Temporal Concept Localization within Video using a Mixture of Context-Aware and Context-Agnostic Segment Classifiers

Shih-hsuan Lee
shuanck@gmail.com

Abstract

This paper describes my solution to the 3rd YouTube-8M Video Understanding Challenge. To deal with the limited number of annotated segments, video-level models were pre-trained on the YouTube-8M frame-level features dataset to create meaningful video representations from frames. The weights of the two models were used to create two types of segment classifiers: context-aware and context-agnostic. The created classifiers were fine-tuned on the YouTube-8M segment-rated frame-level features dataset, with negative label mining, segment expansion, and a custom weighted cross-entropy loss. The final ensemble was created by taking arithmetic means of the class probabilities from six context-aware and six context-agnostic classifiers. It achieved seventh place in the private leaderboard. From the experiment results, a mixture of the two types of the classifier has been shown to have stronger performances on both the public and private test dataset. The code is publicly available at `https://github.com/ceshine/yt8m-2019`.

1. Introduction

In most web searches, video retrieval and ranking are performed by matching query terms to metadata and other video-level signals. However, we know that videos can contain an array of topics that aren't always characterized by the uploader, and many of these miss localizations to brief but important moments within the video. Temporal localization can enable applications such as improved video search (including search within a video), video summarization and highlight extraction, action moment detection, improved video content safety, and many others[2].

Google Research hosted The 3rd YouTube-8M Video Understanding Challenge[2] that asks participants to localize video-level labels to the precise time in the video where the label actually appears, and do it at an unprecedented scale.

The associated YouTube-8M Segments dataset is an extension of the YouTube-8M dataset[1] with human-verified segment annotations. About 237K segments from the validation set of the YouTube-8M dataset were annotated with a total of 1,000 classes.

For each one of these 1,000 classes, the participants are to produce a list of at most 100,000 segments from the test videos that are most likely to be labeled to it. This can be treated as a multi-label classification problem, where each segment can have zero, one, or more associated labels.

There are on average 237 annotated segments per class, which is generally considered to be too few to train even moderately sophisticated models. Therefore, a transfer learning approach is adopted in this paper to avoid overfitting and improve generalization. The video label prediction task from the previous year's YouTube-8M challenge was used to pre-train video-level models. Since the classes used in the segments dataset are a subset of the classes in the video dataset, we can expect pre-trained models to have learned to detect relevant frames to those classes. Further fine-tuning on the segments dataset should help the model to more accurately pinpoint relevant frames in shorter segments.

Directly fine-tuned models are context-agnostic, as they have no information about the other parts of the video. However, for some classes, such context information can be used to make better predictions. To make use of this information, context-aware models are created by combining a video encoder and a segment encoder. The video and segment embedding vectors from the two encoders are concatenated together to predict class probabilities.

On the other hand, context information might have little use to other classes, and training the model with such information could lead to unnecessary overfitting. By using a mixture of context-aware and context-agnostic segment classification models, different characteristics of the 1,000 target classes are expected to be better accommodated, and thus improves overall performance.

2. Related Works

NetVLAD was proposed in 2015[3] to aggregate local image descriptors into a global compact vector. It makes the traditional encoding method VLAD (Vector of Locally Ag-

