ABSTRACT

We tackle the problem of learning concept classifiers from videos on the web without using manually labeled data. Although metadata attached to videos (e.g., video titles, descriptions) can be of help collecting training data for the target concept, the collected data is often very noisy. The main challenge is therefore how to select good examples from noisy training data. Previous approaches firstly learn easy examples that are unlikely to be noise and then gradually learn more complex examples. However, hard examples are much different from easy ones are never learned. In this paper, we propose an approach called multimodal co-training (MMCo) for selecting good examples from noisy training data. MMCo jointly learns classifiers for multiple modalities that complement each other to select good examples. Since MMCo selects examples by consensus of multimodal classifiers, a hard example for one modality can still be used as a training example by exploiting the power of the other modalities. The algorithm is very simple and easily implemented but yields consistent and significant boosts in example selection and classification performance on the FCVID and YouTube8M benchmarks.

1 INTRODUCTION

Learning concept classifiers for videos is a fundamental task for automatic understanding of video and has many practical applications. Recent advances in deep neural networks enable very accurate classifiers to be learned by training on large-scale datasets with manually annotated labels (e.g., Kinetics dataset [23]). However, collecting such datasets involves huge amounts of human labor. Therefore, it is preferable to learn classifiers without using any manually labeled data.

One of the promising approaches to learning without labeled data is learning from web videos containing rich metadata (e.g., titles, descriptions, keywords). Such data are much easier to collect, and datasets consisting of millions of videos are available, e.g., YouTube8M [1], YFCC100M [36]. Although the training data for the target concept can be easily collected by using text-based matching between the concept name and metadata, the collected data are often noisy. The main challenge is therefore how to select good examples from noisy training data.

Liang et al. [26] proposed an approach to learning concept classifiers from web videos with noisy labels. The key idea is to learn easy examples first [2, 21, 25]. That is, easy examples that are very likely to be positive are learned first; then gradually more complex examples are learned. This idea is similar to self-training [40] that is widely used in semi-supervised learning.

Although this approach can select correct examples that are easy, it sometimes misses hard correct examples. Consider hard samples that are much different from any easy sample; a classifier trained on only easy samples will never classify them as positive. In other research fields such as object detection, hard examples have been shown to be useful for improving a classifier. The basic

Figure 1: In webly labeled learning, confident samples are selected as training examples from noisy data. (a) Previous approaches often miss hard examples that would potentially improve classifier performance. (b) The proposed approach can learn hard examples by exploiting the information of other modalities.
approach of hard example mining is to select examples with high loss [12, 16, 30, 31]. However, hard examples with high loss are likely to be noise in learning on noisy data. For example, in Fig. 1a, while the first easy sample can be used as a positive example, the second hard sample cannot be used because it is possible to be noise. 

How can we select good examples that are hard and correct? Our key idea is to use information of multiple modalities. Figure 1 shows an example of learning a ‘dog’ classifier. The dog in the dark (Fig. 1b) is a hard example with only visual information because its appearance is much different from dogs in the light; such an example would not be selected in self-learning based approach. However, by using information of other modalities (e.g., the title ‘a dog is barking’ or sounds of dogs barking), we can confidently select it as a positive example.

On the basis of this intuition, we propose an approach called multimodal co-training (MMCo) for selecting good examples by using multimodal knowledge. MMCo jointly learns the classifiers for multiple modalities and selects examples by voting of multiple classifiers (e.g., taking their max score). Therefore, a hard example for one modality can be used as a training example by exploiting the other modalities. The algorithm is very simple and easily implemented but significantly improve the performance of example selection from noisy training data; precision/recall is improved from 0.66/0.64 to 0.70/0.80 on FCVID benchmark.

In addition, we introduce an online learning algorithm for ‘webly’ labeled learning problem [26]. Although existing approaches [26, 27] uses an alternating optimization that performs joint optimization of the model and the weights of the examples, it has trouble handling large-scale datasets consisting of millions of videos. Here, we extend the original webly labeled learning (WELL) [26] to an online setting by computing the sample weights online for each minibatch. This online WELL can handle a large-scale dataset without degrading accuracy. Online WELL can be extended to multiple classes easily and efficiently like other stochastic gradient descent(SGD)-based classifier learning methods.

