Large-Scale Training Framework for Video Annotation

Seong Jae Hwang*
Univ. of Wisconsin-Madison
sjh@cs.wisc.edu

Joonseok Lee
Google Research
joonseok@google.com

Balakrishnan Varadarajan
Google Research
balakrishnanv@google.com

Ariel Gordon
Google Research
gariel@google.com

Zheng Xu
Google Research
zhengxu@google.com

Apostol (Paul) Natsev
Google Research
natsev@google.com

ABSTRACT

Video is one of the richest sources of information available online but extracting deep insights from video content at internet scale is still an open problem, both in terms of depth and breadth of understanding, as well as scale. Over the last few years, the field of video understanding has made great strides due to the availability of large-scale video datasets and core advances in image, audio, and video modeling architectures. However, the state-of-the-art architectures on small scale datasets are frequently impractical to deploy at internet scale, both in terms of the ability to train such deep networks on hundreds of millions of videos, and to deploy them for inference on billions of videos. In this paper, we present a MapReduce-based training framework, which exploits both data parallelism and model parallelism to scale training of complex video models. The proposed framework uses alternating optimization and full-batch fine-tuning, and supports large Mixture-of-Experts classifiers with hundreds of thousands of mixtures, which enables a trade-off between model depth and breadth, and the ability to shift model capacity between shared (generalization) layers and per-class (specialization) layers. We demonstrate that the proposed framework is able to reach state-of-the-art performance on the largest public video datasets, YouTube-8M and Sports-1M, and can scale to 100 times larger datasets.

CCS CONCEPTS

• **Computing methodologies** → **Visual content-based indexing and retrieval**; **MapReduce algorithms**;

KEYWORDS

Scalability, Distributed framework, Video annotation, MapReduce

ACM Reference Format:
Seong Jae Hwang, Joonseok Lee, Balakrishnan Varadarajan, Ariel Gordon, Zheng Xu, and Apostol (Paul) Natsev. 2019. Large-Scale Training Framework for Video Annotation. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330653

*This work was done during an internship at Google Research.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08.
https://doi.org/10.1145/3292500.3330653

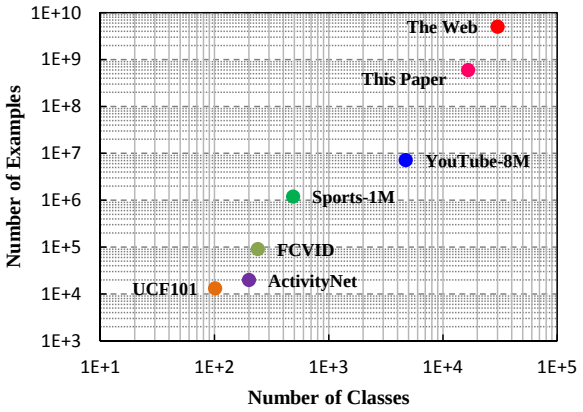


Figure 1: Large-scale video classification datasets.

1 INTRODUCTION

In the last decade, a series of breakthroughs in machine learning and computer vision problems were attributed to the availability of large-scale quality datasets. As the quality and quantity of datasets increased, so did the sophistication of models and their ability to accomplish more complex, high-level tasks such as scene understanding, pixel-level segmentation and depth extraction, Visual-Question-Answering.

In the video classification domain, YouTube-8M [1] is the largest public dataset as of this writing, containing over 7 million videos with 4,716 classes. Classifying thousands of high-level video labels across diverse topics, ranging from objects to activities, requires multi-label classification models that can scale both in the number of classes and number of videos. With millions of video examples, spanning hundreds of thousands of video hours, each epoch of training involves billions of frame-by-frame audio-visual features.

Thanks to modern GPUs and custom hardware accelerators, it is becoming less prohibitive to train machine learning models at this scale, including complex models, such as recurrent deep neural networks [2, 9] and frame-by-frame temporal aggregation networks [1, 19, 27, 29, 37]. Mixtures of binary classifiers have also shown promising multi-label video classification performance at this scale [24].

Nevertheless, even the largest publicly available video datasets lag far behind the volume of public videos on the Internet. YouTube, for example, reached over 1 billion captioned videos in 2017.[1] In addition, videos are growing at an unprecedented scale, with more

[1] https://youtube.googleblog.com/2017/02/one-billion-captioned-videos.html

than 500 hours of video being uploaded to YouTube each minute.[2] Fig. 1 illustrates the scale of several largest video classification datasets and the scale of training data that we are addressing in this paper, which is on the order of 100M videos and tens of thousands of classes, or 1000 times larger than most public datasets. Not only is the volume of online videos large, but so is the variety of topics covered by those videos. Annotating videos at that scale and diversity requires the support of much larger vocabularies than those found in public datasets.

In this paper, we present how we tackle the video annotation problem at scale, with a proposed MapReduce-based distributed framework, which can scale to hundreds of millions of videos with hundreds of thousands of classes or classifier mixtures. The main objective of this paper is to address both annotation quality and scalability at the same time: building a framework that can support training complex video models at web scale. Although it is known that MapReduce is an effective tool for distributed computation at scale [5, 7], to the best of our knowledge, the proposed framework is the first-in-kind application of MapReduce to the problem of large-scale video modeling, supporting both shared (deep) representation learning and specialized per-class (large) mixture modeling.

Specifically, we present a scalable variant of the Deep-Bag-of-Frames (DBoF) model with mixture-of-experts (MoE), one of the top-performing video classification models on YouTube-8M [1, 29]. We design a MapReduce-based framework suitable to train this model, fully taking advantage of *model parallelism* with large mixtures and *data parallelism* through large (even full) mini-batch training:

- **Large Mixture-of-Experts**: Considering the wide range of video topics on the web, it is essential to train a model capable of classifying multiple labels. When the number of possible classes is large, it is generally desirable to increase the number of experts. Without a scalable training framework, however, increasing the number of experts becomes impractical due to computational overhead. For this reason, most previous works have used a small number of (i.e., <5) experts, but this can be sub-optimal, depending on the problem and data diversity. Our framework provides *model parallelism* to allow training large MoEs, with hundreds of thousands of mixtures (across all classes), on hundreds of millions of videos.
- **Large-scale Optimization**: In general, utilizing a larger mini-batch often equates to a superior performance [34]. At modern large-scale datasets, however, considering even 1% batch size (for example, 80K examples in YouTube-8M) becomes infeasible in ordinary settings. Via *data parallelism*, we demonstrate that our framework allows large-batch optimization, namely Resilient Backpropagation (RProp) [33]. We also demonstrate that when the batch size is sufficiently large (e.g., 50%), this traditional approach becomes worth revisiting for its known robustness involving only few parameters.

