Multimodal Contrastive Learning for Unsupervised Video Representation Learning

Anup Hiremath
Avideh Zakhor, Ed.
Gerald Friedland, Ed.

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-206
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-206.html

August 12, 2022

Multimodal Contrastive Learning for Unsupervised Video Representation Learning

by Anup Hiremath

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Avideh Zakhor

Professor Avideh Zakhor
Research Advisor

8/11/2022

(Date)

* * * * * * *

Gerald Friedland

Professor Gerald Friedland
Second Reader

August 12, 2022

(Date)

Contents

1 Abstract

In this report, we propose a multimodal unsupervised video learning algorithm designed to incorporate information from any number of modalities present in the data. We cooperatively train a network corresponding to each modality: at each stage of training, one of these networks is selected to be trained using the output of the other networks. To verify our algorithm, we train a model using RGB, optical flow, and audio. We then evaluate the effectiveness of our unsupervised learning model by performing action classification and nearest neighbor retrieval on a supervised dataset. We compare this triple modality model to contrastive learning models using one or two modalities, and find using all three modalities in tandem provides a 1.5% improvement in UCF101 classification accuracy, a 1.4% improvement in R@1 retrieval recall, a 3.5% improvement in R@5 retrieval recall, and a 2.4% improvement in R@10 retrieval recall as compared to using only RGB and optical flow, demonstrating the merit of utilizing as many modalities as possible in a cooperative learning model.

2 Introduction

The large amount of video data available in the modern internet has led to an increased emphasis on fast and effective ways to classify and learn from repositories of video data. However, training traditional supervised machine learning models on data requires the presence of labels, which are not easily available in most cases. This leads to the task of unsupervised representation learning: training a model that takes an unlabeled data sample as input and yields a vector that represents a sample's useful features as accurately and succinctly as possible. When a supervised task is provided, this pre-trained model can be used as the backbone to speed up training for the supervised task. The unsupervised model can be used as weight initialization for a supervised model, improving the accuracy of supervised learning models without the need for additional labelled data. Alternatively, the unsupervised model can work as an encoder, extracting the relevant features of video to train a fully connected layer on the supervised task.

This report explores ways to leverage the different modalities available in video data to make a more effective unsupervised pre-training method. Our approach uses contrastive learning to create "pseudo-labels" to train the model with, similar to InfoNCE [1]. In contrastive learning, a model is trained on unlabeled data to yield distinct outputs for samples, such as images or video clips, that are found to be dissimilar by some metric, and similar outputs for similar samples. In our case, we classify a pair of sample video clips as being similar or dissimilar using the other modalities in the sample. In one of our steps, we train an unsupervised model on RGB frame data by finding pairs of video clips that are similar in optical flow or audio space. This is effective because while two samples may differ in RGB feature space, they might prove similar in either the flow or audio space. InfoNCE [1] uses only RGB data in a contrastive learning scheme and CoCLR [2] extends it to use RGB and optical flow. In this report, we extend CoCLR [2] to incorporate audio.

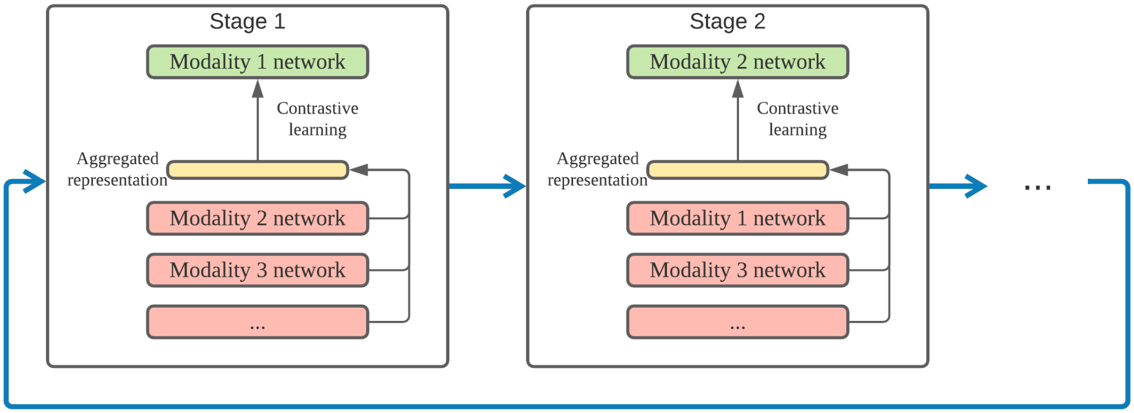

Figure 1: The overall training scheme. The green blocks represent networks that are being trained in a given stage, and the red blocks are the networks that are acting as the ”oracle”. These networks have their weights frozen, and are used to generate aggregated representations that can then be used to train the green network. After each modality network is trained respective to the others, the cycle repeats.

3 Related Work

3.1 Contrastive learning

The field of unsupervised learning is active, and the current state-of-the-art is contrastive learning, in which models are trained by learning to similarly represent augmentations of the same instance, or instances that are known to be similar, and vice versa [1][2][3][4][5][6]. InfoNCE [1] follows this formula, creating a positive pseudo-class by augmenting a sample multiple times, and a negative pseudo-class by augmenting other samples. The model is then trained to similarly represent pairs of samples both from the positive class and distinctly represent all other pairs. Momentum contrast, or MoCo [3] provides a sampling algorithm to run InfoNCE efficiently; negative examples are maintained in a queue in order to prevent having to read them from disk and augment them on every step. SimCLR [5] provides another possible sampling algorithm, in which each sample in a batch is augmented to produce N positive pairs, and then each positive pair is contrasted with the other $2(N-1)$ augmented samples in the batch. DenseCL [4] uses momentum contrast, but rather than training based on a single global representation vector of each sample, it computes a representation vector of each feature of each sample, creating a dense representation to be contrasted. All of the above papers focus on using contrastive learning for image data. In this report, we build upon MoCo [3] for its speed and simplicity, both crucial when dealing with much larger video data.

3.2 Contrastive Video Learning

Much research has been done on adapting contrastive learning to the specific properties of video data [7][8][9][10][11]. CVRL [8] performs both spatial augmentations, consisting of standard image augmentations, and temporal augmentations, which involves selecting different timestamps of the same video. It then trains its model using the InfoNCE [1] loss. VideoMoCo [9] extends momentum contrast specifically for videos, by attenuating past samples in the queue to prefer those more recently added.

