Cross-Class Relevance Learning for Temporal Concept Localization

Junwei Ma *
Layer6 AI
jeremy@layer6.ai

Satya Krishna Gorti *
Layer6 AI
satya@layer6.ai

Maksims Volkovs
Layer6 AI
maks@layer6.ai

Ilya Stanevich
Layer6 AI
ilya@layer6.ai

Guangwei Yu
Layer6 AI
guang@layer6.ai

Abstract

We present a novel Cross-Class Relevance Learning approach for the task of temporal concept localization. Most localization architectures rely on feature extraction layers followed by a classification layer which outputs class probabilities for each segment. However, in many real-world applications classes can exhibit complex relationships that are difficult to model with this architecture. In contrast, we propose to incorporate target class and class-related features as input, and learn a pairwise binary model to predict general segment to class relevance. This facilitates learning of shared information between classes, and allows for arbitrary class-specific feature engineering. We apply this approach to the 3rd YouTube-8M Video Understanding Challenge together with other leading models, and achieve first place out of over 280 teams. In this paper we describe our approach and show some empirical results.

1. Introduction

The problem of video understanding has received increasing attention recently following the rapid advancement in computer vision image models. Temporal concept localization in particular is a challenging but fundamental problem in video understanding where the main goal is to identify video segments that contain a specific concept or action. Just as the rapid advancement in image understanding is aided by the large-scale and diverse ImageNet dataset [10], the recently introduced YouTube-8M dataset [1] brings both scale and diversity to video data. The 3rd YouTube-8M Video Understanding Challenge [1] leverages this data to benchmark video understanding models in a standardized setting.

Video action localization is a closely related problem to concept localization. Action localization generally assumes that action classes are mutually exclusive, meaning that each segment can only contain one class. However, in the real world applications with many classes, a segment may contain multiple actions simultaneously. Moreover, in addition to actions there are also objects of interest that need to be accurately localized. This complexity is captured by the diverse set of classes in the YouTube-8M dataset that include both actions and objects, and where segments are labelled with multiple classes introducing new modelling challenges. At scale, the total number of segment-class pairs becomes prohibitively large. So in this data only a small subset is sampled and labelled. The labels are binary where 1 indicates that segment contains a given class, and 0 indicates that it doesn't contain it. The goal of the challenge is to build accurate segment models that, given a video, can identify all segments that contain each of the classes.

Existing approaches primarily apply recurrent [22] or codebook-based [31] models to extract video and segment features. Then a fully connected classification layer predicts class probabilities for each segment. The idea is to use a common feature extraction backbone, and learn class-specific weights in the last layer. This architecture makes it challenging to incorporate any class information such as hierarchy or class similarity into the model. Furthermore, when number of classes is large and/or few training examples are available per class, this model is prone to over-fitting as it learns a separate set of weights for every class. To deal with these problems we propose a pairwise segment-class model that in addition to segment also takes as input target class and class-related features, and predicts segment to class relevance. This facilitates learning of shared information between classes, and allows for arbitrary class-specific feature engineering.

To avoid running expensive inference for every segment-class pair we further propose a candidate generation

*Authors contributed equally to this work.
[1] www.kaggle.com/c/youtube8m-2019

pipeline. Specifically, we apply video level model to significantly reduce number of candidate segments for each class while preserving high recall. To summarize, our contribution in this paper is two-fold. First, we propose a novel architecture for video concept localization that incorporates target class information as input into the model. We further develop class-specific features that improve localization performance. Second, we demonstrate practical and efficient application of the proposed architecture on the largest video dataset available to date as part of the 3rd YouTube-8M Video Understanding Challenge. Our approach achieves first place out of over 280 teams.

2. Related Work

The temporal concept localization task is closely related to video action classification and action localization. Video action classification task consists of predicting what actions appear in a given video, while action localization aims to find segments where each action appears. However, classes in the YouTube-8M dataset are more diverse than actions, and include everyday objects that exhibit hierarchy and relationships not found in traditional action classification and localization datsets [5, 16, 39].