Note that we focus on improving a *single model* since gaining further improvement from the ensemble procedure of multiple models is a standard follow up step which is typically independent of the single model.

Contributions. Tackling the video classification problem at scale, we propose a variant of the joint DBoF and MoE model, with a newly proposed Self-Weighted Average Pooling (SWAP) approach for temporal pooling of frame-level representations. This model already surpasses the best single model performance reported at the YouTube-8M Large-Scale Video Understanding Challenge. To effectively train this model at an even larger scale, we propose a MapReduce-based training framework, and verify that it can efficiently train the DBoF/MoE model on 100M videos and over 16K classes in a few days. To summarize,

- We present a **MapReduce-based training framework**, designed to train state-of-the-art video annotation models at large scale.
- We explore **algorithmic optimization** schemes, which were not practical previously. Large mixture-of-experts and full-batch fine-tuning reveal that we can improve a converged model after traditional training, achieving state-of-the-art performance on YouTube-8M and Sports-1M datasets.
- We demonstrate that our proposed framework and model is scalable to train on 500M videos with over 16K classes.

2 RELATED WORK

Large-scale datasets containing dense information about various types of data such as ImageNet [21], Places [46], and Visual Genome [20] are being used widely. These datasets have been immensely contributing to the success of diverse machine learning tasks such as image classification [14, 32], scene recognition [45, 46], relational learning [26], and Visual-Question-Answering [48].

Recently, tasks utilizing video data have been explored [8, 38] as large-scale video datasets such as Sports-1M [17] and YouTube-8M [1] began providing millions of videos with frame-by-frame video and audio features along with thousands of video labels. To sequentially process series of frames, many recent successful models incorporated recurrent models (e.g., Long-short Term Memory [13]) [2, 4, 25, 40, 42] or convolution on temporal axis [2, 9]. Also, various attention-based models [1, 18, 19, 27, 29, 39] perform pooling over frames to treat in a "bag-of-words" manner.

With growing data scale, efficient training algorithms are becoming essential as well. One type of approach for neural networks is distributed training, where the model is trained in parallel on different machines with partitioned data, synchronously [44] or asynchronously [6, 30, 43] (see [28] for a list of parallelism strategies in machine learning). Another type of approach is mini-batch scaling method [10, 41] which increases the mini-batch sizes for much faster convergence. However, the mini-batch sizes are still practically bounded (e.g., 32K) for more extreme learning schemes such as full-batch optimization (e.g., RProp [33]).

3 VIDEO ANNOTATION PROBLEM

Given a video of T frames with D dimensional preprocessed frame-level features $\mathbf{X} \in \mathbb{R}^{D \times T}$, the goal of the *video annotation* problem is to predict each of its video-level label $\mathbf{y} \in \{0, 1\}^K$ describing the video content (e.g., gaming, sports), where K is the number of possible labels. As we take a finite number of labels, this problem is

[2] https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/

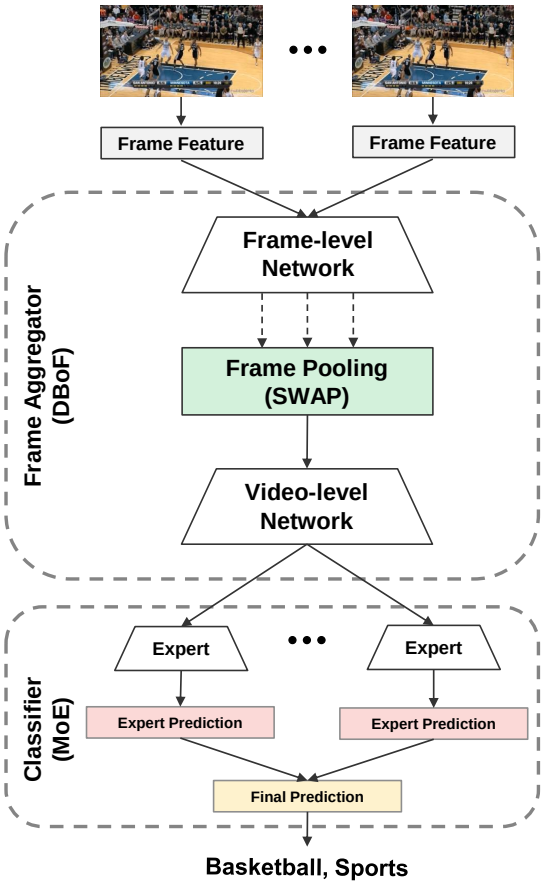


Figure 2: Deep-Bag-of-Frames framework

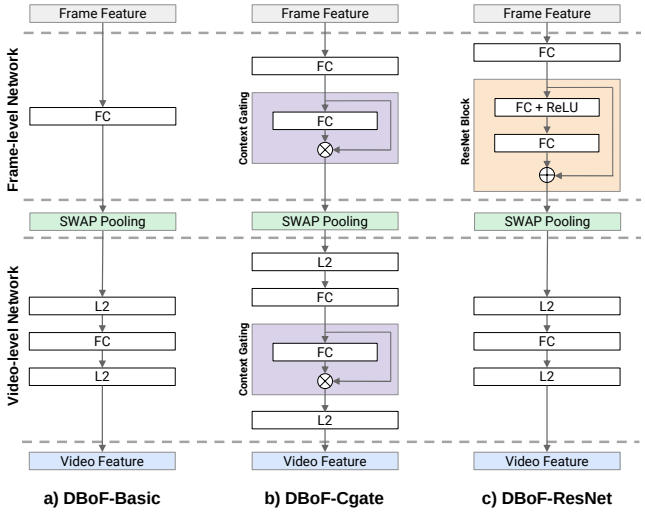


Figure 3: DBoF architectures of frame aggregators

modeled as a multi-label classification problem. Henceforth, we use *video classification* and *video annotation* interchangeably. Naturally, both sequential models (e.g., RNN) and frame-pooling models (e.g., bag-of-frames) become appropriate for this problem. In this work, we focus on the latter type of models which have been the top-performing models on YouTube-8M [1, 3, 29].

3.1 Video Annotation Model

Our model, shown in Fig. 2, consists of the following two components: 1) *Deep-Bag-of-Frames (DBoF)* for aggregating the frame-level features into a video-level feature, and 2) *Mixture-of-Experts (MoE)* for constructing multiple 'expert' binary classifiers for each class.

