DistInit: Learning Video Representations Without a Single Labeled Video

Rohit Girdhar[1*] Du Tran[2] Lorenzo Torresani[2,3] Deva Ramanan[1,4]
[1]Carnegie Mellon University [2]Facebook [3]Dartmouth College [4]Argo AI

Abstract

Video recognition models have progressed significantly over the past few years, evolving from shallow classifiers trained on hand-crafted features to deep spatiotemporal networks. However, labeled video data required to train such models has not been able to keep up with the ever increasing depth and sophistication of these networks. In this work we propose an alternative approach to learning video representations that requires no semantically labeled videos, and instead leverages the years of effort in collecting and labeling large and clean still-image datasets. We do so by using state-of-the-art models pre-trained on image datasets as "teachers" to train video models in a distillation framework. We demonstrate that our method learns truly spatiotemporal features, despite being trained only using supervision from still-image networks. Moreover, it learns good representations across different input modalities, using completely uncurated raw video data sources and with different 2D teacher models. Our method obtains strong transfer performance, outperforming standard techniques for bootstrapping video architectures with image-based models by 16%. We believe that our approach opens up new approaches for learning spatiotemporal representations from unlabeled video data.

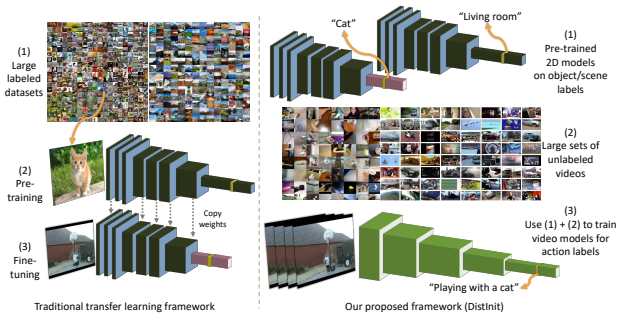


Figure 1: **Learning video representations through transfer.** Traditional approaches to transfer learning follow the process on left: train deep models on large well labeled datasets and finetune on specific task or dataset of choice. This approach, while hugely popular, significantly limits the types of models we can use for our specific task, as they must be "compatible" with the model pre-trained on the large dataset for the learned weights to transfer. This problem is further accentuated in the case of videos, where datasets tend to be small or weakly labeled, and models tend to involve 3D/(2+1)D convolutions, making them "incompatible" with image models. We propose an approach, **DistInit**, to transfer image models to video as shown on the right. DistInit starts from models pre-trained on well labeled image datasets with *object* or *scene* labels, and use them as "teachers" for supervising video models. Hence, the video model is able to learn spatio-temporal features for video understanding, without needing an explicit *action* label for that video.

1. Introduction

Visual recognition tasks have been revolutionized by the resurgence of convolutional neural networks (CNNs) [30, 31] along with the availability of large and well-labeled datasets [32, 40, 60]. This has caused a paradigm shift in computer vision: rather than hand-designing better features [4, 29, 33], most approaches now train deep models that learn features themselves. However, deep learning has been transformative not just because models perform well, but because models also *transfer*. The dominant illustration of this is the use of ImageNet pre-training [40]. It is a near-ubiquitous practice that yields strong improvements across a wide range of tasks, from image classification on small datasets [26] to pixel-labeling tasks like detection and segmentation [19]. Such pre-training is an empirically effective approach to knowledge transfer, where "knowledge" is manifested as labeled and curated datasets.

However, feature learning has not been quite as transformative for video understanding. Early attempts for human action recognition [24] achieved only marginal improvements over previous state-of-the-art hand-crafted features. Since then, numerous deep architectures [8, 9, 15, 16, 44, 51] have been proposed. Interestingly, most performance gains seem to arise from the recent introduction of large-scale video datasets that are carefully curated and annotated [3, 25], enabling effective pre-training. Our work introduces simple but novel approaches for pre-training with

*Work done as a part of an internship at Facebook

*un*labeled, *un*curated videos. Our approach is motivated by the following two questions:

1. What are the "right" labels for a video? Previous work tends to manually defines action ontologies in a top-down fashion [25, 28, 46] or else discovers classes from bottom-up clustering [12, 57]. Action classes may be broad – "washing" may include both washing one's hands or washing a car [36]. Classes may also be fine-grained and nuanced, such as "snuggling with a pillow" [43]. Evidently, the right action vocabulary is unclear and largely up for debate. In contrast, *object* labels seem to much better understood, as they can exploit widespread linguistic knowledge bases such as WordNet. Finally, humans appear able to learn about behaviors and the dynamics of the world even without such explicit action labels. In this work, we answer this question by *making use of objects to label videos*.

2. How do we transfer the knowledge encoded in image datasets [32, 40, 60] into video models? As discussed earlier, the dominant approach is pre-training. However, because spatiotemporal networks structurally expect a space-time video as input, they are difficult to (pre)train on images. As a result, many spacetime networks are intentionally designed with an image "backbone" that allows for pre-training on images. Popular examples include two-stream networks [45] and 3D CNNs that "inflate" 2D kernels to 3D [3, 9, 10]. However, this places *severe* restrictions on the types of video architectures that can be explored. Instead, we introduce a general approach of *knowledge transfer by distillation*, which allows us to transfer knowledge from arbitrary image-based teachers to *any* spatiotemporal architecture (Fig. 1). We refer to our approach as **DistInit**.