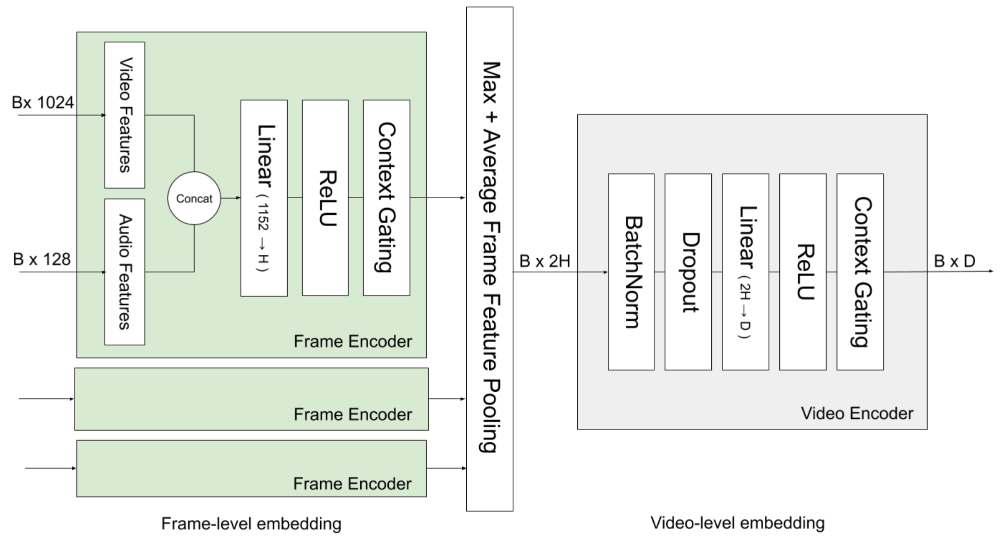


Figure 1. Context-gated DBoF video embedding.

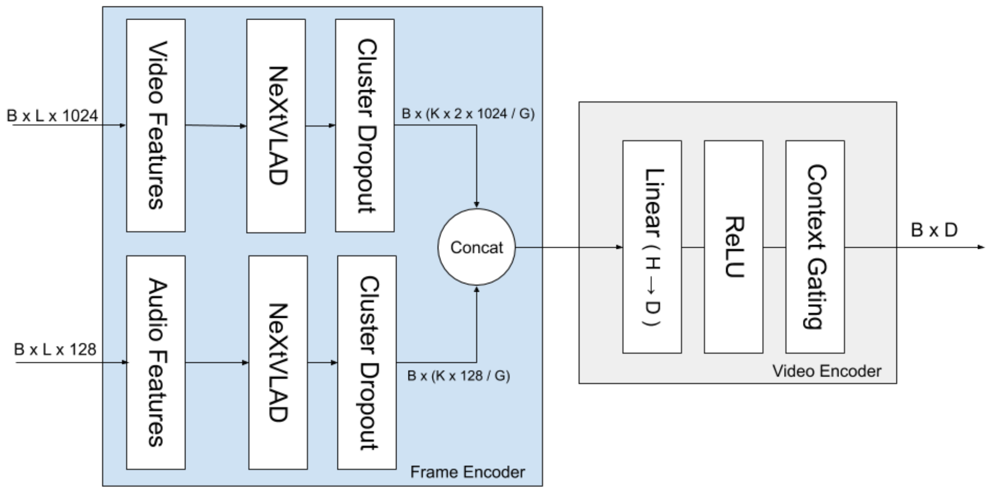


Figure 2. NeXtVLAD video embedding.

gregated Descriptors) differentiable and thus can be trained with stochastic gradient descent.

Miech et al.[11] demonstrates that NetVLAD can also be used to aggregate video features. Ways to make traditional Bag-of-visual-words[15] and Fisher Vectors[13] differentiable were also introduced.

Lin et al.[9] presents a ResNeXt-inspired improvement over NetVLAD called NeXtVLAD. It is shown to be both effective and parameter efficient in aggregating temporal information.

The DBoF (Deep Bag of Frames) approach was introduced as a baseline to the first YouTube-8M challenge[1]. It features an up-projection layer with shared parameters for all frames, followed by a pooling layer. It has recently been shown[12] that by incorporating the context-gating idea introduced by Miech et al.[11], the context-gated DBoF can produce one of the best single models in the second YouTube-8M challenge.

Semi-supervised and active learning approaches targeted at video classification and annotation[14][6][16] have been proposed for this kind of situation where the number of annotated samples is very limited. More general computer vision semi-supervised learning techniques[4] could also potentially be adapted to this problem.

