Rethinking Zero-shot Video Classification: End-to-end Training for Realistic Applications

Biagio Brattoli*
Heidelberg University
biagio.brattoli@iwr.uni-heidelberg.de

Joe Tighe
Amazon
tighe@amazon.com

Fedor Zhdanov
Amazon
fedor@amazon.com

Pietro Perona
Amazon
perona@caltech.edu

Krzysztof Chalupka
Amazon
chalupkk@amazon.com

Abstract

Trained on large datasets, deep learning (DL) can accurately classify videos into hundreds of diverse classes. However, video data is expensive to annotate. Zero-shot learning (ZSL) proposes one solution to this problem. ZSL trains a model once, and generalizes to new tasks whose classes are not present in the training dataset. We propose the first end-to-end algorithm for ZSL in video classification. Our training procedure builds on insights from recent video classification literature and uses a trainable 3D CNN to learn the visual features. This is in contrast to previous video ZSL methods, which use pretrained feature extractors. We also extend the current benchmarking paradigm: Previous techniques aim to make the test task unknown at training time but fall short of this goal. We encourage domain shift across training and test data and disallow tailoring a ZSL model to a specific test dataset. We outperform the state-of-the-art by a wide margin. Our code, evaluation procedure and model weights are available at github.com/bbrattoli/ZeroShotVideoClassification.

1. Introduction

Training image and video classification algorithms requires large training datasets [17, 23, 46, 47, 48]. With no task-specific training data available one may still attempt to train a model using related information and transfer the learned knowledge to classify previously unseen categories. This approach is called zero-shot learning (ZSL) [25, 30] and it is quite successful in the image domain [37, 39, 40, 42, 51].

We focus on ZSL for video action recognition, where data sourcing and annotation is particularly expensive.

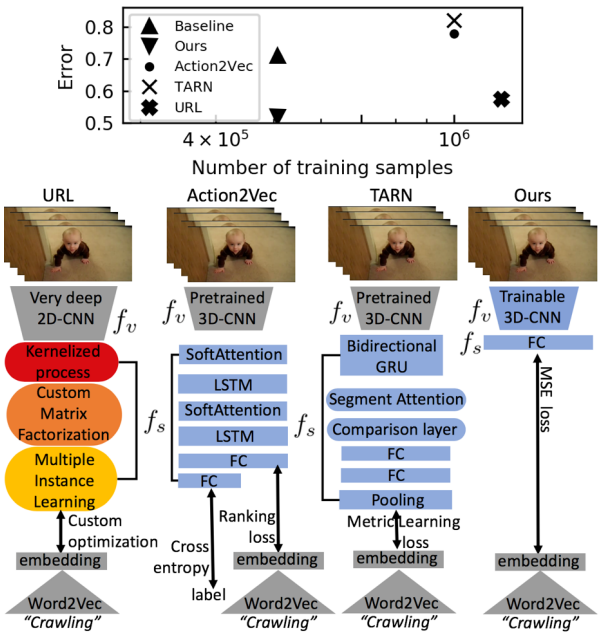

Figure 1: (Top) Our model is state-of-the-art (error computed on the UCF test dataset.) (Bottom) Our e2e model is simple but powerful. URL [59], Action2Vec [14] and TARN [4] are state-of-the-art approaches. Gray blocks represent modules fixed during training. Colors (blue, red, orange, yellow) indicate modules trained in separate stages.

Since the set of possible human actions is huge, action recognition is a great ZSL testbed. Trained on large-scale academic datasets [11, 13, 20, 21, 24, 45], supervised 3D convolutional neural networks (CNNs) proved successful in this domain [12, 46, 47]. How well modern deep networks can recognize human actions in the ZSL setting is, however, an open question.

To our knowledge, all current ZSL methods for video

*Work done during an internship at Amazon.

recognition use pretrained visual embeddings [1, 4, 14, 29, 31, 49, 50, 53, 54, 55, 56, 59]. This provides a good trade-off between training efficiency and using prior knowledge. Shallow trainable models then convert the pretrained representations to ZSL embeddings, as shown in Fig. 1 (Bottom). Low training space complexity of shallow models allows them to benefit from long video sequences [46] and large feature extractors [17].

In contrast, state-of-the-art algorithms in the fundamental CV domains of image classification [17], object detection [32, 34, 44] and segmentation [8, 16, 58] all rely on end-to-end (e2e) training. Representation learning is at the core of deep networks' success across machine learning domains [3], and deeper models can better utilize information available in large datasets [2, 17]. This poses a question: How can an e2e ZSL compete with current methods?

Our contributions involve multiple aspects of ZSL video classification:

Novel Modeling: We propose the first e2e-trained model for zero-shot action recognition. The training procedure is inspired by modern supervised video classification practices. Fig. 1 shows that our method is simple, yet outperforms previous work. Moreover, we devise a novel easy pretraining technique that targets the ZSL scenario for video recognition.

Evaluation Protocol: We propose a novel ZSL training and evaluation protocol that enforces a realistic ZSL setting. Extending the work of Roitberg *et al*. [36], we test a single trained model on multiple test datasets, where sets of training and test classes are disjoint. In addition, we argue that training and test domains should not be identical.

In-depth Analysis: We perform an in-depth analysis of the e2e model and a pretrained baseline. In a series of guided experiments we explore the characteristics of good ZSL datasets.

Our model, training and evaluation code, are available at github.com/bbrattoli/ZeroShotVideoClassification.

2. Related work

We focus on *inductive* ZSL in which test data is fully unknown at training time. There exists a body of literature on *transductive* ZSL [1, 29, 49, 50, 54, 53, 55], where test images or videos are available during training but test labels are not. We do not discuss the transductive approach in this work.

Video classification: Modern, DL-based video classification methods fall largely into two categories: 2D networks [43, 48] that operate on 1-5 frame snippets and 3D networks [5, 6, 7, 12, 15, 27, 41, 46, 47] that operate on 16-128 frames. One of the earliest works of this type, Simonyan and Zisserman [43], trained with only 1-5 frames sampled randomly from the video. At inference many more

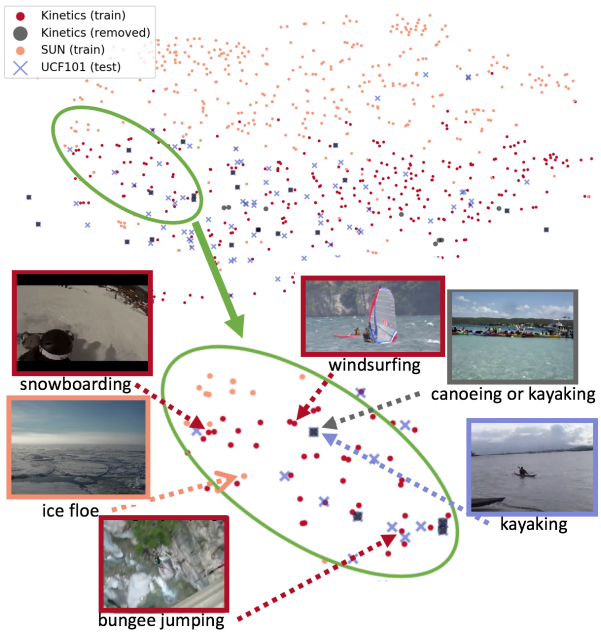

Figure 2: Training and test classes, t-SNE [26] visualization of Word2Vec embeddings. Red dots represent training classes we used, and gray dots training classes we removed in order to separate training and test data. Crosses represent test classes. Pictures are actual dataset videoframes.

frames were sampled and the classifier outputs were averaged across all samples taken for a video clip. This implied that looking at a large chunk of the video was important during inference but wasn't strictly required during training. Wang *et al*. [48] showed that sampling multiple frames throughout the video during training could improve performance, opening the question whether training also requires a large temporal context. However, a body of later work based on more powerful 3D networks [7, 12, 46] showed that for most datasets sampling 16 frames during training is sufficient. Increasing training frame count from 16 to 128 improved performance only marginally.

In this work, we adapt the training-time sampling philosophy of state-of-the-art video classification to the ZSL setup. This allows us to train the visual embedding e2e. As a consequence, the overall architecture and inference procedure are very simple compared to previous work, and the results are state-of-the-art – as shown in Fig. 1.

