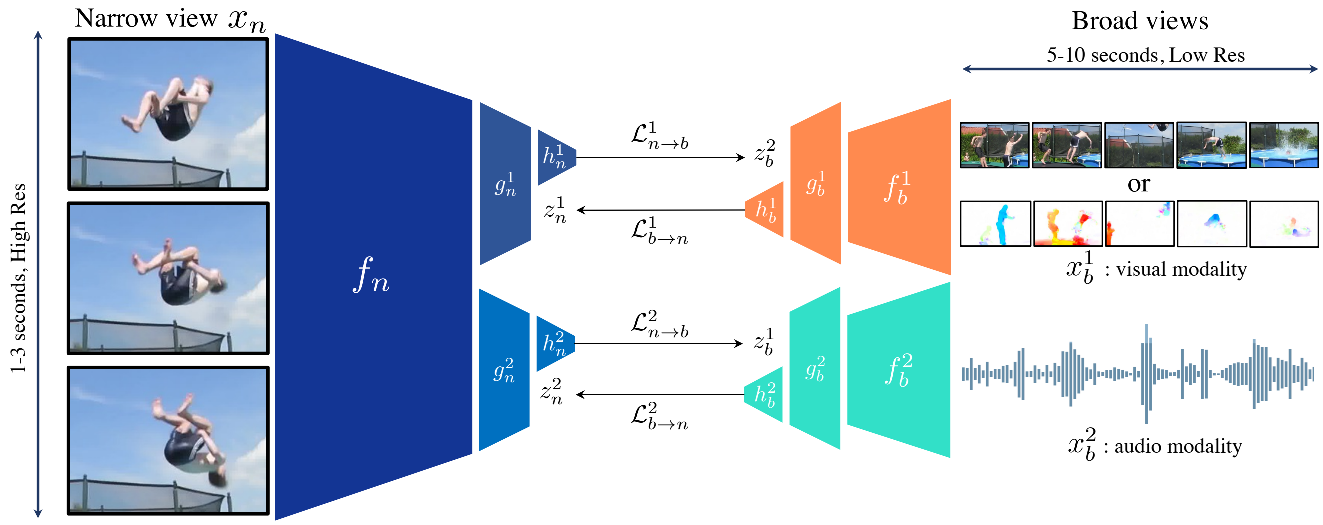

Figure 2. **BraVe**. Given a narrow view x_n spanning a few seconds at high resolution and broad views x_b^1 and x_b^2 covering a larger temporal extent in the video for different modalities, we train independent networks running on the narrow and the broad views to mutually regress each other. This is done by defining two regression losses: $\mathcal{L}_{n\to b}$ to predict a broad view from the narrow view, and $\mathcal{L}_{b\to n}$ enforcing the other way around. To avoid collapse of the learned representations, we introduce three stages of processing as previously done in BYOL [29]: backbone networks (f_n for the narrow view and f_b^1, f_b^2 for the broad views), projector networks (g_n and g_b^1, g_b^2) and predictor networks (h_n and h_b^1, h_b^2). For the broad views, we consider both visual modalities (RGB frames or optical flow) and audio modality.

our framework does not require to define modality-specific losses, is simpler to train (no need to balance the losses), and obtains better performance across the board.

3. Broaden Your Views for Self-Supervised Video Learning

In this section, we detail our approach dubbed **BraVe** for learning self-supervised state-of-the-art representations from a large set of videos, as measured by performance when transferring to downstream tasks. **BraVe**, illustrated in Figure 2, learns by direct regression from a high resolution narrow view that only spans a short clip to a lower resolution broader view which covers a larger temporal context of the video. Multiple options can be considered for the broad view: it can either come from the same modality as the narrow view (RGB in our case) or a different one such as flow or audio. Multiple views can also be combined to further improve performance. Next, we formally describe the learning framework in Section 3.1 and provide intuition why this may be a good self-supervised objective. Then, in Section 3.2, we describe the components and views we use in practice in two standard settings: learning from (i) visual signals alone, and from (ii) visual and audio modalities.

3.1. The BraVe learning framework

General overview. Given a video x that can be composed of multiple modalities, we randomly extract two complementary views: a narrow view x_n that spans a short timeframe in the video (around 1-3 seconds) and a broad view x_b that covers a larger extent of the video (around 5-10 seconds). Details on how these views are obtained are given in Section 3.2. By introducing this temporal asymmetry in the creation of the views, the proposed task consists in extrapolating the full context of the video (the broad view) from only a small portion of the video (the narrow view) as illustrated in Figure 1. We hypothesize that to solve this task, good representations must be learned, which can then be useful for semantic downstream tasks. More formally, we train networks to minimize the training loss $\mathcal{L}$ defined for a given video x as follows:

$$\mathcal{L}(x) = \underbrace{\mathcal{L}_{n\to b}(x)}_{\text{Narrow}\to\text{Broad}} + \underbrace{\mathcal{L}_{b\to n}(x)}_{\text{Broad}\to\text{Narrow}} . \quad (1)$$

This loss is composed of two terms: (i) a prediction loss from the narrow to the broad view, and (ii) a complementary loss to regress the narrow view from the broad view.

BraVe: losses and architectures. For simplicity and computational purposes, we opt for simple regression losses for $\mathcal{L}_{n\to b}$ and $\mathcal{L}_{b\to n}$. This is indeed simpler than standard contrastive losses that require large batches and therefore high compute to work well [18]. One challenge however, is the risk of collapse, since a trivial solution could be to always predict a constant which would lead to perfect regression losses across views. To avoid this, we draw inspiration from recent work [29, 30] in the way we design our networks and losses, as detailed next.

