YouTube-8M Video Understanding

Kun Huang
Department of Computer Science
Stanford University
khuang47@stanford.edu

Abstract

Video understanding is an important branch of computer vision. In this project, I apply NetVLAD, learnable gate and Mixture-of-Experts to large scale video-level classification using the most recently published Youtube-8M dataset. On top of that, I use transfer learning for temporal localization with Youtube-8M segments dataset. The model achieves a global average precision (gAP) of 85% for the video-level classification and a mean average precision (mAP) of 82% for the temporal localization.

1 Introduction

In most web searches, video retrieval and ranking is performed by matching query terms to metadata and other video-level signals. However, 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. Being able to assign video-level labels and localize actions and moments within videos can enable applications such as improved video search (including search within video), video summarization and highlight extraction, action moment detection, improved video content safety, etc.

For the video level classification of this project, the input to the networks is pre-extracted audio-video features of individual frames of the Youtube 8M videos. I use multilayer neural networks that include NetVLAD layer, gating layer, Mixture-of-Experts [10] (MoE) layer.

For the segments detection, the input has human-verified segment labels, and it comes with time-localized frame-level features so classifier predictions can be made at segment-level granularity. I use transfer learning based on the video-level model, then fine-tune the model to create a segment classifier.

2 Related work

Youtube-8m is the largest multi-label video classification dataset, each video is decoded at one-frame-per-second, and used a Deep CNN pre-trained on ImageNet to extract the hidden representation immediately prior to the classification layer [1], which makes the frame features lightweight for download.

For video level classification, I first tried Spatial-Temporal CNN model, with a 3-layer spatial CNN performing 1D convolution over the 1024 single-frame visual features, and a second, 3-layer temporal CNN performing 1D convolution over the frames of each video, with filters of depth 1024 [2]. It achieves 68% gAP in my experiment. LSTM [3] and GRU are also commonly used for solving video recognition problems. The two-stream Bi-directional LSTM model achieves a gAP of 82% [4]. At a high level, either the spatial-Temporal CNN model or the RNN model does not fully exploit the visual and audio features, although they've worked on temporal features, most of the signal extracted from the video still relies on the static features as shown in my experiment.

Besides CNN and RNN, Vector of Locally Aggregated Descriptors (VLAD), which aggregates local descriptors into a compact image representation, is widely applied in image classification [5]. On top of that, NetVLAD [6] integrates VLAD with supervised learning. With NetVLAD pooling the visual and audio features of Youtube 8M dataset, I'm able to achieve 85% gAP from the dataset.

Temporal Action Localization Network (TAL-Net) [7] is a new approach for action localization in video based on Faster R-CNN, which is first proposed to address object detection [8], where given an input image, the goal is to output a set of detection bounding boxes, each tagged with an object class label. At segment level, I finally choose to adopt transfer learning to tackle the temporal localization task, by using the video classification model as a base model, and fine-tuning the video level model on Youtube 8M segment dataset to create a segment classifier, which can be used for segments detection.

3 Dataset and Features

The original YouTube-8M dataset consists of millions of YouTube video IDs, with high-quality machine-generated annotations from a diverse vocabulary of 3,800+ visual entities. It comes with precomputed audio-visual features from billions of frames and audio segments. On average, each video has 3.0 labels. The videos are split into 3 partitions, Train : Validate : Test, with ratios 70%, 20% and 10% [1].

Videos are pre-processed to extract state-of-the-art 1.3 Billion visual and 1.3 Billion audio features. The visual features were extracted using Inception-V3 image annotation model, trained on ImageNet. The audio features were extracted using a VGG-inspired acoustic model on a preliminary version of YouTube-8M. Both the visual and audio features were PCA-ed and quantized to be as lightweight as possible [1].

Google recently released YouTube-8M Segments Dataset, which is an extension of the original YouTube-8M dataset, which includes human verified labels at the 5-second segment level on a subset of YouTube-8M videos, totalling 237k segments covering 1000 categories.