Zero shot video classification: The common practice in zero-shot video classification is to first extract visual features from video frames using a pretrained network such as C3D [46] or ResNet [17], then trains a temporal model that maps the visual embedding to a semantic embedding space [4, 14, 31, 56, 59]. Good generalization on semantic embeddings of class names means that the model can be applied to new videos where the possible output classes are

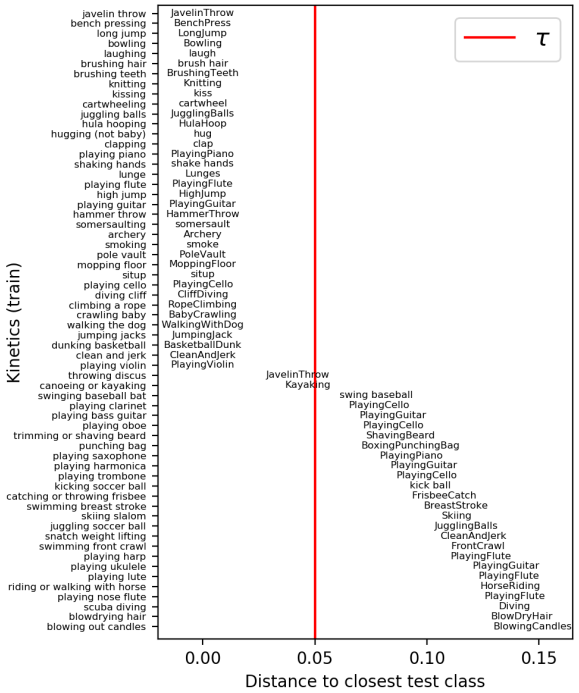

Figure 3: Removing overlapping training and test classes. The y-axis shows Kinetics classes closest to the test sets UCF and HMDB. x-axis shows the distance (see Eq. 4) of the corresponding closest test class. In our experiments, we removed training classes closer than $\tau = 0.05$ to the test set – to the left of the red line in the figure.

not present in training data. Inference reduces to finding the test class whose embedding is the nearest-neighbor of the model's output. Word2Vec [28] is commonly used to produce the ground-truth word embeddings. An alternative approach is to use manually crafted class attributes [19]. We decided not to pursue the manual approach as it harder to apply in general scenarios.

Two effective recent methods, Hahn *et al*. [14] and Bishay *et al*. [4], extract C3D features from 52 clips of 16 frames from each video. They then learn a recurrent neural network [10, 18] to encode the result as a single vector. Finally, a fully connected layer maps the encoded video into Word2Vec embedding. Fig. 1 illustrates this approach. Both [14] and [4] use the same dataset for training and testing, after splitting the available dataset classes into two sets. Using a pretrained deep network is convenient because pre-extracted visual features easily fit in GPU memory, even for a large number of video frames. Alternative approaches use generative models to compensate for the gap between semantic and visual distributions [29, 57]. Unfortunately, performance is limited by the inability to fine-tune the visual embedding. We show fine-tuning is crucial to generalize across datasets.

Our work is similar to Zhu *et al*. [59] in that both methods learn a universal action representation that generalizes across datasets. However, their proposed model does not leverage the potential of 3D CNNs. Instead, they utilize the very deep ResNet200 [17], pretrained on ImageNet [9, 38], which cannot utilize temporal information.

As pointed out by Roitberg *et al*. [36], previous works train their models on actions overlapping with those of the the target dataset, violating ZSL assumptions. For example, Zhu *et al*. [59] train on the full ActivityNet [11] dataset. This makes their results difficult to fairly compare with ours. Under our definition of ZSL (Sec. 3.3), Zhu *et al*. have 23 classes in their training datasets that overlap with the test dataset. The situation is similar for all other methods to varying degrees.

3. Zero-shot action classification

We first carefully define ZSL in the context of video classification. This will allow us to propose not only a new ZSL algorithm, but also a clear evaluation protocol that we hope will direct future research towards practical ZSL solutions. We stay within the inductive setting, as described in Sec. 2.

3.1. Problem setting

A video classification task is defined by a training set (source) $D_s = \{(x_1, c_1), \cdots, (x_{N_s}, c_{N_s})\}$ consisting of pairs of videos x and their class labels c, and a video-label test set D_t. In addition, previous work often uses pretraining datasets D_p as explained in Sec. 2.

Intuitively, ZSL is any procedure for training a classification model on D_s (and possibly D_p) and then testing on D_t *where* D_t *does not overlap with* $D_s \cup D_p$. How this overlap is defined varies. Sec. 3.3 proposes a definition that is more restrictive than those used by previous work, and forces the algorithms into a more realistic ZSL setting.

ZSL classifiers need to generalize to unseen test classes. One way to achieve this is using nearest-neighbor search in a semantic class embedding space.

Formally, given a video x, we infer the corresponding semantic embedding $z = g(x)$ and classify x as the nearest-neighbor of z in the set of embeddings of the test classes. Then, a trained classification model $M(\cdot)$ outputs

$$M(x) = \underset{c \in D_t}{\operatorname{argmin}} \cos\left(g(x), \mathrm{W2V}(c)\right). \tag{1}$$

where cos is the cosine distance and the semantic embedding is computed using the Word2Vec function [28] $\mathrm{W2V}: \mathcal{C} \to \mathbb{R}^{300}$.

The function $g = f_s \circ f_v$ is a composition of a visual encoder $f_v: x \mapsto y$ and a semantic encoder $f_s: y \mapsto z \in \mathbb{R}^{300}$.

3.2. End-to-end training

In previous work, the visual embedding function f_v is either hand-crafted [55, 59] or computed by a pretrained deep network [4, 14, 50, 59]. It is fixed during optimization, forcing model development to focus on improving f_s. Resulting models need to learn to transform fixed visual embeddings into meaningful semantic features and can be very complex, as shown in Fig. 1 (Bottom).

Instead, we propose to optimize both f_v and f_s at the same time. Such e2e training offers multiple advantages:

1. Since f_v provides a complex computation engine, f_s can be a simple linear layer (see Fig. 1).
2. We can implement the full model using standard 3D CNNs.
3. Pretraining the visual embedding on a classification task is not necessary.

End-to-end optimization using the full video is unfeasible due to GPU memory limitations. Our implementation is based on standard video classification methods which are effective even when only a small snippet is used during training, as discussed in detail in Sec 2. Formally, given a training video/class pair $(x, c) \in D_s$ we extract a snippet x^t of 16 frames at a random time $t \leq (\text{len}(x) - 16)$. The network is optimized by minimizing the loss

$$L = \sum_{(x,c)\in D_s} \|\text{W2V}(c) - (f_s \circ f_v)(x^t)\|^2. \qquad (2)$$

Inference procedure is similar but pools information from multiple snippets following Wang *et al*. [48]. Sec. 4.4 details both our training and inference procedures.

To better understand our method's performance under various experimental conditions, we implemented a baseline model that uses identical f_s, f_v and training data, but fixes f_v's weights to values pretrained on the classification task (available out-of-the-box in the most recent PyTorch implementation, see Sec. 4.4). This was necessary since we were not able to access implementations of any of the state-of-the-art methods ([4, 14, 59]). Unfortunately, our own re-implementations achieved results far below numbers reported by their authors, even with their assistance.

3.3. Towards realistic ZSL

To ensure that our ZSL setting is realistic, we extend the methods of [36] that carefully separates training and test data. This is cumbersome to achieve in practice, and has not been attempted by most previous work. We hope that our clear formulation of the training and evaluation protocols will make it easy for future researchers to understand the performance of their models in true ZSL scenarios.

Non-overlapping training and test classes: Our first goal is to make sure that $D_s \cup D_p$ and D_t have "non-overlapping classes". The simple solution – to remove source class names from target classes or *vice-versa* – does not work, because two classes with slightly different names can easily refer to the same concept, as shown in Fig. 3. A distance between class names is needed. Equipped with such a metric, we can make sure training and test classes are not too similar. Formally, let $d\colon \mathcal{C} \to \mathcal{C}$ denote a distance metric on the space of all possible class names $\mathcal{C}$, and let $\tau \in \mathbb{R}$ denote a similarity threshold. A video classification task fully respects the zero-shot constraint if

$$\forall c_s \in D_s \cup D_p, \min_{c_t \in D_t} d(c_s, c_t) > \tau. \qquad (3)$$

A straightforward way to define d is using semantic embeddings of class names. We define the distance between two classes to be simply

$$d(c_1, c_2) = \cos(\text{W2V}(c_1), \text{W2V}(c_2)) \qquad (4)$$

where cos indicates cosine distance. This is consistent with the use of the cosine distance in the ZSL setting as we do in Eq. 1. Fig. 2 shows an embedding of training and test classes after we removed from Kinetics classes overlapping with test data using the procedure outlined above. Fig. 3 shows the distribution of distances between training and test classes in our datasets. There is a cliff between distances very close to 0 and larger than 0.1. In our expeirments we use $\tau = 0.05$ as a natural, unbiased threshold.