As illustrated in Figure 2, we first define a backbone network f_n whose role is to extract a representation from the narrow view x_n. Similarly, we define a backbone network

f_b acting on the broad view x_b. Note that in our framework, the parameters and even the underlying architectures of f_n and f_b can differ since they act on views of a different nature. These representations are then respectively transformed by projectors g_n and g_b, projecting $f_n(x_n)$ and $f_b(x_b)$ to yield the narrow embedding $z_n = g_n(f_n(x_n))$ and the broad one $z_b = g_b(f_b(x_b))$. Inspired by [29], we then define a third stage of processing called the narrow view predictor h_n that takes the projected embedding from the narrow view z_n and produces a prediction $h_n(z_n)$ that is used to regress the broad view z_b using the following loss:

$$\mathcal{L}_{n \to b}(x) = \left\| \frac{h_n(z_n)}{\|h_n(z_n)\|_2} - \texttt{sg}\left[\frac{z_b}{\|z_b\|_2}\right] \right\|_2^2, \qquad (2)$$

where $\texttt{sg}[\cdot]$ denotes the "stop gradient" operator, which operates on its input as the identity, but has zero partial derivatives. Since the loss $\mathcal{L}_{n \to b}$ only depends on the networks associated with the narrow view, we also define a loss to provide training signal for the broad view network. To that end, we introduce a broad view predictor h_b that takes the projected embedding from the broad view z_b and produces a prediction $h_b(z_b)$ that is used to regress the narrow view embedding z_n using the following loss:

$$\mathcal{L}_{b \to n}(x) = \left\| \frac{h_b(z_b)}{\|h_b(z_b)\|_2} - \texttt{sg}\left[\frac{z_n}{\|z_n\|_2}\right] \right\|_2^2. \qquad (3)$$

The role of these predictors is crucial to avoid collapse as found in [29], which we confirm experimentally. The same is true for the stop gradient operator. Differently from [29], we do not use exponential moving averages (EMA) on the weights of the network that process the view being regressed. Unlike [29], who required the moving average for improved performance, we find that this is not necessary in our case.

Intuitions about what needs to be learned by `BraVe`. While the proposed approach avoids plain collapse of the representations, it is also important to question what needs to be learned in order for the loss (1) to be optimized. In particular, we want the narrow backbone to learn to predict the full context represented by the broad view. However, one challenge is to prevent the broad backbone from instead simply learning to throw the broad information away and only keeping the signal contained in the narrow view. To avoid this, we sample the narrow and broad views independently in time when they come from the same visual modality so that it is difficult for the broad backbone to predict what the narrow view is going to be. By doing so, we argue that the best solution to solve the task is for the narrow backbone to extrapolate what is happening in the broad view. We empirically verify the importance of this independent sampling in our experiments in section 4.

Dealing with multiple views and modalities. `BraVe` can be extended to handle K broad views (with $K > 1$) coming from different modalities. To do so and as illustrated in Figure 2, we keep a single narrow backbone network f_n but introduce specific narrow projectors and predictors for each broad views: $\{(g_n^1, h_n^1), \cdots, (g_n^K, h_n^K)\}$. Each additional broad view x_b^k has its own set of backbone, projector and predictor : f_b^k, g_b^k and h_b^k, respectively. Given this, all regression losses are simply aggregated over all pairs composed by the narrow view x_n and the different broad views $\{x_b^k\}_k$:

$$\mathcal{L}(x) = \sum_{k=1}^{K} \mathcal{L}_{n \to b}^k(x) + \mathcal{L}_{b \to n}^k(x). \qquad (4)$$

When using different modalities, the risk for the broad network to only focus on the narrow view is reduced due to the modality gap between the two views. Furthermore, when using audio, syncing helps slightly as previously observed in visual-audio work [45]. We verify this experimentally and report the results in Appendix D.

Final loss. Given a large set of videos $\{x^i\}_{i=1}^N$, we train our model to minimize:

$$\min_{\substack{f_n, g_n, h_n \\ f_b, g_b, h_b}} \sum_{i=1}^{N} \mathcal{L}(x^i). \qquad (5)$$

Next, we provide more details on the specific components that are used when `BraVe` is applied in the unimodal setting and the multimodal setting; as well as how the narrow and broad views are constructed in each case.

3.2. Broad views from visual and audio modalities

In our framework, we regress the representation of a broad backbone which sees a larger context of the video. The broad view is meant to provide information about the full video clip including more temporal context, in order to supervise the narrow backbone f_n. As the different views are processed by different backbones, we can apply a different set of preprocessing and augmentation functions to any of the views. In this section, we first describe the set of transformations that we use when training with visual inputs alone, and then when training with both visual and audio inputs.

Visual modalities. When sampling the broad view from the visual modalities, we aim to cover a large temporal context, the full clip. Accessing more temporal context typically means increasing the number of frames, and thus introducing extra computational complexity. To avoid this overhead, we decrease the spatial resolution of the broad view in order to keep the number of pixels constant. In Section 4 we show the effectiveness of trading temporal context for spatial resolution in the broad view. By keeping the computational cost fixed, we ensure that our method is computationally competitive with alternative self-supervised approaches.

Additionally to the temporal sampling, the set of transformations we consider for use on the narrow and broad views are motivated from two complementary perspectives. First, we can design the transformations $\mathcal{T}_b$ used for the broad view to extract specific features from the input modality, sought to enrich the learned representations $f_n(x_n)$ with a certain type of information. Second, similarly to the use of augmentations in a wide number of machine learning approaches, and in particular in contrastive and regression-based self-supervised learning approaches, we also employ such stochastic transformations to enforce invariance or equivariance constraints on the learned representations. In contrast to the use of augmentations in these self-supervised frameworks however, we emphasize that we do not impose that the set of transformations $\mathcal{T}_n$ used on the narrow view be the same as the set of transformations $\mathcal{T}_b$ used on the broad views. To explore this, we employ a recently introduced augmentation procedure relying on random convolutions [88], by which we augment only the broad view.

Alternatively, we can use optical flow as substitute of RGB in the broad view, which is reminiscent of [76], where the flow network is used to teach the RGB network. Optical flow from sequential images can provide supervision to emphasize motion in the learned representations extracted from the source, which has shown to be important for predicting actions [32, 73, 84]. Optical flow can be extracted using an off-the-shelf unsupervised flow extraction algorithm. As flow is computed once for the full dataset, its computational overhead is negligible compared to training time.

Audio modalities. Our framework can leverage audio as supervisory signal in the broad view. We can either use a single audio broad view or combine a visual broad view and an audio broad view for stronger self-supervision. Audio is a strong supervisory signal, and has been extensively used for self-supervision in videos as it strongly correlates with the visual content, while being easier to process computationally. As pre-processing, we extract spectrograms from consecutive short-time windows on the waveform using Fourier transforms. This approach has been shown to be very effective in obtaining state-of-the-art performance on supervised [24, 44] and unsupervised [3, 41, 42] approaches. For this reason, we encode the audio using a log-mel spectrogram representation as $x_b \in \mathbb{R}^{T_s \times D}$ where T_s is the number of spectrogram frames and D denotes the number of features. Similar to the unimodal setting, we experiment with enlarging the temporal window for the extraction of the audio view, compared with the temporal window of the narrow video view, seeking to increase the amount of context information present in the supervisory signal. Finally, as explained in the previous section, we make sure that the visual narrow view and the audio broad view are in sync at their starting point.