2 ONLINE WEBLY LABELED LEARNING

Our goal is to learn a concept detector from web videos without any manually annotated labels. Liang et al. [26] proposed Webly Labeled Learning (WELL) approach to this problem, where the classifier parameters and sample weights (selected examples) are alternately optimized. However, alternating optimization based on batch training is difficult to apply to a large-scale dataset consisting of millions of videos because all the training data should be processed in every update. We therefore decided to extend WELL to an online setting by selecting samples and computing their weight for each minibatch online. In this section, we first describe the problem of webly labeled learning and review WELL (Sec. 2.1). Then we describe online WELL (Sec. 2.2).

2.1 Background: Webly Labeled Learning

Problem formulation. Webly labeled learning was introduced in [26]. In it, one considers the problem of learning a binary classifier \( g \) for the concept from \( N \) unlabeled web videos \( \{(x_n, t_n)\}_{n=1}^N \), where \( x_n \) is the feature of \( n \)th video, and \( t \) is the metadata (text data) attached to it. The pseudo label of each example \( y_n \in \{0, 1\} \) is inferred by text-based matching between the concept name and the metadata of each video \( t_n \); i.e., \( y_n = 1 \) is assigned if the concept name \( c \) appears in the metadata \( t_n \), and \( y_n = 0 \) if it does not appear. The confidence \( v_n^0 \in \mathbb{R} \) is obtained from the accuracy of matching; a high confidence value is assigned if \( c \) frequently appears in \( t_n \) (for negative-labeled samples). Consequently, the problem reduces to one of learning the classifier given the videos with pseudo labels and their confidences \( \{(x_n, y_n, v_n^0)\} \).

The problem of webly labeled learning is related to semi-supervised [6] and noisy-labeled learning [14]. In noisy-labeled learning, a classifier is learned using samples with noisy (corrupted) labels \( (x_n, y_n) \). While all samples should be treated equally in noisy-labeled learning, in webly labeled learning, the label confidence \( v_n^0 \) of each sample can be obtained from metadata as prior knowledge. In semi-supervised learning, a classifier is learned using both labeled and unlabeled samples. While labeled samples are completely trusted in semi-supervised learning, in webly labeled learning, samples with high confidence labels are not always positive.

Model. We here describe the Webly Labeled Learning (WELL) model [26] that was proposed to solve the problem formulated above. WELL is based on the self-paced curriculum learning theory proposed in [21]. The model parameters \( w \) (e.g., linear classifier weights) and the latent weight variable \( v = [v_1, v_2, \ldots, v_N]^T \) are jointly learned by alternating optimization. The objective function is defined as

\[
\min_{w,v} \mathbb{E} L(w, v; \lambda) = \sum_{n=1}^{N} v_n L(y_n, g(x_n, w)) + f(v; \lambda),
\]

where \( g \) is the classifier, \( L \) is the loss function, and \( v = [v_1, v_2, \ldots, v_N]^T \) denote the latent weight variables that reflect label confidences. Examples with larger weights tend to be learned with high priority. \( f \) is a self-paced regularizer [25] that controls the learning process so that easy samples with small losses tend to be selected earlier. For example, the linear regularizer proposed in [21] is

\[
f(v; \lambda) = \frac{1}{2} \lambda \sum_{n=1}^{N} (v_n^0 - 2v_n)
\]

where \( \lambda \) controls the learning pace; i.e., \( \lambda \) is gradually increased to increase the number of samples used. With a fixed \( w \), the weight variable can be optimized in a closed form as

\[
v_n = \begin{cases} 1 - \frac{2}{\lambda} l_n & \text{if } l_n < \lambda, \\ 0 & \text{if } l_n \geq \lambda, \end{cases}
\]

where \( l_n \) is the loss for the \( n \)th sample. Therefore, examples with losses smaller than \( \lambda \) are used for training.

\( v \) is initialized by the pseudo label’s confidence \( v^0 \) computed by word matching between the concept name and metadata, which corresponds to the intuition of curriculum learning [2] that determines the learning sequence from external prior knowledge. WELL determines the learning sequence by using both prior knowledge in the manner of curriculum learning and the model in the manner of self-paced learning. In this way, WELL can learn from easy examples as well as from more complex ones.

Below, we summarize the implementation of WELL.