3.1.1 Frame Aggregator. Bag-of-words type models have been shown to be promising for sequential data such as videos [22, 39]. Analogously treating a set of frame-level features as a "bag-of-frames", we revise Deep-Bag-of-Frames (DBoF) models [1, 29] as shown in Fig. 2. The overall architecture of DBoF is as follows:

(1) **Frame-level Network**: Given a video and its frame-level features (visual and/or audio) $\mathbf{X} \in \mathbb{R}^{D \times T}$ as stated above, a frame-level network transforms each frame-level feature $\mathbf{x}_j \in \mathbb{R}^D$ of frame j into its new representation, typically to a higher dimensional space. In our work, we use the following three networks illustrated in Fig. 3: (i) fully-connected layer, (ii) fully-connected layer with context gating [29], and (iii) fully-connected layer with a residual block [11].

(2) **Frame Pooling**: Then, the embedded representations of the given video are aggregated to a single video-level feature through a frame pooling layer. Specifically, we use *Self-Weighted Average Pooling (SWAP)* for each video which normalizes the pooling frames $\mathbf{x}_j \in \mathbb{R}^D$ for $j = 1, \ldots, T$ as follows:

$$\mathbf{v} = \sum_{j=1}^{T} \frac{|\mathbf{x}_j|}{\sum_{j'=1}^{T} |\mathbf{x}_{j'}|} \mathbf{x}_j. \tag{1}$$

In other words, the new video-level pooled feature $\mathbf{v}$ is the sum of the frame-level features $\mathbf{x}_j$ weighted by their corresponding activations and normalized over time. We experimented with other popular pooling methods (e.g., average, max, or L_2 pooling), but SWAP performed the best.

(3) **Video-level Network**: The aggregated pooled feature $\mathbf{v}$ goes through another network, embedding the final video-level feature.

While we specify the above components used in this work, we note that DBoF can be generalized with various networks and pooling methods.

3.1.2 Mixture-of-Experts Classifier. Once the video-level feature $\mathbf{v}$ is derived, we train K one-vs-all binary classifiers to estimate the probability $p(\mathbf{y}_k|\mathbf{v})$ of each label $\mathbf{y}_k$ (for $k = 1, \ldots, K$) describing the video $\mathbf{v}$. For each one-vs-all classifier, we use a Mixture-of-Experts (MoE) model [16] which summarizes the 'opinions' $p(\mathbf{y}_k|\mathbf{v}, e)$ from a set of 'experts' $e \in \mathcal{E}_y$ weighted by $p(e|\mathbf{v})$:

$$p(\mathbf{y}_k|\mathbf{v}) = \sum_{e \in \mathcal{E}_y} p(e|\mathbf{v}) p(\mathbf{y}_k|\mathbf{v}, e). \tag{2}$$

We specifically use a binary logistic regression classifier

$$p(\mathbf{y}_k|\mathbf{v}, e) = \sigma(\mathbf{w}_e^T \mathbf{v}) \tag{3}$$

for each expert and let $p(e|\mathbf{v})$ be a softmax over $|\mathcal{E}_k| + 1$ experts with a dummy state for the non-existence of the label $\mathbf{y}_k$.

Similar to DBoF, the choice of classifier is not strictly limited to MoE. For this work, we focus on MoE for the following two reasons: 1) it has been shown to be a powerful classifier among many successful video annotation models [3, 29], and more importantly, 2) it can fully take advantage of our proposed framework in the next section, significantly improving the overall performance at scale.

4 LARGE-SCALE VIDEO ANNOTATION

As mentioned earlier, training a large DBoF model requires significant computational power at scale. In this section, we first describe our distributed training framework based on MapReduce [7], which enables parallelism in both model and data. Then, we show how it applies to the DBoF model to perform scalable operations for the large-scale video annotation task.

4.1 Alternating Large-scale Trainer

A naive implementation of the model in Fig. 2 is not scalable. As the number of model parameters in the classifier part in Fig. 2 grows with the number of labels and experts, backpropagating gradients from the classifier to the video-level network would be the bottleneck. However, it is desirable to have large vocabulary set as well as a large number of experts per classifier, especially for large-scale data to flexibly cover a variety of topics (see Section 4.2 for more discussion).

In order to alleviate this bottleneck, we propose an alternating update scheme between classifier and frame aggregator which updates one while fixing the other. Then, each part can be efficiently updated via model and data parallelism. Our training framework contains three steps:

- **Step 1: Joint Training.** We first jointly train both the frame aggregator and MoE classifier. We intentionally use small MoE (i.e., ≤ 5 experts) to speed up the initial training and optimize via a mini-batch stochastic method (i.e., ADAM) to prevent early overfitting. This is a "warm-start" stage where the performance is solely based on the model itself without distributed computation. After the model converges, proceed to Step 2.
- **Step 2: Large MoE Training.** At this step, the frame aggregator is fixed and not updated. We replace the small MoE from Step 1 with a newly initialized large MoE. Each expert is trained in parallel via *model parallelism*. Upon convergence, proceed to Step 3.
- **Step 3: Frame Aggregator Fine-tuning.** We fix the MoE and fine-tune the frame aggregator via *data parallelism*, namely iRProp$^+$. We do not fine-tune the MoE, although possible, as the benefit is less substantial. Once converged, go back to Step 2.

We repeat Step 2 and 3 until convergence. Both Step 1 and 3 ensure convergence, and Step 2 also converges quickly despite the retraining of MoE because each expert is a very simple classifier (essentially a perceptron). In our experiments, we observe small to no loss of performance after several epochs, and we observe that retraining the MoE repeatedly after each alternation is actually more beneficial than continuously training the MoE. More in-depth observations are described in Section 5.2.1.