4. Experiments

In this section, we evaluate **`BraVe`** and compare its performance against relevant state-of-the-art methods trained on similar data and modalities.

4.1. Experimental setting

Video-only experiments. In the video-only setting, we conduct our experiments on the Kinetics-600 dataset [16]. The dataset has 600 action classes and contains 447k videos at the time of submission, 362k in the train set.

Audio-video experiments. In the crossmodal training setting, we use the AudioSet [25] as pre-training dataset. The dataset has 527 action classes and contains 1.9M videos in the training set at the time of submission.

Architectures. For spatiotemporal volumes such as the sequences of RGB or flow frames, unless specified otherwise, we use the TSM-ResNet50 (TSM-50) [49] architecture for the narrow backbone. For the broad visual backbone we always use a TSM-50 backbone. Video inputs are sampled at 12.5 frames per second (FPS). Unless stated otherwise, we train the narrow backbone on inputs of 16 frames (1.3 seconds) at resolution 224×224, and the broad backbone on inputs of 64 frames at 6.25 FPS (10s) at resolution 112×112. To see how our method scales to different and bigger architectures, we also experiment with different backbones for the narrow network with the R(2+1)D architecture [81], R3D architecture [33] and TSM with twice the number of channels in each layer (TSM-50x2). We also introduce a video variant of the recent NF-Net-F0 architecture [13], by applying the TSM on it (details in Appendix C), which we call TSM-NF-F0. We use these networks only for the narrow view and always use TSM-50 in the broad view. For the broad backbone processing log-mel spectrograms, we use ResNet-50 [36]. All models are trained using a two-layer MLP for the projector heads (g_n and g_b) with a hidden layer of dimension 512, and a three-layer MLP for the predictor heads (h_n and h_b) with hidden layers of dimensions 4096. We use batch normalization after each hidden layer. We use 128 as the output dimension of projectors and predictors.

Feature extraction. For flow extraction, we use the TV-L1 [89] algorithm. We use 80 bins for extracting log-mel spectrograms.

Augmentations. We sample and augment all the visual views independently. For any narrow view, we uniformly sample a temporal offset between 0 and $T - \tau_n$, where T is the duration of the video clip and τ_n denotes the length of the narrow view. We extract the view starting at this offset. For the broad view, we randomly sample the offset between 0 and T. We pad any broad view of insufficient length with a clip extracted from the start of the video sample (*i.e.* looping over the sequence). For all visual modalities (including the flow), we use random cropping and horizontal flipping. For the RGB views, we additionally employ gaussian blurring as well as scale and color jittering. We also explore the use of random convolutions as an augmentation procedure. Following [48], we use He initialization [35] for the weights and fixed zero bias, sampling the size of the kernel uniformly across odd values ranging from 1 to 11. For audio, we use the same starting point as the narrow view, but extend it for a longer time window. If necessary, similarly to the RGB case, we pad the broad audio view with audio extracted from the start of the audio clip. See Appendix A.2 for further details.

Self-supervised training details. We discard labels at training time, and only use them for downstream evaluation. Unless stated otherwise, we employ a batch size of 512 and train for 300k steps, setting the initial learning rate to 0.002. We train all models using AdamW [50], with 5000 warm up steps and cosine learning rate schedule [51]. Following BYOL [29], we multiply the learning rate for all predictors (h_n and h_b) by 10. For batch norm layers, we use a decay rate of 0.9 and epsilon of 1e-5. We use weight-decay of 0.01. More details are given in Appendix A.1.

4.2. Downstream tasks

We use two standard settings to evaluate the quality of the learned visual representations from the narrow backbone f_n: in the *linear* setting, we train a linear layer over frozen features extracted by f_n; in the *fine-tuning* setting, we train f_n and the classifier head end-to-end. During evaluation, we always use 32 frames as inputs at 12.5 FPS, irrespective of the pre-training regime, to be comparable to previous work. We evaluate video representations using the HMDB51 dataset [46], the UCF101 dataset [74] and the Kinetics-600 [17] validation set. The HMBD51 dataset contains 5K videos, corresponding to 51 classes. The UCF101 dataset contains 13K videos, corresponding to 101 classes. The Kinetics-600 validation set contains 28k videos. We also evaluate the learned audio representations from the corresponding broad backbone, f_b, on both the test set of the AudioSet dataset (20K samples, 527 classes) as well as the smaller ESC-50 dataset [66] (2K samples, 50 classes). Following standard procedure, we report top-1 accuracy for all datasets except for Audioset where we report the mean average precision [41]. For the datasets that have official splits (3 for UCF101/HMDB51 and 5 for ESC-50), we follow the standard procedure where split#1 serves as the validation set and the average accuracy over all splits is then reported.

Linear setting. For HMDB51, UCF101 and ESC-50, we extract representations from 10 epochs worth of augmented samples using the learned narrow backbone, and we train a linear SVM using scikit-learn [65] on these frozen features. For Kinetics-600 and AudioSet which are larger, we instead train the linear classifier using the Adam optimizer [43]. In all cases, we use the same augmentations as during unsupervised pre-training except for gaussian blur. Full details are provided in Appendix B. At test time, we average the prediction over 30 clips (10 temporal clips each with 3 spatial crops) as done in [68]. For AudioSet, we follow [41] and use a fully-connected classifier, with one hidden layer of 512 units, in place of the linear classifier.

Fine-tuning setting. In this setting, we add a single, randomly initialized, linear layer at the output of the narrow backbone. We initialize the narrow backbone's weights with those learned using **`BraVe`**, and we fine-tune this architecture end-to-end. Following previous work, we perform this evaluation on the HMDB51 and UCF101 datasets. We use the same test time procedure as for the linear setting. Details are given in Appendix B.

Table 1. **Importance of the broad view.** We evaluate the impact of the temporal extent of the narrow (τ_n) and broad (τ_b) views. M_b is the modality used in the broad view. RC stands for random convolutions. K600 stands for Kinetics-600 and AS for AudioSet.