DistInit leads to a significant 16% improvement over from-scratch training on the HMDB dataset, getting almost half-way to the improvement provided by pretraining on a fully-supervised dataset like Kinetics [25]. From-scratch training is the defacto standard for state-of-the-art architectures that can not be initialized or inflated from image architectures [51]. While large-scale video datasets like Kinetics now provide an alternate path for pre-training, DistInit does so without requiring *any* video data curation. As we show in Section 4.3, it is able to learn competitive representations from an internal uncurated dataset of random web videos. This is in contrast to previous works [7, 13, 35] on unsupervised learning that use ImageNet without labels but still potentially benefit from the data curation.

2. Related Work

Feature learning: Video understanding, specifically for the task of human action recognition, is a well studied problem in computer vision. Analogously to the progress of image-based recognition methods, which have advanced from hand-crafted features [4, 33] to modern deep networks [20, 45, 48], video understanding methods have also evolved from hand-designed models [29, 53, 54] to deep spatiotemporal networks [44, 49]. However, while image based recognition has seen dramatic gains in accuracy, improvements in video analysis have been more modest. In the still-image domain, deep models have greatly benefited from the availability of well-labeled datasets, such as ImageNet [40] or Places [60].

Video datasets: Until recently, video datasets have either been well-labeled but small [28, 43, 46], or large but weakly-labeled [1, 24]. A recently introduced dataset, Kinetics [25], is currently the largest well-annotated dataset, with around 300K videos labeled into 400 categories (we note a larger version with 600K videos in 600 categories was recently released). It is nearly two orders of magnitude larger than previously established benchmarks in video classification [28, 46]. As expected, pre-training networks on this dataset has yielded significant gains in accuracy [3] on many standard benchmarks [28, 43, 46], and have won CVPR 2017 ActivityNet and Charades challenges. However, it is worth noting that this dataset was collected at a significant curation and annotation effort [25].

Video labels: The challenge in generating large-scale well-labeled video datasets stems from the fact that a human annotator has to spend much longer to label a video compared to a single image. Previous work has attempted to reduce this labeling effort through heuristics [59], but these methods still require a human annotator to clean up the final labels. There has also been some work in learning unsupervised video representations [34, 42], however has typically lead to inferior results compared to supervised features.

Pre-training: The question we pose is: since labeling images is faster, and since we already have large, well-labeled image datasets such as ImageNet, can we instead use these to bootstrap the learning of spatiotemporal video architectures? Unsurprisingly, various previous approaches have attempted this. The popular two-stream architecture [44] uses individual frames from the video as input. Hence it initializes the RGB stream of the network with weights pre-trained on ImageNet and then fine-tunes them for action classification on the action dataset. More recent variants of two-stream architectures have also initialized the flow stream [55] from weights pretrained on ImageNet by viewing optical flow as a grayscale image.

Inflation: However, such initializations are only applicable to video models that use 2D convolutions, analogous to those applied in CNNs for still-images. What about more complex, truly spatiotemporal models, such as 3D convolutional architectures [49]? Until recently, such models have largely been limited to pre-training on large but weakly-labeled video datasets, such as Sports1M [24]. Recent work [3, 9] proposed a nice alternative, consisting of inflating standard 2D CNNs kernels to 3D, by simply replicating the 2D kernels in time. While effective in getting

strong performance on large benchmarks, on small datasets this approach tends to bias video models to be close to static replicas of the image models. Moreover, such initialization constrains the 3D architecture to be identical to the 2D CNN, except for the additional third dimension in kernels. This effectively restricts the design of video models to extensions of what works best in the still-image domain, which may not be the architectures for video analysis.

Distillation: Our approach is inspired by techniques that distill teacher networks into student models [21]. However, distillation typically trains smaller student models on the same data distribution used to train the teacher (with the goal of increasing efficiency). Our approach instead focuses on representation learning through pre-training. Our students are larger (3D vs 2D CNNs) and geared for different data domains (videos vs images). The most related work may be cross-modal distillation [18], which transfers supervision from RGB to flow or depth modalities. Importantly, such work focuses on the *same* end task, such as object detection. In contrast, we focus on *task distillation*, where tasks are quite different (object detection versus action classification). From this perspective, our philosophy aligns with taskonomies [58], which advocates the use of different pre-training tasks for a variety of target tasks. But rather than advocating pre-training, we pursue distillation since it allows for arbitrary changes in network topology. Other works have also used similar distillation frameworks for action recognition tasks, such as [5] which transfers supervision from frames to videos by solving a correspondence problem. Our approach is much simpler, and directly transfers supervision by matching label distributions. Finally, prior works have also shown improvements using the 2D image-based networks ('teachers', in our context) directly, as additional features for action recognition [6, 14], hence reinforcing our observation that scene/object information is a very useful cue for video understanding.

Domain adaptation: Our work is also related to techniques for domain adaptation (DA) [37, 41], where the goal is adapting a network to a new data distribution. Our formulation differs in that we also adapt to new tasks (object classification vs action recognition) and network architectures (2D CNNs vs 3D CNNs). We show extensive experiments with standard benchmarks and show significant improvements over inflation and other previous approaches in learning video representations for action recognition.

3. Our Approach

We now describe our approach in detail. To reiterate, our goal is to learn video representations without using any video annotations. We do so by leveraging pre-trained 2D networks, using them to supervise or "teach" the video models. Hence, we refer to the 2D pre-trained networks as "teachers" and our target video network as "student".