Different training and test video domains: We argue that video domains of $D_s \cup D_p$ and D_t should differ. In previous work, the standard evaluation protocol is to use one dataset for training and testing, using 10 random splits. This does not account for domain shifts that happen in real world scenarios due to data compression, camera artefacts, and so on. For this reason ZSL training and test datasets should ideally have disjoint video sources.

Multiple test datasets: A single ZSL model should perform well on multiple test datasets. As outlined above, previous works train and test anew for each available dataset (typically UCF and HMDB). In our experiments, training happens only once on the Kinetics dataset [21], and testing on all of UCF [45], HMDB [24] and ActivityNet [11].

3.4. Easy pretraining for video ZSL

In a real-world scenario a model is trained once and then deployed on diverse unseen test datasets. A large and diverse training dataset is crucial to achieve good performance. Ideally, the training dataset would be tailored to the general domain of inference – for example, a strong ZSL surveillance model to be deployed at multiple unknown locations would require a large surveillance and action recognition dataset.

Sourcing and labeling domain-specific video datasets is, however, very expensive. On the other hand, annotating images is considerably faster. Therefore, we designed a simple

Dataset	VisualFeat	UCF	HMDB	Activity
URL [59]	ResNet200	42.5	51.8	-
DataAug [55]	-	18.3	19.7	-
InfDem [35]	I3D	17.8	21.3	-
Bidirectional [50]	IDT	21.4	18.9	-
FairZSL [36]	-	-	23.1	-
TARN [4]	C3D	19	19.5	-
Action2Vec [14]	C3D	22.1	23.5	-
Ours(605classes)	C3D	41.5	25.0	24.8
Ours(664classes)	C3D	43.8	24.7	-
Ours(605classes)	R(2+1)D_18	44.1	29.8	**26.6**
Ours(664classes)	R(2+1)D_18	**48**	**32.7**	-

Table 1: Comparison with the state-of-the-art on standard benchmarks. We evaluate on half test classes following Evaluation Protocol 1 (Sec. 4.3). Ours(605classes) indicates we removed all training classes that overlap with UCF, HMDB, or ActivityNet. Ours(664classes) indicates we removed only training classes overlapping with UCF and HMDB. We outperform previous work in both scenarios. Sec. 2 argues that URL's results are not compatible with other works as their training and test sets overlap and their VisualFeat is an order of magnitude deeper.

Method	UCF		HMDB		Activity	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
URL [59]	34.2	-	-	-	-	-
664classes	**37.6**	**62.5**	**26.9**	**49.8**	-	-
605classes	35.3	60.6	24.8	44.0	**20.0**	**42.7**

Table 2: Evaluation on all test classes. In contrast to Table 1, here we report results of our method applied to all three test datasets using Evaluation Protocol 2 (Sec. 4.3). We applied a single model trained on classes dissimilar from all of UCF, HMDB and ActivityNet. Nevertheless, we outperform URL [59] on UCF101. URL authors do not report results on full HMDB51. Remaining previous work do not report results on neither full UCF101 nor full HMDB51.

dataset augmentation scheme which creates synthetic training videos from still images. Sec. 5 shows that pretraining our model using this dataset boosts performance, especially if available training data is small.

We convert images to videos using the Ken Burns effect: a sequence of crops moving around the image simulates video-like motion. Sec. 4.1 provides more details.

Our experiments focus on the action recognition domain. In action recognition (as well as in many other classification tasks), location and scenery of the video is strongly predictive of action category. Because of this we choose SUN [52], a standard scene recognition dataset. Fig. 2 shows the complete class embedding of our the scene dataset's class names.

4. Experimental setup

To facilitate reproducibility, we describe our training and evaluation protocols in detail. The protocols propose one way of training and evaluating ZSL models that is consistent with our definitions in Sec. 3.3.

4.1. Datasets

UCF101 [45] has 101 action classes primarily focused around sports, with 13320 videos sourced from YouTube. *HMDB51* [24] is divided into 51 human actions focused around sports and daily activities and contains 6767 videos sourced from commercial videos and YouTube. *ActivityNet* [11] contains 27,801 untrimmed videos divided in 200 classes focusing on daily activities with videos sourced using web search. We extracted only the labeled frames from each video. *Kinetics* [21] is the largest currently available action recognition dataset, covering a wide range of human activity. The first version of the dataset contains over 200K videos divided in 400 categories. The newest version has 700 classes for a total of 541624 videos sourced from YouTube. *SUN397* [52] (see Sec. 3.4) is a scene understanding image dataset. It contains 397 scene categories for a total of over 100K high-resolution images. We converted it to a simulated video dataset using the Ken Burns effect: To create a 16-frame video from an image, we randomly choose "start" and "end" crop locations (and crop sizes) in the image, and linearly interpolate to obtain 16 crops. Each of them are then resized to 112×112.

4.2. Training protocol

Our experiments in Sec. 5 use two training methods:

Training Protocol 1: Remove from Kinetics 700 all the classes whose distance to any class in UCF $\cup$ HMDB is smaller than τ (see Eq. 4). This results in a subset of Kinetics with 664 classes, which we call Kinetics 664. As explained in Sec. 3.3, this setting is already more restrictive than that of the previous methods, which train new models for each test dataset.

Training Protocol 2: Remove from Kinetics 700 all the classes whose distance to any class in UCF $\cup$ HMDB $\cup$ $ActivityNet$ is smaller than τ (see Eq. 4). This results in a subset of Kinetics with 605 classes which we call Kinetics 605. This setting is even more restrictive, but is closer to true ZSL. Our goal is to show that it is possible to train a single ZSL model that applies to multiple diverse test datasets.

Figure 2 shows a t-SNE projection of the semantic embeddings of all Kinetics 700 classes, as well as the 101 UCF

classes and the classes we removed to obtain Kinetics 664.

4.3. Evaluation protocol

We tested our model using two protocols: the first follows Sec. 3.3 to emulate a true ZSL setting, the second is compatible with previous work. Both Evaluation Protocols apply the same model to multiple test datasets.

Evaluation Protocol 1: In order to make our results comparable with previous work, we use the following procedure: Randomly choose half of the test dataset's classes, 50 for UCF and 25 for HMDB. Evaluate the classifier on that test set. Repeat ten times and average the results for each test dataset.

Evaluation Protocol 2: Previous work uses random training/test splits of UCF [45] and HMDB [24] to evaluate their algorithms. However, we train on a separate dataset Kinetics 664 / 605 and can test on full UCF and HMDB. This allows us to return more realistic accuracy scores. The evaluation protocol is simple: evaluate the classifier on all 101 UCF classes and all 51 HMDB classes.

4.4. Implementation details

In our experiments, f_v (see Sec. 3.1) is the PyTorch implementation of R(2+1)D_18 [47] or C3D[46]. In the pretrained setting, we use the out-of-the-box R(2+1)D_18 pretrained on Kinetics 400[21], while C3D is pretrained on Sports-1M[20]. In the e2e setting, we initialize the model with the pretrained=False argument. The visual embedding $f_v(x)$ is BxTx512 where B is the batch size and T is the number of clips per video. We use $T = 1$ for training, and $T = 25$ for evaluation in Tables 1 and 2. The clips are 16 frames long and we choose them following standard protocols established by Wang *et al*. [48]. We average $f_v(x)$ across time (video snippets) similarly to previous approaches [46, 59]. f_s is a linear classifier with 512x300 weights. The output of $f_s \circ f_v$ is of shape Bx300.

We follow standard protocol in computing semantic embeddings of class names [4, 53, 59]. Word2Vec [28] – in particular, the gesim [33] Python implementation – encodes each word. We average multi-word class names. In rare cases of words not available in the pretrained W2V model (for example, 'rubiks' or 'photobombing') we manually change the words (see the code for more details). Formally, for a class name consisting of N words $c = [c^1, \cdots, c^N]$, we embed it as $\text{W2V}(c) = \sum_{i=1}^{N} \text{W2V}(c^i) \in \mathbb{R}^{300}$. We set τ to 0.05 following the analysis in Sec. 3.3 based on Fig. 3.