MapReduce framework [7] has been recognized as a versatile solution to efficiently manage large-scale problems through distributed computing, when a problem can be broken down into independent pieces. The **Map** step distributes the pieces to multiple workers which run in parallel. Then, once their jobs are complete, the **Reduce** step aggregates the results to proceed with the next global operation. This "divide-and-conquer" approach scales well given a sufficient number of available workers. Our framework effectively utilizes MapReduce to perform Step 2 and 3 efficiently:

(1) **Model Parallelism**: As we freeze the frame aggregator in Step 2, only the classifier for each class becomes trainable. Then, since the classes are independent of each other, the MoEs are also independent of each other. This allows the *models to be trained in parallel* which in effect allows larger MoEs to be trainable. Specifically, our framework **Maps** the partitioned classes and their respective models to the workers, updates their parameters in parallel, and then **Reduces** them back to a single DBoF model. Given well trained frame aggregating networks, this scheme allows the MoEs to scale tens of thousands of classes.
(2) **Data Parallelism**: In machine learning, samples are often assumed to be independent and identically distributed (i.i.d.), and gradients are computed within a mini-batch of randomly chosen hundreds of samples, assuming they can reasonably represent the entire dataset. With billions of examples, however, it becomes harder to represent the entire dataset unless we significantly increase the mini-batch size, which is also prohibitive. Our framework allows the gradient computation in parallel (**Map**) from a larger pool of independent examples and aggregates it (**Reduce**) with a large batch size [34]. With this, we even take full-batch gradient computation with billions of examples.

Given our scalable framework, we next describe the algorithmic aspect of the model and training parallelism described above. Ultimately, our goal is to identify and verify that these simple approaches could consistently improve the DBoF models that already perform solidly.

4.2 Large Mixture-of-Experts

Compared to global classifiers that classify all classes with equally structured classifier models, the key advantage of using a set of local classifiers such as MoE is its ability to flexibly train based on unique characteristics of the class. Consequently, having more experts becomes especially useful as the number of classes gets larger and as those classes cover a wide variety of topics. Zhu et al. [47] presented a promising result that larger MoE actually yields higher classification accuracy with video-level features on YouTube-8M dataset.

It is not trivial, however, to increase the number of experts with a large-scale dataset. With respect to the DBoF framework given K possible labels, constructing a DBoF model with MoE of $|\mathcal{E}|$ binary classifier experts for *each* label requires $K|\mathcal{E}|$ experts in total. This quickly becomes problematic with a large-scale dataset having thousands of labels (i.e., $K = 4{,}716$ for YouTube-8M) with a

moderate hidden representation size (e.g., 2,048) resulting in a MoE with approximately 10M×$|\mathcal{E}|$ variables to train.

Fortunately, the weights $\mathbf{w}_e$ in Eq. (3) of each and every expert $e \in \mathcal{E}_k$ for all $k = 1, \ldots, K$ labels can be trained independently from each other. In practice, we partition K classes into M workers to train the experts corresponding to the classes, drastically reducing the training time proportional to the number of available workers in $O(|\mathcal{E}|K/M)$ as we evenly distribute the classes to the workers.

4.2.1 Adaptive Mixture-of-Experts. We also notice that the classes with a different number of positive examples need a different number of experts. That is, labels with a small number of examples require fewer experts to avoid overfitting or to reduce unnecessary experts. To alleviate this, for each label $\mathbf{y}_k$ we bound the maximum number of experts to be $|\mathcal{E}_{\max}|$ and determine the adjusted number of experts $|\mathcal{E}_k|$ based on the number of positive examples in the dataset as follows:

$$|\mathcal{E}_k| = \min\left\{\frac{\sum_{i=1}^{N} \mathbb{1}[\mathbf{y}_k^{(i)} = 1]}{q}, |\mathcal{E}_{\max}|\right\} \tag{4}$$

where the summation in the numerator is the number of samples with the k'th label $\mathbf{y}_k$ and $q > 0$ is a constant scaling the number of experts to that sum. Thus, increasing q enforces the classes with small number of examples to have fewer experts. This adaptive scheme controls the balance between the number of examples per class and its number of experts.

4.3 Full-batch Fine-tuning

Previous work [41] have acknowledged the value of large batch training for faster convergence but could not further increase the mini-batch size (i.e., 32K) under practical limitations. Given the efficient data parallelism with our scalable framework, however, we strategically apply the large batch optimization as follows.

First, we train the model with a standard mini-batch solver (Step 1 in Section 4.1) to obtain fast initial training while minimizing early overfitting which is more detrimental. This is a safe and secure approach as demonstrated by other DBoF models [1, 3, 29].

Upon convergence, the model becomes sensitive to further updates, so robustness is the key for performing effective fine-tuning. Thus, we further fine-tune the model with a robust *full-batch optimization*, namely the *Improved Resilient Backpropagation* (RProp [33]) called *iRProp*$^+$ [15]. We explore this traditional full-batch optimization method for its robustness with very few parameters and performance competitive to even second-order methods. Briefly, the full-batch gradient is computed by summing over the gradient with respect to every training example in the entire training dataset. Then, depending on the gradient direction compared to the previous iteration, the learning rate of each weight changes.

Our framework allows an efficient full-batch gradient computation via MapReduce, described in Algorithm 1. Given the current weights $\mathbf{w}$, we first compute the gradient $[\partial E/\partial \mathbf{w}]_{\mathbf{x}=\mathbf{x}^{(i)}}$ for each training example $\mathbf{x}_i$ across the *entire* full-batch *in parallel* in the **Map** step. Then, following the RProp gradient computation scheme, we collect those distributed gradients in the **Reduce** step and sum them up to compute the full-batch gradient $\mathbf{g}$. The proceeding update step follows iRProp$^+$ (see [15] for details), but we note that any optimization scheme which could benefit from full-batch gradients (e.g., full-batch SGD) may integrate to this framework. For N examples with $|\mathbf{w}|$ weights and M workers, the framework can compute the gradient in $O(|\mathbf{w}|N/M)$ once we assign similar number of examples to each worker.

Algorithm 1 MapReduce-based iRProp$^+$ Algorithm

Given:
1) Learning rate coefficients η^+ and η^-
2) Learning rate bounds $\Delta_{\max} = 10$ and $\Delta_{\min} = 0$
3) Initial network weights $\mathbf{w}$
for $t = 1, 2, \ldots$ **do**
  Map: Compute gradient for each sample $\mathbf{x}^{(i)}$ over the full-batch in parallel
    $\mathbf{g}^{(i)} \leftarrow \left[\frac{\partial E}{\partial \mathbf{w}}\right]_{\mathbf{x}=\mathbf{x}^{(i)}} \quad \forall i = 1, \cdots, N$
  Reduce: $\mathbf{g} \leftarrow \sum_{i=1}^{N} \mathbf{g}^{(i)}$ over the full-batch
  $\mathbf{w} \leftarrow \text{iRProp}^+\left(\mathbf{g}, \eta^+, \eta^-, \Delta_{\max}, \Delta_{\min}\right)$
end for

5 EXPERIMENTS

We evaluate the video annotation performance of our models on two public benchmark datasets. We also analyze how our MapReduce framework allows the models to scale up efficiently and further improves the performance by experimenting with actual YouTube scale samples.