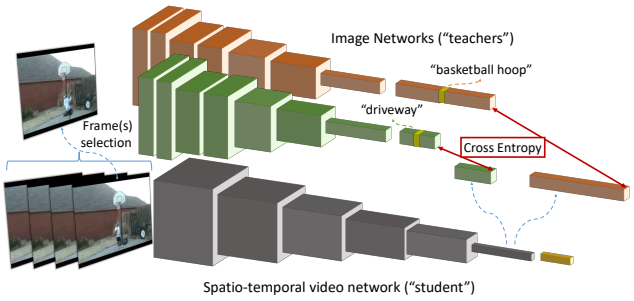


Figure 2: **DistInit network architecture.** We use random frames from the input clip to generate soft-labels for the video model, using an arbitrary number of image-based teachers networks. The student tries to match the targets provided by the teachers.

We make no assumption over the respective architectures of these models, i.e., we do no constrain the structure of the 3D network to be merely a 3D version of the 2D networks it learns from or to have a structure compatible with them.

Figure 2 depicts the network architecture used to train the student network. We start with teacher networks trained on standard image-level tasks, such as image classification on ImageNet. While in this work we primarily focus on classification, our architecture is generic and can also benefit from teachers trained on spatial tasks such as detection, keypoint estimation and so on, with the only difference being the definition of the distillation loss function. Also, our architecture is naturally amenable to work with an arbitrary number of teachers, which can be used in a multi-task learning framework to distill information from multiple domains into the student. Throughout the training process, these teacher networks are kept fixed, in "test" mode, and are used to extract a feature representation from the video to be used as a "target" to supervise the student network.

Since teacher networks are designed to find *objects* in images, it is not obvious how to use them to extract features for *actions* in video. We propose a simple solution: pre-train the spatiotemporal action network for finding objects in frames of a video. However, our teacher networks are designed to work over images, so how do we apply them on a video? We experiment with standard approaches from the literature, including uniform or random sampling of frames [44], as well as averaging predictions from multiple different frames [55]. In this work we use the last-layer features in the form of normalized softmax predictions or (unnormalized) logits. In case of multiple frames, we average the teacher logits before computing a normalized prediction target. The *student* network then takes the complete video clip as input. We train it to be able to predict the features or probability distribution produced by the teacher. For this purpose, we define the last layer in the student network to be a linear layer that takes the final spatiotemporally averaged feature tensor and maps it to a number of units that

matches the dimensionality of the output generated by the teacher. In case of multiple teachers, we define a linear layer per teacher, and optimize all losses jointly.

To formalize, let us denote a video as $x = \{x_1, \dots x_T\}$, where x_t is the t^{th} frame. In our problem formulation, we have access to a teacher that reports a prediction label at each frame. For simplicity, we assume that the teacher returns a (softmax) distribution over K classification labels. We generate a distribution over labels for a video by (1) averaging the per-frame logits for the k^{th} class $z_k(x_t)$ and (2) passing the average through a softmax function with temprature τ (typically $\tau = 1$):

$$y_k(x) = \frac{e^{\frac{1}{\tau T}\sum_t z_k(x_t)}}{\sum_l e^{\frac{1}{\tau T}\sum_t z_l(x_t)}} \quad \textbf{[Temporal Averaging]} \quad (1)$$

The resulting distribution is then used as soft targets for training weights w associated with a student network f of arbitrary architecture by means of the following objective:

$$\text{Loss}(w) = -\sum_k y_k(x) \log f_k(x; w) \quad \textbf{[Soft Targets]} \quad (2)$$

where $f_k(x; w)$ is the student softmax distribution for label k. Finally, we explore multi-source knowledge distillation by adding together losses from different image-based teachers:

$$\min_w \sum_i \text{Loss}_i(w) \quad \textbf{[Multi-Source Distillation]} \quad (3)$$

In our experiments, we explore teachers trained for object classification (ImageNet) and scene classification (Places).

We train the network using loss functions inspired from the network distillation literature [2, 21]. When using the teacher network to produce a probability distribution, we train the student to produce a matching distribution by incurring a cross entropy loss between the two distributions. As suggested in [21], we also tried using different values of *temperature* (τ) to scale the logits before computing softmax and cross entropy, but found temperature value of 1 to perform best in our experiments. We also experimented with the a mean squared loss on the logits (before softmax normalization), as suggested in [2], and compare performance in Section 4.

Architecture Details: We use recent, state-of-the-art, network architectures for all experiments and comparisons. For the still-image *teacher* networks, we use the ResNet-50 [20] architecture, trained on different image datasets such ImageNet 1K [40] and Places 365 [60]. For the spatiotemporal (video) *student* architectures, we first experiment with a variant of the Res3D [50] architecture. Res3D is an improved version of the popular C3D [49] using residual connections. We denote a N-layer Res3D model as Res3D-N, which is compatible with the standard ResNet-N [20] architecture. Since there is a one-to-one correspondence between such 2D and 3D models, the 3D models can also be initialized by *inflating* the learned weights from 2D models (e.g., for each channel, replicate the 2D filter weights along the temporal dimension to produce a 3D convolutional filter). Similar ideas of inflating 2D models to 3D have been proposed previously for Inception-style architectures [3], along with initialization techniques from corresponding 2D models [3, 9, 10]. The existence of a 1-to-1 mapping between the 2D and 3D models used in our experiments allows us to compare our approach to the method of inflation for initialization. However, we stress that unlike inflation, our method is applicable even in scenarios where such 1-to-1 mapping does not hold.