To minimize the loss of Eq. 2 we use the Adam optimizer [22], starting with a learning rate of $1e{-}3$. Batch size is 22 snippets, with 16 frames each. The model trained for 150 epochs, with a tenfold learning rate decrease at epochs 60 and 120. All experiments are performed on the Nvidia Tesla V100 GPU.

Following [46], we reshaped each frame's shortest side to 128 pixels, and cropped a random 112x112 patch on training and the center patch on inference.

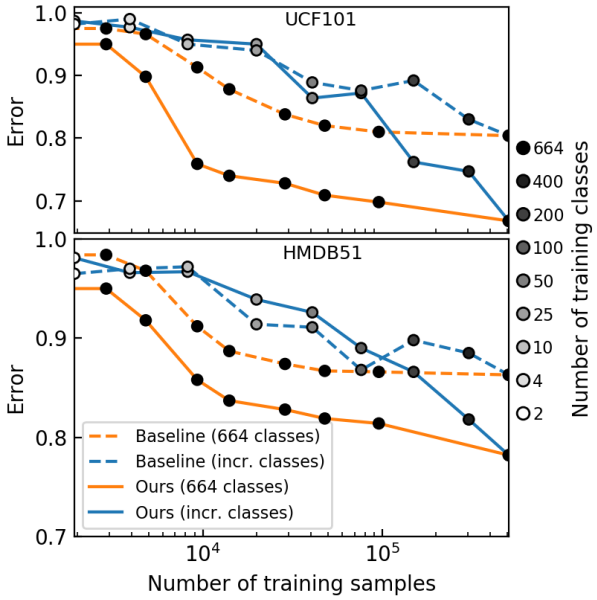

Figure 4: Number of training classes matters in ZSL. Orange curves show performance on subsets of Kinetics 664, as we keep all the training classes and increase the subset size. The blue curves, whose markers become progressively brighter, indicate a separate experiment where we increased the number of training classes starting from 2, all the way up to 664 (Sec. 5.2). For any given training dataset size, performance on test data is much better with more training classes. In addition, when few training classes are available the e2e model is not able to outperform the baseline.

5. Results

Our experiments have two goals: compare our method to previous work and investigate our method's performance vs the baseline (see Sec. 3.2.) The first is necessary to validate that e2e ZSL on videos can outperform more complex approaches that use pretrained features. The latter will allow us to understand under what conditions e2e training can be particularly beneficial.

5.1. Comparison to the state of the art

Table 1 compares our method to existing approaches. We followed our Training and Evaluation Protocol 1, as described in Sections 4.2 and 4.3. Our protocols are more restrictive than that of previous methods: we removed training classes that overlap with test classes, introduced domain shift, and applied one model to multiple test datasets. Despite this, we outperform previous video-based methods by a large margin. Furthermore, when testing on UCF we outperform URL [59] which uses a network an order of mag-

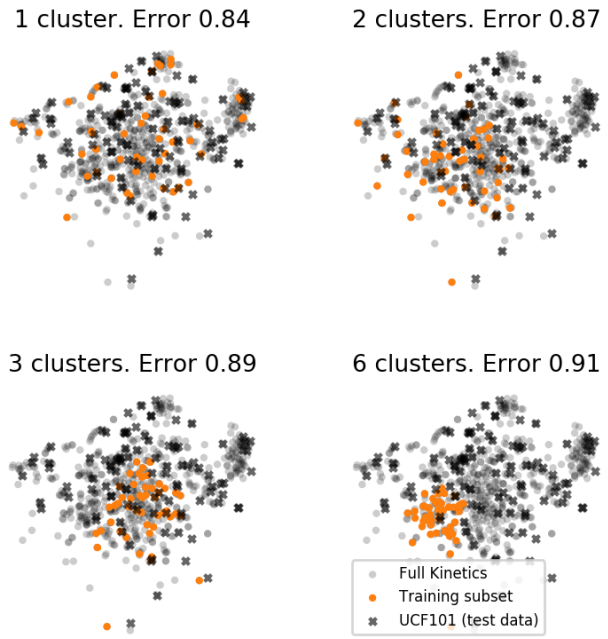

Figure 5: Diverse training classes are good for ZSL. Here we trained our algorithm on subsets of 50 Kinetics 664 classes. (Top left) Training classes picked uniformly at random. (Top right) We clustered Word2Vec embeddings of classes into two clusters, then trained and evaluated separately using each cluster, and averaged the results. (Bottom) Here we averaged the results of training using three and six clusters. The figure shows that the more clusters, the less diverse the training classes were semantically. At the same time, less diversity caused higher errors.

nitude deeper than ours – 18 vs 200 layers – and 23 classes overlap between training and testing (see Sec. 2).

5.2. Comparison to a baseline method

Our baseline method described in Sec. 3.2 uses a fixed, pretrained visual feature extractor but is otherwise identical to our e2e method. This allows us to study the benefits of e2e training under Evaluation Protocol 2, (see Sections 4.2 and 4.3). Using all test classes provides a more direct evaluaition of the method.

Training dataset size: To investigate the effect of training set size on performance we subsampled Kinetics 664 uniformly at random, then re-trained and re-evaluated the model. Fig. 4 shows that the e2e algorithm consistently outperforms the baseline on both datasets. Both algorithms' performance is worse with smaller training data. However, the baseline flattens out at about 100K training datapoints, whereas our method's error keeps decreasing. This is expected, as the e2e model has more capacity.

Number of training classes: In many video domains diverse data is difficult to obtain. Small datasets might not

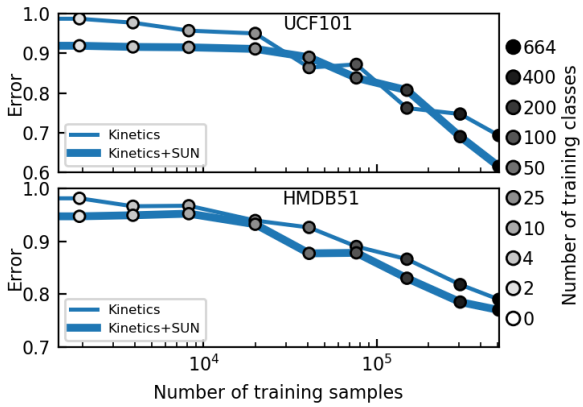

Figure 6: Augmented pretraining with videos-from-images. We trained our algorithm on progressively smaller subsets of Kinetics 664 classes (Sec. 5.2). We compared the results to training on the same dataset, after pretraining the model on our synthetic SUN video dataset (Sec. 5.3). The pretraining procedure boosts performance up to 10% points.

only have few datapoints, but also contain only a few training classes. We show that the number of training classes can impact ZSL results as much as training dataset size.

To obtain Fig. 4 we subsampled Kinetics 664 classwise. We first picked 2 Kinetics 664 classes at random, and trained the algorithm on those classes only. We repeated the procedure using 4, 10, 25, 50, 100, 200, 400 and all 664 classes. Naturally, the fewer classes the fewer datapoints the training set contained. This results are compared in Fig. 4 with the procedure described above, where we removed Kinetics datapoints at random – independent of their classes.

The figure shows that it is better to have few training samples from a large number of classes rather than many from a very small number of classes. This effect is more pronounced for the e2e model rather than the baseline.

Training dataset class diversity: We showed that ZSL works better with more training classes. If we have a limited budget for collecting classes and datapoints, how should we choose them? We investigated whether the set of training classes should emphasize fine differences (e.g. "shooting basketball" vs "passing basketball" vs "shooting soccerball" and so on) or diversity.

In Fig. 5 we selected 50 training classes in four ways: (Top Left) We randomly choose 50 classes from the whole Kinetics 664 dataset, trained the algorithm on these classes, and ran inference on the test set. We repeated this process ten times and averaged inference error. (Top Right) We clustered the 664 classes into 2 clusters in the Word2Vec embedding space, and chose 50 classes at random within one of the clusters, trained and ran inference. We then repeated the procedure ten times and averaged the result.