We train our model to yield similar representations invariant to temporal transformations; that is, we expect the same representation from two clips cut from the same video. Recent approaches such as Jenni et al. [10], instead train their model to specifically recognize temporal information. Behrmann et al. [11] use InfoNCE to train a model to recognize whether one clip takes place after another or vice versa, and use this specific temporal information to yield a representation learner. RSPNet [7] augments samples by playing them at different speeds, and trains a model to similarly represent samples from different videos that play at the same speed. None of these recent papers take into account other modalities represented in the data, and instead represent new ways of working with only RGB frame data.

3.3 Multimodal Video Learning

There are existing approaches that work with different modalities present in unlabeled video data [12][13][14][15][16][17]. One approach is to devise specific cross-modality tasks that can be used to construct a loss function. Arandjelović et al. [12] train a model to detect audio and video clips that correspond to the same time stamp. Korbar et al. [13] use this idea to train a representation learner to use in downstream supervised tasks. Piergiovanni et al. [14] construct many such cross-modality tasks, and incorporate the modalities into the formulation of the loss. Another approach, taken by XDC [15] and Asano et al. [16] is to use the different modalities to construct an effective clustering model and use the resulting clusters as pseudo-classes. GDT [17] treats the extraction of a modality from a video as an augmentation and incorporates augmentations into the loss function itself, thus learning to recognize samples from the same video even if the samples are of different modalities.

Our proposed approach is similar to that of CoCLR [2], which trains a neural network for each modality cooperatively, alternating the selected network to train using the other networks as an oracle. CoCLR [2] extracts RGB and dense optical flow, and uses a modified version of the InfoNCE [1] loss to train an RGB net and a flow net against one another. To train the RGB net, positive pairs for some given sample are extracted by getting the top k clips with the representation closest to that sample according to the optical flow network. CMA, or Cross-Modal Agreement, [6] uses RGB and audio instead, and rather than alternating one of the networks as the oracle, uses both the RGB and the audio representations of a sample to find similar samples. In this report, we extend the alternating-oracle pattern of CoCLR [2] to three modalities, thus increasing the information available to the models.

4 Approach

Our approach involves two stages: pre-training and training. In the pre-training step, we train three encoders using InfoNCE [1] with momentum contrast [3] for each one of our three modalities: RGB, optical flow, and audio. In the training step, we use the pre-trained encoders in a cooperative contrastive learning scheme similar to CoCLR [2].

4.1 Pre-Training

Our algorithm, as illustrated in Figure 1, involves alternating selection of one of the modalities on which to train a encoder net, while freezing the encoders corresponding to other modalities. This is similar to the way CoCLR [2] alternates between RGB and optical flow, extended to three modalities. The frozen encoders act as an "oracle", determining which other samples in the dataset are most similar to a given sample. In order for those frozen encoders to serve as reasonable "oracles" to train the other encoders with, we need to first make sure they are trained to represent their respective modalities accurately. This is done using the InfoNCE [1] loss. For our encoder f, let $x \in \mathcal{D} = x_1, x_2, ...x_S$ be one modality of a video clip sampled from dataset $\mathcal{D}$. For some video x, let a_1 and a_2 be two random augmentations of x. These could be spatial augmentations, such as blurs, masking, etc., or temporal augmentation, consisting of changing the timestamp in the source video a clip is selected from. Assume a_1 and a_2 form a *positive pair*, i.e. our encoder needs to learn that these two vectors are similar. Then, our negative set $\mathcal{N}$ is composed of random augmentations of other samples from the dataset; vectors that form *negative pairs* with a_1. The InfoNCE [1] loss is:

$$\mathcal{L}_{\text{InfoNCE}} = -log[\frac{exp(f(a_1) \cdot f(a_2)/\tau)}{exp(f(a_1) \cdot f(a_2)/\tau) + \sum_{a_n \in \mathcal{N}} exp(f(a_1) \cdot f(a_n)/\tau)}] \tag{1}$$

Note that this is simply a softmax cross-entropy classification loss with temperature τ: the "classes" are all the samples in the dataset, the "logits" are $[f(a_1) \cdot f(a)$ for $a \in \{a_2, \mathcal{N}\}]$, and the "label" is the index of the logit corresponding to a_2. In other words, if the samples in the dataset are "classes", the correct "class" is the sample that forms a positive pair with a_1. This makes the encoder learn to recognize different random augmentations from the same video as being similar, leading it to encode only important features of the given clip.

In practice, fetching samples from $\mathcal{N}$ can be computationally inefficient, since for every time-step, negative examples $f(a_n)$ need to fetched from disk, augmented, and run through the encoder. Since we are computing the gradient with respect to x, we do not need to back-propagate through the computation of $f(a_n)$. Instead, a naive solution is to maintain a queue $\mathcal{B}$ of past encoder outputs $f(a_1)$ in memory and use this queue as our negative set $\mathcal{N}$, greatly speeding up the training process [3]. The problem with this approach is that as the encoder f updates with successive gradient steps, past encoder outputs saved in the queue will no longer exactly imitate the output of the current encoder being trained. This is problematic if the encoder makes large gradient steps and acts differently from timestep to timestep.

This problem is mitigated by momentum contrast, or MoCo [3], whose overall block diagram can be found in Figure 2. MoCo maintains two encoders: one to be actively trained, f_q, and the other to encode samples to be saved in the queue, f_k. The modified loss is now given as:

$$\mathcal{L}_{\text{InfoNCE with MoCo}} = -log[\frac{exp(f_q(a_1) \cdot f_k(a_2)/\tau)}{exp(f(a_1) \cdot f(a_2)/\tau) + \sum_{b \in \mathcal{B}} exp(f_q(a_1) \cdot b/\tau)}] \quad (2)$$

After each timestep, the following is carried out [3]:

1. $f_k(a_2)$ is enqueued in the buffer $\mathcal{B}$, and the least recent entry is dequeued.

2. f_k is updated with a momentum update: $f_k \leftarrow m * f_k + (1-m) * f_q$, where m is the momentum hyperparameter

Maintaining f_k this way keeps it from changing too drastically from step to step, but also keeps it from growing stagnant as f_q changes.