(2+1)D CNNs: More recently, full 3D models have been superseded by (2+1)D architectures [51], where each 3D kernel is decomposed into a 2D spatial component followed by a 1D temporal filter. Similar models have also been proposed previously [47], and are also known as P3D [38] or S3D [56] architectures. These models have proven to be more efficient, with much fewer parameters, and more effective on various standard benchmarks [51, 56]. However, these models no longer conform to standard 2D architectures because they contain additional `conv` and `batch_norm` layers that extend over time. These parameters do not exist in corresponding 2D models and so cannot be initialized with images. Nevertheless, our distillation remains applicable even in this scenario. In this work, we refer to such networks using R(2+1)D-N notation, for N-deep architecture.

Implementation Details: For all experiments, we use the hyperparameter values described in [51]. For distillation pre-training, we use the hyper-parameter setup for "Kinetics from-scratch training." We use distributed Sync-SGD [17] over 16×4 GPUs, starting with LR=0.01, and dropping it by $10\times$ every 10 epochs. Weight decay is set to 10^{-4}. We train for a total of 45 epochs, where each epoch is defined as 1M iterations. The video model is learned on 8-frames clips of 112 pixels. The network has depth of 18, which enables faster experimentation compared to the best model reported in [51] which uses 32 frames and has a depth of 34 layers. The batch size used for Kinetics training is 32/GPU, which we reduce to 24/GPU to accommodate the additional memory requirements for the teacher networks. For the fine-tuning experiment on smaller datasets like HMDB, we use Sync-SGD with 8×2 GPUs, starting with LR=0.002, an dropping it by $10\times$ every 2 epochs. The weight decay is set to 5×10^{-3}. We train 8 epochs, with each epoch defined as 40K steps. When training from scratch, we use initial LR of 0.01 with a step every 10 epochs, trained for total of 45 epochs.

4. Experiments

We now experimentally evaluate our system. We start by introducing the datasets and benchmarks used for training and evaluation in Section 4.1. We then compare DistInit with inflating 2D models for initialization in Section 4.2. Next we ablate the various design choices in DistInit in Section 4.3, and finally compare to previous state of the art on UCF-101 [46] and HMDB-51 [28] in Section 4.4.

4.1. Datasets and Evaluation

Our method involves two stages, as typical in video understanding literature [3]: pre-training on a large, unlabeled corpus of videos using still-image models as "teachers", followed by fine-tuning on the training split of a labeled target dataset ('test bed'). After training, we evaluate the performance on the test set of the target dataset.

Unlabeled source videos: We experiment with a variety of different unlabeled video corpuses in Section 4.3, including Kinetics [25], Sports1M [24] and an internal set of videos. While some of these datasets do come with semantic (action) labels, we ignore such annotations and only use the raw videos. Kinetics and Sports1M contain about 300K and 1.1M videos, respectively. In this work, we drop any labels these datasets come with, and only use the videos as a large, unlabeled corpus to train video representations. The internal video set includes 1M videos without any semantic labels and randomly sampled from a larger collection. We use these diverse datasets to show that our method is not limited to any form of data curation, and can work with truly in-the-wild videos.

Target test videos: HMDB-51 [28] contains 6766 realistic and varied video clips from 51 action classes. Evaluation is performed using average classification accuracy over three train/test splits from [23], each with 3570 train and 1530 test videos. UCF-101 [46] consists of 13320 sports video clips from 101 action classes, also evaluated using average classification accuracy over 3 splits. We use the HMDB split 1 for ablative experiments, and report the final performance on all splits for HMDB and UCF in Section 4.4.

4.2. DistInit vs Inflation

Inflation: We first compare our proposed approach to inflation [3, 9], i.e., initializing video models from 2D models by inflating 2D kernels to 3D via replication over time. Note that inflation is constrained to operate on 3D models that have a one-to-one correspondence with the 2D model. Hence, we use a Res3D-18 model, which is compatible for direct inflation from ResNet-18 models. We experiment with publicly available ImageNet and PlaceNet models. We compare it with our distillation approach in Table 1, trained using an ImageNet pretrained model as the teacher. Distillation improves performance by 15% over a model trained from scratch, and 4% over a model trained with inflated weights (the current best-practice for training such models).

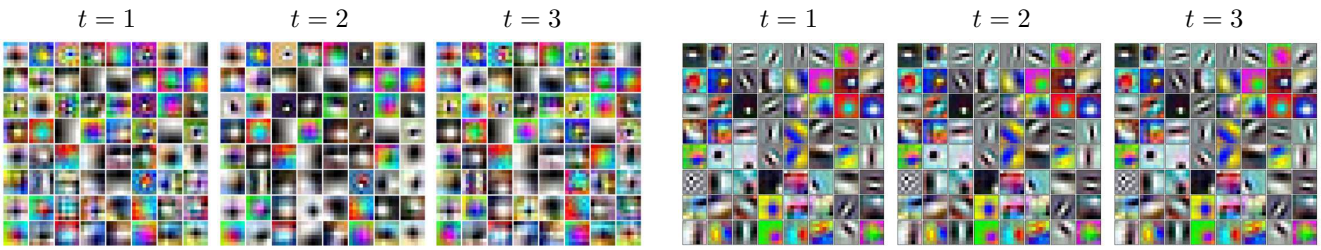
Model	Initialization	Per clip	Top 1	Top 5
Res3D-18	Scratch	24.6	25.4	55.2
Res3D-18	ImageNet inflated	32.5	35.8	66.2
Res3D-18	PlaceNet inflated	32.5	35.6	66.2
Res3D-18	DistInit (ours)	**36.6**	**39.9**	**73.5**
R(2+1)D-18	Scratch	22.0	24.1	53.1
R(2+1)D-18	DistInit (ours)	**37.8**	**40.3**	**74.4**
R(2+1)D-18	Kinetics pre-training	-	51.0	-