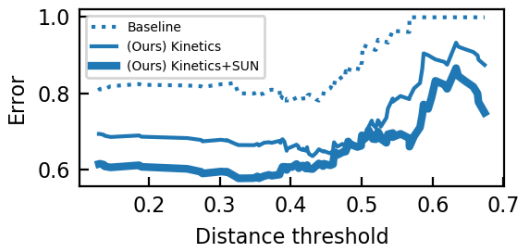

Figure 7: Error as test classes move away from training. For each UCF101 test class, we computed its distance to 10 nearest neighbors in the training dataset. We arranged all such distance thresholds on the x-axis. For each threshold, we computed the accuracy of the algorithms *on test classes whose distance from training data is larger than the threshold*. In other words, as x-axis moves to the right, the model is evaluated on cumulatively smaller, but harder test sets.

(Bottom) Here we chose 50 classes in one of 3 clusters (Left) and one of 6 clusters (Right), trained, and averaged inference results of 10 runs. The figure shows that test error for our method increases as class diversity decreases. This result is not obvious, since the task becomes harder with increasing class diversity.

5.3. Easy pretraining with images

Previous section showed that class count and diversity are important drivers of ZSL performance. This inspired us to develop the pretraining method described in Sec. 3.4: we pretrain our model on a synthetic video dataset created from still images from the SUN dataset. Fig. 6 shows that this simple procedure consistently decreases test errors by up to 10%. In addition, Fig. 7 shows that this initialization scheme makes the model more robust to large domain shift between train and test classes. The following section describes the latter finding in more detail.

5.4. Generalization and domain shift

A good ZSL model generalizes well to classes that differ significantly from training classes. To investigate the performance of our models under heavy domain shift, we computed the accuracy on subsets of test data with a growing distance from the training dataset. We first trained our model on Kinetics 664. Then, for a given distance threshold τ (see Sec. 3.3), we computed accuracy on the set of UCF classes whose mean distance from the closest 10 Kinetics 664 classes is larger than τ. Fig. 7 shows that the baseline model's (not trained e2e) performance drops to zero at around $\tau \sim 0.57$. Our method performs much better, never dropping to zero accuracy for high thresholds. Finally, using the SUN pretraining further increases performance.

UCF101 accuracy			50 classes		101 classes	
e2e	Augment	Multi	Top-1	Top-5	Top-1	Top-5
			26.8	55.5	19.8	40.5
✓			43.0	68.2	35.1	56.4
✓	✓		45.6	73.1	36.8	61.7
✓		✓	48.0	74.2	37.6	62.5
✓	✓	✓	**49.2**	**77.0**	**39.8**	**65.6**

Table 3: Ablation study. Numbers represent classification accuracy. "50 classes" uses Evaluation Protocol 1 (Sec. 4.3.) "101 classes" uses Evaluation Protocol 2. e2e: training the visual embedding as opposed to fixed, pretrained baseline (Sec. 3.2). Augment: pretrain using the SUN augmentation scheme (Sec. 5.3). Multi: At test time, extract multiple snippets from each video and average the visual embeddings (Sec. 4.4).

5.5. Ablation study

Table 3 studies contributions of different elements of our model to its performance. The performance is low when the visual embedding is fixed. The e2e approach improves the performance by a large margin. Our class augmentation method further boosts performance. Finally it helps to extract linearly spaced snippets from a video on testing, and average their visual embeddings. Using 25 snippets improves considerably the performances without influencing the training time of the model.

6. Conclusion

We followed practices from recent video classification literature to train the first e2e system for video recognition ZSL. Our evaluation protocol is stricter than that of existing work, and measures more realistic zero-shot classification accuracy. Even under this stricter protocol, our method outperforms previous works whose performance was measured with training and test sets overlapping and sharing domains. Through a series of directed experiments, we showed that a good ZSL dataset should have many diverse classes. Guided by this insight, we formulated a simple pretraining technique that boosts ZSL performance.

Our model is easy to understand and extend. Our training and evaluation protocols are easy to use with alternative approaches. We made our code available at github.com/bbrattoli/ZeroShotVideoClassification to encourage the community to build on our insights and create a strong foundation for future video ZSL research.

Acknowledgement. We thank Amazon for providing the computational means and Alina Roitberg for a productive discussion about the evaluation protocol.

References

[1] Ioannis Alexiou, Tao Xiang, and Shaogang Gong. Exploring synonyms as context in zero-shot action recognition. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 4190–4194. IEEE, 2016. 2

[2] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 2

[4] Mina Bishay, Georgios Zoumpourlis, and Ioannis Patras. Tarn: Temporal attentive relation network for few-shot and zero-shot action recognition. *arXiv preprint arXiv:1907.09021*, 2019. 1, 2, 3, 4, 5, 6

[5] Biagio Brattoli, Uta Büchler, Anna-Sophia Wahl, Martin E. Schwab, and Björn Ommer. Lstm self-supervision for detailed behavior analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[6] Uta Büchler, Biagio Brattoli, and Björn Ommer. Improving spatiotemporal self-supervisionby deep reinforcement learning. In *IEEE Conference on European Conference on Computer Vision (ECCV)*, 2018. 2

[7] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 2

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 2

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3

[10] Rahul Dey and Fathi M Salemt. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017. 3

[11] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015. 1, 3, 4, 5

[12] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. *arXiv preprint arXiv:1812.03982*, 2018. 1, 2

[13] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, volume 1, page 3, 2017. 1

[14] Meera Hahn, Andrew Silva, and James M Rehg. Action2vec: A crossmodal embedding approach to action learning. *arXiv preprint arXiv:1901.00484*, 2019. 1, 2, 3, 4, 5

[15] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018. 2

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 2, 3

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 3

[19] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos in the wild. *Computer Vision and Image Understanding*, 155:1–23, 2017. 3

[20] 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, 6

[21] Will Kay, Joao 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, 4, 5, 6

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[24] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563. IEEE, 2011. 1, 4, 5, 6

[25] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *AAAI*, volume 1, page 3, 2008. 1

[26] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 2

[27] Brais Martinez, Davide Modolo, Yuanjun Xiong, and Joseph Tighe. Action recognition with spatial-temporal discriminative filter banks. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 3, 6

[29] Ashish Mishra, Vinay Kumar Verma, M Shiva Krishna Reddy, S Arulkumar, Piyush Rai, and Anurag Mittal. A gen-

erative approach to zero-shot and few-shot action recognition. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 372–380. IEEE, 2018. 2, 3

[30] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pages 1410–1418, 2009. 1

[31] AJ Piergiovanni and Michael S Ryoo. Learning shared multimodal embeddings with unpaired data. *CoRR*, 2018. 2

[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2

[33] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en. 6

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2

[35] Alina Roitberg, Ziad Al-Halah, and Rainer Stiefelhagen. Informed democracy: voting-based novelty detection for action recognition. *arXiv preprint arXiv:1810.12819*, 2018. 5

[36] Alina Roitberg, Manuel Martinez, Monica Haurilet, and Rainer Stiefelhagen. Towards a fair evaluation of zero-shot action recognition using external data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 2, 3, 4, 5

[37] Karsten Roth, Biagio Brattoli, and Bjorn Ommer. Mic: Mining interclass characteristics for improved metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8000–8009, 2019. 1

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 3

[39] Artsiom Sanakoyeu, Vasil Khalidov, Maureen S. McCarthy, Andrea Vedaldi, and Natalia Neverova. Transferring dense pose to proximal animal classes. In *CVPR*, 2020. 1

[40] Artsiom Sanakoyeu, Vadim Tschernezki, Uta Büchler, and Björn Ommer. Divide and conquer the embedding space for metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[41] Nawid Sayed, Biagio Brattoli, and Björn Ommer. Cross and learn: Cross-modal self-supervision. In *German Conference on Pattern Recognition (GCPR)*, 2018. 2

[42] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 1

[43] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 2

[44] Bharat Singh, Mahyar Najibi, and Larry S Davis. Sniper: Efficient multi-scale training. In *Advances in Neural Information Processing Systems*, pages 9310–9320, 2018. 2

[45] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 1, 4, 5, 6

[46] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. 1, 2, 6

[47] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018. 1, 2, 6

[48] 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 *European conference on computer vision*, pages 20–36. Springer, 2016. 1, 2, 4, 6

[49] Qian Wang and Ke Chen. Alternative semantic representations for zero-shot human action recognition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 87–102. Springer, 2017. 2

[50] Qian Wang and Ke Chen. Zero-shot visual recognition via bidirectional latent embedding. *International Journal of Computer Vision*, 124(3):356–383, 2017. 2, 4, 5