Dataset	M_b	τ_n	τ_b	HMDB51	UCF101
K600	RGB+RC	10s	10s	56.7	78.3
K600	RGB+RC	1.3s	1.3s	57.4	87.6
K600	RGB+RC	1.3s	5s	61.7	89.0
K600	RGB+RC	1.3s	10s	**63.3**	**89.5**
AS	Audio	1.3s	1.3s	67.6	92.0
AS	Audio	1.3s	5s	**68.1**	**92.4**
AS	Audio	1.3s	10s	67.1	**92.4**

Table 2. **Visual transformation for the broad view.** We compare various augmentations for the visual input of the broad view, when pre-training on Kinetics-600. We use $\tau_n = 1.3s$ (narrow extent) and $\tau_b = 10s$ (broad extent). RC stands for random convolutions.

M_b	HMDB51	UCF101
RGB	59.6	87.8
RGB+RC	63.3	89.5
Flow	**65.9**	**91.6**

4.3. Ablation study

In this section, we study the effect of the different components of **`BraVe`** on the performance of the narrow backbone f_n. Specifically, we study four main elements: **(i)** the effect of the temporal extents of the narrow and broad views, **(ii)** the improvements brought by different choices of transformations for the visual modality, **(iii)** the importance of having separate weights for the narrow view and the broad view network components and **(iv)** the effect of temporally syncing the narrow view and the broad view. By default, we conduct this analysis using the HMDB51 and UCF101 benchmarks in the linear setting.

Importance of the broad view. We study the effect of the temporal extent of the narrow and broad views in the RGB-only setting (using random convolutions RGB+RC for the broad view) and the multimodal setting (using audio spectrogram for the broad view). We report results in Table 1. First, in the unimodal setting, we find that for a narrow view extent τ_n of $1.3s$, performance improves significantly across the two downstream tasks as we increase the duration of the broad view τ_b from $1.3s$ to $10s$, (*e.g*. from 57.4 to 63.3

Table 3. **Weight sharing.** We explore the effect of sharing weights across different components of the models. Models are trained on the Kinetics-600 dataset using RGB visual input in the broad view.

Separate Backbone	Separate Projector	Separate Predictor	HMDB51	UCF101
✓	✓	✓	**59.6**	**87.8**
✗	✓	✓	56.4	86.5
✗	✗	✓	51.4	82.5
✗	✗	✗	51.8	83.0

Table 4. **Sync study.** Effect of syncing the narrow and broad views.

Dataset	Sync	M_b	HMDB51	UCF101	K600
K600	✗	RGB+RC	63.3	**89.5**	**66.9**
K600	✓	RGB+RC	**65.0**	86.8	60.0

on HMDB51). This empirically supports our intuition that broader views can provide better supervision. Second, we find that using temporally large views of $10s$ for both the narrow view and the broad view degrades performance, as the task becomes significantly easier and we are unlikely to get rich embeddings. In the multimodal setting, we find that increasing the context from $1.3s$ to $5s$ also brings an improvement, although it is smaller than in the visual setting. We do not see further improvements when extending the broad view to $10s$, and hence choose $5s$ for the temporal extent of the audio broad view.

Visual transformation for the broad view. In Table 2, we investigate the effect of using different visual inputs in the broad view. First, we see that using Random Convolutions (RC) [88] on the RGB frames significantly improves performance, compared to using standard RGB frames. `BraVe` enables the use of such an aggressive augmentation since it has a dedicated backbone for that view. Moreover, only using this augmentation on the broad view ensures that the backbone trained on the narrow view does not suffer from shift in distribution of intensities [88]. Furthermore, using optical flow for the broad view leads to further improvement when compared to using RC augmentation. This demonstrates a surprisingly high effectiveness of leveraging hand-designed feature extraction process, probably because this allows important factors – here motion and segmentation information – to be included in the desired representation.

Weight sharing. In Table 3, we study the effect of sharing weights across the different components of our model. First, we observe a significant decrease in performance when sharing the backbone networks. In this case, to solve the task we propose, the single backbone may need to split its capacity to extract features useful for both prediction tasks, from the narrow to the broad, and vice versa; which visibly hurts performance on the downstream task. While we could increase the capacity of the shared backbone, this would not provide the flexibility of separate backbones for processing different broad modalities and views. Next, we see an even larger drop, when additionally sharing the projector. Finally, when sharing all components, the important performance gap overall compared with our approach confirms our intuition that integrating information from local and global temporal context by only doing data augmentation as in the image case [18] is not a good strategy for videos, and further highlights the benefit of our proposed approach.

Syncing views. In Table 4, we study the effect of having the same temporal starting point for the narrow and the broad view. As expected, when using a broad visual modality, syncing significantly decreases performance in UCF101 (-2.7%) and Kinetics-600 (-6.9%) but slightly benefits HMDB51 ($+1.7\%$). We hypothesise that when both views are in sync, the broad network can simply focus its prediction only on the narrow view since the relative position of the views is deterministic hence making the self-supervised task easier as explained in the intuition paragraph of Section 3.1. As such, the network specialises in predicting short clips which would explain the slight improvement on the short clips of HMDB51 and the important decrease in performance for Kinetics and UCF101 that have longer clips.

4.4. Comparison with the state-of-the-art

We compare `BraVe` against the state-of-the-art for self-supervised video representation learning in Table 5. Note that when evaluating in visual tasks, we only use the RGB modality to be comparable to previous work.

Visual only on Kinetics600. In the setting where we use only the video modality combined with random convolutions in the broad view, we find that our TSM-50 model outperforms the current state-of-the-art CVRL approach [68] on UCF101 finetuning, and on HMDB51 linear and finetuning, despite having less parameters in our network (23.5M versus 33.3M). When integrating the flow modality we further increase the performance on UCF101 and HMDB51 to set a new state-of-the-art when training only from Kinetics-600 from the visual modality alone. On the Kinetics-600 linear evaluation, we obtain lower performance (66.9 versus 70.4) that we hypothesize is due to the advantage of contrastive-based approaches for such in-domain tasks. We also compare to using the same backbone (R3D50) as CVRL but observe slightly worse performance that we hypothesize to be due to our setting being more adapted to TSM-50.