Table 1: **Distillation vs Inflation.** As described in Section 4.2, our distillation approach outperforms training video models from scratch or initializing them by inflating 2D models. We evaluate using percentage accuracy on the HMDB-51 dataset, Split 1. The models used are 18-layer Res3D and R(2+1)D, over 8-frame input, trained with cross-entropy loss (described in Section 4.3). The DistInit training is done using 2D network trained on ImageNet.

(2+1)D: More importantly, our approach can also be used to initialize state-of-the-art temporal architectures such as R(2+1)D [51], which do not have a natural 2D counterpart. In such a setting, the current best practice is to initialize such networks from scratch. Here, distillation improves performance by 16%. Finally, we also report the model trained using actual Kinetics labels, and as expected, that yields higher performance. Hence there is clear value to the explicit manual supervision provided in such large-scale datasets, but distillation appears to get us "half-way" there.

Visualizations: At this point, it is natural to ask why the distilled model outperforms current best-practices such as inflation? We visualize the learned representation by plotting the first layer `conv` filters in Figure 3. It can be seen that our distilled model learns truly spatiotemporal filters that vary in time, whereas inflation simply *copies* the same filter over time. Such dynamic temporal variation is readily present in the videos used for distillation, even when they are not labelled with spatiotemporal action categories. Filters pre-trained with inflation initialization never see actual video data, and so cannot encode such variation. In Figure 4 we also compare the filters learned by our R(2+1)D model via distillation vs via fully-supervised training. Our filters look quite similar to those learned through supervised learning, showing the effectiveness of our approach. In some sense, the improved performance of distillation can be readily explained by more data – networks learning from scratch see no data for pre-training, inflation networks see ImageNet, while distilled networks see both Imagenet and unlabeled videos. Our practical observation is that one can use image-based teachers to pre-train on massively large, unlabeled video datasets.

We can also analyze the effectiveness of distillation pre-training, by visualizing the correlation of the representation



(a) Ours (b) ImageNet inflated

Figure 3: **Learned Res3D filters.** We compare the learned first layer representation using our distillation approach, to inflation. For each, we show the 64 `conv_1` filters, for each time instance of the filter. As described in Section 4.2, our filters change in value over time, indicating that they learn to look for some amount of temporal dynamics in the input video. This clearly contrasts with ImageNet-inflated filters, which are *exact* copies over time, and so do not respond to any temporal change in pixel values.

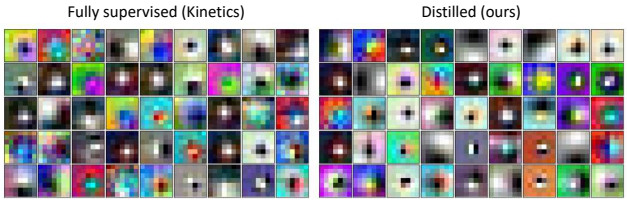


Figure 4: **Learned R(2+1)D filters.** Similar to Figure 3, we show the first layer conv filters for the R(2+1)D models. Note that 2.5D conv layer contains a 2D convolution in space followed by 1D convolution in time, and in this visualization we are only showing the former, i.e. the 45 2D conv filters that operate on the RGB image. We observe that our distillation approach learns spatiotemporal representations that are relatively similar to the fully supervised model (compared to filters learned from Imagenet in Fig. 3).

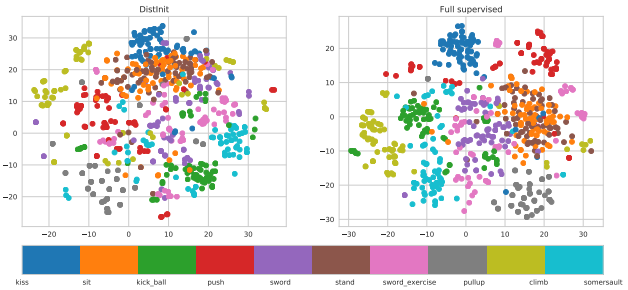


Figure 5: **Learned high-level representation.** While the filter maps in Figure 3 and 4 can be used to interpret the low-level representation learned by our model, we now try to probe the high level representation by visualizing the last layer features. This figure shows tSNE [52] visualization of averaged last layer features from the model trained with DistInit, and trained with full Kinetics supervision. Each dot represents a video from HMDB training set, and is color coded by the class of that video. For ease of visualization, we picked 10 **random** classes to plot. Note that DistInit is already able to segregate many videos into clusters correlated with their action classes, without ever being trained on any action labels! The fully supervised model naturally does better as it has been trained on a large action dataset, Kinetics. This further suggests DistInit leads to useful representation for classifying actions.

we learn with the classes in the task of action recognition. As explained in Figure 5, we can see that the last layer features for the same action class tend to cluster together when projected to 2D using tSNE [52].