4 Network Architecture

4.1 Video-level classification

The video-level classifier consists of a NetVLAD layer , a fully connected layer, a gating layer and a final Mixture-of-Expert (MoE) classification layer. The network has a similar architecture with the one described in Learnable pooling with Context Gating for video classification [9].

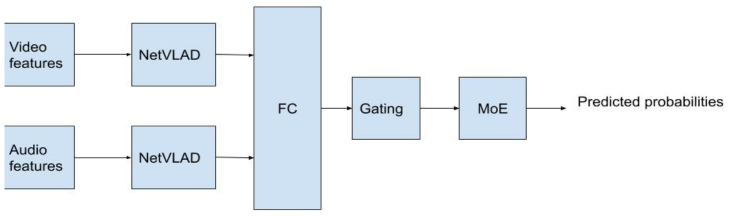


Figure 1: Video-level model. NetVLAD layer for features aggregation, MoE for the final classification.

4.1.1 Feature aggregation with NetVLAD

In computer vision, Vector of Locally Aggregated Descriptors (VLAD), which aggregates local descriptors into a compact image representation, is widely applied in image classification. The essential idea of VLAD is to accumulate, for each cluster center c_k, the difference of the vectors X_i

assigned to c_k. In other words, compute the distribution of data with regards to the class centers.

$$V(i,k) = \sum_{n=1}^{N} a_k(X_i)(x_i(j) - c_k(j))$$

The cluster centers c_k are predefined for VLAD, as a_k denotes if feature X_i belongs to the center k. For supervised learning, it's possible to learn the cluster centers c_k. And instead of using hard-assignment a_k, we can use softmax function for a soft-assignment [6]. NetVLAD is described as:

$$V(i,k) = \sum_{n=1}^{N} \frac{e^{w_k^T X_i + b_k}}{\sum e^{W_{k'} + b_{k'}}} (x_i(j) - c_k(j))$$

w_k, b_k and c_k are learnable parameters. Figure 1 demonstrates the logic flow and implementations of NetVLAD [6] for visual data of a single frame, while audio data follows a similar path.

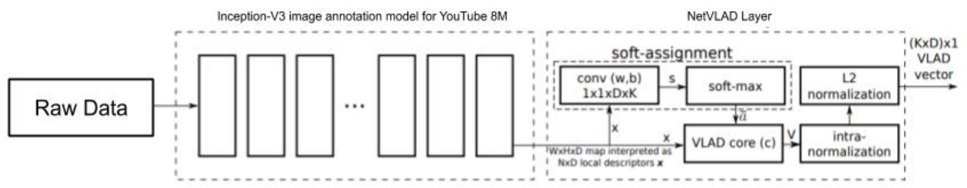


Figure 2: Raw visual/rgb features pass through a CNN architecture, and the extracted features are aggregated in NetVLAD layer.

4.1.2 Gating Layer

The gating layer transforms the input feature representation X into a new representation Y:

$$Y = \sigma(W * X + b) * X$$

where X is the input feature vector and σ is the element-wise sigmoid activation. W and b are trainable parameters. Similar to the gates in LSTM and GRU, the gating layer is to let the neural network itself learn the relevance of each input feature, and upweight relevant activations and suppress irrelevant activations.

4.1.3 Mixture of Expert

The idea of Mixture of Experts (MoE) is to train a number of neural nets, each of which specializes each part of the data [10]. A manager neural net will look into the input data and decide which specialist to give it to.

$$p(c_k) = \sum_{j=1}^{E} p(c_k|e_j)p(e_j)$$

MoE doesn't make very efficient use of data because the data is fractionated over all these different experts, so with small dataset it can't be expected to do very well. But it can make very good use of extremely large datasets. Figure 3 shows the implementation of MoE, the manager comes with a softmax layer to pick the experts. The experts are non-mutually exclusive multilabel classifiers.

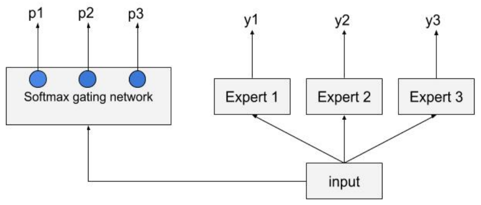


Figure 3: MoE: experts specifies in different regimes, manager determines the relevance of experts.

