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

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

Joseph 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 [21, 27, 51, 52, 53]. 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) [29, 34] and it is quite successful in the image domain [41, 42, 44, 45, 47, 56].

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

Since the set of possible human actions is huge, action recognition is a great ZSL testbed. Trained on large-scale academic datasets [11, 17, 24, 25, 28, 50], supervised 3D convolutional neural networks (CNNs) proved successful in this domain [12, 51, 52]. 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
recognition use pretrained visual embeddings [1, 4, 18, 33, 35, 54, 55, 58, 59, 60, 61, 64]. 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 [51] and large feature extractors [21].

In contrast, state-of-the-art algorithms in the fundamental CV domains of image classification [21], object detection [36, 38, 49] and segmentation [8, 20, 63] 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, 21]. 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. [40], 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, 33, 54, 55, 58, 59, 60], 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 [48, 53] that operate on 1-5 frame snippets and 3D networks [5, 6, 7, 12, 19, 31, 46, 51, 52] that operate on 16-128 frames. One of the earliest works of this type, Simonyan and Zisserman [48], trained with only 1-5 frames sampled randomly from the video. At inference many more 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. [53] 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, 51] 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 [51] or ResNet [21], then trains a temporal model that maps the visual embedding to a semantic embedding space [4, 13, 14, 15, 16, 18, 35, 61, 64]. Good generalization on semantic embeddings of class names means that the model can be applied to new videos where the possible
output classes are not present in training data. Inference reduces to finding the test class whose embedding is the nearest-neighbor of the model’s output. Word2Vec [32] is commonly used to produce the ground-truth word embeddings. An alternative approach is to use manually crafted class attributes [23]. We decided not to pursue the manual approach as it harder to apply in general scenarios.

Two effective recent methods, Hahn et al. [18] 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, 22] 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 [18] 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 [33, 62]. 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. [64] 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 [21], pretrained on ImageNet [9, 43], which cannot utilize temporal information.

As pointed out by Roitberg et al. [40], previous works train their models on actions overlapping with those of the target dataset, violating ZSL assumptions. For example, Zhu et al. [64] 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) $D_s = \{(x_1, c_1), \ldots , (x_N, c_N)\}$ 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

$$M(x) = \arg\min_{c \in D_t} \cos(g(x), W2V(c)).$$  \hspace{1cm} (1)$$

where $\cos$ is the cosine distance and the semantic embedding is computed using the Word2Vec function [32] $W2V : \mathcal{C} \rightarrow \mathbb{R}^{300}$.

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 [60, 64] or computed by a pretrained deep network [4, 18, 55, 64]. 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

\[
L = \sum_{(x, c) \in D_s} \| W2V(c) - (f_s \circ f_v)(x^t) \|^2. \tag{2}
\]

Inference procedure is similar but pools information from multiple snippets following Wang et al. [53]. 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, 18, 64]). 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 [40] 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 \( d: C \rightarrow C \) denote a distance metric on the space of all possible class names \( C \), and let \( \tau \in \mathbb{R} \) denote a similarity threshold. A video classification task fully respects the zero-shot constraint if

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

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

\[
d(c_1, c_2) = \cos(W2V(c_1), W2V(c_2)) \tag{4}
\]

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 experiments 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 [25], and testing on all of UCF [50], HMDB [28] 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
4.1. Datasets

with our definitions in Sec. 3.3. 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 [57], a standard scene recognition dataset. 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 [57], 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 [50] has 101 action classes primarily focused around sports, with 13320 videos sourced from YouTube. HMDB51 [28] 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 [25] 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 [57] (see Sec. 3.4) is a scene understanding image dataset. It contains 397 scene categories for each video.

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.

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 [57], 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 [50] has 101 action classes primarily focused around sports, with 13320 videos sourced from YouTube. HMDB51 [28] 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 [25] 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 [57] (see Sec. 3.4) is a scene understanding image dataset. It contains 397 scene categories for each video.

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.

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 ∪ 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 ∪ HMDB ∪ 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

<table>
<thead>
<tr>
<th>Method</th>
<th>UCF Top-1</th>
<th>UCF Top-5</th>
<th>HMDB Top-1</th>
<th>HMDB Top-5</th>
<th>Activity Top-1</th>
<th>Activity Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL [64]</td>
<td>34.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>664classes</td>
<td>37.6</td>
<td>62.5</td>
<td>26.9</td>
<td>49.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>605classes</td>
<td>35.3</td>
<td>60.6</td>
<td>24.8</td>
<td>44.0</td>
<td>20.0</td>
<td>42.7</td>
</tr>
</tbody>
</table>

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 [64] 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.
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 [50] and HMDB [28] 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 \(_{18} \) [52] or C3D[51]. In the pretrained setting, we use the out-of-the-box R(2+1)D \(_{18} \) pretrained on Kinetics 400[25], while C3D is pretrained on Sports-1M[24]. In the e2e setting, we initialize the model with the pretrained=False argument. The visual embedding \( f_v(x) \) is \( B \times T \times 512 \) 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. [53]. We average \( f_v(x) \) across time (video snippets) similarly to previous approaches [51, 64]. \( f_s \) is a linear classifier with \( 512 \times 300 \) weights. The output of \( f_s \circ f_v \) is of shape \( B \times 300 \).

We follow standard protocol in computing semantic embeddings of class names [4, 58, 64]. Word2Vec [32] – in particular, the gesim [37] 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 \( c = [c^1, \ldots, c^N] \), we embed it as \( W2V(c) = \sum_{i=1}^{N} W2V(c^i) \in \mathbb{R}^{300} \). We set \( \tau \) 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 [26], 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 [51], we reshaped each frame’s shortest side to 128 pixels, and cropped a random 112x112 patch on training and the center patch on inference.

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 [64] 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.

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 evaluation 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 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 class-wise. 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.
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 $\tau$ (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 $\tau$. 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.

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 generously supporting the project, and Alina Roitberg for a productive discussion on the evaluation protocol.
References


[58] Xin Xu, Timothy Hospedales, and Shaogang Gong. Semantic embedding space for zero-shot action recognition. In