4.3. Diagnostic Analysis

Design choices for the teacher network: We now ablate the design choices for the teacher networks. Teacher networks are required to generate a target label to supervise the video model being trained, by using images from the video clip. We experiment with picking the center frame, a random frame, or multiple random frames from the clip to compute the targets. In case of multiple frames, we average the logits before passing them through softmax to generate the target distribution. We compare these methods in Table 2, and observe higher performance when picking frames randomly. This improvement can be due to less overfitting through label augmentation. We use it in our final model.

Distillation loss: Next, we evaluate the different choices for the loss function in distillation. As already explained in Section 3, previous work has suggested different loss functions for distillation tasks. We compare two popular approaches: KL divergence over distribution and l_2 loss over logits. In the case of the former, we compute the softmax distribution from the teacher networks, as well as from the student branch that attempts to match that teacher, and use a cross entropy between the two softmax distributions as the

Model	Pick strategy	Per clip	Top 1	Top 5
R(2+1)D-18	Center	37.8	40.3	74.4
R(2+1)D-18	Random	39.9	43.2	73.9
R(2+1)D-18	2 Random	39.6	**44.0**	73.5

Table 2: **Video to Image.** We compare different strategies of converting the video into image(s) for extracting the target label. We find strongest performance when picking random frames to generate the target distribution. Model used here is 18-layer R(2+1)D, over 8-frame input, trained with cross-entropy loss (Section 4.3); evaluated using percentage accuracy on HMDB-51 split 1.

Loss function	Per clip	Top 1	Top 5
cross-entropy (over softmax)	37.8	40.3	74.4
mean squared error (over logits)	35.6	39.9	70.5

Table 3: **Loss function for distillation.** We compare different loss functions for distillation, and find that the performance was relatively stable with different choices. The model used here is a 18-layer R(2+1)D, over 8-frame input, evaluated using percentage accuracy on HMDB-51 split 1.

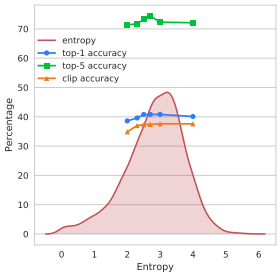


Figure 6: **Accuracy variation with entropy.** Our results suggest that if we use DistInit only on videos for which the teacher is sufficiently sure of its predictions (entropy $<$ 2.7), we obtain slightly better performance while ignoring 50% of the input videos, making training faster.

objective to optimize. We find this objective can be well optimized using the standard hyper-parameter setup used for Kinetics training in [51]. In the case of the latter, we skip the softmax normalization step and directly compute the mean squared error between the last linear layers as the objective. Since the initial loss values are much higher, we needed to drop the learning rate by a factor of 10 to optimize this model, with all the other parameters kept the same. As Table 3 shows, we observe nearly similar downstream performance with both.

Selecting confident predictions: As some recent work [39] has shown, distillation techniques can benefit from using only the most confident predictions for training the student. We use the entropy of the predictions from the teachers as a notion of their confidence. We implement this confidence thresholding by setting a zero weight for the loss on each example, for which the teacher is not confident, or has high-entropy predictions; effectively dropping parts of the training data that are confusing for the teacher. We show the performance on dropping different amounts of data in Figure 6. The red curve shows a kernel density estimate (a PDF) of entropy values for an ImageNet teacher on the Kinetics data. At any given entropy value (e), it shows the relative likelihood of a data point to have that entropy value, and $\int_{-\infty}^{e} f(x)dx$ (area under the curve from $-\infty$ to e) is the percentage of data with entropy $\leq e$. We experiment with setting different thresholds for dropping the low-confidence data points during DistInit, and show the downstream HMDB-51 split-1 performance in the line plots. We found slightly better performance, even after dropping nearly half the data, making training faster.

Varying the unlabeled dataset: We now try to evaluate whether our method is dependent on any specific video data source, and if it can benefit from additional data sources. We evaluate this in Table 4 and observe nearly similar performance when using different sets of videos (without labels) like Kinetics [25] and Sports1M [24]. We also experiment with an internal set of videos downloaded from the web, and still get strong DistInit performance. This shows our method is not limited to any form of data curation, and can learn from truly in-the-wild videos.

Model	Unlabeled set	Size	Per clip	Top 1	Top 5
R(2+1)D-18	Kinetics [25]	0.3M	37.8	40.3	74.4
R(2+1)D-18	Sports1M [24]	1.1M	37.5	39.9	73.3
R(2+1)D-18	Kinetics+Sports1M	1.4M	38.0	41.8	75.3
R(2+1)D-18	Internal videos	1.0M	38.2	41.2	72.0

Table 4: **Unlabeled data for distillation.** This table shows that our model is not limited to any specific source of unlabeled data, and can also benefit from multiple sources of data. Size denotes the number of unlabeled videos used from that set. Performance reported on HMDB-51 split 1.

Model	Initialization	Per clip	Top 1	Top 5
Res3D-18	Scratch	30.7	38.7	70.0
Res3D-18	ImageNet mean inflated	33.5	43.9	73.9
R(2+1)D-18	DistInit (ours)	**42.6**	**49.2**	**81.2**

Table 5: **DistInit on Optical Flow.** This table shows that our model is also applicable to other modalities, like optical flow. Note that the inflated initialization for the first layer (`conv_1`) was performed by averaging the kernel on channel dimension, and then replicating it two times. Reported on HMDB-51 split 1.