4.2 Training

Once the encoders corresponding to each modality are trained we begin the cooperative training part of the algorithm, which is an extension of InfoNCE [1] and CoCLR [2] and the main contribution of this report. Our overall training scheme can be found in Figure 1: Given M pre-trained encoders corresponding to M modalities we select one encoder f for each stage of training, and use the other $M-1$ encoders as oracles to train f. In this report we use $M = 3$, with the modalities RGB, optical flow, and audio. With $M = 2$, this algorithm reduces to CoCLR [2], and with $M = 1$, this algorithm reduces further to InfoNCE [1]. Our proposed scheme outlined in Figure 1 has three stages corresponding to the three modalities, and we summarize a single stage of training in Figure 2.

Similar to the InfoNCE [1] loss, our loss uses cross-entropy to train an encoder that encodes positive pairs similarly and encodes all other pairs differently. The main difference between the single-modality pre-training algorithm and our multimodal algorithm is the way we select positive pairs. Rather than providing only a single positive pair (a_1, a_2), we select k more positive matches for a_1 to form the positive set $\mathcal{P} = \{a_2, a_{p_1}, a_{p_2}, ..., a_{p_k}\}$, as in CoCLR [2].

In order to select these k positive samples, we use the other $M-1$ encoders, which we refer to as *samplers* and whose outputs we concatenate to form an *oracle*. We freeze the weights of these samplers; they are only there to mine positive pairs for the encoder f being trained. When we load in x, which is a single modality of a video clip, we also load in the other $M-1$ modalities of the same video clip: $z_1, z_2, ..., z_{M-1}$. We extract the oracle output o from the samplers $f_{s_1}, f_{s_2}, ..., f_{s_{M-1}}$ as follows:

$$o = concat(f_{s_1}(A(z_1)), f_{s_2}(A(z_2)), ..., f_{s_{M-1}}(A(z_{M-1}))) \tag{3}$$

where $A(z_i)$ is a random augmentation of z_i, specific to the modality of z_i. This process of deriving o is outlined in the upper right of Figure 2. We then similarly compute the oracle output o_i corresponding to every other sample $x_i \in \mathcal{D}$. Of all the pairs (o, o_i), we then find the k pairs with the highest similarity scores; that is, we find the k clips in the dataset that the oracle encodes most similarly to our current clip. The process of selecting positive pairs using multiple oracles is our main contribution over CoCLR [2].

The loss for our input x_i is the same as CoCLR [2]:

$$\mathcal{L}_{\text{CoCLR}} = -log[\frac{\sum_{x_p \in \mathcal{P}} exp(f(a_1) \cdot f(a_p)/\tau)}{\sum_{x_p \in \mathcal{P}} exp(f(a_1) \cdot f(a_p)/\tau) + \sum_{x_n \in \mathcal{N}} exp(f(a_1) \cdot f(a_n)/\tau)}] \tag{4}$$

This is also a softmax cross-entropy classification loss, except instead of only one correct "label" for the encoder to classify, there are now $k+1$ correct "labels". The encoder trains to represent a_i similarly to the way it represents the $k+1$ samples in $\mathcal{P}$ and differently to the way it represents the samples in $\mathcal{N}$.

We modify this loss to incorporate momentum contrast [3]. As illustrated in Figure 2, in addition to the queue $\mathcal{B}$ to store past encoder outputs, we also maintain a queue $\mathcal{O}$ to store past oracle outputs. To derive our positive set $\mathcal{P}$, instead of searching through the entire dataset $\mathcal{D}$ for the k samples with the most similar oracle output, we can approximate the result by searching through $\mathcal{O}$. In addition to avoiding to re-compute many oracle outputs on every step, this has the added benefit of speeding up the top-k search. The loss function now becomes:

$$\mathcal{L}_{\text{CoCLR with MoCo}} = -log[\frac{\sum_{x_p \in \mathcal{P}} exp(f_q(a_1) \cdot f_k(a_p)/\tau)}{\sum_{x_p \in \mathcal{P}} exp(f_q(a_1) \cdot f_k(a_p)/\tau) + \sum_{b \in \mathcal{B}} exp(f_q(a_1) \cdot b/\tau)}] \tag{5}$$

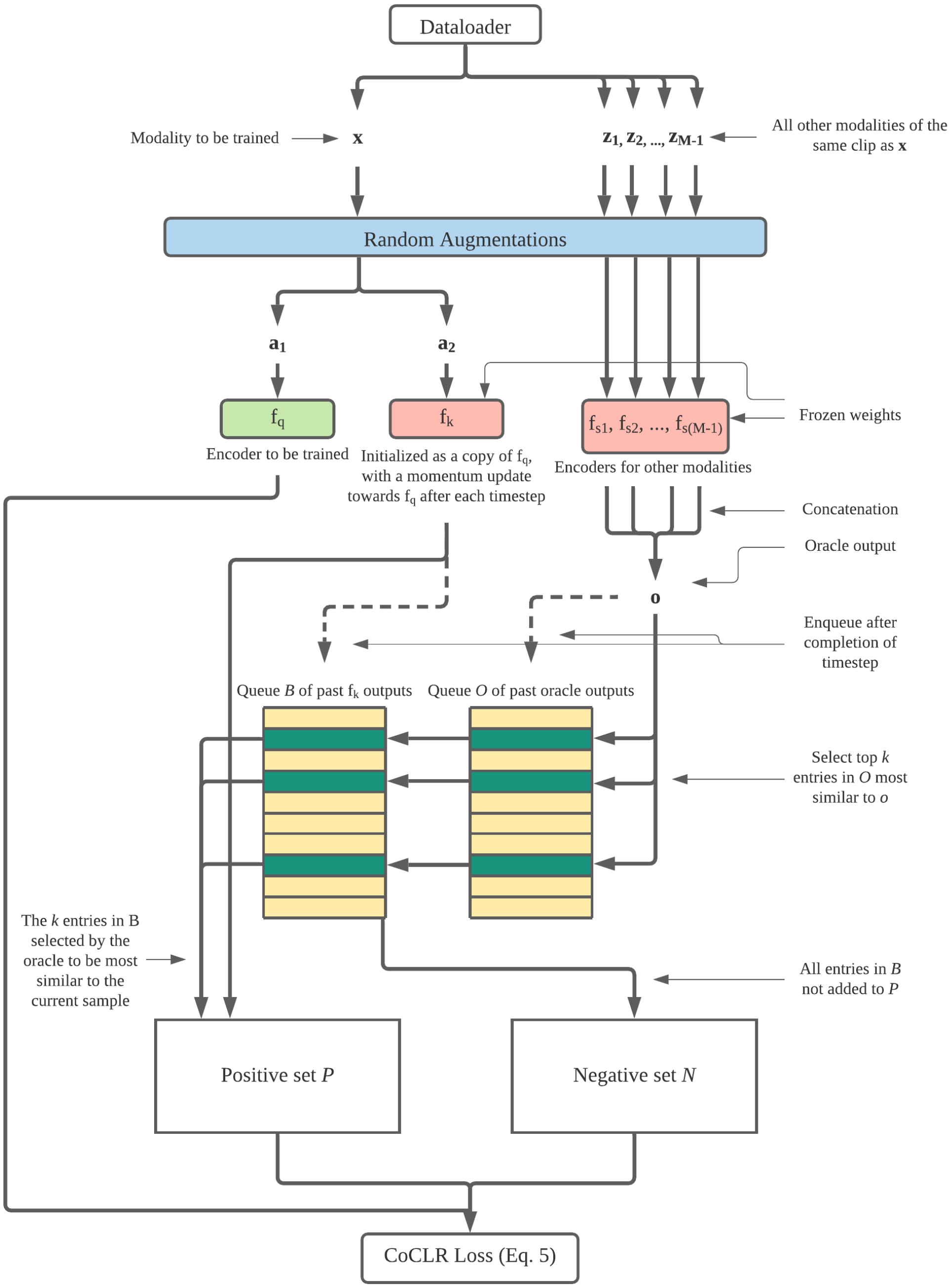
Dataloader
Modality to be trained
x
z1, z2, ..., zM-1
All other modalities of the same clip as x
Random Augmentations
a1
a2
fq
Encoder to be trained
fk
Initialized as a copy of fq, with a momentum update towards fq after each timestep
fs1, fs2, ..., fs(M-1)
Frozen weights
Encoders for other modalities
Concatenation
Oracle output
o
Enqueue after completion of timestep
Queue B of past fk outputs
Queue O of past oracle outputs
Select top k entries in O most similar to o
The k entries in B selected by the oracle to be most similar to the current sample
All entries in B not added to P
Positive set P
Negative set N
CoCLR Loss (Eq. 5)