Multimodal on AudioSet. We also compare our approach in the multimodal (visual and audio modalities) setting by training `BraVe` on AudioSet. In that setting, we train for 620k steps instead of 300k, as AudioSet is significantly larger than Kinetics-600. We also increase the number of input frames of the narrow network from 16 to 32 frames (at 12.5FPS) and the number of input frames of the broad network from 64 (at 6.25FPS) to 128 (at 12.5FPS) during

Table 5. **Comparison of learnt representations against the state-of-the-art.** We report the performance in the linear and fine-tuning (FT) settings, on three vision benchmarks: UCF101, HMDB51, Kinetics-600 (K600); as well as on two audio benchmarks: ESC-50 and AudioSet (AS). K400 is Kinetics-400, YT8M is Youtube-8M [1], IG65M is Instagram-65M [26]. We specify dataset sizes in years. We denote the modalities $\mathcal{M}$ used for training by: V for RGB, F for flow and A for audio. **All models use only RGB for the visual downstream tasks.**

					UCF101		HMDB51		K600	ESC-50	AS
Method	Backbone (#params)	Dataset	Years	$\mathcal{M}$	Linear	FT	Linear	FT	Linear	Linear	MLP
MEM-DPC [31]	R-2D3D (32.6M)	K400	0.07	VF		78.1		41.2		/	/
VDIM [19]	custom (17.3M)	K600	0.1	V		79.7		49.2		/	/
CoCLR [32]	S3D (9.1M)	K400	0.07	VF	74.5	87.9	46.1	54.6		/	/
CVRL [68]	R3D50 (33.3M)	K600	0.1	V	90.6	93.4	59.7	68.0	**70.4**	/	/
BraVe:V↔V (ours)	TSM-50 (23.5M)	K600	0.1	V	89.5	93.5	63.3	**70.9**	66.9	/	/
BraVe:V↔F (ours)	TSM-50 (23.5M)	K600	0.1	VF	**91.6**	**93.8**	**65.9**	69.7	66.3	/	/
BraVe:V↔V (ours)	R3D50 (33.3M)	K600	0.1	V	88.8	92.6	61.8	69.2	66.6	/	/
BraVe:V↔F (ours)	R3D50 (33.3M)	K600	0.1	VF	91.1	93.4	65.6	70.6	65.6	/	/
AVTS [45]	MC3 (11.7M)	AS	1	VA		89.0		61.6		80.6	
ELo [67]	R(2+1)D-50 (46.9M)	YT8M	13	VFA		93.8	64.5	67.4			
AVID [58]	R(2+1)D-50 (46.9M)	AS	1	VA		91.5		64.7		89.2	
GDT [64]	R(2+1)D-18 (33.3M)	AS	1	VA		92.5		66.1		88.5	
MMV [3]	R(2+1)D-18 (33.3M)	AS	1	VA	83.9	91.5	60.0	70.1	55.5	85.6	29.7
XDC [4]	R(2+1)D-18 (33.3M)	AS	1	VA		93.0		63.7		84.8	
XDC [4]	R(2+1)D-18 (33.3M)	IG65M	21	VA		95.5		68.9		85.4	
BraVe:V↔A (ours)	R(2+1)D-18 (33.3M)	AS	1	VA	89.9	94.1	64.8	71.1	63.6	90.4	34.7
BraVe:V↔A (ours)	TSM-50 (23.5M)	AS	1	VA	93.0	94.8	69.4	72.6	70.1	90.5	34.4
BraVe:V↔FA (ours)	TSM-50 (23.5M)	AS	1	VFA	93.1	95.4	70.0	74.6	69.3	90.1	34.5
BraVe:V↔FA (ours)	R(2+1)D-50 (46.9M)	AS	1	VFA	92.5	95.1	68.3	73.6	69.4	**91.6**	34.5
BraVe:V↔FA (ours)	TSM-NF-F0 (71.5M)	AS	1	VFA	**94.1**	**95.8**	**71.4**	73.1	**72.6**	90.2	34.5
BraVe:V↔FA (ours)	TSM-50x2 (93.9M)	AS	1	VFA	93.1	95.7	70.5	**77.8**	71.4	91.1	**34.8**
Supervised [11, 44, 67, 87]						96.8	71.5	75.9	82.4	94.7	43.9

pretraining. We use $\tau_b = 5s$ for the audio broad view. We make four important observations. **(i)** Under this setting, **BraVe** outperforms all state-of-the-art methods when using the same pretraining data and same backbones. In particular, when using R(2+1)D-18 we outperform the current state-of-the-art XDC [4] on UCF101 (93.0 vs 94.1) and MMV [3] on HMDB51 (71.1 vs 70.1). **(ii)** Interestingly, we observe that **BraVe** benefits from using two broad views coming from two different modalities, audio and flow (going from 94.8 to 95.4 on UCF101 and 72.6 to 74.6 on HMDB51 finetuning regime). **(iii) BraVe** benefits from using larger visual backbones. Our TSM-NF-F0 (71.5M parameters) sets a new state-of-the-art on UCF101 finetuning (95.8) even beating the best XDC model that is using 21 times more data. The performance of this model is particularly striking in the linear setting with 94.1 for UCF101 and 71.4 on HMDB51 which is on par with the best previous finetuned results. This is an important practical achievement as it enables the use of our models off-the-shelf, without the need for fine-tuning. Our TSM-50x2 model (93.9M parameters) is the best overall, setting a new state-of-the-art on HMDB51 finetuning with 77.8 even outperforming the best supervised results published to date (75.9 from [87]). **(iv)** When evaluating the performance of the broad audio network we also significantly outperform previous state-of-the-art on two challenging benchmarks, ESC-50 and Audioset. Notably, we significantly improve the performance in AudioSet, the hardest of the audio tasks.

5. Conclusion

In this paper, we introduced **BraVe**, a self-supervised learning framework for video. Our method efficiently learns its representation by supervising a temporally narrow view with a general broad view, which can be either computed from RGB, flow or audio. Our model achieves state-of-the-art performance when trained on datasets such as Kinetics or AudioSet. Notably, when trained with larger backbones, **BraVe** outperforms the previous best supervised transfer result on the challenging HMDB51 benchmark.

Acknowledgments

The authors would like to thank Antoine Miech, Bilal Piot, Evan Shelhamer and Sander Dieleman for fruitful discussions as well as Andy Brock for his help with the NFNet implementation.