(0) Initialize the sample weights \( v \) from the label confidence \( v^0 \).
(1) Update the model \( w \) with fixed sample weights \( v \).
(2) Compute the losses of all examples by using the updated model.
(3) Compute the weights \( v \) of all examples by using Eq. 3.
(4) Increase age \( \lambda \). Repeat (2)–(5) until convergence.

2.2 Online Webly Labeled Learning

Although WELL learns the classifier by alternating minimization, it is not appropriate for the training on a large-scale dataset. In alternating optimization, all training samples are used to optimize the model \( w \) in every iteration, wherein the sample weights \( v \) are fixed when optimizing the model. If the training data is very large (e.g., millions of samples), it takes a very long time to complete one iteration, which makes convergence very slow. In addition, it is generally infeasible to put whole dataset in memory. Hence, we decided to extend the webly labeled learning model in [26] to the online learning based on stochastic gradient decent (SGD).

The main problem is how to select optimal samples and compute their weights in a minibatch. If the sample weights \( v \) are given, the model can be stochastically optimized over the minibatch in the manner of SGD. Although alternating optimization generally requires all samples in order to optimize each variable, in the binary scheme of WELL, the weight \( v \) can be computed using only the loss of the sample \( l_n \) and the model age \( \lambda \) using Eq. 3. Therefore, we can compute the sample weight \( v \) online after computing the loss of each sample \( l_n \), where the model is updated with the weighted loss \( v l_n \). The age is determined by the scheduler described below.

More specifically, the online webly labeled learning algorithm is as follows.

(0) Initialize model \( w \) and age \( \lambda \).
(1) Randomly sample a minibatch with \( N_m \) examples.
(2) Compute the losses \( l_1, \ldots, l_{N_m} \) using the model \( w \). Cross entropy loss is used in this study.
(3) Compute the weights \( v_1, \ldots, v_{N_m} \) of the samples in the minibatch with Eq. 3 and the computed losses (the confidences of the pseudo labels \( v \) are used in the first few iterations.)
(4) Update \( w \) by using the weighted losses, i.e., \( \sum_{i=1}^{N_m} v_i l_i \)
(5) Increase the age \( \lambda \) based on the age scheduler described below.

Repeat (2)–(5) until convergence.

We should note that the training cost of online WELL is less than that of WELL. WELL needs to compute the loss twice in each iteration in order to update the weights \( w \) and the model \( v \). On the other hand, we have to compute the loss only once; sample weights \( w \) are computed using the loss, and the model \( v \) is updated using the weighted loss.

**Age scheduling.** Model age \( \lambda \) is an important parameter that determines which samples to select as examples. Manually determining the age schedule (initial value, early stopping value, etc.) is hard; in practice, the age is controlled in accordance with the rate of used samples in the positive pool \( p \) (e.g., 30% positive samples \( p_{0.3} \) are first used, and \( \lambda \) is gradually increased until 60% positive samples \( p_{0.6} \) are used), which can be specified more intuitively. The age \( \lambda \) is computed from \( p \) by sorting all positive samples, where the loss at the 100th percentile is \( \lambda \). However, in online learning, not all of the training samples are available for the updates, the age cannot be computed from \( p \).

To control \( \lambda \) in online learning, we introduce a FIFO queue that stores the losses of the previous iterations. We can compute \( \lambda \) from \( p \) by regarding the samples in the queue as a whole training set. There are two approximations here. First, the samples in the queue approximate the whole set of training samples; the size of the queue should be large enough to hold enough samples. Second, the losses in the queue are computed from the previous model; they are different from the loss computed by the current model. The size of the queue should not be too large because of this. Although there is a trade-off between these two approximations, the same performance as in batch training can be achieved by selecting an appropriate queue size.

**Multiple classes.** Online WELL is easily extended to multiple classes like other multilabel classification models learned by SGD. Given a dataset \( \{ (x_n, y_n) \}_{n=1}^{N} \), the goal is to learn a multilabel classifier that outputs the confidence of \( C \) classes, where \( y_n = [y_{n,1}, \ldots, y_{n,C}]^T \in \{0,1\}^C \) are labels for \( C \) classes. Note that we do not assume that the classes are mutually exclusive; therefore, the samples can be positive for multiple classes. For each iteration, the age \( \lambda_c \) is determined for each class independently following the age scheduling described above, whereby the weights of the samples \( v \in \mathbb{R}^{N \times C} \) are computed for every class. The loss in a minibatch can be computed as

\[
\sum_{n=1}^{N} \sum_{c=1}^{C} v_{n,c} l(x_n, y_{n,c}, g(x_n, w, c)),
\]

where \( g(\cdot, \cdot) \) is the prediction for the \( c \)th class. By sharing the computation, this algorithm enables multiple classifiers to be learned much more efficiently than learning each classifier independently, especially computationally heavy classifiers like deep neural networks.