4.2 Temporal Localization

The segment localization model makes use of the video-level model, and it comes with two types of segment classifiers: a context-ignore classifier, which is just video-level model fine-tuned on the segment dataset, and a context-aware classifier. The final prediction is the average of the two.

4.2.1 Context-aware model

Context-aware model consists of three parts: a video embedding generator, a segment embedding generator and a fully connected classifier which is a fully connected layer with a Relu activation followed by a final MoE layer. A video embedding is created by removing the last layer (MoE layer) of a video-level model and feeding it with an entire video. A segment embedding is created by following the same strategy but fed with a segment. The embeddings are concatenated and are passed to the fully connected layer and the MoE layer. During training, the parameters of the video embedding generator, which is the context of the segment, will keep frozen in order to preserve the context features learned from the video level model. While the learning rate of the segment embedding generator is set to be half the learning rate of fully connected classifier.

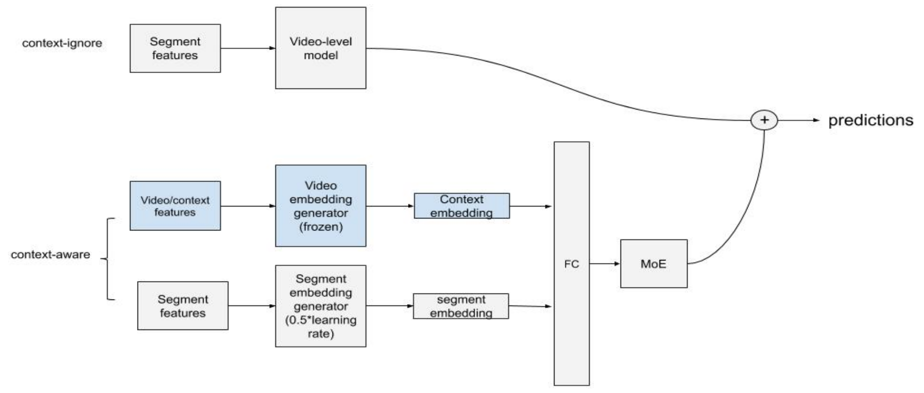


Figure 4: Temporal localization architecture.

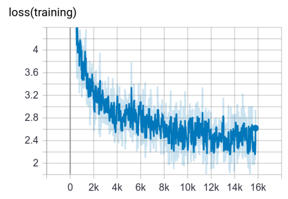
5 Experiments

Models are built using Tensorflow [11]. Pandas [12] is used to ingest the dataset vocabulary. For video-level model, the learning rate of 0.0002 with the learning rate decay of 0.8 every 1000000 steps gives the best performance. The batch size for training is 128 which gives a reasonable training speed and converge speed. For each batch, batch normalization is applied during training. Adam optimizer with L2 regularization is used during training.

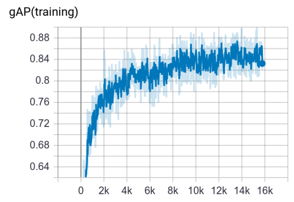
The experiments with gradient clip gives similar results. Parts of the reason could be that the ground truth is skewed, that only several individual classes out of the total 1000 classes are labeled as 1 for each video, so after the model learned how to distinguish negative labels, which is relatively easy for the classifier compared with recognizing positive labels, the cost mainly comes from positive labels which is small.

Moreover, Loss is calculated using cross entropy loss function: α * ŷ * log(y) + (1-ŷ) * log(1-y). Attempts were made to amplify the loss of the positive labels, that is, making α greater than 1. But no performance improvements are observed this way, due to the fact that the model learned very well on classifying negative labels, which makes the second part of the formula close to zero.