References

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 8

[2] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Ivan Laptev, Josef Sivic, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *CVPR*, 2016. 2

[3] Jean-Baptiste Alayrac, Adrià Recasens, Rosalia Schneider, Relja Arandjelović, Jason Ramapuram, Jeffrey De Fauw, Lucas Smaira, Sander Dieleman, and Andrew Zisserman. Self-supervised multimodal versatile networks. In *NeurIPS*, 2020. 1, 2, 5, 8, 13

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

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

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

[7] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020. 2

[8] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Neural Information Processing Systems*, 2019. 2

[9] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *NeurIPS*, 2019. 2

[10] Miguel A. Bautista, Artsiom Sanakoyeu, Ekaterina Sutter, and Björn Ommer. Cliquecnn: Deep unsupervised exemplar learning. In *NeurIPS*, 2016. 2

[11] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021. 8

[12] Andrew Brock, Soham De, and Samuel L. Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. In *ICLR*, 2021. 13

[13] Andrew Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021. 5, 13

[14] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018. 1, 2

[15] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020. 2

[16] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018. 5

[17] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the Kinetics dataset. In *CVPR*, 2017. 6

[18] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 1, 2, 3, 7

[19] R Devon et al. Representation learning with video deep infomax. *arXiv preprint arXiv:2007.13278*, 2020. 2, 8

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

[21] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *ICCV*, 2017. 2

[22] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS*, 2014. 2

[23] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. Vse++: Improving visual-semantic embeddings with hard negatives. In *BMVC*, 2017. 2

[24] Logan Ford, Hao Tang, François Grondin, and James R Glass. A deep residual network for large-scale acoustic scene analysis. In *InterSpeech*, 2019. 5

[25] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *ICASSP*, 2017. 5

[26] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *CVPR*, 2019. 8

[27] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Learning representations by predicting bags of visual words. In *CVPR*, 2020. 2

[28] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 2

[29] 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, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020. 2, 3, 4, 6, 12

[30] Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning*, pages 3875–3886. PMLR, 2020. 3

[31] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. In *ECCV*, 2020. 2, 8

[32] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. In *NeurIPS*, 2020. 2, 5, 8

[33] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can

spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018. 5

[34] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1, 2

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 5, 12

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 5

[37] Olivier J Hénaff, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. In *ICML*, 2020. 1, 2

[38] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018. 2

[39] Jiabo Huang, Qi Dong, Shaogang Gong, and Xiatian Zhu. Unsupervised deep learning by neighbourhood discovery. In *ICML*, 2019. 2

[40] Rishabh Jain, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 2

[41] Aren Jansen, Daniel PW Ellis, Shawn Hershey, R Channing Moore, Manoj Plakal, Ashok C Popat, and Rif A Saurous. Coincidence, categorization, and consolidation: Learning to recognize sounds with minimal supervision. In *ICASSP*, 2020. 5, 6, 13

[42] Aren Jansen, Manoj Plakal, Ratheet Pandya, Daniel PW Ellis, Shawn Hershey, Jiayang Liu, R Channing Moore, and Rif A Saurous. Unsupervised learning of semantic audio representations. In *ICASSP*, 2018. 5, 13

[43] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[44] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. In *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020. 5, 8

[45] Bruno Korbar, Du Tran, and Lorenzo Torresani. Cooperative learning of audio and video models from self-supervised synchronization. In *NeurIPS*, 2018. 1, 2, 4, 8, 14

[46] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011. 6

[47] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 2020. 2

[48] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *ICLR*, 2020. 5

[49] Ji Lin, Chuang Gan, and Song Han. TSM: Temporal shift module for efficient video understanding. In *ICCV*, 2019. 5, 13

[50] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6, 12

[51] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 6

[52] Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, and Kevin Murphy. What's cookin'? Interpreting cooking videos using text, speech and vision. *NAACL*, 2015. 2

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

[54] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-End Learning of Visual Representations from Uncurated Instructional Videos. In *CVPR*, 2020. 2

[55] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100M: Learning a text-video embedding by watching hundred million narrated video clips. In *ICCV*, 2019. 2

[56] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020. 2

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

[58] Pedro Morgado, Nuno Vasconcelos, and Ishan Misra. Audio-visual instance discrimination with cross-modal agreement. *arXiv preprint arXiv:2004.12943*, 2020. 2, 8

[59] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 2

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

[61] Andrew Owens and Alexei A Efros. Audio-visual scene analysis with self-supervised multisensory features. In *ECCV*, 2018. 2

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

[63] Viorica Pătrăucean, Ankur Handa, and Roberto Cipolla. Spatio-temporal video autoencoder with differentiable memory. In *ICLR (Workshop)*, 2016. 2

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

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 6, 12

[66] Karol J Piczak. ESC: Dataset for environmental sound classification. In *ACM Multimedia*, 2015. 6

[67] AJ Piergiovanni, Anelia Angelova, and Michael S Ryoo. Evolving losses for unsupervised video representation learning. In *CVPR*, 2020. 2, 8

[68] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. *arXiv preprint arXiv:2008.03800*, 2020. 1, 2, 6, 7, 8

[69] Pierre H Richemond, Jean-Bastien Grill, Florent Altché,

Corentin Tallec, Florian Strub, Andrew Brock, Samuel Smith, Soham De, Razvan Pascanu, Bilal Piot, et al. Byol works even without batch statistics. In *NeurIPS (SSL Workshop)*, 2020. 1, 2

[70] Ozan Sener, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. Unsupervised semantic parsing of video collections. In *ICCV*, 2015. 2

[71] Arda Senocak, Tae-Hyun Oh, Junsik Kim, Ming-Hsuan Yang, and In So Kweon. Learning to localize sound source in visual scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[72] Gunnar A Sigurdsson, Jean-Baptiste Alayrac, Aida Nematzadeh, Lucas Smaira, Mateusz Malinowski, João Carreira, Phil Blunsom, and Andrew Zisserman. Visual grounding in video for unsupervised word translation. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[73] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *ICLR*, 2014. 5

[74] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 6

[75] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015. 2

[76] Jonathan Stroud, David Ross, Chen Sun, Jia Deng, and Rahul Sukthankar. D3d: Distilled 3d networks for video action recognition. In *WACV*, 2020. 5

[77] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. VideoBERT: A joint model for video and language representation learning. In *ICCV*, 2019. 2

[78] Kai Tian, Shuigeng Zhou, and Jihong Guan. Deepcluster: A general clustering framework based on deep learning. In *ECML/PKDD*, 2017. 2

[79] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 2

[80] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. In *NeurIPS*, 2020. 2

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

[82] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *NIPS*, 2016. 2

[83] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *ECCV*, 2018. 2

[84] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *ECCV*, 2016. 2, 5

[85] Donglai Wei, Joseph J. Lim, Andrew Zisserman, and William T. Freeman. Learning and using the arrow of time. In *CVPR*, 2018. 2

[86] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016. 2

[87] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, 2018. 8

[88] Zhenlin Xu, Deyi Liu, Junlin Yang, and Marc Niethammer. Robust and generalizable visual representation learning via random convolutions. *arXiv preprint arXiv:2007.13003*, 2020. 5, 7, 12

[89] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint pattern recognition symposium*, 2007. 5

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

Appendix

In this appendix, we provide additional details useful for reproduction of the results. In Section A we present the details of our training pipeline, including architecture and hyperparameter details (A.1), data augmentation and feature extraction (A.2). In Section B we detail the linear and fine-tuning evaluation procedures. Section C describes in more detail the TSM-NF-F0 architecture used in the paper. Section D evaluates the importance of syncing video and audio.