3 MULTIMODAL CO-TRAINING

In webly labeled learning, easy examples with small losses are preferentially used as training data. Correct but hard examples that would potentially improve performance are never learned. As a solution to this problem, we propose multimodal co-training (MMCo) as a way to use multimodal information in web videos (e.g., visual, audio, and metadata), which enables us to exploit hard examples. For example, a sample with a small loss in the audio modality but large loss in the visual modality could be a hard (but correct) sample from the visual perspective (a dog at night maybe). MMCo can utilize such samples by selecting examples on the basis of the consensus of the classifiers of individual modalities. Hard examples for one modality can still be used as a training example by exploiting the power of the other modalities. In this way, the classifiers of individual modalities mutually enhance each other.

The underlying motivation of our approach is very close to that of co-training in semi-supervised learning [3], which is discussed in Sec. 3.3.

3.1 Model

We jointly learn multiple modality classifiers. Given a dataset consisting of multiple modality features and pseudo labels \( \{(x_1^m, \ldots, x_n^M, y_n)\}_{n=1}^{N} \), the goal is to learn classifier parameters for multiple modalities \( (w^1, \ldots, w^M) \), where \( M \) is the number of modalities. The classifier
of each modality $g^m(x^m_n, w^m)$ is learned with the loss computed in the same way as single modality (Eq. 4). The only difference is the way of computing the weight variable $v_n$. $v_n$ reflects the label confidence, and thus does not depend on modality $m$, i.e., the same weight is used across all modalities. Therefore, we determine the selected samples and their weights by the consensus of classifiers of all modalities.

Sample selection. We here describe how to select examples in MMCo, i.e., how to compute sample weights $v_n$. In a previous approach, the weight is computed from the loss using Eq. 3, in which the loss is computed using the cross entropy between the confidence $g(x, w)$ and label $y$. The confidence is computed by the voting of multiple classifiers. Specifically, the sample weight $v_n$ is calculated as follows:

$$v_n = \max(0, 1 - \frac{L(s_n, y_n)}{\lambda}),$$

(5)

where the confidence $s_n$ of the sample is

$$s_n = h(g^1(x^1_n, w^1), ..., g^M(x^M_n, w^M)),$$

(6)

where $h$ is the voting function that takes the input of multiple classifier confidences $\{g^m(x^m_n, w^m)\}_{m=1}^M$. We introduce three types of simple voting scheme, namely, max ($\max_m g^m(x^m_n, w^m)$), average ($\frac{1}{M} \sum_m g^m(x^m_n, w^m)$), and product ($\prod_m g^m(x^m_n, w^m)$). This enable us to use hard examples of each classifier. For example, for max voting, if $g^1(x^1_n, w^1)$ is small and $g^2(x^2_n, w^2)$ is large, a large weight is assigned to this sample although it is a hard example for $g^2$. Our discussion for a single class can be extended to multiple classes in the same manner of Sec. 2.2.

3.2 System Design

The modalities used in this study are RGB, motion, audio, and metadata (see Sec. 4.1 for the details of the features). In addition, the concatenated features of all modalities extracted from video (RGB, motion, and audio) are used as another modality, which we call concat feature. In training, the classifiers of all modalities are used and mutually enhanced by our co-training approach. In testing, only the classifier of the concat feature is used; metadata information is not used, because we assume that metadata is not available for the test videos (It is also possible to use a metadata classifier for classifying the videos with metadata). We found that using the classifier for concatenated features worked better than using classifiers for multiple modalities.

The important point here is that a modality that cannot be used in testing can be used for training. For example, we can train the classifiers for the metadata modalities even if metadata is not available in the testing; a video-content classifier can be improved by co-training with a metadata classifier. On the other hand, the original WELL is based on self-paced learning, and thus, it cannot fully exploit metadata modality information (it can only be used to extract pseudo labels).