To speed up the training process, only videos labeled with the 1000 categories from the segment dataset were used in training. For segments, in addition to the 5 human-labeled frames, the previous and later 3 frames are also included to the training example. The feature data is normalized using l2 norm. Again, due to the fact that the labels are skewed towards negative labels, I can't use precision or accuracy as metrics. Instead, I adopt global average precision (gAP) for video-level prediction, and Mean average precision (mAP) for segment-level prediction where relevant segments are predicted for each class.

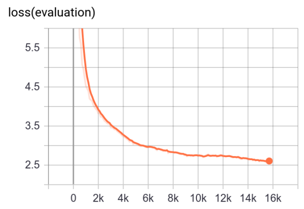


(a) loss

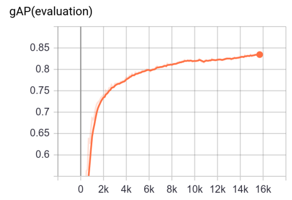


(b) gAP

Figure 5: Training process of the video-level classification

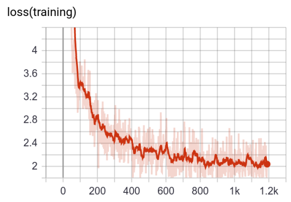


(a) loss

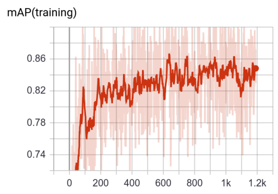


(b) gAP

Figure 6: Evaluation process of the video-level classification

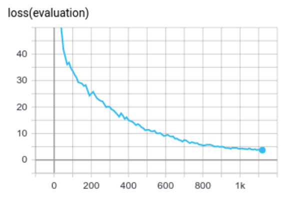


(a) loss

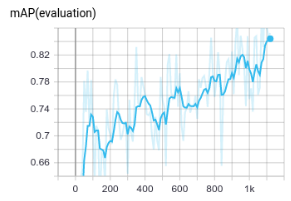


(b) mAP

Figure 7: Training process of the segment-level detection



(a) loss



(b) mAP

Figure 8: Evaluation process of the segment-level detection

6 Conclusion

With YouTube-8M dataset, I created a video-level classifier with NetVLAD layer, a fully connected layer with gating, and a Mixture-of-Experts layer for the final classification, which achieves gAP of 85% . The context-aware model and the context-ignore model combined is able to achieve mAP of 82% for the temporal localization task.

The algorithm is focused on static features, but it might be a good idea to also incorporate temporal features. Future experiments can include both the NetVLAD-associated model and a recurrent model (such as LSTM and GRU), and let the neural network to figure out the relevance of each model.

A caveat of my project is that the models are too large. The video-level model is 3.72G and the segment-level model is 10G, making it impractical to load the models to phones and other devices. It's important to have compact models that meet memory and computational requirements, and this is true even if working in cloud computational environments.

References

[1] Abu-El-Haija, Sami, et al. "Youtube-8m: A large-scale video classification benchmark." arXiv preprint arXiv:1609.08675 (2016).

[2] Gauthier, Alexandre, and Haiyu Lu. "YouTube-8M Video Classification." (2017).

[3] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[4] Li, Fu, et al. "Temporal modeling approaches for large-scale youtube-8m video understanding." arXiv preprint arXiv:1707.04555 (2017).

[5] Jegou, Herve, et al. "Aggregating local image descriptors into compact codes." IEEE transactions on pattern analysis and machine intelligence 34.9 (2011): 1704-1716.

[6] Arandjelovic, Relja, et al. "NetVLAD: CNN architecture for weakly supervised place recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[7] Chao, Yu-Wei, et al. "Rethinking the faster r-cnn architecture for temporal action localization." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[8] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

[9] Miech, Antoine, Ivan Laptev, and Josef Sivic. "Learnable pooling with context gating for video classification." arXiv preprint arXiv:1706.06905 (2017).

[10] West, David. "Neural network credit scoring models." Computers & Operations Research 27.11-12 (2000): 1131-1152.

[11] Abadi, Martín, et al. "Tensorflow: A system for large-scale machine learning." 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016.

[12] McKinney, Wes. "pandas: a foundational Python library for data analysis and statistics." Python for High Performance and Scientific Computing 14 (2011).