[51] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4582–4591, 2017. 1

[52] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492. IEEE, 2010. 5

[53] Xun Xu, Timothy Hospedales, and Shaogang Gong. Semantic embedding space for zero-shot action recognition. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 63–67. IEEE, 2015. 2, 6

[54] Xun Xu, Timothy Hospedales, and Shaogang Gong. Transductive zero-shot action recognition by word-vector embedding. *International Journal of Computer Vision*, 123(3):309–333, 2017. 2

[55] Xun Xu, Timothy M Hospedales, and Shaogang Gong. Multi-task zero-shot action recognition with prioritised data augmentation. In *European Conference on Computer Vision*, pages 343–359. Springer, 2016. 2, 4, 5

[56] Bowen Zhang, Hexiang Hu, and Fei Sha. Cross-modal and hierarchical modeling of video and text. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 374–390, 2018. 2

[57] Chenrui Zhang and Yuxin Peng. Visual data synthesis via gan for zero-shot video classification. *arXiv preprint arXiv:1804.10073*, 2018. 3

[58] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In

Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2881–2890, 2017. 2

[59] Yi Zhu, Yang Long, Yu Guan, Shawn Newsam, and Ling Shao. Towards universal representation for unseen action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9436–9445, 2018. 1, 2, 3, 4, 5, 6

SUPPLEMENTARY MATERIAL
Rethinking Zero-shot Video Classification: End-to-end Training for Realistic Applications

Network	Train	classes	UCF 50	UCF 101	HMDB 25	HMDB 51
C3D	K400	361	33.7	25.7	17.0	13.3
R3D_18	K400	361	37.2	29.0	20.4	16.8
R(2+1)D_18	K400	361	38.7	30.6	22.0	18.1
C3D	K700	664	40.3	33.1	22	17.0
R3D_18	K700	664	41.2	34.2	23.6	19.0
R(2+1)D_18	K700	664	**43.0**	**35.0**	**25.8**	**20.6**
R(2+1)D_18	K400	400	50.1	44.5	27.2	22.5
R(2+1)D_18	K700	700	54.6	49.7	30.5	25.6

Table 1: Accuracy of different backbone architectures trained on the first (K400) and last (K700) version of Kinetics [19]. The models are evaluated on a single clip (16 frames).

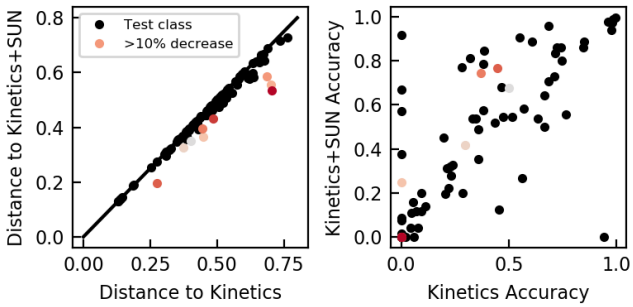

Figure 1: Each dot represents a UCF101 test class. Test class accuracy (right) and distance (left) to the train set (Kinetics664) for two models: one with random initialization and one pretrained on SUN (see Sec 3.4, Main Text). A colored dot indicates a test class that reduces its distance to the train set by more than 10% when SUN is included on training.

1. Backbone choice

Supplementary Table 1 compares the accuracy of three 3D convolutional backbones on two kinetics versions using our Training Protocol 1 (Sec. 4.2, Main Text). For this comparison we also tried using the full Kinetics 400/700 datasets, without removing overlapping test classes. The table shows that adding the 6% of the training classes most overlapping with the test set yields an unexpected >40% accuracy boost for UCF and 25% on HMDB. This proves that the zero-shot learning constraint is non-trivial.

2. SUN pretraining: easier task or better representation?

Section 3.4 (Main Text) shows that pretraining on a scenes dataset (SUN397) improves ZSL performance. In this section, we ask whether the boost is due to better model generalization or simply because the source domain becomes closer to the target domain.

Per each UCF101 test class, Sup. Fig. 1 shows the W2V distance to Kinetics train classes as well as (Kinetics + SUN) train classes. Test classes that got more than 10% closer to training data are marked in color. The right subplot, however, shows that the model trained on (Kinetics + SUN) boosts the accuracy of many classes – in particular, the accuracy of many classes that are *not* among the colored ones rose significantly. The model pretrained on SUN data increases performance on many classes which are not close to SUN data. We conclude that pretraining on SUN allows the model to generalize better over almost all test classes, not only the ones close to SUN data.

3. Training class diversity

We expand the analysis of Sec. 5.2 and Fig. 5, Main Text, by testing the influence of training class diversity on both UCF and HMDB. Sup. Fig. 2 correlates model performance with training class density. For this experiment, we selected 50 train classes with different density in the Word2Vec space, using the same clustering approach we used in Sec. 5.2. Per each diversity value, we select 50 classes and train a model multiple times to compute the standard deviation. Sup. Fig. 2 shows that test error de-

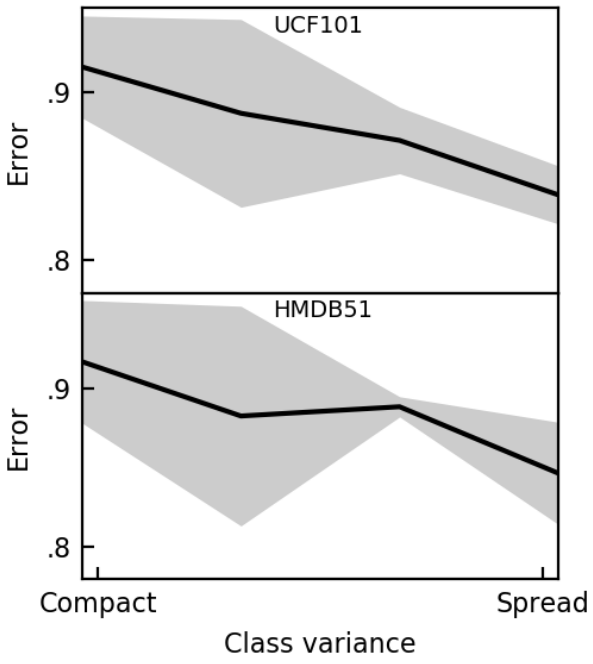

Figure 2: Performance of the e2e model trained on 50 Kinetic664 classes and tested on UCF and HMDB. The 50 classes are chosen based on diversity in their W2V embedding (see Fig. 5, Main Text, for details). The more semantically diverse the training classes, the lower the error.

creases as training classes become more diverse. At the same time, the standard deviation decreases, indicating that for compact classes, the performance highly depends on where in the class space we sample the classes, which is something we only know once the test set is available.

This outcome is not obvious, since we might expect the task to become harder when class variance increases (given the same number of training datapoints). However, we do not observe decrease in performance. Therefore, we can conclude that the model can only benefit from a high variety within the train class distribution. This new insight can be useful during training dataset collection.

4. Analyze the model capability action per action

What does better or worse accuracy indicate for specific classes? We break down the change in performance between models for each UCF101 test class.

4.1. Direct comparison by sorting classes

In Sec. 5, Main Text, we evaluated the model using error aggregated over all the test classes. It is also interesting to know whether the network is getting better at recognizing specific classes, or improves across the board?

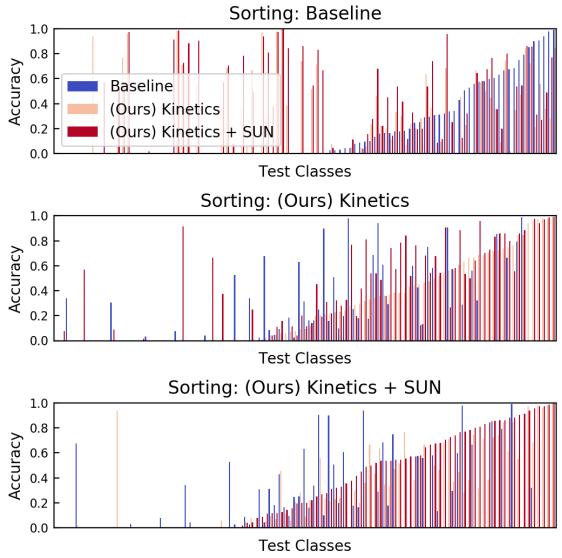

Figure 3: Accuracy on each UCF101 test class, for three models. Each subplot uses different model's accuracies to sort the classes, otherwise the numbers are the same.