3. Network Architecture

This section describes the neural network architecture used to pre-train models on the video frames dataset and later fine-tune them on the segments dataset.

3.1. Video-level Pre-training

The video classifier shown in Figure 3 has two parts: the video embedding model and the mixture of experts classifier. The video embedding sub-model takes the frames from a video with an arbitrary length and maps them to a vector with a fixed size D. The mixture of experts classifier then takes this vector to predict the probability of a label being assigned to this video.

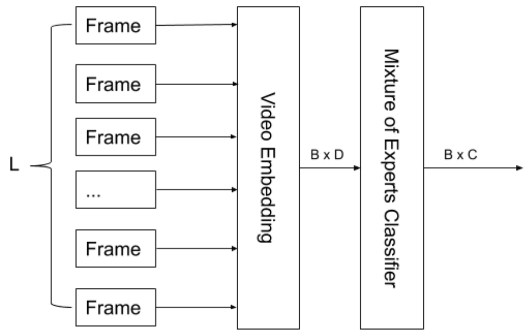


Figure 3. The architecture of the video/segment classifier.

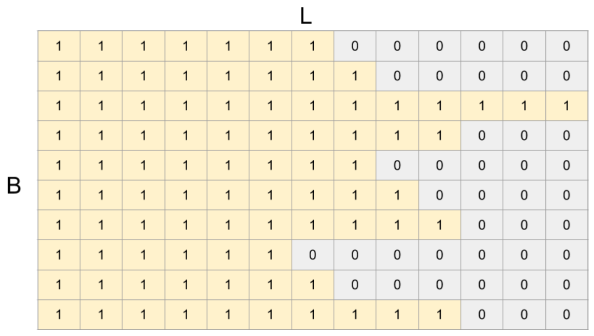
In practice, B videos are grouped into a batch of shape $(B, L, 1024 + 128)$ (video and audio features are concatenated). The batch length L is set to be the length of the longest video inside the batch. Shorter videos are padded and a mask is created for this batch (Figure 4). The mask will be used by the video embedding sub-model to mask out the padded frames.

For regularization and memory saving purposes, a maximum video length L_m is set for most of the models. A random sampling without replacement will be performed to retrieve L_m frames for videos with more than L_m frames.

This is a multi-label classification problem. The loss function used is the standard multi-label cross entropy (Equation 1).

$$L(\boldsymbol{p}, \boldsymbol{y}) = -\frac{1}{|C|} \sum_{c \in C} y_c \log(p_c) + (1 - y_c) \log(1 - p_c) \tag{1}$$

Only the videos tagged with one of the 1,000 labels used in the segments dataset were used in the pre-training. The decision was made to reduce the training time and make the mixture of experts classifier reusable in the later fine-tuning. Training with all video labels might be helpful, but it was not implemented due to the time constraint.



1	1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	0	0	0

Figure 4. An example of the mask for a batch. The value zero means the frame was padded.

Two types of video embedding architecture were used to make the ensemble more diverse: context-gated DBoF and NextVLAD. The final ensemble was created by taking arithmetic means of the class probabilities.

3.2. Video Embedding: Context-Gated DBoF

The context-gated DBoF (deep bag of frames) architecture[12] is used with some modifications for this paper (Figure 1). The frame pooling method is a combination of max pooling and average pooling. The masking of frames is conducted inside the pooling layer. No L2 normalization is applied before nor after the pooling.

Context-gated DBoF, comparing to NeXtVLAD, is more memory-efficient. Therefore, a larger maximum video length L_m can be used, and it could provide some diversity to the ensemble.

3.3. Video Embedding: NeXtVLAD

The NeXtVLAD layer[9] is both effected and parameter efficient in aggregating temporal information comparing to the NetVLAD layer (Figure 2). Only one change is made inside the NeXtVLAD layer for this paper — the batch normalization on the cluster assignment activation is performed cluster-wise (the number of channels is the number of cluster K) instead of globally (the number of channels is the number of cluster K times the number of groups G).

