Logistic Regression is Still Alive and Effective: The 3^{rd} YouTube 8M Challenge Solution of the IVUL-KAUST team

Merey Ramazanova [1], Chen Zhao[1], Mengmeng Xu[1], Humam Alwassel[1], Sara Rojas Martinez[1], Fabian Caba[2], and Bernard Ghanem[1]

[1]King Abdullah University of Science and Technology, Saudi Arabia
[2]Adobe Research, San Jose, CA, USA

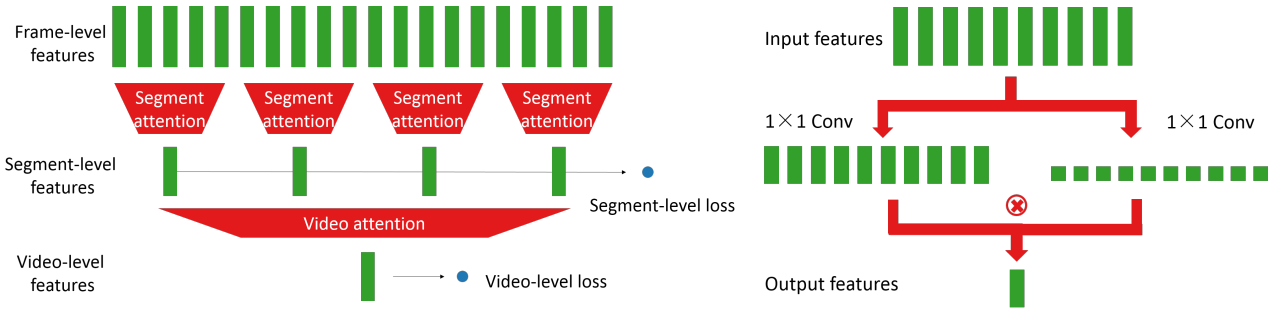

Figure 1: **Architecture of the attention models. Left:** overall hierarchical architecture; **Right:** implementation of the attention module. Frame-level features in each segment go through a segment-level attention module and produce segment-level features, which are used to compute a segment-level loss with segment-level annotations. All segment-level features in a video are input into a video-level attention module and produce a video-level features, which are used to compute a video-level loss with video-level annotations.

Abstract

In this report, we present our solution for the 3^{rd} YouTube-8M Video Understanding Challenge for a task of temporal localization of topics within a video. Our team achieves the 9^{th} place in the Public Leaderboard and the 11^{th} place in the Private Leaderboard with a difference of 4.5×10^{-4} from the 10^{th} gold medal winner. Overall, we train 20 different models independently and use their ensemble to predict segment scores. Along with a video classifier, we generate final scores for each segment. We use one-loss or two-loss training strategies for different models to make full use of video-level annotations and segment-level annotations. Furthermore, we adopt a teacher-student model and deep clustering to generate pseudo-labels to increase the amount of fully-annotated data.

1. Brief problem description

The task of the 3^{rd} YouTube-8M Video Understanding Challenge is to predict a topic for a 5-second length segment within a video. Videos are annotated with over 3800 topics, but only 1000 are included in the final evaluation for segment-level predictions [1]. The training set only includes video-level annotations, whereas the validation set incorporates both video-level and segment-level annotations.

Predictions are evaluated based on the Mean Average Precision (mAP) @ 100000, which is computed as follows

$$mAP@100000 = \frac{1}{C}\sum_{c=1}^{C}\frac{\sum_{k=1}^{n} P(k) \times rel(k)}{N_c},$$

where C is the number of topics (classes), $P(k)$ is the precision for top-k sorted predictions, n is the number of seg-

ments predicted per class, $rel(k)$ equals 1 if the item at rank k is a correct class, and 0 otherwise, and N_c is the number of positively-labeled segments for the each class.

2. Our solution

Our overall solution is illustrated in Fig. 2. We design three different architectures (blue, green and red blocks), each with multiple model variations, for segment prediction and train each model separately. Some of our models (e.g., the logistic model) only have one loss computed using the segment-level annotations in the validation set, whereas some models (e.g., the attention models) have two losses computed using both segment-level and video-level annotations in the training and validation sets. For prediction, we first use the model ensemble technique to combine the results of different segment-prediction models, then fuse the segment scores with video scores that are predicted by a video classifier trained on the whole training set.

To make full use of the validation set with segments-level annotations, we split it into two subsets of equal size: V_1 to be used for training and V_2 for validation. For models with only the segment-level loss, we use V_1 for training and V_2 for validation. For models with two losses, we train on both the training set and V_1 and validate on V_2.

To increase the amount of fully (segment-level) annotated data, we generate pseudo-labels.

The models will be elaborated in Section 3, and the training and prediction strategies in Section 4.

3. Models overview

We introduce our first two architectures for the segment prediction task: the segment-loss logistic model and the double-loss attention models. We will introduce these two categories of models in Section 3.1 and Section 3.2. In addition, we also adopt a video classification model to predict scores for each video as a whole, which are utilized to refine the segment scores. This will be described in Section 3.3.