Feature extraction. Feature extraction is a common step in both tasks. Early works extract hand-crafted features by tracking pixels over time [38]. Recent works instead employ deep neural networks that use spatio-temporal convolutions such as I3D [6], C3D [35] and two-stream approaches [11, 6] to capture visual and motion features. The YouTube-8M video and segment datasets [1] used in our experiments consist of both video and audio features extracted from publicly available Inception network [33] and VGG-inspired acoustic model [15] respectively.

Action Classification. Recurrent models such as LSTMs [12] and GRUs [9] are natural candidates for video level action classification as they are well suited to extract temporal features across time. Applications of recurrent models on this task can be seen in [31] and [34]. Other popular approaches for action classification [18, 27] are based on Fisher Vectors (FV) [28] and Vector of Locally aggregated Descriptors (VLAD) [17]. These methods rely on maintaining a hand-crafted codebook. NetVLAD [3] and NetFV [22] are variations of VLAD and FV methods that learn the encoding end-to-end using deep networks. These methods achieve state-of-the-art performance on the YouTube-8M video classification task [22, 31, 34, 25], and we also leverage them in our pipeline.

Action Localization. Action localization task can be partitioned into two groups: *fully-supervised* and *weakly-supervised*. In the fully supervised setting, models are trained using frame-level annotations indicating action boundaries. [7, 30] perform fully supervised action localization is a two stage manner, first predicting proposals and further classifying proposals into action classes. GTAN [20] learns a set of Gaussian kernels used to predict intervals of actions in a single-stage manner. The network is optimized end-to-end using classification loss and two regression losses to predict action boundaries.

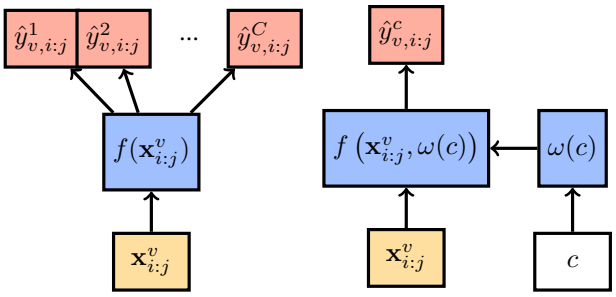
On the other hand, weakly supervised action localization models predict action classes and boundaries only based on high level action category labels. This can be formulated as multi-instance multi-label learning problem [41], where multiple instances describe an example having multiple labels. For example, W-TALC [26] extends the work of STPN [24] which introduces attention module to identify sparse subset of video frames associated with actions by having novel objective functions based on multiple-instance learning and activity similarity.

Datasets. Some notable datasets for action classification include Moments in Time [23], Something-Something [13], UCF101 [32], Sports1M [19]. Thumos [16], MultiThumos [39] and ActivityNet [5] are popular datasets used in recent work for action localization. Another category of datasets used for spatio-temporal localization include MSRActions [40], AVA [14] and UCFSports [29]. Compared to these, Youtube-8M [1] is by far the largest multi-class dataset with over 3.8K classes that contain a very broad and diverse set of actions and objects.

Pairwise Relevance Learning. It is common in structured prediction tasks to design a pairwise compatibility function over input-output pairs [36]. This can be seen for example in [2], where joint image-class model is trained to embed classes and measure compatibility between images and classes for zero-shot classification. Our approach follows similar intuition, and we aim to learn a general pairwise model to predict segment to class relevance.