A. Pre-training details

A.1. Architecture and model hyperparameters

Each predictor is a three-layer MLP with hidden dimensions of 4096; each projector is a two-layer MLP with hidden dimension of 512. To train our models, we use the AdamW optimizer [50] with cosine decay on the learning rate, with 5000 steps of linear warm up (starting from 0.0 to the initial learning rate value). All the models are trained with initial learning rate $2 \cdot 10^{-3}$ and batch size 512. We use weight decay with value 0.01. Following [29], we multiply the learning rate of the predictor MLPs by 10. We train all models for 300k steps except for the models with audio reported in Table 5, which are trained for 620k steps. We use 16 Cloud TPUs to train all models except for the TSM-50x2 and the R(2+1)D-50 which we train with 32 Cloud TPUs.

A.2. Data augmentation and feature extraction

RGB: Unless stated othwerwise, we subsample training videos to 12.5 FPS. For the broad views of the visual only models and the narrow view of the ablation using a 10s narrow view (first row, Table 1), we subsample training videos to 6.25 FPS. In terms of spatial augmentations, we use random cropping, random flipping, color jittering, scale jittering and gaussian blurring; sampling their parameters independently for each view. Given the original frame, cropping is performed by sampling a bounding box with aspect ratio ranging between $\frac{1}{2}$ and 2.0 and area between 30% and 100% of the full image. This bounding box is used to crop all frames of the video consistently in time. We horizontally flip all the frames with probability 0.5. With probability of 0.8 we apply color randomization in brightness, saturation, contrast and hue. This is done by adjusting brightness and hue by an additive offset, each uniformly sampled in respectively $[-32/255, 32/255]$ and $[-0.2, 0.2]$ on a per-sample basis; and similarly, adjusting contrast and saturation by a multiplicative factor, each sampled in $[0.6, 1.4]$. After this preprocessing, we clip the pixel values in the range $[0, 1.0]$. Furthermore, with probability 0.2, we convert the RGB sequence to a grayscale sequence. Finally, we apply gaussian blur with standard deviation σ uniformly sampled in $[0.1, 2.0]$ and with kernel size equal to $\frac{1}{10}$th of the crop side.

Flow: Temporal sampling of the flow is performed similarly to the RGB case for the broad view. In terms of spatial augmentations, we use random cropping, sampling the crop independently from the narrow view. We also horizontally flip all the frames with probability 0.5. We resize the shortest size of the original frame to 128 and uniformly sample a 112×112 crop. We find that scale jittering in flow does not improve performance; as a result, we do not employ this augmentation.

Random Convolutions: Following [88], we use He initialization [35] for the weights, fixed zero bias and dimension-preserving padding. We sample the size of the kernel uniformly across odd values ranging from 1 to 11. All sampling of kernel size and weights is performed on a per-sample basis. We refer the reader to the original paper for further details and illustration of the augmentation procedure.

Spectrograms: The audio is sampled at 48k Hz. We take 80 bins of log-mel spectrograms extracted with Hanning windows of size 320 (6.67 ms) at a stride of 160 (3.33 ms).

B. Downstream task evaluation

Linear evaluation on HMDB51, UCF101 and ESC-50. For the linear evaluation on HMDB51, UCF101, and ESC-50 we use the SVM implementation of SciKit-Learn [65]. For all three datasets, we use the same augmentations as during the pre-training stage except for gaussian blurring, and process 10 epochs worth of augmented samples. For each sample, we extract features using the pre-trained backbone. When evaluating the TSM-50, R3D and R(2+1)D visual backbones and the RN-50 audio backbone, we find it helpful to rescale the features using a batch norm layer with fixed scaling and offset parameters (respectively of 1 and 0), collecting training statistics over the extracted features. We sweep the value for the regularization parameter of the SVM in the following set of values: $\{10^{-5}, 3 \cdot 10^{-5}, 10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}, 10^{-2}\}$. When evaluating TSM-NF-F0 and TSM-50x2, we find it more effective to remove this normalization procedure. In these cases, we sweep the value for the regularization parameter of the SVM in the following set of values: $\{1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.09, 0.08,$ $0.07, 0.06, 0.05, 0.04, 0.03, 0.02, 0.01\}$. For all models and downstream tasks, we use the first split to pick the optimal value and report the average of all the splits in that regime. At test time, we do not apply any augmentation. We subsample test videos to 12.5 FPS. For HMDB51 and UCF101, given a test video, we resize the minimum side to 256 and then average the predictions over 30 clips of size 224×224 (10 temporal clips regularly spaced within the video each providing 3 random spatial crops). For HMDB51 and UCF, we use clips of 32 frames. For ESC-50 we use a single window of 5s at test time. Finally, one special case is the ablation with a 10s narrow view (first row, Table 1), which is trained with 64 frames at 6.25 FPS and 112×112 crops. For fairness, we evaluate it with clips of size 112×112 (minimum side 128) of 64 frames subsampled at 6.25 FPS (same frame rate than in training).