Although we can use any type of classifier for each modality, from the aspect of large-scale retrieval, it is better to use a linear classifier in testing than a more complex classifier such as a deep neural network. Large-scale retrieval is one of the main applications of video content classifier learning. To retrieve from large amounts of video, the classifier should be scalable. A linear classifier is easy to scale for a large database, whereas a deep neural network is difficult to scale. Although the scalable classifier should be used in testing, any type of classifier can be used for another modality in training. Therefore, our approach can exploit the power of a strong classifier (e.g., a deep neural networks) to select good examples from noisy training data, while previous self-paced learning-based approaches select examples using only the testing classifier.

3.3 Relationship with prior work

Self-training, co-training; Self-training [40] and co-training [3] are commonly used techniques in semi-supervised learning. In self-training, a classifier is first learned with a small amount of labeled data; then, it is applied to unlabeled data, wherein the most confident samples are added as new training examples. On the other hand, co-training iteratively train classifiers on two different views, where new training examples of each classifier are determined by the prediction of the other classifier. While the idea of WELL [26] is based on self-training that selects examples for the classifier by using its own predictions, MMCo incorporates the idea of co-training; examples for each classifier are selected on the basis of the predictions of other classifiers as well as its own predictions. Unlike co-training in semi-supervised learning, MMCo does not involve an iterative process, so it can easily handle more than two classifiers.
Multimodal ensemble learning: Our approach selects examples on the basis of the consensus of classifiers for multimodal features, which is the same idea as ensemble learning [10] that makes predictions by combining multiple classifiers. It is known that an effective ensemble learning system should consist of diverse classifiers [33, 34], and several studies have investigated the use of complementary multimodal information in videos for ensemble learning [5, 11, 19]. While ensemble learning is generally used to improve prediction accuracy, our approach uses an ensemble of multimodal classifiers to select good examples from noisy training data. Although we investigated only simple voting of classifiers, other ensemble learning techniques [4, 15, 39] could be integrated into our approach.

Learning from search engine results: While we tackle webly labeled examples from web videos with weak annotations, there is another type of webly supervised classifier learning which directly uses results returned by search engines [7–9, 13, 41]. This type typically uses a search engine (e.g., YouTube) to search by concept name and learns a classifier by using the retrieved results as positive samples. Since the challenge is similar to our problem, i.e., how to handle noise in the search results, our approach could be applied for this problem by using the search results as positive-labeled noisy training data and the search rank to compute the initial label confidence (i.e., top-ranked videos are likely to be positive).

4 EXPERIMENTS

4.1 Experimental Settings

Datasets. The main evaluation used Fudan-columbia Video Dataset (FCVID) [22] as the main evaluation. FCVID contains 91,223 videos from YouTube, totaling 4232 hours of videos. The dataset contains ground truth annotations at the video level for 239 categories including activities, objects, scenes, etc. We used the standard partitioning of the training/test set provided by [22]. We did not use the manual labels in training. Instead, we inferred pseudo labels from the metadata (title and description) in the same manner as [26]. In addition, we used the YouTube8M dataset [1] in the larger scale evaluation.

Features. We first extracted features from videos and used them as classifier inputs. In the evaluation of FCVID, we used the following features for each modality. RGB: Frame-level features were first extracted from keyframes by VGG [32] trained on ILSVRC, and a video-level feature was then computed by average pooling; these were the same features used in [26]. Motion: P3D ResNet features were extracted in the manner described in [29]. We used the model trained on the Kinetics dataset [23], which was provided by 

\[\text{Concat} \]

Audio: We extracted 128-dimensional VGGish features produced from a VGG-like audio classification model [18]. We used the model trained on the YouTube8M dataset [1] provided by Google.2

Metadata: TextCNN features were extracted from the title of the YouTube video following [38]. We used the model provided by [38], which was trained on the YouTube8M dataset.3 Concat: The features of RGB, Motion, and Audio are concatenated.

In the YouTube8M evaluation, we used the officially provided RGB and audio features (1024-dimensional GoogLeNet [35] feature and 128-dimensional VGGish feature). For metadata, we used the same features as FCVID evaluation.

Classifiers. We used a linear classifier for all modalities except audio. A multi-layer perceptron with one hidden layer (4096 units) was used for audio feature classifier.