3. Approach

Given a video v, we use $\mathbf{x}_i^v$ to denote the i'th frame in v. We follow [1] and represent video by a sequence of frames $\mathbf{x}^v = [\mathbf{x}_1^v, \mathbf{x}_2^v, ..., \mathbf{x}_{F_v}^v]$, where F_v is the total number of frames in v. We assume that features have been extracted for each frame so $\mathbf{x}_i^v$ is a vector in $\mathcal{R}^d$. Segment $\mathbf{x}_{i:j}^v = [\mathbf{x}_i^v, \mathbf{x}_{i+1}^v, ..., \mathbf{x}_j^v]$ is a sub-sequence of consecutive frames from i to j with $j > i$. The temporal concept localization problem is formulated as predicting relevance of a given segment $\mathbf{x}_{i:j}^v$ to class c. The ground truth binary label for this task is denoted by $y_{v,i:j}^c \in \{0, 1\}$, where positive label 1 indicates that $\mathbf{x}_{i:j}^v$ is relevant to c and negative label 0 indicates that $\mathbf{x}_{i:j}^v$ is not relevant to c. For a large database of videos and classes, it's impractical to label all segment-class pairs. Typically only a subset of pairs is sampled and labelled, so $y_{v,i:j}^c$ is unknown for most segments $\mathbf{x}_{i:j}^v$. In the following sections we outline our approach to this problem and show empirical results.



(a) Traditional architecture (b) CCRL

Figure 1: (a) shows commonly used classification architecture where a vector of class probabilities is predicted for each segment. (b) shows our proposed CCRL architecture where segment $\mathbf{x}^v_{i:j}$ and class features $\omega(c)$ are jointly used as input to a binary relevance model.

3.1. Candidate Generation

At scale, the number of segments can become prohibitively large making model inference very expensive. To address this, we propose a candidate generation stage to significantly reduce the set of segments that need to be scored for each class. We use video level candidate generation and select videos that are likely to contain segments with a given class. The main motivation for this is that predicting whether video contains a class is significantly easier than localizing where exactly it appears. This is evidenced by the performance of video models from the previous edition of the YouTube-8M challenge that focused on video classification. Winning solutions were able to achieve nearly 0.9 in mean average precision on this task [31] even though there were over 3.8K classes. Moreover, labelling videos is significantly easier than labelling segments with precise start and end times. YouTube-8M dataset has video level labels for more than 6M videos but only 47K videos have segment labels. Using video models thus also allows us to leverage information from the much larger video dataset and transfer it to segment prediction.

Our candidate generation stage uses a video level model to predict class probabilities for every video in the corpus. Then, given a target class c, we sort videos according to probability for c and take top-K. All segments from the top-K videos are treated as candidates for c and are passed to the segment model. By leveraging leading approaches for video classification [31] we are able to reduce the number of candidate segments by over 20x while still achieving near perfect 99.7% average segment recall across classes.

3.2. Traditional Approach to Classification

Typical architecture for segment/video classification consists of feature extraction backbone followed by a classification layer where a separate set of weights is learned for each class. The main idea is that the backbone learns to extract visual information generally useful for prediction, and classification layer learns to transform this information into class-specific predictions. Formally, given a segment $\mathbf{x}^v_{i:j}$ the model outputs a prediction vector:

$$\hat{\mathbf{y}}_{v,i:j} = f(\mathbf{x}^v_{i:j}) \tag{1}$$

where $\hat{\mathbf{y}}_{v,i:j} = [\hat{y}^1_{v,i:j}, \hat{y}^2_{v,i:j}, ..., \hat{y}^C_{v,i:j}]$ contains predictions for all target classes. This architecture is illustrated in Figure 1a.

Despite its popularity, this architecture has several disadvantages. First, in many real-world applications classes often have close hierarchical relationships. An example of this is shown in Figure 2. Here, we show t-SNE [21] visualization of the bag-of-words descriptions for a subset of classes in YouTube-8M. From the figure we can observe complex and hierarchical inter-class relationships common in these domains. For example, a number of classes are related to automobiles. These are sub-divided into economy, sports and luxury, and can be further partitioned by manufacturer etc. Incorporating this information or any other class related features is non-trivial in the traditional model. It typically requires loss modification and/or manual weight manipulation in the classification layer [2]. Second, class-specific weights are only updated when examples of that class are encountered during training. For classes with few training examples this can result in under/over-fitted models. Accurately labelling video segments is a difficult task and most datasets including YouTube-8M have highly sparse labels.