After the NeXtVLAD layer, a random 25% of the clusters are dropped during training by setting all the associated residuals to zero.

The masking is performed in the way described by Lin et al.[9], i.e., the attention weights are set to zero to padded frames.

3.4. Mixture of Experts Classifier

Some modifications to the mixture of experts classifier is made for this paper (Figure 5). A batch normalization layer and a dropout layer is added to both the gating network and the expert network. It is done so that fewer assumptions are made to the distribution of the input features, and to add some regularization that is independent of the video embedding model.

Instead of assigning different weights on each expert for different classes, the same set of weights is applied to all classes. This is to reduce the chance of overfitting, as the mixture of experts classifier is shared between video and context-agnostic segment classifiers.

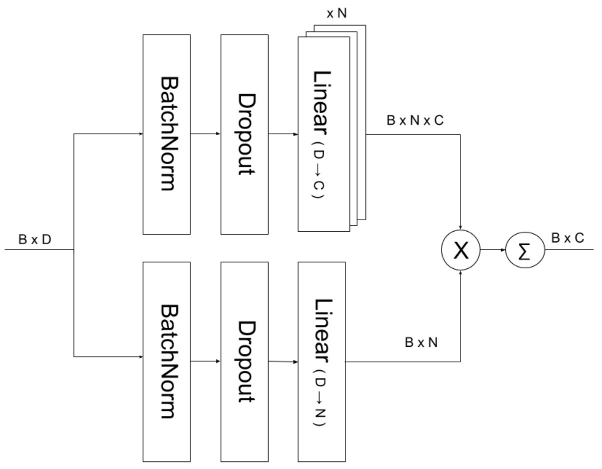


Figure 5. The architecture of the mixture of experts classifier. There's also a dummy "expert" that always predicts zero, but is not shown here.

3.5. SE Context Gating

The SE Context Gating[9] has been modified to make it more flexible and consistent with other parts of the model (Figure 6). A batch normalization layer is added at the start, and the batch normalization layer in the middle is placed after the ReLU activation.

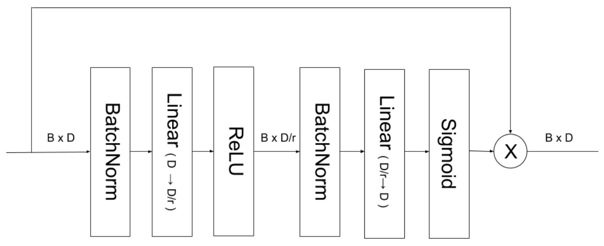


Figure 6. The architecture of the SE context gating layer.

3.6. Negative Label Mining in the Segments Dataset

Unlike in the video dataset, where all labels were evaluated when an annotator reviews a video, only one label was evaluated at a time for the segments dataset. That leaves 99.9% of the labels unattended. To make use of the video-level labels and make the model converge faster, negative label mining was used to create a list of highly-unlikely labels.

Almost always the segment-level labels come from the video-level labels, so we can say that any other labels not from the union of the set of video-level labels S_v and the set of (positive) segment labels S_s are very unlikely to be used in any of the segments in this video. These labels are then treated as the negative labels to every segment.

The loss function used in the fine-tuning stage is a combination of binary cross-entropy and multi-label cross-entropy. For a segment label j with a binary score y, the loss is calculated as in Equation 2. S contains all the 1,000 possible labels in the dataset. A stronger weight is placed on the positive examples to improve recall.

$$\begin{aligned} L(\boldsymbol{p}, y, j) = & -(\frac{3}{2} y \log(p_j) + (1-y)\log(1-p_j)) \\ & - \frac{1}{2} \frac{1}{|S - S_v \cup S_s|} \sum_{c \notin S_v \cup S_s} \log(1-p_c) \end{aligned} \tag{2}$$

3.7. Segment Expansion

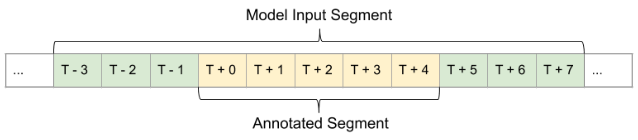


Figure 7. The segment expansion scheme.