Training. The model age \(\lambda\) was determined from the rate of used positive examples \(p\); \(p\) was initialized as 0.3, which is increased by 0.05 every five epochs and early stopped at \(p=0.6\). The training labels (i.e., pseudo labels \(y_n\) and their confidences \(c_n^p\)) were obtained by word hard matching in the same way as [26]. Adam optimizer [24] was used to update the model with initial learning rate \(10^{-3}\).

4.2 Comparison with Baselines

We compared final classification performances of the classifiers trained on the web videos on FCVID. The main objective of this comparison was to determine whether online WELL achieves comparable performance to the conventional WELL based on alternating optimization and whether MMCo improves performance. To this end, we reproduced WELL with the same features and parameters and compared it with the online WELL with and without MMCo. We used concat features in our reproduction of WELL for a fair comparison.

Table 1 shows the results. Online WELL achieves comparable performance to WELL using the same features. MMCo with max voting improves mAP by 4.2 points relative to WELL without MMCo. It also significantly outperforms the other baselines [25, 28, 37]. Although the different features are used for the comparison, our approach improves even on WELL that is much better than the others with the same VGG features. In addition, WELL-MM [27] and MMCo are complementary extensions of WELL. WELL-MM infers pseudo labels and their confidences using multimodal knowledge instead of word matching, which can easily be incorporated into our approach simply by changing pseudo labels.

4.3 Evaluation of Example Selection

To show that MMCo selects good examples from noisy training data, we evaluated the performance of example selection. In webly
labeled learning, the samples with the loss smaller than the age were used as positive samples as described in Eq.3; we measured the quality of selected positive samples by recall, precision, and AP. Recall and precision were computed as the number of selected positive samples divided by the total number of positive samples and the total number of selected samples, respectively, and AP was computed by the ranking of losses for all positive samples. We used online WELL and compared with and without MMCo. Voting schemes were also compared. The baseline (w/o MMCo) computed the sample weights using one classifier for concat features while the proposed approach (w/ MMCo) computed weights by the voting of classifiers of all five modalities.

Figure 3 shows the precision/recall curve and performance during the training. The proposed MMCo with max voting (MMCo (max)) performed significantly better than the baseline. When the rate of used samples was 0.6 (in Fig. 3b), the precision/recall of MMCo (max) was 0.70/0.80, which is much better than the results of the baseline without MMCo (0.66/0.64) and the results using all samples with pseudo labels (0.49/0.85). These results clearly demonstrate the effectiveness of MMCo. The experiments in Sec. 5.1 provide a more detailed analysis of example selection and discuss why max voting performs the best.

4.4 Large-Scale Evaluation on YouTube-8M
The YouTube-8M dataset contains millions of videos; it is too big to be handled by the conventional approach based on alternating minimization. We conducted an evaluation on the YouTube-8M dataset to determine whether the performance of online WELL gets better on larger-scale datasets. We varied the size of the dataset by randomly selecting samples. We evaluated mAP for the 500 most frequently appearing classes from all 4716 classes, because samples from many classes in YouTube8M rarely appear when the size of the dataset is small. Figure 4 shows the performance of our approach with and without MMCo (max voting) for different subset sizes. mAP increases along with the size of the noisy training data up to two million samples. MMCo consistently improves mAP for any dataset size (about a 5% relative improvement). The results show that online WELL has the potential to increase its accuracy, because it can handle an unlimited number of videos and noisy training data is easy to collect.

| mAP | Recall |
|-----|--------|
|     | easy | normal | hard |
|     | easy | normal | hard |
| w/ MMCo | 0.655 | 0.642 | 0.506 | 0.823 | 0.709 | 0.579 |
| w/o MMCo | 0.700 | 0.592 | 0.256 | 0.857 | 0.648 | 0.299 |

Table 2: Sample selection performance in terms of the difficulty (easiness) of samples. The same number of positive-labeled samples is used for both conditions (p=0.6).

5 DETAILED ANALYSIS
We conducted a more detailed analysis to better understand the behavior of MMCo. We used max voting unless otherwise specified.
5.1 Are hard samples actually selected?

The motivation of MMCo is to select hard positive examples that cannot be selected by the previous approach. We performed three analyses to validate MMCo really select hard samples.