5.1 Experimental Setup

5.1.1 Datasets. We use two large-scale public video datasets. First, **YouTube-8M**[3] [1] is the largest public video dataset of 7M+ YouTube videos (400K+ hours) over 4,716 possible classes (3.4 classes per video on average) totaling up to 3.2B features. 70% and 20% of the dataset are publicly available as training and validation sets respectively, and we randomly select 10% of the validation set (2% of the full dataset) to evaluate our models, and use the rest for training, similarly to WILLOW [29]. **Sports-1M** [17] has 1.2M YouTube video URLs with labels on 487 sports activity classes.

For both datasets, the video features are extracted from Inception-v3 features [36] trained on the ImageNet dataset [21] followed by PCA (with whitening) and 8-bit quantization for one frame per second (1 fps). The audio features for each second of the video are extracted using an acoustic ResNet-50 [12], followed by a short-time Fourier transform for ResNet [11]. The final dimensions of the video and audio features are 1024 and 128 respectively (see [1, 23] for details).

5.1.2 Model Specifications. We construct three DBoF models with different DBoF frame aggregators (the top box in Fig. 2) with the architectures shown in Fig. 3. **DBoF-Basic** (Fig. 3a) consists of a fully-connected (FC) layer for the frame-level network before SWAP pooling (Eq.(1)) and another FC layer as the video-level network. **DBoF-Cgate** (Fig. 3b) adds a context gating layer [29], aiming at modeling interdependencies between features, to the frame-level and video-level networks of DBoF-Basic. Lastly, in **DBoF-ResNet**

[3] We use YouTube-8M ver. 2, released in February 2017 to compare against reported results with the same version. We also experimented with ver. 3 (May 2018) and observed similar improvement.

(Fig. 3c), a standard ResNet block with a skip-connection [11] is added to the frame-level network of the DBoF-Basic.

For each FC layer, the output dimensions are 2,048 for all models, and batch normalization and sigmoid function are applied except for the first FC in the ResNet block.

5.1.3 Training Procedure. For the joint training stage, we randomly sample 30 frames for each video and used ADAM with $\alpha = 0.9$ and $\beta = 0.999$, initial learning rate of 0.005, decay rate of 0.95 every 5,000 mini-batches of size of 512 and 0.9 momentum. The maximum number of experts $|\mathcal{E}_{\max}|$ is set to 5 for all models for the joint training. For the fine-tuning, we use iRProp$^+$ (also referred to as RProp for simplicity) with the global learning rate of 10^{-5} and the standard parameter values as shown in Algorithm 1. We minimize the cross-entropy loss across all classes.

5.1.4 MapReduce Setups. We use NVIDIA Tesla p100 GPU to perform matrix/tensor computations in the joint training stage (Step 1) with small MoE. For the large MoE training (Step 2), we partition the video classes into 500 partitions and trained them in parallel. For the full-batch fine-tuning (Step 3), we use 500~5000 workers each computing gradients for a small group of samples in parallel.

5.1.5 Evaluation Metrics. We evaluate our models with four metrics listed below.

1) Global Average Precision (GAP) computes the area under the precision-recall curve, evenly taking all video-entity pairs into account together:

$$GAP = \frac{100}{K|V|} \sum_{v \in V} \sum_{y \in Y} P(r_{v,y}) \Delta R(r_{v,y}) \tag{5}$$

where $r_{v,y}$ is the rank of estimated score on video v for label y, when all scores are sorted globally. $P(i)$ and $R(i)$ are the precision and recall levels at i respectively. Recall that K is the number of possible classes. Following the YouTube-8M benchmark [1], we evaluate the top $k = 20$ predicted labels per video:

$$GAP_k = \frac{100}{K|V|} \sum_{v \in V} \sum_{y \in Y} P(r_{v,y}) \Delta R(r_{v,y}) \mathbb{1}[r_y^{(v)} \le k] \tag{6}$$

where $r_y^{(v)}$ is the video-specific rank of an estimated label y.

2) Mean Average Precision (MAP), on the other hand, takes the mean of per-class average precisions:

$$MAP_k = \frac{100}{K|V|} \sum_{v \in V} \sum_{y \in Y} P_y(r_v^{(y)}) \Delta R_e(r_v^{(y)}) \mathbb{1}[r_y^{(v)} \le k] \tag{7}$$

where $r_v^{(y)}$ is the entity-specific rank of an estimated label y, $P_y(i)$ and $R_y(i)$ are the precision and recall with respect to the label y at position i. When the classes are uniformly distributed over the test set, GAP and MAP become equivalent. In reality, videos tend to have long-tailed distributions over classes, so those with fewer examples will get penalized by MAP more than by GAP.

3) Precision at Equal Recall Rate (PERR) is similar to MAP but instead of using a fixed $k = 20$, it considers the precision up to the number of ground truth classes in each video:

$$PERR = \frac{100}{K} \sum_{y \in Y} \left[\frac{1}{|V_y|} \sum_{v \in V_y} \mathbb{1}[r_v^{(y)} \le |V_y|] \right] \tag{8}$$

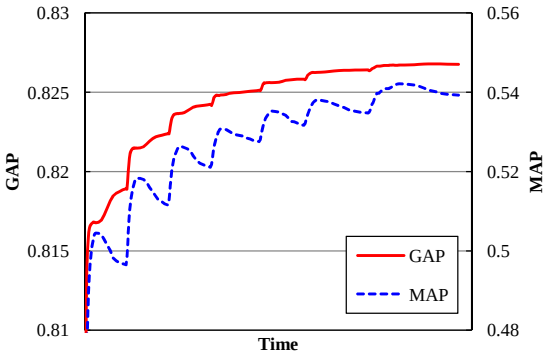


Figure 4: An example of validation error during alternating MoE retraining (Step 2) and RProp fine-tuning (Step 3)

where V_y is the set of videos with true label y and $\mathbb{1}[r_v^{(y)} < |V_y|]$ is the number of correct predictions made within the top $|V_y|$.

4) Hit@k is the ratio of the test samples with at least one ground truth label in top k predictions. We use $k = 1, 5$ for our evaluation.

5.2 Result and Analysis

We first show the results of applying either large MoE or RProp fine-tuning to observe their individual benefits. We use the YouTube-8M dataset for these analyses.