Sup. Figure 3 shows the accuracy on each UCF test class for three models: baseline, e2e trained on Kinetics, and e2e pretrained on SUN397 and then trained on Kinetics. We sorted the classes from hardest to easiest for each model. Sup. Fig. 4 shows the same information, zoomed in on worst and best actions only. The two plots show that some of the actions which are difficult for the baseline model are correctly classified by our e2e models. On the other hand, the inverse situation is rare. In addition, the actions which are correctly classified by the baseline are also easily identified by our models.

In addition, the results of e2e trained on Kinetics and e2e pretrained on SUN and trained on Kinetics are highly correlated, but the second achieves overall better performances. This suggests that SUN provides *complementary* information to Kinetics which are useful for the target task. On the other hand, the baseline is less correlated with the e2e results, suggesting that the fixed visual features have a lot to learn and *should be fine-tuned*.

4.2. Confusion matrices

Sup. Figures 6 - 8 show confusion matrices of the three models we evaluated on UCF101. Sup. Fig. 5 shows the three CM directly compared with each other. In particular, we show the L2 distance computed pair-wise between

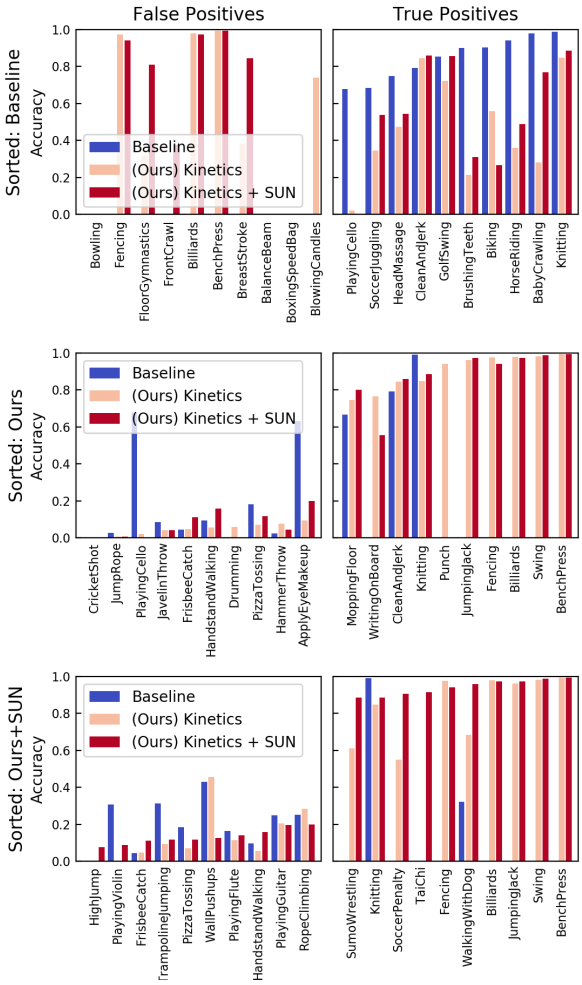

Figure 4: Accuracy on best and worst 10 classes for each model.

the CMs. This shows biases present in the Baseline model, which were removed by e2e training. Some interesting biases we discovered:

Playing: All models confuse the classes starting with the word "Playing". This issue probably comes from the way we embed the class name into the semantic embedding – simply averaging the words. Future work might focus on tackling this problem by using a different semantic encoder. This bias is less pronounced in e2e models.

JumpingRope: The baseline model wrongly classifies many actions as *JumpingRope*.

HandStandWalking: Our model trained on only Kinetics has a bias towards *HandStandWalking* class. This is attenuated by pre-training on SUN.

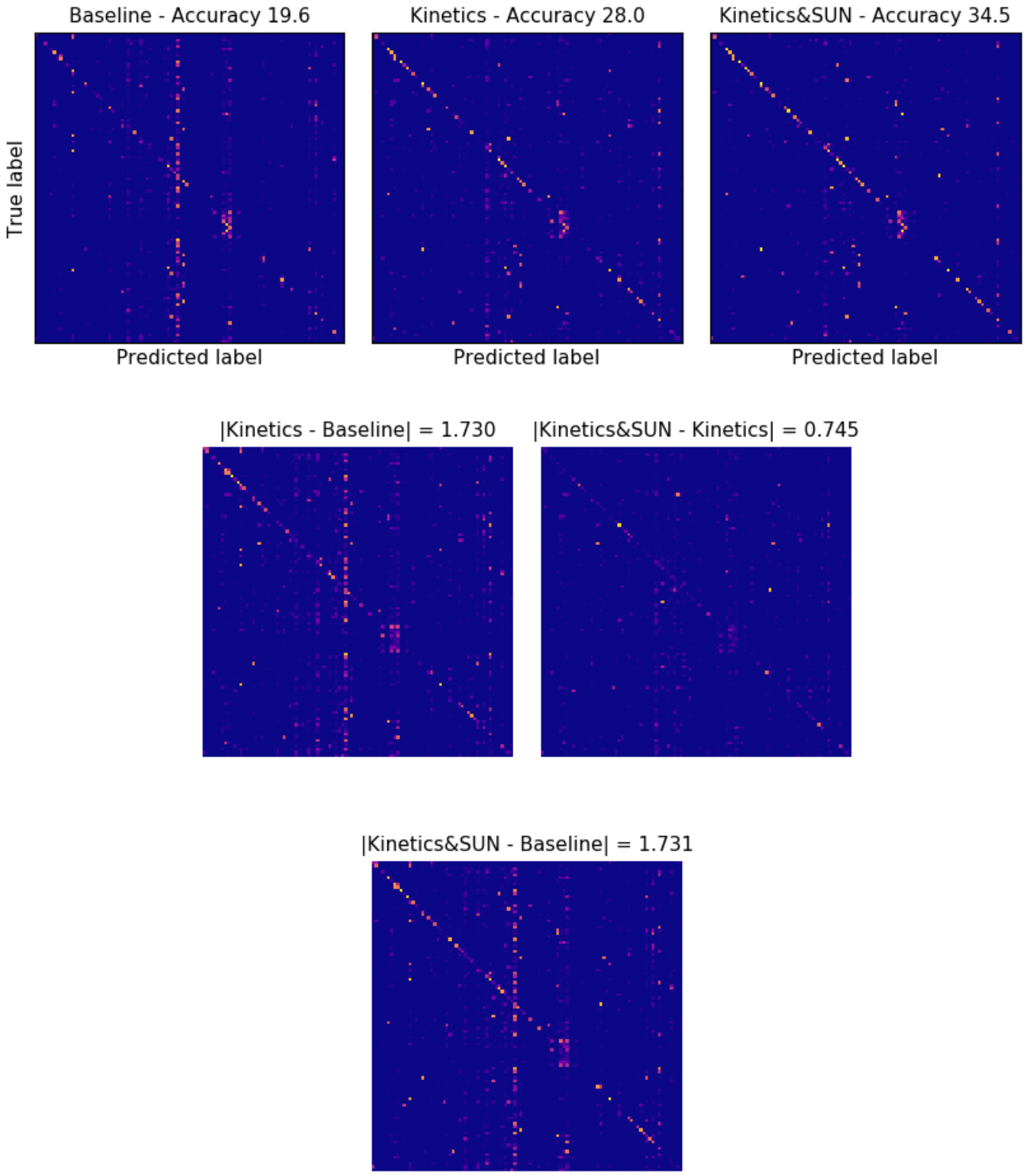

Figure 5: **Top**: UCF101 confusion matrices. **Middle and Bottom**: Pairwise L2 distances between the CMs, with average score indicated in the title.