Figure 2: A single timestep of our entire multimodal contrastive learning algorithm for a single stage. We use Momentum Contrast [3] along with the CoCLR loss [2]. In our paper, $M = 3$, so there are two other modalities z_1 and z_2, and two corresponding encoders f_{s1} and f_{s2}. The entire algorithm described is computed in batches in order to speed up computation.

After the completion of one such stage, a different modality is selected and the next stage begins. The queues are cleared, and the recently trained encoder f_q becomes one of the frozen samplers to make up the next oracle.

5 Experimental Setup

5.1 Modality Extraction

For unsupervised learning, we use the Kinetics-400 dataset [18]. Kinetics is an action classification dataset, consisting of 400 action classes and 306,245 videos. Each sample in the dataset corresponds to a roughly 10 second YouTube video of someone performing a labelled action. Kinetics has become a standard benchmark for video learning tasks, and because of its immense size and variety is commonly used by unsupervised video learning projects to pre-train their models. Then, a different action classification dataset, UCF101 [19], is used for the supervised evaluation of the unsupervised model.

Since we use Kinetics simply as a giant repository of unlabelled videos, we discard the labels given in the dataset. We then clean the data by removing all videos with silent audio, and all videos that are not exactly 300 frames long; since the videos are recorded at 30 frames per second, this is 10 seconds of video. This left us with 202,331 remaining video clips.

After downloading the data from YouTube, we extract the three modalities we are using: rgb, optical flow and audio. 300 RGB frames are extracted from each video and re-sized to 128 by 128. Optical flow is computed on these frames using the denseflow library [20], which calls the OpenCV-CUDA GPU implementation of Zach et al.'s TVL1 dense optical flow algorithm [21]. For each pair of consecutive RGB frames, this algorithm yields flow in the x and y-directions, which is placed into the first and second channels of a 3-channel 128 by 128 image. This is the same technique used by CoCLR [2]: making the RGB and optical flow modalities share the same shape greatly simplifies the implementation of the resulting model, since the representation learner's backbone architecture and augmentation pipeline can be shared between the two modalities.

Audio is handled separately, since it contains fewer spatial dimensions than the other two modalities we work with. We start by extracting a mel-scaled spectrogram using librosa [22]. This is a representation of the audio in which the vertical axis corresponds to different frequencies, with low frequencies on the bottom to high frequencies on top, and the horizontal axis corresponds to time. The spectrogram is computed such that the vertical axis consists of 128 frequencies and scaled horizontally such that 128 pixels corresponds to one second. We then amplify the raw amplitude signal provided by converting the spectrogram's units from power to decibels, and save the spectrograms as greyscale images, setting all of the channels of the image to be equal. This allows us to use an out-of-the-box image classification backbone for our audio data.

5.2 Data Preparation

All of the data is packed into the Lightning Memory-Mapped Database (lmdb) format [23]: a serialized binary format designed to improve access speed for very large datasets. Without this, we would have to read in 2 × $batch\_size$ × $frames\_per\_clip$ JPEG images for each each gradient step for RGB alone. This number of disk accesses proved prohibitively slow, with each batch taking a few minutes to get from disk. By using lmdb [23] we are able to greatly speed up the I/O portion of computation, allowing us to train our model on the full cleaned dataset of more than 200,000 video clips.