Performance by difficulty: We evaluated the performance of example selection in terms of the difficulty, or easiness, of the samples. The objective of this experiment was to show that the samples that cannot be detected by initial classifiers can be rescued by the proposed approach. Therefore, we define easiness as a classification score using the classifier trained on the top 30% of samples with a high pseudo label’s confidence $c_0$. We divided the positive samples in the training data into three sets in terms of by easiness, i.e., easy, normal, and hard, the samples were sorted by easiness score for each class, where the 30, 40, and 30th percentiles of samples are selected as easy, normal, and hard, respectively. Table 2 shows the example selection performance of each difficulty set. MMCo significantly improves the performance of hard samples in terms of both mAP (from 0.256 to 0.506) and recall (0.299 to 0.579), while there is no big difference for easy and normal samples.

Distribution of difficulties of selected examples: We performed a more detailed analysis of selected examples. We compare the distribution of easiness (defined above) with and without MMCo. Figures 3a and 3b shows the results with and without MMCo, respectively, where the x-axis shows the rank of easiness scores among all positive-labeled examples, which is normalized to [0,1]. The MMCo selected positive hard examples with the scores in the range [0.05], while the method without MMCo did not select them often. Note that there is not a big difference in the number of selected negative examples (i.e., false positive samples), which shows MMCo selects only true positive examples from hard samples.

Qualitative analysis: We qualitatively analyzed the selected examples. We picked ‘cello performance’ class of FCVID and watched the detail of the selected examples. We again used the classification scores of the classifier trained on top 30% examples, i.e., the same condition under which samples were first selected by MMCo. Figure 6a shows the metadata and concat classifier scores of positive labeled training samples, where true and false positive examples are depicted in red and blue. The selected samples of each voting scheme is shaded in color (when using only metadata and concat scores). The results suggest why max voting performed better than other voting schemes. We can see that the samples with high confidence for one modality are likely to be positive, while the samples with low confidence are not always negative. For example, there are some positive samples with zero confidence for one modality (lower right and upper left in the figure), which are hard samples for the modality but selected by using max voting. Figure 6b shows examples of these hard positives. Metadata classifier assigns high scores to simple titles (second row) and low scores to long titles that are difficult to parse (first row). By learning with such hard examples, metadata classifiers learns the key words or phrases of ‘cello performance’ such as ‘cellist’, ‘playing’, and ‘cello lessons’. Video-content classifier (concat) is also improved by learning with examples that are hard from the visual perspective (second row).

5.2 Which modalities improve performance?

To investigate which modalities improve performance, we tested with varied combination of modalities. Since our main interest is the improvement in example selection performance, we only changed training modalities that are used to compute scores for example selection. We used the concat feature in the testing in all experiments. Table 3 shows the results for various modality combinations; rows 1–5 are for only a single modality, while rows 6–11 are for multiple modalities. The results provide several important findings. 1) The results using both video-content and metadata modality (rows 7–11) are much better than those using only either type of features (rows 1–6). This suggests that using modalities with complementary information is more important than using just multiple modalities. 2) Among the video-content related features, P3D contributed significantly to performance improvements, while audio did not contribute very much (row 8–10). 3) The results for Concat + Metadata (row 7) are comparable to those of using more than two classifiers, which is a practically reasonable choice in terms of the trade-off between the training cost and test accuracy.
We evaluated the robustness of our approach to noise by manually controlling the amount of noise in the training data. Since the training data of our task is noisy, there are true and false positives among the positive-labeled samples. We varied the noise level $n_{FP}/(n_{TP} + n_{FP})$ in the range of 0.2 to 0.9, where $n_{TP}$ and $n_{FP}$ are the number of true and false positives, respectively. We fixed the true positive samples and randomly selected false positive samples as artificial noisy samples. Figure 7 compares the performances of our approach with and without MMCo on FCVID for different noise levels. Our approach (w/ MMCo) was consistently better than the baseline (w/o MMCo) with any amount of noise. It scored 0.63 in mAP at a noise level of 0.9, which demonstrates that it can work on a very noisy dataset.