3.1. Logistic model

We start with training a simple segment-wise logistic model (**S-Logi**) provided in the started code. We use V_1 for training the model. We then modify the original version of the model to predict a label for each frame instead of each segment and then mean-pool the predicted scores of all 5 frames in each segment to obtain its segment-level prediction. We refer to this model as the frame-wise logistic model (**F-Logi**).

3.2. Attention models

Inspired by weakly-supervised action detection [4], we design a hierarchical double-attention architecture as shown in Fig. 1 to take advantage of the video-level annotations in both the training set and the validation set and the segment-level annotations in the validation set. There are two attention modules: a segment-level attention module and a video-level attention module; each module computes a loss based on the corresponding annotations: segment-level loss and video-level loss. Apart from the basic implementation of the attention modules (Section 3.2.1) shown in the right of Fig. 1, we also design two different variations for the video-level module: Non-local Attention Module (Section 3.2.2), and Squeeze-Excitation Attention Module (Section 3.2.3). We also consider other module designs within this hierarchical architecture, such as FCFC described in Section 3.2.4.

3.2.1 Basic attention models

In the basic attention model (refered to as **Att^2, Loss^2**), we adopt the attention implementation shown in the right part of Fig. 1 for both segment-level attention and video-level attention modules.

3.2.2 Non-Local attention model

In the non-local attention model (**NL**), the segment-level attention module is the same as the one in the basic model. For the video-level attention, to make use of correlations between different frames, we insert a temporal non-local block [7] at the beginning of the video-level attention module. So the input features to the video-level attention module are the output of the non-local block.

3.2.3 Squeeze-Excitation attention model

In the squeeze-excitation attention model (**SE**), the segment-level attention module is the same as the one in the basic model. For the video-level attention, to aggregate video global information, we insert a temporal squeeze-and-excitation block [3] at the beginning of the video-level attention module. So the input features to the video-level attention module are the output of the squeeze-and-excitation block.

3.2.4 FCFC model

The FCFC model (**FCFC**) uses the same video-level attention module as the non-local attention model and uses a double fully-connected strategy to implement the segment-level module. In its segment-level module, it first applies fully-connected computation for features of each frame independently and then applies a second fully-connected layer with respect to each feature channel of all frames. In this way, we combine information across different channels and across all frames in the segment without introducing as

many parameters as a feature-flatten/fully-connected layer would do.

3.3. Video classification model

As some of our models (such as the logistic model) are trained with segment-level annotations only, they are not able to take advantage of the large training set which only has video-level annotations. Therefore, We expect a good video classification model fully trained on the training set to help refine predictions for segments. The obtained video-level classification scores are assumed to provide effective guidance for predicting segments in the same video. The segments in the video are less likely to belong to low-scored video categories since low scores indicate that these categories probably don't exist in the video at all. And vice versa.

Concretely, we use last year's 1st place model [5] to predict for each video in the test set. The predicted video scores are fused with segment scores predicted by the ensemble of the above-mentioned models (model ensemble will be described in Section 4.3). We use simple element-wise multiplication of both scores and have tried different combinations, such as $p_{final} = p_{video} \times p_{segment}^2$, $p_{final} = p_{video}^2 \times p_{segment}$, and $p_{final} = p_{video} \times p_{segment}$, where p refers to a score in one category.

4. Training and prediction strategies

4.1. Data expansion via pseudo-labels

Considering that only a small portion of the entire dataset has segment-level annotations, we try to expand the annotated data by producing pseudo segment labels so that the trained models can generalize better. We consider two methods to generate pseudo labels: 1) teacher-student model [6, 9]; 2) deep clustering [2].

1. **Teacher-student model** uses a well-trained segment classifier (e.g., the logistic model) to predict pseudo labels for the unlabeled segments in both the training set and validation set. The pseudo labels are post-processed to clean noise using the video-level annotations [8].

2. **Deep clustering** labels each frame by clustering. It first labels each frame in the annotated segments with their segment-level labels, then clusters all the video frames based on their deep features using k-means. In each cluster, the labeled frames vote to generate a label list in a decreasing-count order. The unlabeled frames are assigned the top label from the list if this label also exists in the frame's video-level annotations. If not, the following ones from the list will be considered until a label is assigned or the list is finished.

4.2. Training

For logistic models, we train them only on the validation set with one (segment-level) loss. However, for the attention models, we use both segment-level and video-level labels and optimize for two losses.

Considering that only a part of the training data has segment-level annotations, we only use the annotated segments to compute the segment-level loss by masking out all the others, whereas we use all the frames in one video to compute the video-level loss. We now describe three different ways to train the whole network.

1. **One loss training (S1).** It only uses the annotated segments in the validation set to train the logistic model.

2. **Two loss: two-step training (S2).** First, we pre-train the whole double-attention network using only the video-level loss on both the training set and V_1. Then, using the pre-trained model as initialization, we finetune the segment module using segment-level loss on v_1 (the last fully-connected layer to generate the segment-level scores is initialized using the last pre-trained fully-connected layer that generates the video-level scores). We reduce the learning rate for finetuning.