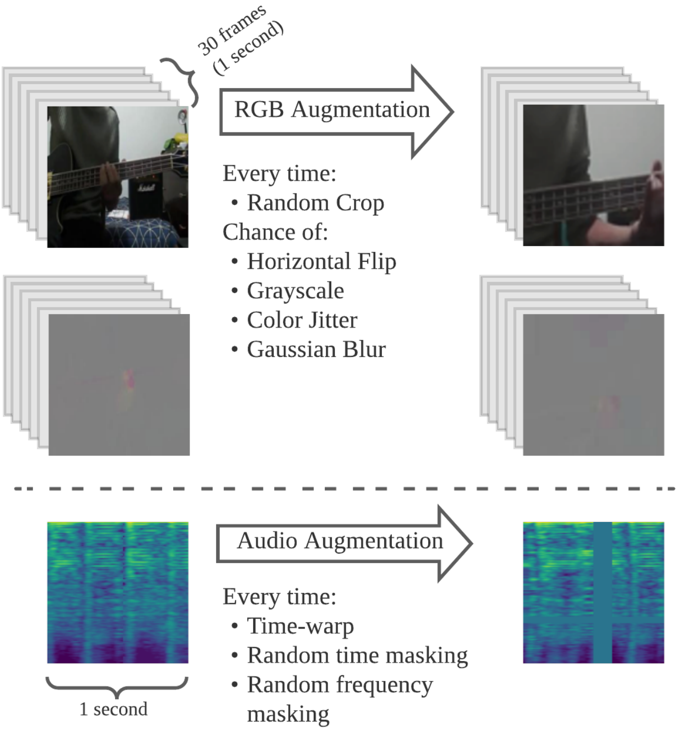

Figure 3: The augmentation types used on the three modalities in each sample: RGB frames, optical flow frames, and an audio spectrogram

5.3 Data Augmentations

In contrastive unsupervised representation learning algorithms such as InfoNCE [1] and SimCLR [5], the choice of data augmentation is crucial. Unlike in supervised tasks where data augmentation is an add-on to make models more robust, in contrastive learning, augmentations are an essential part of the learning process. In the absence of labels, the model is trained to recognize two inputs as different augmentations of the same clip. Without augmentations that accurately simulate the transformations and noise present in real data, the model will not be able to learn the essential features of its training data effectively.

We first perform temporal augmentation. At each iteration of the dataloader, we load the three pre-computed modalities of a video: RGB frames, optical flow, and audio. However, instead of simply sampling, augmenting, and returning a one-second clip from the video, it randomly samples two one-second clips, each from different parts of the same ten second video. The second of these clips is effectively a temporal augmentation of the first; it is the first clip shifted in time. Training the encoder to produce the same output invariant of this augmentation means different parts of the same video should yield the same representation. This temporal augmentation is performed half of the time; the other half of the time the same clip is duplicated and returned, corresponding to no temporal augmentation.

After selecting two clips, spatial transformations are performed as shown in Figure 3. We use the same spatial augmentations for RGB and optical flow as CoCLR [2]. Every clip is randomly cropped and resized back to 128×128, and there is a 50% chance of a horizontal flip. Then, there is a 30% chance that the sample is subjected to additional augmentations: the sample is color-jittered, which randomly modifies the clip in saturation, contrast, brightness, and hue, Gaussian-blurred, and turned greyscale 20% of the time.

The audio portion of the two clips returned from temporal augmentation is augmented using SpecAugment [24], a data augmentation system designed specifically for audio. First, a time-warp is applied; the spectrogram is randomly perturbed in the time axis using a dense image warp. Then, time and frequency masking are applied, corresponding to a vertical and a horizontal band of the spectrogram being randomly replaced with the mean value of the spectrogram. Figure 4 shows some samples of the augmentations applied on the three modalities we work with.

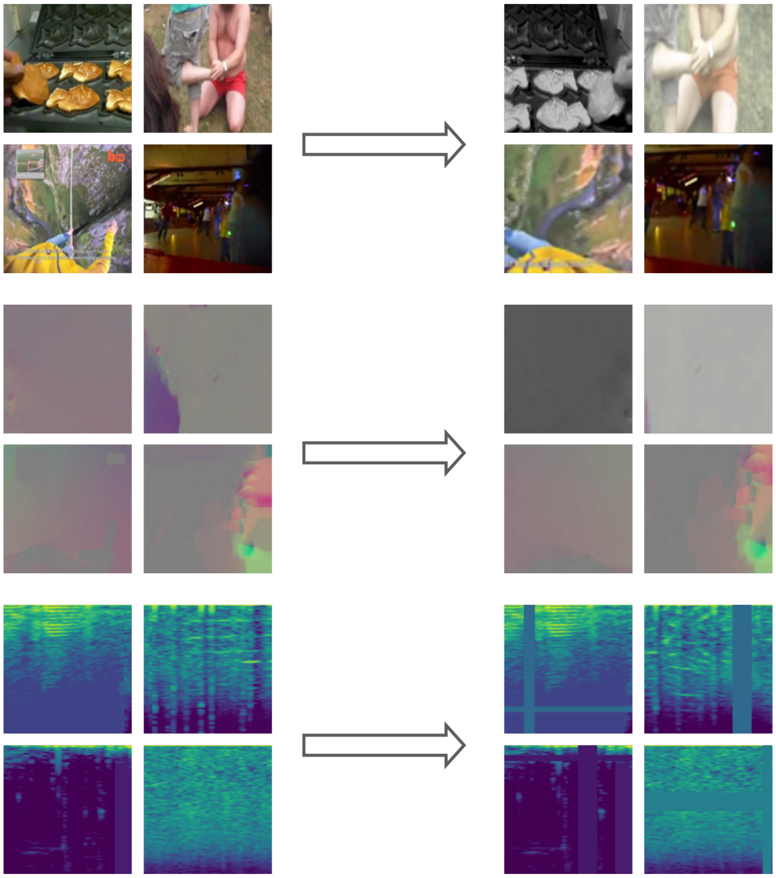

Figure 4: Examples of randomly selected and augmented samples. Many of the optical flow frames appear plain grey or very nearly grey; this corresponds to videos with little to no movement during the sampled second.

In the end, our dataset output at each step consists of six elements: three modalities from each of the two clips sampled from each video. The RGB and flow each have shape $(batch\_size, 30, 3, 128, 128)$, and the audio has shape $(batch\_size, 1, 128, 128)$. This scheme helps maximize the amount of information provided to the model from each video clip.

5.4 Network Architectures

The model is set up in PyTorch, making use of the torch.distributed module to train data across multiple GPUs. For the RGB and Flow networks, we use the S3D network architecture [25] as the backbone. This was chosen because S3D is noted to be specifically well-suited to both RGB and Optical Flow inputs; likely because of this versatility, CoCLR [2] gets their best results using S3D. For the audio network, we use Resnet-18 [26], because it remains a simple, fast, and effective network architecture for image-based tasks. Ultimately, the choice of backbone is not the focus of this report, since the contrastive learning algorithm we describe is agnostic of the backbone used. Both the 3D and 2D backbones can be easily swapped out as more effective network architectures emerge.