Finetuning evaluation on HMDB51 and UCF101. For fine-tuning, we use the SGD optimizer with momentum set to 0.9. We use a batch size of 256 for all methods except for R(2+1)D-50 and TSM-50x2 where we use a smaller batch size of 128 due to their high memory requirements. The batch is distributed over 32 workers. Although we use cross replica batch norm during pre-training (*i.e.* the statistics are accumulated over the 32 workers), during finetuning, we find it better to only compute statistics of batch norm within each worker. We hypothesize that this has a regularization effect on these small datasets. We use a linear warm up for the learning rate for 50 epochs (starting from 0.0 to the initial learning rate value). Learning rate is then decreased using a cosine decay for 550 epochs. Weight decay is employed on the weights of the network (except bias and batch norm parameters). We also apply dropout before the last linear layer mapping the representation to the logits of the classes. We cross validate the value of the initial learning rate (taking values in $\{0.03, 0.1, 0.3\}$), the weight decay (taking values in $\{0., 10^{-7}\}$) and dropout rate (taking values in $\{0.1, 0.5\}$). Similarly to the linear setting, we select hyperparameters on split 1 of each downstream task and report averaged performance values across splits. We noticed that TSM-NF-Net needed slightly different parameters (probably due to the fact that this model does not use any form of normalization) so we adapted the range of the following hyperparameters: higher value of dropout in $\{0.5, 0.8\}$ and smaller learning rate on HMDB51 in $\{0.01, 0.03\}$. The values of hyperparameters found for all networks are given in Table 6. For training, we apply the following augmentation procedure in this order: temporal sampling, scale jittering, resizing the minimum side to 256, extracting a random crop of 224×224 and random horizontal flipping. For temporal sampling, we randomly sample in time a subclip of 32 frames from the original video clip. For scale jittering, we independently scale width and height by a value uniformly sampled from $[0.8, 1.2]$. At test time, we resize the minimum side to 256 and then average the predictions over 30 clips of size 224×224 (10 temporal clips regularly spaced within the video each providing 3 random spatial crops). We use the same FPS as during pre-training, *i.e.* 12.5 FPS.

Linear evaluation on Kinetics600. Since Kinetics600 is too large to fit in memory, we cannot use Scikit-Learn directly. Instead we train the linear layer for 80 epochs using the SGD optimizer with momentum set to 0.99 with a batch size of 256. We found it beneficial to apply batch norm and L2 normalization before the linear layer. We use a linear warm up for the learning rate for 5 epochs (starting from 0.0 to the initial learning rate value). Weight decay is employed on the linear layer's weights (excluding bias parameters). We also apply dropout just before the linear layer (after batch norm and L2 normalization). We cross validate the value of the initial learning rate (taking values in `np.logspace(-0.5, 0, 4)`), the weight decay (taking values in $\{0., 10^{-8}\}$) and dropout rate (taking values in $\{0.0, 0.05\}$) on a small held out set from the training set (4K videos). For training, we apply the following augmentations in this order: temporal sampling, resizing the minimum side to 256, extracting a random crop of 224×224 and random horizontal flipping. For temporal sampling, we randomly sample in time a subclip of 32 frames from the original video clip. At test time, we resize the minimum side to 256 and then average the prediction over 30 clips of size 256×256 (10 temporal clips linearly spaced within the video each with 3 spatial crops). We do not apply scale jittering or horizontal flipping during test time. We use the same FPS as during pre-training, *i.e.* 12.5 FPS. We report the top 1 accuracy on the validation set of Kinetics600.

Shallow classifier evaluation on AudioSet. Following the protocols in [3, 41, 42], we evaluate our audio representations by training a shallow MLP on AudioSet. The MLP has 1 hidden layer with 512 units, and is trained with the Adam optimizer using a batch size of 512 for 20 epochs. We use batch normalization layers on the frozen audio features and after the hidden layer. A ReLU activation function is applied after the second batch normalization. We use a linear warm up of 5000 steps starting from 0.0 to the initial learning rate of 2×10^{-4}, which then decays following a cosine function. At test time, we use 10 overlapping crops of length 5s regularly spaced throughout the audio clip.

C. Architecture details about TSM-NF-F0

Normalizer Free Networks (NF-Nets in short) are a recently introduced family of networks [12] that do not use any form of normalization and are the current state-of-the-art on the ImageNet benchmark [13]. We adapt this architecture to video by applying the Temporal Shift Module [49] algorithm. In details, we insert the temporal shift module in all Normalizer Free blocks at the beginning of the residual branch, following same approach as for ResNets [49]. In our work, we use the smallest network out of the NF-Net family (NF-Net-F0). As shown in Table 5 of the main paper, we obtain remarkable performance in the linear setting when using these networks even though their latent dimension is not much larger than our TSM-RN50 (3072 vs 2048). This may be due to the fact that these networks do not employ

Table 6. Hyperparameters for finetuning on HMDB51 and UCF101.

Method	Backbone	Dataset	HMDB51			UCF101		
			Dropout	LR base	Weight decay	Dropout	LR base	Weight decay
BraVe:V↔V	TSM-50	K600	0.5	0.3	10^{-7}	0.1	0.3	10^{-7}
BraVe:V↔F	TSM-50	K600	0.1	0.1	10^{-7}	0.5	0.1	0.0
BraVe:V↔V	R3D50	K600	0.1	0.1	10^{-7}	0.5	0.1	0.0
BraVe:V↔F	R3D50	K600	0.5	0.3	10^{-7}	0.5	0.1	10^{-7}
BraVe:V↔A	R(2+1)D-18	AS	0.5	0.1	10^{-7}	0.1	0.1	0.0
BraVe:V↔A	TSM-50	AS	0.5	0.3	0.0	0.5	0.3	0.0
BraVe:V↔FA	TSM-50	AS	0.5	0.3	10^{-7}	0.5	0.1	10^{-7}
BraVe:V↔FA	R(2+1)D-50	AS	0.1	0.1	10^{-7}	0.1	0.1	10^{-7}
BraVe:V↔FA	TSM-NF-F0	AS	0.8	0.03	0.0	0.8	0.1	10^{-7}
BraVe:V↔FA	TSM-50x2	AS	0.1	0.1	0.0	0.1	0.1	10^{-7}

Table 7. **Sync study.** Effect of syncing the narrow and broad views.

Dataset	Sync	M_b	HMDB51	UCF101	K600
AS	✗	Audio	67.2	92.1	69.0
AS	✓	Audio	**68.1**	**92.4**	**69.2**

any form of normalizer which might make them more suited for linear evaluation.

D. Syncing audio and video

Table 7 shows the performance of a model trained with broad audio view when the narrow view and the broad view start at the same temporal instant (*sync*) or are independently randomly sampled in time (*async*). As discussed in Section 3.1 in the paper, this experiment supports already established evidence [45] that syncing audio and video is beneficial for the resulting model in self-supervised learning.