In the following section we propose a different architecture for segment classification that avoids most of these problems.

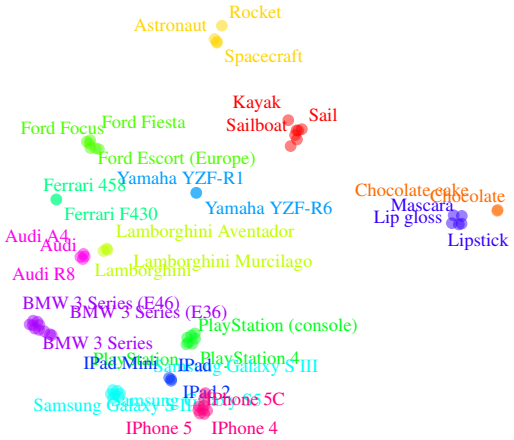


Figure 2: t-SNE reduced bag-of-words description vectors for a subset of Youtube-8M classes. Different colors represent clusters found by applying k-means. Clear hierarchical relationships between classes can be observed from this diagram.

Table 1: Performance of the segment candidate generation models evaluated by Recall at 100K segments for each class.

Model	Recall@$100K$
Logistic	0.9713
LSTM	0.9931
Ensemble [31]	**0.9969**

3.3. Cross-Class Relevance Learning

To incorporate class information into the model and promote cross-class information sharing, we propose a pairwise architecture. Specifically, our architecture takes both segment and target class as input and predicts relevance probability. We train a *single* pairwise model that captures general notion of "relevance" between all segments and classes. This architecture is shown in Figure 1b. Using $\omega(c)$ to denote extracted features for class c, our model outputs relevance probability between segment $\mathbf{x}^v_{i:j}$ and c:

$$\hat{y}^c_{v,i:j} = f\left(\mathbf{x}^v_{i:j}, \omega(c)\right) \tag{2}$$

Pairwise architecture allows us to encode arbitrary class information into the model in a principled way. By training a single model for all classes we can automatically learn relationships between classes that are useful for the target task. To further illustrate this point, consider an example where f is a decision tree model and $\omega(c)$ simply outputs class index c. The decision tree can use class index to group classes in each sub-tree and automatically learn an implicit hierarchy. This is not possible with traditional classification architectures. Another advantage is that a learned model can potentially generalize to new unseen classes during training as long as $\omega(c)$ can reasonably encode them.

We train this model with a binary cross entropy objective using all labelled segments:

$$\begin{aligned}\mathcal{L} = -\sum_{y^c_{v,i:j}} & y^c_{v,i:j} \log\left(f(\mathbf{x}^v_{i:j}, \omega(c))\right) \\ & + (1 - y^c_{v,i:j}) \log\left(1 - f(\mathbf{x}^v_{i:j}, \omega(c))\right)\end{aligned} \tag{3}$$

During inference, given a segment $\mathbf{x}^v_{i:j}$ we need to determine which classes are relevant for $\mathbf{x}^v_{i:j}$. Naive approach requires C forward passes through the model, one for every class. This quickly becomes prohibitively expensive, especially for datasets such as YouTube-8M that have thousands of classes. However, we note that after candidate generation step we only need to consider a small subset of segments for each class. The two-stage approach makes inference in our model practical and scalable, and we run it without difficulties on the large scale YouTube-8M data.