5.2.1 Trainer Behavior. Fig. 4 shows an example of validation error trajectory during alternating MoE retraining (Step 2) and RProp fine-tuning (Step 3). We observe that it steeply improves in both GAP and MAP, then GAP continues to get better with a lower rate but MAP drops slightly, and this pattern repeats.

The periods showing the steep improvements in GAP and MAP correspond to the large MoE training (Step 2). When our trainer proceeds to the large MoE training from either joint training (Step 1) or frame aggregator fine-tuning (Step 3), the old MoE is replaced with a new large MoE (adaptively, according to Eq. (4)) and trained from scratch. Although the model may initially suffer, we observe that in practice the performance drop is quickly recovered within the same epoch, thus not noticeable in the figure.

The epochs where GAP increases but MAP decreases correspond to the frame-aggregator fine-tuning (Step 3). This is because the full-batch gradient in RProp, by construction, is the sum over samples with equal weights, so the frame-aggregator is optimized to maximize GAP, while the model is overfit in terms of MAP. (Recall that GAP treats *samples* equally whereas MAP treats *classes* equally.) Despite the drop in MAP, frame aggregator fine-tuning (Step 3) is crucial for the subsequent MoE retraining and the overall model performance.

5.2.2 MoE Size. For each of the three DBoF models, we first perform the joint training with 5 experts per class until it converges. This becomes the model-specific *baseline*, where we start further optimization with large-scale approaches. We then train large MoE with $|\mathcal{E}_{\max}| \in \{10, 20, 50\}$ with $q = 3$ in Eq. (4), while fixing the frame aggregator. As shown in Table 2, we see that GAP consistently improves as the size of MoE increases, though the benefit diminishes with more complex models. On the other hand, MAP improves by a large margin in all models. Since MAP measures the precision with

Table 1: Results on YouTube-8M

Model	MoE	GAP	MAP	PERR	Hit@1	Hit@5
DBoF-Basic (baseline)	5	82.75	46.67	74.79	86.83	93.98
DBoF-Basic + RProp	5	83.00	46.72	75.05	87.05	94.10
DBoF-Basic + Large MoE	50	83.16	51.35	75.37	87.31	94.16
DBoF-Basic + RProp + Large MoE	50	**83.26**	**51.56**	**75.47**	**87.34**	**94.20**
DBoF-Cgate (baseline)	5	83.10	47.39	75.11	87.21	94.10
DBoF-Cgate + RProp	5	83.32	47.60	75.43	87.43	94.15
DBoF-Cgate + Large MoE	50	83.26	**51.74**	75.48	87.49	94.18
DBoF-Cgate + RProp + Large MoE	50	**83.42**	51.59	**75.63**	**87.57**	**94.24**
DBoF-ResNet (baseline)	5	83.48	47.86	75.55	87.58	94.28
DBoF-ResNet + RProp	5	83.73	48.14	75.73	87.58	94.29
DBoF-ResNet + Large MoE	50	83.67	51.92	75.76	87.51	94.26
DBoF-ResNet + RProp + Large MoE	50	**83.83**	**52.16**	**75.96**	**87.71**	**94.35**
monkeytyping [40]	16	81.06	-	-	-	-
FDT [3]	4~16	81.78	-	-	-	-
WILLOW [29]	2	83.00	-	-	-	-

Table 2: MoE size vs. GAP and MAP

MoE	GAP			MAP		
Size	Basic	Cgate	ResNet	Basic	Cgate	ResNet
5	82.75	83.10	83.48	46.67	47.39	47.86
10	83.04	83.23	83.62	50.10	50.80	51.32
20	83.07	83.26	83.63	50.78	51.16	51.59
50	**83.14**	**83.29**	**83.67**	**51.35**	**51.74**	**51.92**

respect to *each* label, this suggests that the less frequent classes benefit significantly from the large and/or adaptively-sized MoEs. This is desirable in practice where rare but crucial classes must be detected with high precision. The training time for 50 experts compared to 5 experts increases by only less than two folds despite the 10 times larger MoE size.

5.2.3 RProp Batch Size. The DBoF models with RProp after the joint training (DBoF + RProp in Table 1) consistently gain 0.22~0.25 GAP with respect to their baseline models. We note that the baseline DBoF models have already converged, but the RProp fine-tuning provides further improvement within 1 hour for each RProp iteration by covering the *entire* training set using *all frames* of the videos (compared to 30 frames sampled in the joint training).

We vary the batch size from 0.01% (~600 samples, close to the ADAM mini-batch size of 512) to 100% (full-batch) for RProp on the joint trained DBoF-Basic. Fig. 5 shows the performance curves of the RProp trainings using {0.01%, 0.1%, 1%, 10%, 50%, 100%} of the training set. Starting at 82.75 GAP, we clearly see improvements with increasing batch size.

5.2.4 Scalability. For the GPU-based joint training, it takes a few hours to complete one epoch, depending on the embedding size and the number of frames sampled per video. The full-batch fine-tuning only takes about 20 minutes for each gradient compute step, which is done over the entire data.

The learning curves in Fig. 5 indicate that with our MapReduce framework training speed does *not* grow directly proportional to the batch size, i.e., the 100% batch training (45 iterations in 15 hrs) does not take 10K times longer than the 0.01% batch training (109 iterations in 15 hrs). Also, after further training, we observe consistent benefits of using larger batch sizes similar to Fig. 5 and achieve the best performances when we use at least 50% (~3M) of the full-batch. This large batch training on a conventional non-parallel framework is theoretically possible by accumulating the gradients over millions of samples, but it may take days to weeks. Our framework, on the other hand, allows such large batch training in hours for the equal number of iterations.

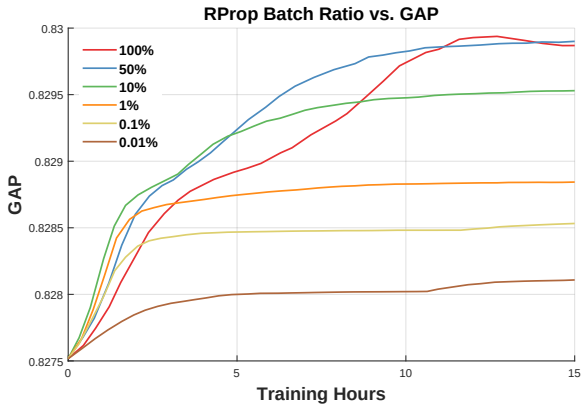


Figure 5: RProp batch size vs. GAP

5.3 Evaluation on Public Benchmarks

We compare against several state-of-the-art models with a couple of the largest public benchmark datasets: YouTube-8M and Sports-1M.