The hypothesis is that while the human annotators only saw the 5 seconds in the segment, the vicinity of the segment might provide useful information about the segment (for example, the preparation of the goalkeeper before the penalty kick and the celebration after the kick). For a segment start at time T, the previous t and the next t frames are also included as the input of the segment classifier.

The value t was set to be 3 (as demonstrated in Figure 7) from experiments in an earlier stage of the competition, which showed the best performance among candidates including $t = 2, 3, 5$. However, the local cross-validation was not properly implemented at the time, so this choice might have caused some overfitting to the public leaderboard of the competition.

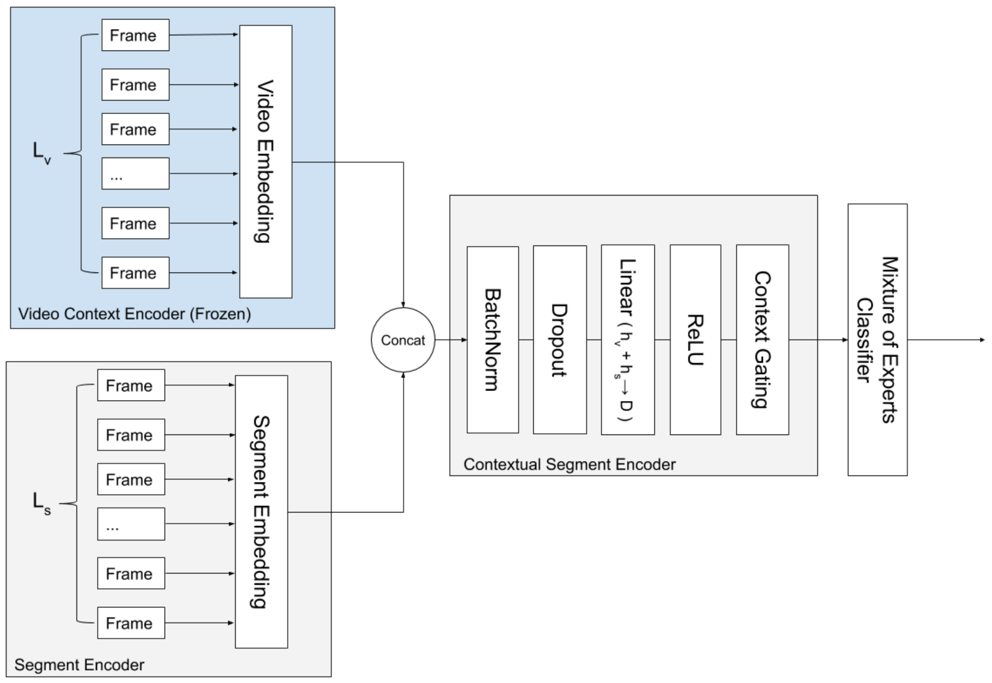


Figure 8. The architecture of the context-aware segment classifier

3.8. Context-Agnostic Segment Classifier

Because the pre-training task uses the same set of labels as in the segments dataset, we can use the pre-trained model, without any modification, to predict the class probabilities of a segment. The input of the model changes from (sampled) frames from a video to frames from expanded segments described in the previous section. Further fine-tuning is performed to make the model adapt to this shorter form of input.

Since the length of segments is fixed at $2t + 5 = \mathbf{11}$, the masking mechanism inside the model is turned off.

The model can only see the (expanded) segment itself, not the video containing the segment, hence the ”context-agnostic” in the name.