Table 2: Private leaderbooard performance of segment models with and without candidate generation (CG)

Model	Without CG	With CG
ConvNet [4]	0.7454	0.8036
LSTM [31]	0.7681	0.8023
Transformer [37]	0.6524	0.7955
NetVLAD [3]	0.7243	0.8023
NetFV [22]	0.7203	0.8028
CCRL	**0.7711**	**0.8091**
Ensemble	-	**0.8329**

3.4. Class Features

We conducted extensive investigation on the types of class features that can be encoded with $\omega(c)$. Here, we present one interesting feature that significantly improves performance. Given a segment $\mathbf{x}^v_{i:j}$ and target class c, our goal is to compare $\mathbf{x}^v_{i:j}$ to all labelled relevant and irrelevant segments for class c. The main idea is similar to clustering in that if $\mathbf{x}^v_{i:j}$ is relevant for c then it should be "close" to other relevant segments and "far" from irrelevant ones. We formalize this as follows:

$$\text{SIM}_{\text{pos}}(\mathbf{x}^v_{i:j}, c) = \sum_{u \neq v} \sum_{y^c_{u,k:l}=1} \text{dist}(\mathbf{x}^v_{i:j}, \mathbf{x}^u_{k:l})$$

$$\text{SIM}_{\text{neg}}(\mathbf{x}^v_{i:j}, c) = \sum_{u \neq v} \sum_{y^c_{u,k:l}=0} \text{dist}(\mathbf{x}^v_{i:j}, \mathbf{x}^u_{k:l})$$

where $\text{dist}(\mathbf{x}^v_{i:j}, \mathbf{x}^u_{k:l})$ is a distance function between segments $\mathbf{x}^v_{i:j}$ and $\mathbf{x}^u_{k:l}$. SIM_{pos} and SIM_{neg} compute total distances to relevant (positive) and irrelevant (negative) segments respectively. Any similarity function can be used for dist. Empirically, we found cosine of the angle between segment feature encodings (e.g. RNN hidden states) to work well. Adding these features significantly improved localization accuracy, and further demonstrates the flexibility of the proposed architecture that allows to explore virtually any class-related features.

4. Experiments

We conduct experiments on temporal localization task as part of the 3rd YouTube-8M Challenge. This challenge dataset is an extension of the original YouTube-8M dataset [1], which contains 6.1M videos described by 2.6B audio-visual frame features. The videos are labelled across 3862 classes, and have an average of 3 labels per video. The extended dataset contains 237K human-verified segment class labels from 47K videos with approximately 5 segments per video. The videos in the extended dataset are labelled with 1000 classes which is a subset of the original 3862 classes.

Table 3: Private leaderboard performance for top-5 teams as well as our individual models. Our team "Layer6 AI" achieved first place.

Team	Leaderboard	Rank
Layer6 AI	**0.8329**	**1**
BigVid Lab	0.8262	2
RLin	0.8255	3
bestfitting	0.8171	4
Last Top GB Model	0.8046	5
ConvNet	0.8036	6
LSTM	0.8023	7
Transformer	0.7955	8
NetVLAD	0.8023	7
NetFV	0.8028	7
CCRL	**0.8091**	**5**

Challenge data is split into a publicly available labelled set and two held-out test sets (public and private leaderboards). For all experiments, we split the labelled set into 90% $\mathcal{S}_{\text{train}}$ and 10% $\mathcal{S}_{\text{val}}$ sets and tune hyperparameters using $\mathcal{S}_{\text{val}}$ set. We use the chosen hyperparameters to the retrain on the full 100% labelled set. Results for the held-out set are reported by submitting our model predictions to the leaderboard through the Kaggle platform. We report results on the private leaderboard which corresponds to the larger held-out test set that was used to rank teams at the end of the competition.

The challenge task is to predict a ranked list of up to 100K segments for each class. This list is evaluated against the ground truth using mean average precision (mAP):

$$\text{mAP} = \frac{1}{C}\sum_{c=1}^{C}\frac{\sum_{i=1}^{n}\text{Prec}(i)\times rel(i)}{N_c} \tag{4}$$

where $\text{Prec}(i)$ is precision at rank i and $rel(i)$ is 1 if segment at rank i is relevant and 0 otherwise; N_c denotes the total number of relevant segments in class c. In addition to mAP, we use average class recall to evaluate the quality of the segment candidate generation model.