5.5 Training Implementation

We first pre-train three single-modality unsupervised models using InfoNCE [1]: one on RGB data, one on audio data, and one on optical flow data. We train each of these encoders for 200 epochs on Kinetics-400.

In order to evaluate the effectiveness of our multimodal training scheme, we then train a contrastive learning model using all three of these modalities. We train this model for two cycles each of 3 stages each, with each stage being 200 epochs - every modality encoder was trained using the other encoders as the oracle for a total of 400 epochs. We set $k = 5$; that is, the oracle selects the top 5 most similar samples from its queue of past outputs. Both queues in our algorithm are 16,384 elements long. We determine this length and all our other hyperparameters empirically; while smaller sized queues result in faster training time, they also result in worse models since the oracle might not be able to find similar samples in a smaller queue.

This model was trained using an Azure compute instances, with Tesla V100 GPU, 110GB of RAM, an Intel Xeon E5-2690 CPU and a 1TB disk. The limiting factor in this was disk usage, as the data preparation process constantly ran into the 1TB limit. For our three-modality model, one training iteration took roughly 3 days, each 3-iteration stage took around 9 days, and the full two stages we ran took around 3 weeks to train.

6 Results

We evaluate our unsupervised training method by evaluating our trained model on two supervised tasks. The first is action classification, in which our model is further trained on the UCF101 action classification dataset [19], and then evaluated against the UCF101 test set. The second is clip retrieval, in which the model retrieves the top N clips with the most similar encoder output to a given test clip, and we evaluate whether any of the retrieved clips share the same class as the test clip. Though all three modalities are used in the training process, the effectiveness of our approach is evaluated using only the encoder dedicated to RGB. Through our cooperative learning method, our RGB encoder is able to infer information about audio and optical flow despite only being supplied RGB data during supervised evaluation.

6.1 Evaluation on Action Classification

We evaluate our model in the same way as past work on this topic [2] [6]: we fine-tune and evaluate our RGB encoder on the supervised action classification task UCF101 [19]. UCF101 consists of around 13,000 clips labelled in 101 different action classes, and is mostly a visual-only dataset; the lack of consistent audio present in most of its clips is the main reason we chose to perform our unsupervised training on Kinetics.

We train our supervised classification models in two ways:

1. Full training: no weights are frozen, and our encoder is trained end-to-end.

2. Linear probe: The entire encoder's weights are frozen except for a final fully connected layer.

In both evaluation approaches, we compare against approaches evaluated using K400 training for a more fair comparison with our model, since training using larger datasets will naturally result in improved classification accuracy. For instance, Elo [14], which is trained with YouTube8M, reaches a 93.8% accuracy, outperforming our model by 4.4%.

UCF101 Classification Accuracy after Full Training

			Modality			
Model	**Date**	**Training Dataset**	**RGB**	**Flow**	**Audio**	**Top-1 Test Acc**
InfoNCE [1]	2018	K400	✓			79.5
CBT [27]	2019	K400	✓			79.5
SpeedNet [28]	2020	K400	✓			81.1
XDC [15]	2020	K400	✓		✓	84.2
CMA [6]	2021	K400	✓		✓	87.5
CoCLR [2]	2021	K400	✓	✓		87.9
TCLR [29]	2022	K400	✓	✓		88.2
STS [30]	2021	K400	✓			89.0
GDT [31]	2021	K400	✓		✓	89.3
RSPNet [7]	2021	K400	✓			**93.7**
Ours	2022	K400	✓	✓	✓	89.4

Table 1: Test accuracy after full training - the unsupervised model serves as weight initialization. State-of-the art accuracy is provided for reference.

In the first method, our unsupervised algorithm on Kinetics serves to initialize weights for our supervised training on UCF101. Since the entire model can be trained end-to-end using the supervised data, this provides the highest classification accuracy. Our model is compared against the state-of-the-art in Table 1, where we see that in full training, our model outperforms InfoNCE [1], which uses only RGB, by 9.9%, and CoCLR [2], which uses RGB and optical flow, by 1.5%. Our model also outperforms the models using only RGB and audio, XDC [15] and CMA [6], by 5.2% and 1.9% respectively, although neither of them use the alternating-oracle method used by our model and CoCLR.

RSPNet [7], which trains its model to recognize the relative speeds at which clips are played, is able to outperform our model using only RGB data by running its model for 1,000 epochs. Our model might be able to build on the results of RSPNet in the future by incorporating different speeds of clips as a temporal augmentation, and by using RSPNet instead of InfoNCE [1] as a more effective pre-training method.

UCF101 Classification Accuracy after Linear Probe

			Modality			
Model	**Date**	**Dataset**	**RGB**	**Flow**	**Audio**	**Top-1 Test Acc**
CBT [27]	2019	K400	✓			54.0
MemDPC [32]	2020	K400	✓			54.1
CoCLR [2]	2021	K400	✓	✓		74.5
Ours	2022	K400	✓	✓	✓	**76.5**

Table 2: Test accuracy after linear probe - the unsupervised model's weights are frozen, except for a single fully connected layer as the end. State-of-the-art accuracy is provided for reference.

In Table 2, using the linear probe, our RGB encoder is truly acting as an encoder: our frozen RGB encoder encodes UCF101 clips as feature vectors and these feature vectors are used to train a single linear layer. This method of evaluation is less common, but provides better insight on the state of the unsupervised model before the introduction of labelled data. In Table 2, using the linear probe, our model outperforms CoCLR [2] by 2%. Thus, our improvement over CoCLR's results shows that audio information can be incorporated successfully into a contrastive learning framework.

6.2 Evaluation on Retrieval

6.2.1 Quantitative Retrieval Results

We also evaluate unsupervised representation learners on the task of video retrieval. In this task, no supervised fine-tuning is done; the model being tested is trained purely using unsupervised data. Similar to classification, only the RGB encoder is used; the other encoders only serve to train the RGB encoder. First, for every sample in both the train and test splits of UCF101, we use the RGB encoder being evaluating to extract a feature vector representation of that sample. Then, for each feature vector from the test set, we look for the k vectors from the training set with the highest cosine similarity. In effect, we ask our model to retrieve the k most similar clips in the train set to each test clip. We evaluate this retrieval by computing the recall R@k: the fraction of clips in the test set for which at least one of the k clips retrieved by our model is of the same action class as the test clip.