3.9. Context-Aware Segment Classifier

To make use of the context of a segment, two video embedding models taken from pre-trained video-level models (they can be from the same pre-trained model) are combined, with one acts as video context encoder, and the other as segment encoder. The concatenated embedding vectors are sent to a fully-connected layer with optional context gating to model the interaction between the context and the segment. And , a mixture of experts classifier is appended to obtain the class probabilities (Figure 8).

The weights of the video context encoder are frozen, i.e., they won’t be updated and their gradients will not be calculated. Fine-tuning the encoder did not improve cross-validation performance in the preliminary experiments, and it takes much more time to train, so I decided to keep them frozen.

The random sampling and masking described in section 3.1 were also applied to the video frame inputs. The first six segments and the final two segments were never annotated, judging from exploratory data analysis and confirmed by the organizer, as a way to avoid potential intro and credit sections. These eight segments are also dropped from every video before the frame sampling.

The learning rate of the segment encoder is set to be half the learning rate of the contextual segment encoder and the mixture of experts classifiers, in an attempt to mitigate catastrophic forgetting [8].

4. Experiment Results

This section describes the implementation details and presents the experiment results on the Youtube-8M Segments Dataset.

4.1. Datasets

For the Youtube8M frame-level features dataset that was used in the video-level pre-training, only 3658 shards out of 3884 shards in the training set and 744 shards out of 3884 shards in the validation set were used due to resource constraints. The lists of shards used are provided in the Github repository associated with this paper (`https://github.com/ceshine/yt8m-2019`).

The Youtube8m segment-rated frame-level features were split into 8 folds by shards. The segment classifiers were trained using data from different folds as the validation set.

4.2. Implementation Details

PyTorch 1.3.0 was used to implement models and create data loading pipelines. The list of shards (tfrecord files) is shuffled at the start of each epoch, and the examples are then read sequentially. Four workers with different random seeds were used to improve loading speed and create more randomness.

The AdamW[10] optimizer implemented as by Gugger et al.[5] was used with slanted triangular learning rates[7]. The warm-up stage occupies 25% of the total training steps. The max learning rate was .0003 for video-level models and .0002 for the segment classifiers (for the segment encoder in the context-aware classifier, the max learning rate was .0001). The weight decay was 0.1 for video-level models, and 0.02 for segment classifier. Biases and batch normalization weights were exempted from decaying. Dropout probability is set to be 50% except for the 25% cluster dropout probability in NeXtVLAD models.

The pre-training of a NeXtVLAD model ("nxvlad-2" in Table 2) took 15 hours on a Tesla P100 GPU. And the pre-training of a context-gated DBoF model ("dbof-3" in Table 1) took about 13 hours on a Tesla T4 GPU (this is a rough estimate from logs because the GCP instance was preempted several times during training).

The training/fine-tuning of context-agnostic models took 4 to 5 minutes per 1,000 steps for both NeXtVLAD and context-gated DBoF models on a GTX 1070 GPU.

The training of context-aware models took 8 to 12 minutes per 1,000 steps on a Tesla T4 GPU.

When predicting the test segments dataset, the first six segments and the last two segments were dropped as in the fine-tuning stage. This trick can improve the test $MAP@100,000$ by about 0.003. The predicted class probabilities from each model were discretized from $[0., 1.]$ into $[0, 9999]$ by linear scaling and rounding, and then dumped to a Numpy memory-mapped file.

The discretized probabilities of each class were loaded from disk, averaged across models, rounding to the nearest integer, and then put into 10,000 buckets. The top 100,000 probable segments for each class were extracted after all segments are predicted. The segments inside the same buckets were shuffled at the end to avoid accidentally using leaked information.