### 5.3 Noise robustness

We evaluated the robustness of our approach to noise by manually controlling the amount of noise in the training data. Since the training data of our task is noisy, there are true and false positives among the positive-labeled samples. We varied the noise level $n_{FP}/(n_{TP} + n_{FP})$ in the range of 0.2 to 0.9, where $n_{TP}$ and $n_{FP}$ are the number of true and false positives, respectively. We fixed the true positive samples and randomly selected false positive samples as artificial noisy samples. Figure 7 compares the performances of our approach with and without MMCo on FCVID for different noise levels. Our approach (w/ MMCo) was consistently better than the baseline (w/o MMCo) with any amount of noise. It scored 0.63 in mAP at a noise level of 0.9, which demonstrates that it can work on a very noisy dataset.

### 5.4 Qualitative examples

Figure 8 shows the top retrieval results on FCVID by using the classifiers trained by webly labeled learning. The results with and without MMCo are compared to qualitatively demonstrate the effectiveness of our approach. While the approach without MMCo detects many false positives of very similar concept, the approach with MMCo rejects such false positives by effectively using the metadata information in training. Since our trained classifier is a simple linear classifier, this retrievals can be performed in less than one second from 48,435 videos. It is also easy to extend million or billion-scale retrievals by using the techniques of approximate nearest neighbor search [20].

### 6 CONCLUSION

We presented a multimodal co-training (MMCo) algorithm, a simple and effective method for learning concept classifier from videos on the web. MMCo can select good examples from noisy training data by jointly learning classifiers of multiple modalities and selecting examples based on their consensus. We conducted an extensive experimental analysis to demonstrate the effectiveness of MMCo, which demonstrated better example selection and consistent improvements in webly labeled learning on FCVID dataset over baseline methods. Our detailed analysis from multiple aspects further examined the behavior of MMCo and provided several useful insights about how to exploit multimodal information effectively in webly supervised learning.

### REFERENCES

[1] Sami Abu-El-Haija, Nisarg Kothari, Junsook Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheerendra Vajjayanarasimhan. 2016. YouTube-8M: A Large-Scale Video Classification Benchmark. In arXiv preprint.

[2] Yoshua Bengio, Jérome Bouardouz, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In IML.

[3] Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In COLT.

[4] Leo Breiman. 1996. Bagging predictors. Machine Learning, 24, 2 (1996), 123–140.

[5] Shih-Fu Chang, Dan Ellis, Wei Jiang, Keesha Lee, Akira Sanagawa, Alexander C. Leui, and Jielo Luo. 2007. Large-scale multimodal semantic concept detection for consumer video. In Proceedings of the international workshop on Workshop on multimedia information retrieval - MIR '07.

[6] Alexander Chapelle, Olivier and Scholkopf, Bernhard and Zien. 2009. Semi-Supervised Learning. IEEE Transactions on Neural Networks 20, 3 (2009), 542–542.

[7] Ken Chatfield, Relja Arandjelović, Andrew Zisserman, Relja Arandjelović, Omkar Parkhi, and Andrew Zisserman. 2015. On-the-fly learning for visual search of large-scale image and video datasets. International Journal of Multimedia Information Retrieval 4, 2 (2015), 75–93.

[8] Ken Chatfield and Andrew Zisserman. 2012. VISOR : towards on-the-fly large scale object category retrieval. In ACCV.

[9] Xinlei Chen and Abhinav Gupta. 2015. Webly supervised learning of convolutional networks. In ICCV.

[10] Thomas G. Dietterich. 2002. Ensemble Learning. Vol. 2. 110—125 pages.

[11] Jianping Fan, Hangzai Luo, Yuli Gao, and Ramesh Jain. 2007. Incorporating concept ontology for hierarchical video classification, annotation, and visualization. IEEE Transactions on Multimedia 9, 5 (2007), 939–957.

[12] Pedro Felzenszwalb, David Mcallester, Deva Ramanan, and U C Irvine. 2008. A Discriminatively Trained Multiscale, Deformable Part Model. In CVPR.

[13] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. 2005. Learning object categories in webly supervised learning. In COLT.

[14] Benoit Fréhys and Michel Verleysen. 2014. Classification in the presence of label noise: A survey. IEEE Transactions on Neural Networks and Learning Systems 25, 5 (2014), 845–869.
Figure 8: Top retrievals on FCVID dataset. Retrievals were performed by using the classifier trained with the webly label, where the results are sorted by classification score and marked as negative (red) or positive (green).