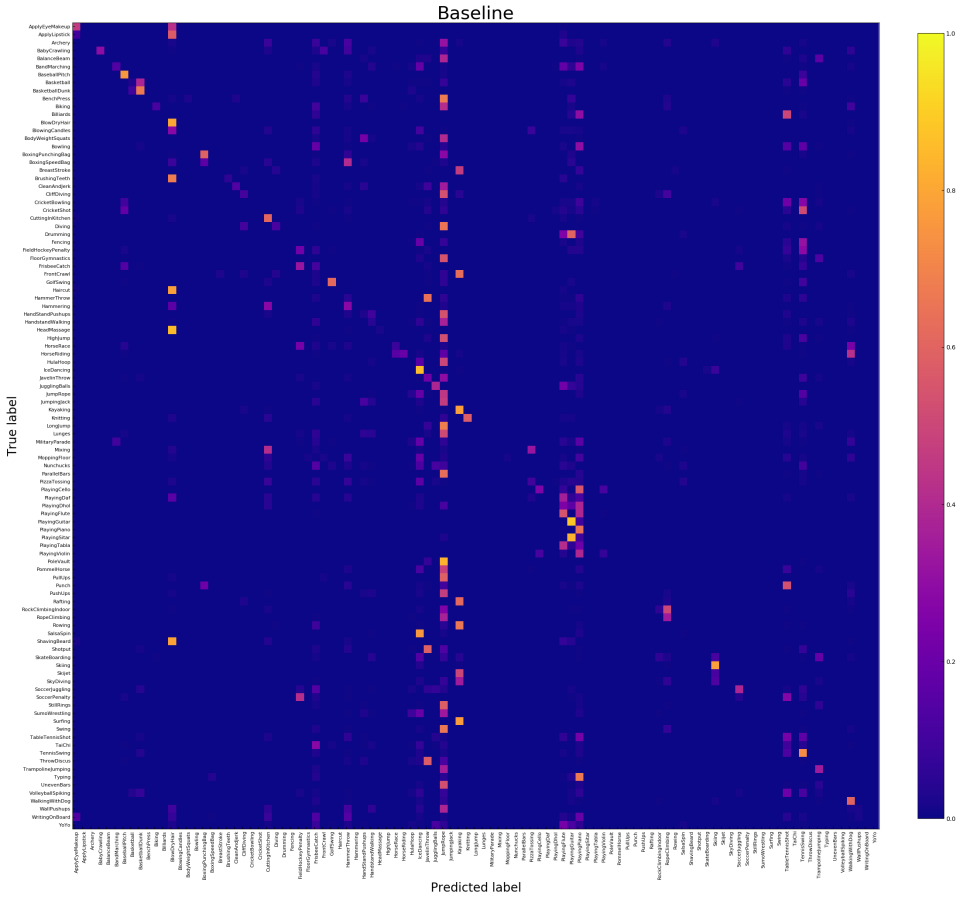

Figure 6: Confusion matrix on UCF101 using our baseline model (see Sec. 3.2 in the main paper). (Figure better seen zoomed in on the digital version)

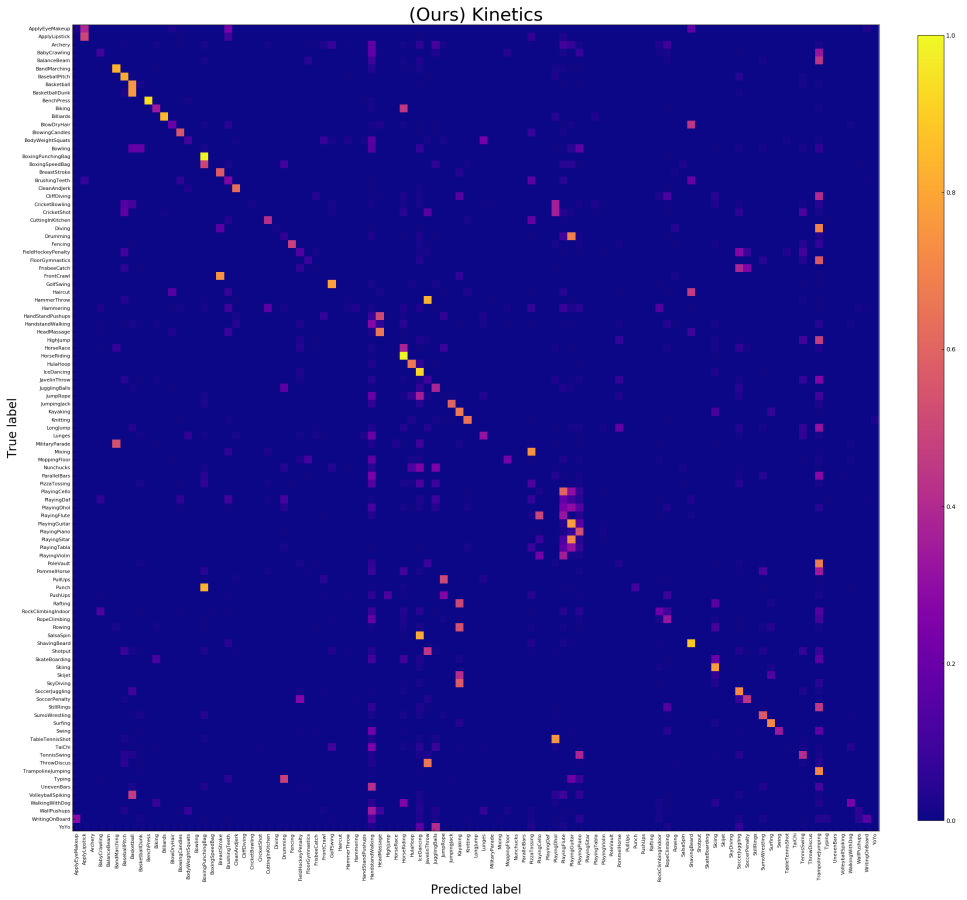

Figure 7: Confusion matrix on UCF101 using our e2e model trained on Kinetics. (Figure better seen zoomed in on the digital version)

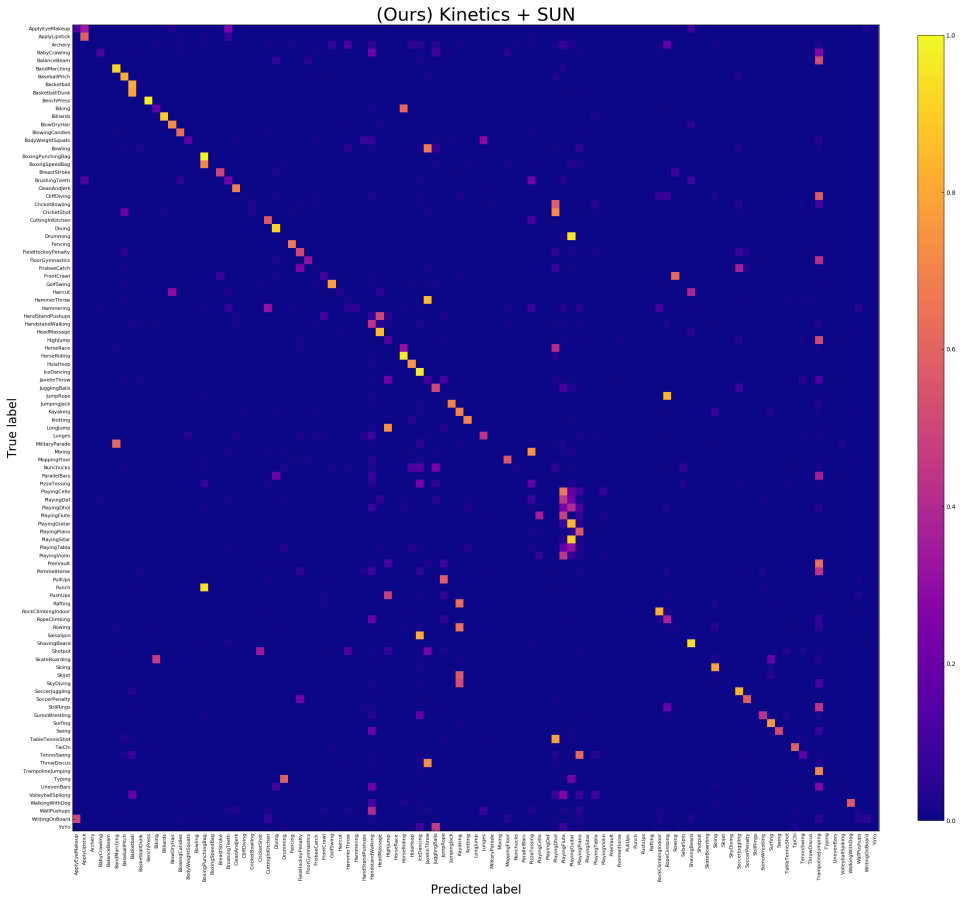

Figure 8: Confusion matrix on UCF101 using our e2e model pretrained on SUN397 (see Sec. 3.4 in the main paper) and fine-tuned on Kinetics. (Figure better seen zoomed in on the digital version)