4.1. Implementation

We compare popular temporal localisation models against our proposed CCRL architecture. To make comparison fair all models share the same candidate segment generation pipeline.

Candidate Generation Models For segment candidate generation we consider logistic and LSTM models that were part of the winning solution for the YouTube-8M video classification challenge [31]. Logistic model described in [1] is a fully-connected architecture with a sigmoid activation to output class probability predictions. The input features are composed by concatenating RGB and audio inputs for each

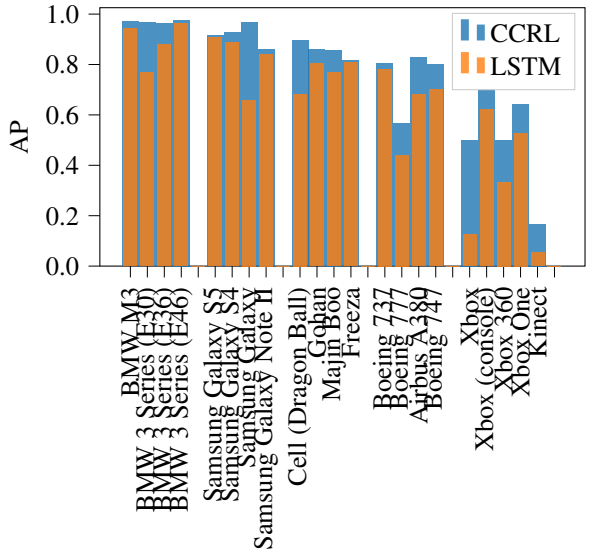


Figure 3: Average precision performance for individual classes for CCRL and LSTM models.

frame. LSTM model is a stack of bi-directional LSTM layers followed by a uni-directional LSTM layer with a cell size of 1024. A fully-connected layer is then used for classification from the final hidden state of the uni-directional LSTM layer. We also evaluate model ensemble obtained through knowledge distillation [31].

Segment Models For segment classification we consider latest neural network architectures for temporal learning. These include Convolutional Neural Networks (ConvNet) [4], LSTM [31], Transformer [37], NetVLAD [3] and NetFV [22]. In ConvNet model, two layers of width 2 convolutions are applied along the video frames. Frame representations from last layer are then combined via average or max pooling and passed to classification layer. Transformer model first compresses frames into segment representations using a convolutional layer, then three blocks of self-attention are applied followed by a fully connected classification layer. LSTM is initialised with pre-trained video LSTM model used for candidate generation. We then fine-tune it for localisation task using segment level binary cross entropy loss. NetVLAD [3] and NetFV models are based on [22]. We modify the original implementation by removing batch norm layers and switching from mixture-of-experts [1] to logistic regression. These modifications lead to more stable learning and better accuracy.

For CCRL, we use tree-based gradient boosting machines (GBMs) to learn non-smooth class relationships that are difficult to capture with deep learning. We use predictions from segment candidate generation models as additional input features for each segment. Class features include one-hot class encoding and various versions of similarity features outlined in Section 3.4. We use the XG-