Using other teachers: Just as our model is capable of learning from more data, our model is also capable of using diverse supervision. We experiment with replacing the ImageNet teacher with a model trained on PlaceNet [60], and obtain 36.8% HMDB fine-tuning performance as opposed to 40.3% before with ImageNet. Apart from the fact that our model can learn from diverse sources of supervision, this result shows that objects $\rightarrow$ actions semantic transfer is more effective than scenes $\rightarrow$ actions. This makes sense as human actions are typically informed more by the objects in their environment, than the environment itself. We also tried training with both ImageNet and PlaceNet teachers jointly, and obtained a top-1 accuracy of 40.7%, suggesting that there is little benefit of adding scene cues (from Places) given object information (from ImageNet). However, teachers from unrelated domains are likely to provide more complementary information and lead to higher improvements.

Different input modalities: One of the biggest advantages of our method is that it is applicable to learn representations for any arbitrary input data modality. We experiment with optical flow, which still contributes to significant performance improvements on video tasks, even with modern video architectures, across different datasets [3]. Previous work [3, 55] has used ImageNet initialization for networks

Model	Architecture	#frames	Pre-training	Split 1	3-split avg
Misra *et al.* [34]	AlexNet [27]	1	Scratch	-	13.3
Misra *et al.* [34]	AlexNet [27]	1	Tuple verify [34]	-	18.1
Misra *et al.* [34]	AlexNet [27]	1	ImageNet	-	28.5
Two-stream (RGB) [11, 44]	VGG-M [45]	1	ImageNet	-	40.5
C3D [3]	Custom	16	Scratch	24.3	-
LSTM [3]	BN-Inception [22]	-	ImageNet	36.0	-
Two stream (RGB) [3]	BN-Inception [22]	1	ImageNet	43.2	-
I3D (RGB) [3]	BN-Inception [22]	64	ImageNet	49.8	-
Ours (RGB)	R(2+1)D-18 [51]	32	DistInit	**54.9**	**54.8**

(a) HMDB-51

Model	Architecture	#frames	Pre-training	Split 1	3-split avg
Misra *et al.* [34]	AlexNet [27]	1	Scratch	-	38.6
Misra *et al.* [34]	AlexNet [27]	1	Tuple verification [34]	-	50.2
Two-stream (RGB) [11, 44]	VGG-M [45]	1	ImageNet	-	73.0
C3D [3]	Custom	16	Scratch	51.6	-
LSTM [3]	BN-Inception [22]	-	ImageNet	81.0	-
Two stream (RGB) [3]	BN-Inception [22]	1	ImageNet	83.6	-
I3D (RGB) [3]	BN-Inception [22]	64	ImageNet	84.5	-
STC [5]	3D-ResNet	16	Knowledge Tx	82.1	-
STC [5]	STC-ResNext [5]	16	Knowledge Tx	84.7	-
Ours (RGB)	R(2+1)D-18 [51]	32	DistInit	**85.7**	**85.8**

(b) UCF-101

Table 6: **Comparison with previous work on HMDB and UCF.** We split the tables based on the base architecture for fair comparison. In the first section, we report architectures with comparable depth as ours, and in the second we report other approaches using deeper architectures. Our model out-performs all these previous methods. Note that we do not compare to Kinetics pre-trained models. Using Kinetics for pre-training with I3D [3] gets 74.3% and 95.1% 3-split avg on HMDB and UCF, but is not comparable to our unsupervised approach which does not use those labels.

accepting flow as input. This is far from ideal since flow has much different statistics than RGB images. DistInit, on the other hand, is agnostic to the input data modality of the student network. We train the student network to learn from the input flow modality, while the teacher uses a random RGB frame from the same clip to generate the distillation target. As we show in Table 5, DistInit still produces strong initialization and improves over training from scratch or the ImageNet inflated initialization. However, due to high computational cost of computing flow, we ignore this input modality for the final comparisons.

4.4. Comparison with previous work

Finally, in Table 6 we compare our model to other standard models and initialization methods on HMDB and UCF. For these comparisons, we use the 32-frames model, tested using dense predictions (instead of uniformly sampled 10-clips) for each testing video. Here we only compare to RGB-based models for computational speed, though our approach is applicable to flow or other modalities as shown in Section 4.3. We obtain strong performance compared to standard methods, and other unsupervised feature learning techniques [5, 34]. Finally, in Figure 7 we show which classes benefit the most from the initialization provided by DistInit compared to that computed by inflation.

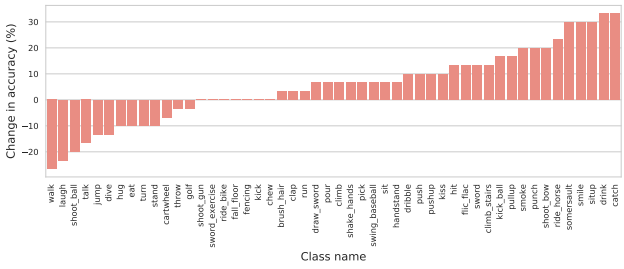


Figure 7: **HMDB classes with largest gain using DistInit instead of inflation.** The plot shows HMDB per-class accuracy difference between the two finetuned models tested. It suggests that our method is most useful for classes that require understanding motion, such as "catching" and "smiling." On the other hand, classes like "shoot_ball" or "eat" are easy to recognize from single frames.