UCF101 Nearest-Neighbor Retrieval Recall

			Modality			**Retrieval Recall**			
Model	**Date**	**Dataset**	**RGB**	**Flow**	**Audio**	**R@1**	**R@5**	**R@10**	**R@20**
SpeedNet [28]	2020	K400	✓			13.0	28.1	37.5	49.5
MemDPC [32]	2020	K400	✓			20.2	40.4	52.4	64.7
STS [30]	2021	K400	✓			38.3	59.9	68.9	77.2
CoCLR [2]	2021	K400	✓	✓		44.5	60.6	68.4	77.0
Ours	2022	K400	✓	✓	✓	45.9	64.1	70.8	77.0
InfoNCE [2]	2018	UCF	✓			36.0	52.0	61.8	71.0
CoCLR [2]	2021	UCF	✓	✓		**53.3**	**69.4**	**76.6**	**82.0**

Table 3: Recall on the task of UCF101 nearest-neighbors retrieval. The state-of-the-art is provided for reference.

In Table 3, we find that our model outperforms the state-of-the-art models in UCF101 retrieval with the exception of CoCLR [2]. This is due to the fact that CoCLR's reported retrieval in the last row of Table 3 is evaluated on UCF101 after pre-training with UCF101, unlike our model, which pre-trains on Kinetics-400.

In order for an apples-to-apples comparison with CoCLR to evaluate the contribution of audio information to our training scheme, we also ran UCF101 retrieval on CoCLR pre-trained on Kinetics-400. As expected, incorporating audio into our contrastive learning framework shows a 1.4% improvement in R@1 retrieval, a 3.5% improvement in R@5 retrieval, and a 2.4% improvement in R@10 retrieval performance.

6.2.2 Qualitative Retrieval Results

In Figure 5, we show two examples of how our RGB augmentations result in successful retrieval on UCF101 video clips. Though the clips in the both rows are oriented differently and are different colors, the random flips and color jitter applied as data augmentations enables the RGB encoder to recognize the videos as the same activity. Our model and CoCLR [2] yield similar results for these test clips, since both approaches use RGB augmentations in the same way to train their encoders.

In Figure 6, we show two examples of how incorporating optical flow affects retrieval results. In the top row, we can see that the model has learned to identify biking and horse riding as similar activities due to the similar motion in the videos, despite the differing backgrounds. Likewise, the clips in the bottom row all share a swinging motion despite the variety of activities. Both our model and CoCLR [2] again yield similar results, since both models use optical flow in their approaches.

In Figure 7, we show two examples of how incorporating audio information improves retrieval results, which is the main contribution of this report. In the top row, videos containing music are linked with one another, even if the scene and the motions within them are completely different. This is because music is easily identifiable on a spectrogram: notes and their harmonics show up as sharp, regularly spaced horizontal lines. In the second row, clips of drums and punches are retrieved together because of their similar percussive sounds. Here, there is a noticeable difference between our results and that of CoCLR [2]. Since CoCLR is never supplied audio data, it has not learned to associate clips containing music or percussive sounds with one another, and so it returns classes that are unrelated to the test clip.

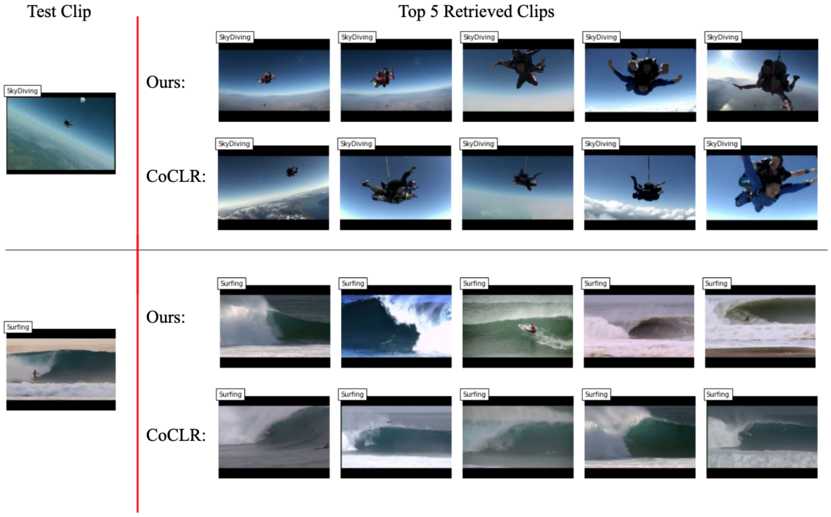

Figure 5: Qualitative effects of incorporating RGB information on UCF101 retrieval results. Our triple modality model is compared against CoCLR [2], both pre-trained on Kinetics-400.

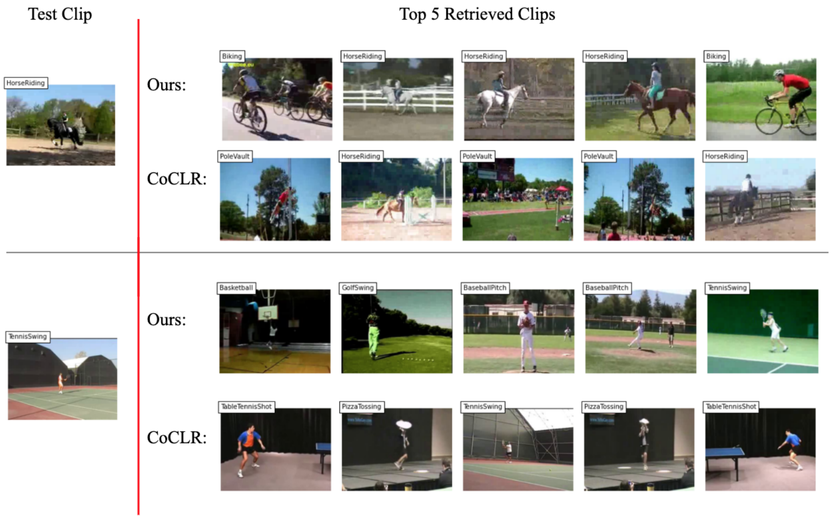

Figure 6: Qualitative effects of incorporating optical flow information on UCF101 retrieval results. Our triple modality model is compared against CoCLR [2], both pre-trained on Kinetics-400.