4.3. Model Evaluation

AUC (area under ROC curve) was used locally to evaluate validation performance for segment classifiers. Only the annotated labels were used to calculate the AUC (i.e., mined negative labels were not included). For the pre-training task, recall and precision rates of the positive labels with 0.5 cutoff were also recorded for reference.

$$MAP@100,000 = \frac{1}{C}\sum_{c=1}^{C}\frac{\sum_{k=1}^{n}P(k)\times rel(k)}{N_c} \qquad (3)$$

Three pre-trained context-gated DBoF models were used in the final ensemble (Table 1). Only one NeXtVLAD model (nxvlad-2) was used (Table 2). The results in Table 1 and Table 2 have shown that training with bigger batch sizes and more steps at the expense of the maximum length can produce better performances on the validation set. Generally speaking, better pre-trained models make better segment classifiers. The pre-trained models shown here are the best ones I was able to get at the end of the competition.

The batch size of 128 was used to train all segment classifiers. Local experiments showed that using smaller batch sizes (e.g. 32) hurts validation performance.

Three NeXtVLAD-based and three DBoF-based context-agnostic models were trained (Table 3). NeXtVLAD-based models have better AUC than DBoF-based ones, but their validation loss (Equation 2) is more unstable.

The final ensemble has six context-aware models with NeXtVLAD segment encoders, and a model with DBoF segment encoder is also shown in the last row in Table 1 for reference. The total parameter counts and trainable (not frozen) parameter counts are shown in Table 5.

The final ensemble (first row in Table 6) got $MAP@100,000 = 0.80102$ for the private test set, which landed it at the seventh place on the private leaderboard. Post-competition analysis shows that only the two best pre-trained models — "dbof-3" and "nxvlad-2" — were needed to get to the seventh place (last two rows in Table 6). It also shows that ensembles with only context-aware or context-agnostic models have worse performance than a mixture of two, strengthening the theory that the mixture of two can better accommodate the different characteristics of the target labels.

5. Conclusions

This paper presents a transfer learning approach to The 3rd YouTube-8M Video Understanding Challenge. Video-level models are pre-trained on the frame-level features dataset from the previous Youtube-8M challenges[1]. Two types of segment classifier are created from the video-level models and fine-tuned on the segment dataset. Negative label mining and a custom weighted cross entropy loss function are introduced to improve the fine-tuning. The experiment results have demonstrated that while using only one

Name	Batch Size	Steps	Params	H	D	Max Length	N	Log Loss	Recall	Precision
dbof-1	32	120k	38M	4096	2048	∞	5	0.003230	63.5	82.56
dbof-2	32	100k	36M	4096	2048	200	4	0.003314	61.84	83.44
dbof-3	128	100k	36M	4096	2048	150	4	0.002805	70.11	82.22

Table 1. Video-level Context-gated DBoF Models and their validation scores. (H: frame feature size; D: video feature size; N: # of MoE mixtures)

Name	Batch Size	Steps	Params	C	G	Max Length	N	Log Loss	Recall	Precision
nxvlad-1	32	120k	26M	64	32	200	4	0.003101	61.15	87.16
nxvlad-2	48	200k	33M	64	16	150	4	0.002744	65.33	87.70

Table 2. Video-level NeXtVLAD models and their validation scores. (C: # of cluster; G: # of groups; N: # of MoE mixtures.)

Base	Steps	Fold	Loss	AUC
nxvlad-2	12k	2	0.5769	83.00
nxvlad-2	8k	1	0.5010	83.57
nxvlad-2	12k	0	0.5570	82.96
dbof-3	8k	2	0.4965	82.71
dbof-3	8k	1	0.5104	82.56
dbof-3	9k	0	0.5233	82.34

Table 3. Context-Agnostic Models and their local CV scores. (Batch Size = 128)

Video	Segment	Steps	Fold	Loss	AUC
dbof-1	nxvlad-2	12k	6	0.4961	82.44
dbof-1	nxvlad-2	12k	7	0.4889	82.89
dbof-2	nxvlad-2	12k	5	0.4878	82.84
nxvlad-2	nxvlad-2	15k	5	0.5106	82.73
dbof-3	nxvlad-2	9k	3	0.4852	82.58
dbof-3	nxvlad-2	9k	4	0.4848	82.47
nxvlad-2	dbof-3	9k	7	0.4972	82.58

Table 4. Context-Aware Models and their local CV scores. The last row did not make it into the final ensemble. (Batch Size = 128. FC Dimension = 2048. Max length = 150.)