3. **Two loss: end-to-end training (S3).** This strategy trains the whole network from end to end in one pass with the two losses at the same time. We use equal weights for both losses.

4.3. Prediction

We use the model ensemble strategy to generate the final prediction scores of each segment. We train all 20 models independently. When inferring on the test set, we average the class score from a set of segment classifiers in a weighted manner. Table 1 shows our five weighting strategies.

Table 1: Weights for the model ensemble

Arch.	Num.	E1	E2	E3	E4	E5
NL	2	0.09	0.06	0.11	0.08	0.05
SE	2	0.09	0.06	0.11	0.08	0.05
Att^2	2	0.09	0.06	0.11	0.08	0.05
FCFC	1	0.08	0.05	0.1	0.07	0.04
$Loss^2$	1	0.05	0.03	0.05	0.04	0.02
F-Logi	3	0.09	0.14	0.05	0.08	0.12
S-Logi	1	0.06	0.14	0.04	0.05	0.12
Cluster	6	0	0	0	0.12	0.16

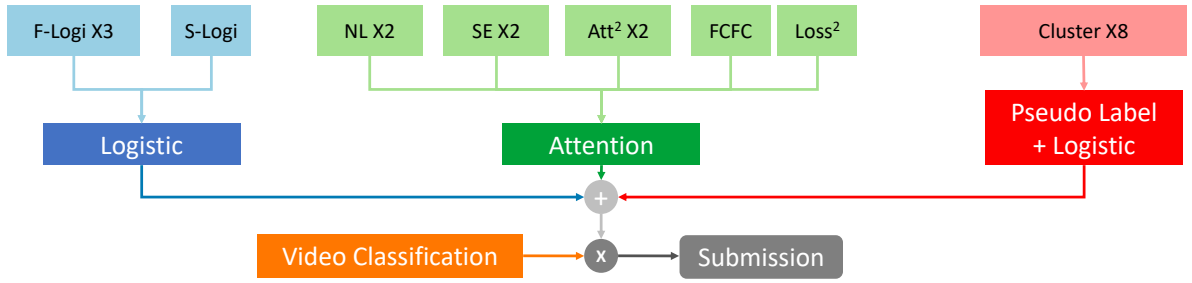

Figure 2: **Training and ensemble schema.** We train $4 + 8 + 8$ models for three types of architectures independently and multiply the averaged segment scores with their corresponding video classification score to generate the finial submission file.

5. Experiments

We use different training strategies (S1, S2, or S3) to train different models after testing different combinations of these strategies on the models mentioned above. Based on our experimental observation, the most efficient and effective way is to use S1 to train a simple network that does not contain too many parameters and use S2 or S3 to train a deep neural network to mitigate overfitting.

5.1. Hyperparameter search

To search for hyperparameters, we train models on the V_1 and evaluate their performance on V_2 simultaneously. Once the evaluation metric (e.g., the mean average precision) plateaus, we retrain the model using the strategy S1 with the same hyperparameters but doubling the training iterations.

5.2. Performance on the test set

5.2.1 Individual models

Table 2 summarizes the mean average precision (mAP) for each model.

Table 2: **Performance (mAP) of individual models**.

Architecture	Model	Training	mAP
Logistic	F-Logi	S1	0.791
	S-Logi	S1	0.789
Attention	NL	S2	0.789
	SE	S2	0.788
	Att^2	S2	0.788
	FCFC	S2	0.784
	Loss^2	S3	0.778
Pseudo Label	Cluster	S2	0.786

5.2.2 Ensemble of different models

Table 3 shows the mAPs of ensemble of different models. We use E1 and E4 as our final scores in the Kaggle leaderboard.

Table 3: **Performance (mAP) of model ensemble**

Model	E1	E2	E3	E4	E5
mAP (Pub)	**0.7957**	0.7868	0.7874	0.7941	0.7932
mAP (Priv)	**0.7864**	0.7774	0.7778	0.7846	0.7831

6. Acknowledgements

The authors would like to thank Ali Thabet and David Pugh for their fruitful discussions, the IBEX team at King Abdullah University of Science and Technology for help us use the computing resources, and the Google team for providing the Youtube-8M Tensorflow Starter Code.

References

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[2] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*, 2018.

[3] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[4] Phuc Nguyen, Ting Liu, Gautam Prasad, and Bohyung 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.

[5] Miha Skalic and David Austin. Building a size constrained predictive models for video classification. In *European Conference on Computer Vision*, pages 297–305. Springer, 2018.

[6] He-Da Wang, Teng Zhang, and Ji Wu. The monkeytyping solution to the youtube-8m video understanding challenge. *CoRR*, abs/1706.05150, 2017.

[7] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[8] Mengmeng Xu, Yancheng Bai, and Bernard Ghanem. Missing labels in object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[9] Chen Zhao and Bernard Ghanem. Thumbnet: One thumbnail image contains all you need for recognition. *arXiv preprint arXiv:1904.05034*, 2019.