5.3.1 YouTube-8M. Table 1 summarizes the results of all the DBoF models we trained on the YouTube-8M dataset, including 1) base model, 2) with RProp, 3) with large MoE, and 4) alternating large MoE + RProp training. We compare against several top-performing models [3, 29, 40] from the YouTube-8M Large-Scale Video Understanding Challenge. While their final ensemble model achieved

Table 3: Results on Sports-1M (GAP, MAP, PERR)

Model	MoE	GAP	MAP	PERR
DBoF-Basic (Baseline)	5	76.74	73.39	71.34
DBoF-Basic + RProp	5	**78.04**	74.28	**72.63**
DBoF-Basic + Large MoE	50	77.53	74.21	72.37
DBoF-Basic + RProp + Large MoE	50	77.94	**74.30**	72.52

Table 4: Comparison against top-performing models on Sports-1M (with 30 frames)

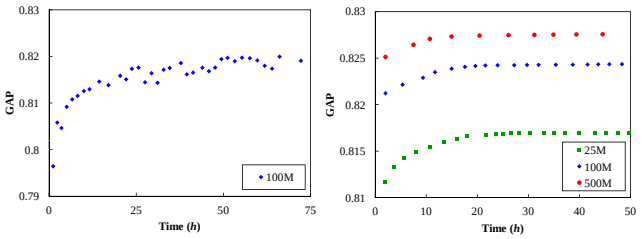
Model	Hit@1	Hit@5
P3D ResNet [31]	66.4	87.4
Convolutional Pooling [42]	71.7	90.4
DBoF-Basic + RProp + Large MoE (Ours)	**73.0**	**90.8**

higher GAP (84.69), we compare against their reported best single models for fair comparison. We list our observations below:

- We first see that DBoF-ResNet performs best, out of the 3 baseline DBoF models, followed by DBoF-Cgate. Their relative order is the same, with or without large MoEs, and with or without full-batch fine-tuning.
- **RProp Only** and **Large MoE Only**: We observe that for all three DBoF models, both large MoE (shown with finer sizes in Table 2) and RProp fine-tuning individually show improvements over their corresponding baseline joint training models for all metrics. We note that the baseline DBoF-Basic (82.75 GAP) was originally worse than the WILLOW model (83.00 GAP) but the RProp fine-tuning itself has improved to be competitive (83.00 GAP) with no changes to the model itself. The GAPs for DBoF-Cgate and DBoF-ResNet both consistently improve by 0.22 and 0.25 respectively from fine-tuning only, making it a worthwhile consideration for a converged model. Alternating between RProp and the initial small MoE training (i.e., without Step 2) does not provide a notable improvement. The MAPs also improve, especially by a large margin with larger MoEs.
- **Full model (RProp + Large MoE)**: Except for the MAP of DBoF-Cgate, the best performances of all the DBoF models for all metrics are achieved with the full model (DBoF + RProp + Large MoE) which performs both MoE and RProp fine-tuning together in the alternating optimization scheme. In practice, we need to perform two or three alternations to achieve the overall convergence.

5.3.2 Sports-1M. Our large-scale framework generalizes well to another popular benchmark set, Sports-1M, as shown in Table 3. Overall, we see even larger improvements compared to the YouTube-8M results. Also, we observe larger impact by RProp than by large MoE, probably due to the relatively smaller dataset scale compared to YouTube-8M.

Table 4 compares our full model against a couple of state-of-the-art models on Sports 1M. To follow their experimental setting, we use 30 frames for evaluation and evaluate in Hit@$\{1, 5\}$. We see that our model outperforms both state-of-the-art models.



(a) Frame-level **(b) Video-level**

Figure 6: Validation performances (in GAP) on large-scale datasets.

5.4 Evaluation on Scalability

Lastly, we demonstrate the scalability of the proposed framework on a larger dataset with public videos on the web. We sample 500M videos from YouTube for training and additional 10M for evaluation. We use 16,675 classes, and ground truth labels extracted from an internal video annotation system, which leverages metadata, content, and user signals to annotate the main topics of a video. To represent each video, we extract 1500-dimensional visual features for each frame using an in-house image classification model similar to Inception-v4 [35], 2048-dimensional audio features for each second of audio [12], and a 256-dimensional CDML video-level embedding [23].

We first train the full DBoF model illustrated in Fig. 3 with frame-level features for 30 randomly chosen frames on a subset of 100M videos. Fig. 6 shows the Global Average Precision (GAP) on a validation set during the first 2–3 days of training. Fig. 6a shows that with our framework the DBoF model converges within a reasonable time (~72 hours).

For even faster training, we fix the frame-level network in Fig. 3 and train on average-pooled video-level features. We observe that all three models converge within a day; the model trained on the smallest dataset (25M videos) takes about 15 hours, the intermediate one (100M) about 20 hours, and the largest dataset (500M) takes about 24 hours until convergence. We also observe that larger training data leads the model to converge with higher annotation accuracy. Compared to the full frame-level model, this simplified model converges in < 24 hours, with a reasonable performance.[4]

6 CONCLUSION

In this work, we present how to annotate videos at industry scale, with a MapReduce-based training framework redesigned for large-scale complex video classification models. With billions of videos in tens of thousands of label space, we show our proposed framework trains state-of-the-art models within a day. Also, we empirically demonstrate that our DBoF-based models significantly outperform on large-scale public benchmark datasets, thanks to large mixture of experts (model parallelism) as well as full-batch fine-tuning with RProp (data parallelism). Naturally, our future work is to explore the applicability of our framework to other large-scale datasets of different kinds and to more diverse networks.

[4]Note that we do not compare the GAP between these two setups since their validation sets are not equivalent, and the purpose of this experiment is purely to test the scalability of the proposed framework beyond the scale of YouTube-8M.

REFERENCES

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. 2016. Youtube-8M: A Large-scale Video Classification Benchmark. *arXiv preprint arXiv:1609.08675* (2016).