Video	Segment	Total	Trainable
dbof-1	nxvlad-2	64M	36M
dbof-2	nxvlad-2	62M	35M
dbof-3	nxvlad-2	65M	37M
nxvlad-2	nxvlad-2	59M	35M
nxvlad-2	dbof-3	67M	43M

Table 5. Parameter counts of context-aware models.

of the context-agnostic or context-aware architecture alone can already get seventh place on the private leaderboard, a mixture of context-agnostic and context-aware segment classifiers can give a further 0.006 to 0.01 improvements on the $MAP@100,000$ metric.

The hyper-parameters (e.g., training steps, batch sizes) of the models were not fully tuned due to time constraints. Most of the models in the final ensemble were trained in the last two days of the competition. Small improvements can be expected by further tuning the hyper-parameters.

Aware	Agnostic	Public	Private
6	6	**0.81217**	**0.80102**
0	6	0.80333	0.79020
6	0	0.80304	0.79388
3	3 (dbof)	0.80775	0.79630
3	3 (nxvlad)	**0.81051**	**0.80009**

Table 6. Ensemble Results. (Aware: # of context-aware models; Agnostic: # of context-agnostic models; Public & Private: $MAP@100K$ in the respective leaderboard.

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. 2016. Available at `http://arxiv.org/abs/1609.08675`.

[2] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. The 3rd youtube-8m video understanding challenge, 2019. `https://www.kaggle.com/c/youtube8m-2019/overview`.

[3] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. nov 2015. Available at `https://arxiv.org/abs/1511.07247`.

[4] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. MixMatch: A Holistic Approach to Semi-Supervised Learning. 2019. Available at `http://arxiv.org/abs/1905.02249`.

[5] Sylvain Gugger and Jeremy Howard. Adamw and superconvergence is now the fastest way to train neural nets. 2018. Available at `https://www.fast.ai/2018/07/02/adam-weight-decay/`.

[6] Guo-jun Qi, Yan Song, Xian-Sheng Hua, Hong-Jiang Zhang, and Li-Rong Dai. Video annotation by active learning and cluster tuning. pages 114–114, June 2006.

[7] Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. jan 2018. Available at `http://arxiv.org/abs/1801.06146`.

[8] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L. Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 3390–3398, 2018.

[9] Rongcheng Lin, Jing Xiao, and Jianping Fan. NeXtVLAD: An efficient neural network to aggregate frame-level features for large-scale video classification. 2018. Available at `https://arxiv.org/abs/1811.05014`.

[10] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. 2017. Available at `http://arxiv.org/abs/1711.05101`.

[11] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with Context Gating for video classification. pages 1–8, 2017. Available at `http://arxiv.org/abs/1706.06905`.

[12] Paul Natsev. Context-gated dbof models for youtube-8m. 2018. Available at `https://research.google.com/youtube8m/workshop2018/natsev.pdf`.

[13] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for imagecategorization. *CVPR, 2007*, 2007.

[14] Tomás Sabata, Petr Pulc, and Martin Holena. Semi-supervised and Active Learning in Video Scene Classification from Statistical Features. *Ial@Pkdd/Ecml*, 2192:24–35, 2018.

[15] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. *ICCV, 2003.*, 2003.

[16] Shangfei Wang, Huan Lin, and Yongjie Hu. Affective classification in video based on semi-supervised learning. In Derong Liu, Huaguang Zhang, Marios Polycarpou, Cesare Alippi, and Haibo He, editors, *Advances in Neural Networks – ISNN 2011*, pages 238–245, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.