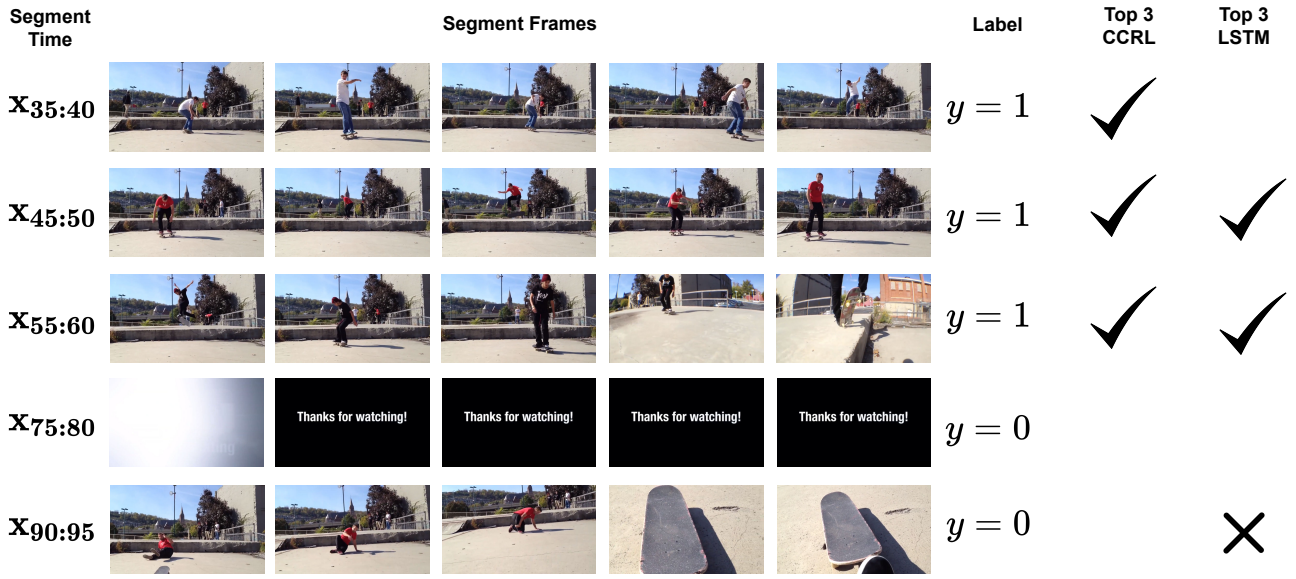


Figure 4: Qualitative example of instance localization for CCRL and LSTM models. Top-3 segments for class "skateboarding" are shown for each model. LSTM makes a mistake on segment $\mathbf{x}_{90:95}$ where frames resemble skateboarding but actually show a skateboarder falling. CCRL is able to correctly identify all relevant segments.

Boost [8] package for GBM training in all experiments.

4.2. Results

Table 1 summarizes results for segment candidate generation models. We focus on recall here since the primary aim for candidate generation is to capture as many relevant segments as possible. From the table we see that candidate generation is able to reduce total number of segments from around 2.2M to 100K and still achieve near perfect recall. The model ensemble in particular is able to achieve recall of 0.9969 which indicates that virtually all relevant segments are included in the candidate set.

Table 2 compares leaderboard model performance for all segment models before and after applying candidate generation. We observe significant increase in performance with up to 20% gain in mAP after candidate generation. This indicates that candidate generation is able to successfully transfer information from the much larger video classification dataset and improve localization. Notably, models that perform worse on their own, such as transformer, benefit more from the candidate generation. We also see that CCRL is the best performing model both with and without candidate generation. Strong performance suggests that CCRL is a promising approach for concept localization and potentially other related tasks. Model ensemble further improves performance and gives our best accuracy of 0.8329. We experimented with various ensemble techniques but found that simple averaging worked best here.

Figure 3 shows average precision performance for CCRL and LSTM models for a subset of classes clustered according to their bag-of-words descriptions. Here, we can see the benefits of cross-class learning. For example, LSTM performs significantly worse for classes "BMW 3 Series (E30)" and "Samsung Galaxy", while related classes have higher performance. However, CCRL is able to achieve more consistent accuracy across classes in a given cluster. One explanation for the more consistent scores can be attributed to better information sharing between classes. However, also observe that for some clusters such as the Xbox products, our model struggles to achieve consistent performance. Further architectural improvements or additional features can be beneficial here and we leave it for future work.

Table 3 shows the final leaderboard scores for top-5 teams as well as individual models. Our team "Layer6 AI" achieved first place outperforming the second team by 0.67 points in mAP. We also see that individual models have strong performance and all place in the top-10 on the leaderboard. Our best single model places 5'th on the leaderboard and can be used on its own, particularly in production environments where model ensembles can be difficult to manage.