5. Conclusion

We describe a simple approach to transfer knowledge from image-based datasets labeled for object or scene recognition tasks, to learn spatiotemporal video models for human action recognition tasks. Much previous work has addressed this problem by constraining spatiotemporal architectures to match 2D counterparts, limiting the choice of networks that can be explored. We describe a simple approach, DistInit, based on distillation that can be used to initialize any spatiotemporal architecture. It does so by making use of image-based teachers that can leverage considerable knowledge about objects, scenes, and potentially other semantics (e.g., attributes, pose) encoded in richly-annotated image datasets. Unlike previous unsupervised learning works that depend on the curated ImageNet dataset, albeit without labels, we show our model even works on truly in-the-wild uncurated videos. We demonstrate significant improvements over standard best practices for initializing spatiotemporal models. That said, our results do not match the accuracy of models pretrained on recently-introduced, large-scale *supervised* video datasets. But we note that these were collected and annotated with significant manual effort. Because our approach requires only *unsupervised* videos, it has the potential to make use of massively-larger data for learning accurate video models.

Acknowledgments: RG and DR were partly supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number D17PC00345. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright annotation theron. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied of IARPA, DOI/IBC or the U.S. Government.

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. 2

[2] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *KDD*, 2006. 4

[3] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1, 2, 4, 5, 7, 8

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1, 2

[5] Ali Diba, Mohsen Fayyaz, Vivek Sharma, M Mahdi Arzani, Rahman Yousefzadeh, Juergen Gall, and Luc Van Gool. Spatio-temporal channel correlation networks for action classification. In *ECCV*, 2018. 3, 8

[6] Ali Diba, Vivek Sharma, and Luc Van Gool. Deep temporal linear encoding networks. In *CVPR*, 2017. 3

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

[8] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Subhashini Venugopalan Marcus Rohrbach, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 1

[9] Christoph Feichtenhofer, Alex Pinz, and Richard Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016. 1, 2, 4, 5

[10] Christoph Feichtenhofer, Alex Pinz, and Richard P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *CVPR*, 2017. 2, 4

[11] Christoph Feichtenhofer, Karen Simonyan, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition (online). http://www.robots.ox.ac.uk/~vgg/software/two_stream_action/. 8

[12] David F. Fouhey, Weicheng Kuo, Alexei A. Efros, and Jitendra Malik. From lifestyle VLOGs to everyday interactions. In *CVPR*, 2018. 2

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

[14] Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. A Better Baseline for AVA. *arXiv preprint arXiv:1807.10066*, 2018. 3

[15] Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. Video Action Transformer Network. In *CVPR*, 2019. 1

[16] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. ActionVLAD: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017. 1

[17] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 4

[18] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *CVPR*, 2016. 3

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 4

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3, 4

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 8

[23] Yu-Gang Jiang, Jingen Liu, Amir Roshan Zamir, Ivan Laptev, Massimo Piccardi, Mubarak Shah, and Rahul Sukthankar. THUMOS challenge: Action recognition with a large number of classes. http://www.thumos.info/, 2013. 5

[24] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 1, 2, 5, 7

[25] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 1, 2, 5, 7

[26] Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. In *ICLR*, 2016. 1

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 8

[28] Hilde Kuehne, Hueihan Jhuang, Estibaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 2, 5

[29] Ivan Laptev. On space-time interest points. *IJCV*, 2005. 1, 2

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015. 1

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *IEEE*, 1998. 1

[32] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 1, 2

[33] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 1, 2

[34] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and Learn: Unsupervised Learning using Temporal Order Verification. In *ECCV*, 2016. 2, 8

[35] Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, and Hamed Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *CVPR*, 2018. 2

[36] Paul Over, Jon Fiscus, Greg Sanders, David Joy, Martial Michel, George Awad, Alan Smeaton, Wessel Kraaij, and Georges Quénot. Trecvid 2014–an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *TRECVID*, 2014. 2

[37] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis.

In *IJCAI*, 2009. 3

[38] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017. 4

[39] Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. In *CVPR*, 2018. 7

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 1, 2, 4

[41] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010. 3

[42] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, 2018. 2

[43] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016. 2

[44] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1, 2, 3, 8

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2, 8

[46] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CRCV-TR-12-01*, 2012. 2, 5

[47] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015. 4

[48] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. In *ICLR Workshops*, 2016. 2

[49] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 2, 4

[50] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*, 2017. 4

[51] 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. 1, 2, 4, 5, 7, 8

[52] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing high-dimensional data using t-sne. *JMLR*, 2008. 6

[53] Heng Wang, Alexander Kläser, Cordelia Schmid, and Liu Cheng-Lin. Action Recognition by Dense Trajectories. In *CVPR*, 2011. 2

[54] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 2

[55] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 2, 3, 7

[56] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. *arXiv preprint arXiv:1712.04851*, 2017. 4

[57] Yang Yang, Imran Saleemi, and Mubarak Shah. Discovering motion primitives for unsupervised grouping and one-shot learning of human actions, gestures, and expressions. *TPAMI*, 2013. 2

[58] Amir R. Zamir, Alexander Sax, William Shen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018. 3

[59] Hang Zhao, Zhicheng Yan, Heng Wang, Lorenzo Torresani, and Antonio Torralba. SLAC: A sparsely labeled dataset for action classification and localization. *arXiv preprint arXiv:1712.09374*, 2017. 2

[60] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million Image Database for Scene Recognition. *TPAMI*, 2017. 1, 2, 4, 7