[2] Vladimir Aliev, Pavel Ostyakov, Roman Suvorov, Gleb Sterkin, Elizaveta Logacheva, Oleg Khomenko, and Sergey Nikolenko. 2018. Label Denoising with Large Ensembles of Heterogeneous Neural Networks. In *Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding.*

[3] Shaoxiang Chen, Xi Wang, Yongyi Tang, Xinpeng Chen, Zuxuan Wu, and Yu-Gang Jiang. 2017. Aggregating Frame-level Features for Large-Scale Video Classification. In *Proc. of the CVPR Workshop on YouTube-8M Large-Scale Video Understanding.*

[4] Choongyeun Cho, Benjamin Antin, Sanchit Arora, Shwan Ashrafi, Peilin Duan, Dang The Huynh, Lee James, Hang Tuan Nguyen, Moji Solgi, and Cuong Van Than. 2018. Axon AI's Solution to the 2nd YouTube-8M Video Understanding Challenge. In *Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding.*

[5] Cheng-Tao Chu, Sang K Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Kunle Olukotun, and Andrew Y Ng. 2007. Map-Reduce for machine learning on multicore. In *Advances in neural information processing systems (NIPS).*

[6] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems (NIPS).*

[7] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (2008), 107–113.

[8] Basura Fernando, Efstratios Gavves, José Oramas, Amir Ghodrati, and Tinne Tuytelaars. 2017. Rank pooling for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 4 (2017), 773–787.

[9] Shivam Garg. 2018. Learning Video Features for Multi-Label Classification. In *Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding.*

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

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR).*

[12] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. 2017. CNN architectures for large-scale audio classification. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*

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

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely Connected Convolutional Networks.. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[15] Christian Igel and Michael Hüsken. 2003. Empirical evaluation of the improved Rprop learning algorithms. *Neurocomputing* 50 (2003), 105–123.

[16] Michael I Jordan and Robert A Jacobs. 1994. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* 6, 2 (1994), 181–214.

[17] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proc. of the IEEE conference on Computer Vision and Pattern Recognition (CVPR).*

[18] Eun-Sol Kim, Jongseok Kim, Kyoung-Woon On, Yu-Jung Heo, Seong-Ho Choi, Hyun-Dong Lee, and Byoung-Tak Zhang. 2018. Temporal Attention Mechanism with Conditional Inference for Large-Scale Multi-Label Video Classification. In *Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding.*

[19] Sebastian Kmiec and Juhan Bae. 2018. Learnable Pooling Methods for Video Classification. In *Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding.*

[20] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision* 123, 1 (2017), 32–73.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems.*

[22] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. 2008. Learning realistic human actions from movies. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[23] Joonseok Lee, Sami Abu-El-Haija, Balakrishnan Varadarajan, and Apostol Natsev. 2018. Collaborative Deep Metric Learning for Video Understanding. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

[24] Joonseok Lee, Apostol Natsev, Walter Reade, Rahul Sukthankar, and George Toderici. 2018. The 2nd YouTube-8M Large-Scale Video Understanding Challenge. In *Proc. of the European Conference on Computer Vision (ECCV).*

[25] Fu Li, Chuang Gan, Xiao Liu, Yunlong Bian, Xiang Long, Yandong Li, Zhichao Li, Jie Zhou, and Shilei Wen. 2017. Temporal modeling approaches for large-scale Youtube-8M video understanding. In *Proc. of the CVPR'17 Workshop on YouTube-8M Large-Scale Video Understanding.*

[26] Xiaodan Liang, Lisa Lee, and Eric P Xing. 2017. Deep variation-structured reinforcement learning for visual relationship and attribute detection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[27] Rongcheng Lin, Jing Xiao, and Jianping Fan. 2018. NeXtVLAD: An Efficient Neural Network to Aggregate Frame-level Features for Large-scale Video Classification. In *Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding.*

[28] Tie-Yan Liu, Wei Chen, and Taifeng Wang. 2017. Distributed machine learning: Foundations, trends, and practices. In *Proceedings of the 26th International Conference on World Wide Web Companion.* 913–915.

[29] Antoine Miech, Ivan Laptev, and Josef Sivic. 2017. Learnable Pooling with Context Gating for Video Classification. In *Proc. of the CVPR Workshop on YouTube-8M Large-Scale Video Understanding.*

[30] Xinghao Pan, Maximilian Lam, Stephen Tu, Dimitris Papailiopoulos, Ce Zhang, Michael I Jordan, Kannan Ramchandran, and Christopher Ré. 2016. Cyclades: Conflict-free asynchronous machine learning. In *Advances in Neural Information Processing Systems.* 2568–2576.

[31] Zhaofan Qiu, Ting Yao, and Tao Mei. 2017. Learning spatio-temporal representation with pseudo-3d residual networks. In *2017 IEEE International Conference on Computer Vision (ICCV).* IEEE, 5534–5542.

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS).*

[33] Martin Riedmiller and Heinrich Braun. 1993. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE International Conference on Neural Networks.*

[34] Samuel L Smith, Pieter-Jan Kindermans, and Quoc V Le. 2017. Don't Decay the Learning Rate, Increase the Batch Size. *arXiv preprint arXiv:1711.00489* (2017).

[35] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, Inception-ResNet and the impact of residual connections on learning.. In *Proc. of the AAAI Conference on Artificial Intelligence.*

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR).*

[37] Yongyi Tang, Xing Zhang, Jingwen Wang, Shaoxiang Chen, Lin Ma, and Yu-Gang Jiang. 2018. Non-local NetVLAD Encoding for Video Classification. In *Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding.*

[38] Gül Varol, Ivan Laptev, and Cordelia Schmid. 2018. Long-term temporal convolutions for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 6 (2018), 1510–1517.

[39] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. 2009. Evaluation of local spatio-temporal features for action recognition. In *Proc. of the British Machine Vision Conference (BMVC).*

[40] He-Da Wang, Teng Zhang, and Ji Wu. 2017. The Monkeytyping Solution to the Youtube-8M Video Understanding Challenge. In *Proc. of the CVPR Workshop on YouTube-8M Large-Scale Video Understanding.*

[41] Yang You, Igor Gitman, and Boris Ginsburg. 2017. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888* (2017).

[42] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. 2015. Beyond short snippets: Deep networks for video classification. In *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR).*

[43] Ruiliang Zhang and James Kwok. 2014. Asynchronous distributed ADMM for consensus optimization. In *Proc. of the International Conference on Machine Learning (ICML).*

[44] Sixin Zhang, Anna E Choromanska, and Yann LeCun. 2015. Deep learning with elastic averaging SGD. In *Advances in Neural Information Processing Systems.*

[45] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid scene parsing network. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[46] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[47] Linchao Zhu, Yanbin Liu, and Yi Yang. 2017. UTS submission to Google YouTube-8M Challenge 2017. In *Proc. of the CVPR'17 Workshop on YouTube-8M Large-Scale Video Understanding.*

[48] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*