Figure 4 shows a qualitative example comparing CCRL and LSTM predictions on one video. Each row represents one of five labelled segments for this video with class "skateboarding". To evaluate localization, we examine the top-3 predicted segments from each model. From the fiigure we see that LSTM makes a mistake on segment $\mathbf{x}_{90:95}$ where frames resemble skateboarding but actually show a skateboarder falling. CCRL is able to correctly identify all

relevant segments.

5. Conclusion

We propose cross-class relevance learning approach for temporal concept localization. Our approach uses a pairwise model that takes both segment and target class as input and predicts relevance. This promotes information sharing between classes, and allows for extensive class feature engineering. We apply our approach to the YouTube-8M video understanding challenge together with other leading models and achieve first place. In the future, we aim to explore additional features to describe classes as well as pairwise architectures for joint prediction.

References

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 1, 2, 4, 5

[2] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for attribute-based classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 2, 3

[3] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016. 2, 4, 5

[4] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018. 4, 5

[5] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015. 2

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

[7] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1130–1139, 2018. 2

[8] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016. 6

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 2

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[11] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941, 2016. 2

[12] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. 1999. 2

[13] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The something something video database for learning and evaluating visual common sense. In *ICCV*, volume 1, page 3, 2017. 2

[14] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6047–6056, 2018. 2

[15] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson. Cnn architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017. 2

[16] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The thumos challenge on action recognition for videos in the wild. *Computer Vision and Image Understanding*, 155:1–23, 2017. 2

[17] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR 2010-23rd IEEE Conference on Computer Vision & Pattern Recognition*, pages 3304–3311. IEEE Computer Society, 2010. 2

[18] V. Kantorov and I. Laptev. Efficient feature extraction, encoding and classification for action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 2

[19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 2

[20] F. Long, T. Yao, Z. Qiu, X. Tian, J. Luo, and T. Mei. Gaussian temporal awareness networks for action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 344–353, 2019. 2

[21] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 3

[22] A. Miech, I. Laptev, and J. Sivic. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017. 1, 2, 4, 5

[23] M. Monfort, A. Andonian, B. Zhou, K. Ramakrishnan, S. A. Bargal, Y. Yan, L. Brown, Q. Fan, D. Gutfreund, C. Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 2

[24] P. Nguyen, T. Liu, G. Prasad, and B. Han. Weakly supervised action localization by sparse temporal pooling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6752–6761, 2018. 2

[25] P. Ostyakov, E. Logacheva, R. Suvorov, V. Aliev, G. Sterkin, O. Khomenko, and S. I. Nikolenko. Label denoising with large ensembles of heterogeneous neural networks. *CoRR*, abs/1809.04403, 2018. 2

[26] S. Paul, S. Roy, and A. K. Roy-Chowdhury. W-talc: Weakly-supervised temporal activity localization and classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 563–579, 2018. 2

[27] X. Peng, L. Wang, Y. Qiao, and Q. Peng. Boosting vlad with supervised dictionary learning and high-order statistics. In *European Conference on Computer Vision*, pages 660–674. Springer, 2014. 2

[28] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007. 2

[29] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, volume 1, page 6, 2008. 2

[30] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5734–5743, 2017. 2

[31] M. Skalic and D. Austin. Building a size constrained predictive model for video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 1, 2, 3, 4, 5

[32] K. Soomro, A. R. Zamir, and M. Shah. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2012. 2

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 2

[34] Y. Tang, X. Zhang, J. Wang, S. Chen, L. Ma, and Y. Jiang. Non-local netvlad encoding for video classification. *CoRR*, abs/1810.00207, 2018. 2

[35] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 2

[36] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484, 2005. 2

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017. 4, 5

[38] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *International journal of computer vision*, 103(1):60–79, 2013. 2

[39] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 126(2-4):375–389, 2018. 2

[40] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2442–2449. IEEE, 2009. 2

[41] Z.-H. Zhou, M.-L. Zhang, S.-J. Huang, and Y.-F. Li. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320, 2012. 2