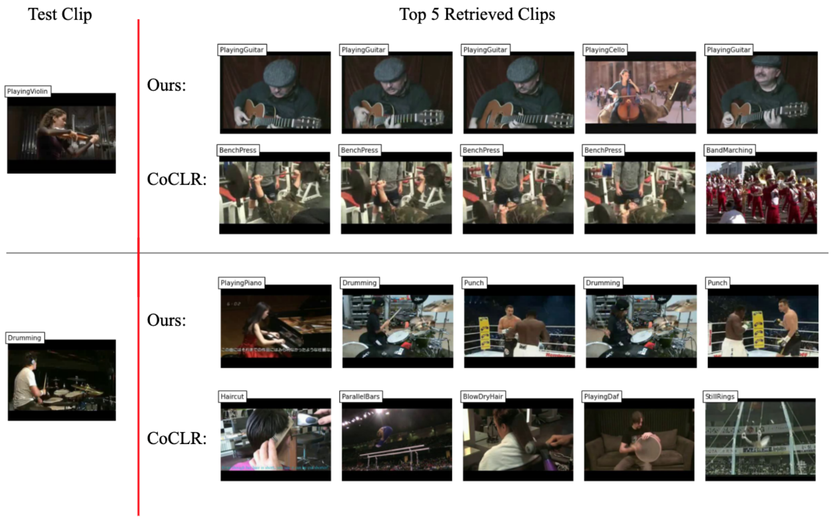

Figure 7: Qualitative effects of incorporating audio information on UCF101 retrieval results. Our triple modality model is compared against CoCLR [2], both pre-trained on Kinetics-400.

7 Conclusions and Future Work

Our approach extends the existing idea of cooperative contrastive video learning from unlabelled video data [2] to incorporate three modalities. We train a model using RGB, optical flow, and audio, and evaluate its performance against existing contrastive learning frameworks with fewer modalities to determine whether incorporating additional modalities using our scheme is effective. We train these models on Kinetics-400 and evaluate them on UCF101, using the tasks of action classification and nearest-neighbors retrieval. We found that in both tasks, our triple-modality model outperformed the contrastive learning models with fewer modalities and compared favorably to the state-of-the-art. Further improvements could be found in simply running our training scheme for more cycles, or by trying out other recent backbone architectures like SlowFast [33]. Our training scheme can be easily extended to other modalities such as text: one could add a fourth encoder with an NLP backbone if subtitles or narration were available for video clips.

References

[1] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018.

[2] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning, 2021.

[3] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2019.

[4] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training, 2021.

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

[6] Pedro Morgado, Nuno Vasconcelos, and Ishan Misra. Audio-visual instance discrimination with cross-modal agreement, 2021.

[7] Peihao Chen, Deng Huang, Dongliang He, Xiang Long, Runhao Zeng, Shilei Wen, Mingkui Tan, and Chuang Gan. Rspnet: Relative speed perception for unsupervised video representation learning, 2021.

[8] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning, 2021.

[9] Tian Pan, Yibing Song, Tianyu Yang, Wenhao Jiang, and Wei Liu. Videomoco: Contrastive video representation learning with temporally adversarial examples, 2021.

[10] Simon Jenni and Hailin Jin. Time-equivariant contrastive video representation learning, 2021.

[11] Nadine Behrmann, Juergen Gall, and Mehdi Noroozi. Unsupervised video representation learning by bidirectional feature prediction, 2020.

[12] Relja Arandjelović and Andrew Zisserman. Look, listen and learn, 2017.

[13] Bruno Korbar, Du Tran, and Lorenzo Torresani. Cooperative learning of audio and video models from self-supervised synchronization, 2018.

[14] AJ Piergiovanni, Anelia Angelova, and Michael S. Ryoo. Evolving losses for unsupervised video representation learning, 2020.

[15] Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering, 2020.

[16] Yuki M. Asano, Mandela Patrick, Christian Rupprecht, and Andrea Vedaldi. Labelling unlabelled videos from scratch with multi-modal self-supervision, 2021.

[17] Mandela Patrick, Yuki M. Asano, Polina Kuznetsova, Ruth Fong, João F. Henriques, Geoffrey Zweig, and Andrea Vedaldi. On compositions of transformations in contrastive self-supervised learning, 2021.

[18] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017.

[19] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild, 2012.

[20] Shiguang* Wang, Zhizhong* Li, Yue Zhao, Yuanjun Xiong, Limin Wang, and Dahua Lin. denseflow. https://github.com/open-mmlab/denseflow, 2020.

[21] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l1 optical flow. volume 4713, pages 214–223, 09 2007.

[22] Brian McFee, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, Ayoub Malek, Dana, Kyungyun Lee, Oriol Nieto, Dan Ellis, Jack Mason, Eric Battenberg, Scott Seyfarth, Ryuichi Yamamoto, viktorandreevichmorozov, Keunwoo Choi, Josh Moore, Rachel Bittner, Shunsuke Hidaka, Ziyao Wei, nullmightybofo, Adam Weiss, Darío Hereñú, Fabian-Robert Stöter, Lorenz Nickel, Pius Friesch, Matt Vollrath, and Taewoon Kim. librosa/librosa: 0.9.2, June 2022.

[23] Howard Chu. Lightning memory-mapped database manager (lmdb), 2011.

[24] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. *Interspeech 2019*, Sep 2019.

[25] Da Zhang, Xiyang Dai, Xin Wang, and Yuan-Fang Wang. S3d: Single shot multi-span detector via fully 3d convolutional networks, 2018.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[27] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Learning video representations using contrastive bidirectional transformer, 2019.

[28] Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T. Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. Speednet: Learning the speediness in videos, 2020.

[29] Ishan R. Dave, Rohit Gupta, Mamshad Nayeem Rizve, and Mubarak Shah. TCLR: temporal contrastive learning for video representation. *CoRR*, abs/2101.07974, 2021.

[30] Jiangliu Wang, Jianbo Jiao, Linchao Bao, Shengfeng He, Wei Liu, and Yunhui Liu. Self-supervised video representation learning by uncovering spatio-temporal statistics. *CoRR*, abs/2008.13426, 2020.

[31] Mandela Patrick, Yuki Asano, Polina Kuznetsova, Ruth Fong, Joao F. Henriques, Geoffrey Zweig, and Andrea Vedaldi. Multi-modal self-supervision from generalized data transformations, 2021.

[32] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning, 2020.

[33] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition, 2019.