UTS submission to Google YouTube-8M Challenge 2017

Linchao Zhu Yanbin Liu Yi Yang
University of Technology Sydney
{zhulinchao7, csyanbin, yee.i.yang}@gmail.com

Abstract

In this paper, we present our solution to Google YouTube-8M Video Classification Challenge 2017. We leveraged both video-level and frame-level features in the submission. For video-level classification, we simply used a 200-mixture Mixture of Experts (MoE) layer, which achieves GAP 0.802 on the validation set with a single model. For frame-level classification, we utilized several variants of recurrent neural networks, sequence aggregation with attention mechanism and 1D convolutional models. We achieved GAP 0.8408 on the private testing set with the ensemble model. The source code of our models can be found in `https://github.com/ffmpbgrnn/yt8m`.

1. Introduction

The basic methodology towards untrimmed video classification can be 1) frame-level/clip-level feature generation; 2) leveraging video context information; 3) temporal aggregation. In the YouTube-8M dataset, two frame-level features are provided, which are static image features extracted by the Inception network [13] pre-trained on ImageNet and audio features extracted by a VGG-inspired acoustic model [7] trained on the first version of YouTube-8M. The original testing videos were not available during the competition, and new features could not be extracted. We thus focus on 2) and 3) in the paper. First, each frame is conditioned on the previous frames and the orders of the frames need to be leveraged. We then utilize aggregation methods which eliminate order information but abstract discriminative representations for classification. We first present our approach in Section 2 and the results are shown in Section 3. The conclusion is drawn in Section 4.

2. Our Approach

We first show our initial analysis of the dataset. We then present our approach using video-level and frame-level features.

2.1. Dataset Analysis

Videos have **multiple labels**. To calculate the loss, a simple method is to replace the softmax function with the sigmoid function. Second, we can use the softmax function but the labels need to be smoothed. Third, regarding the video tag assignment problem in a sequence to sequence scenario, label prediction can be generated at each decoding step with a softmax function. There is no order relation between the labels but we manually sort the labels in the vocabulary order. From our preliminary results, the sigmoid function always performs best.

The YouTube-8M dataset is **imbalance** that some categories (excluding top-level categories) have over 50k positive examples, *e.g.*, "Minecraft", while some categories have only 100 positive examples, *e.g.*, "Mammal" (Figure 1). We tried to keep the label balanced in a minibatch during training, *i.e.*, the number of positive examples are similar for the categories in a minibatch. However, it results in worse performance. We also tried to normalize the label weight with its frequency, *i.e.*, higher frequency labels have lower weights, but it does not help either. Our explanation is that the videos are imbalanced in the training, validation and testing set, and the loss calculated by random sampling estimates the "true" distribution better. In Figure 2, we show the label co-occurrence matrix on the training and validation set, which have very similar distribution.

The YouTube-8M dataset is also **noisy**. Abu-El-Haija *et al.* [1] reported that the precision and recall of labels are about 78.8% and 14.5%, respectively. Missing and noisy data are common in this dataset. To tackle this issue, we attempted to remove some noisy labels or complete the missing labels for each class in a certain amount, *e.g.*, 5%. Both methods have no influence on the performance. We plan to investigate this in the future. We did not deal with label noise or use positive negative sampling in the following models.

2.2. Video-level Classification

In video-level classification, one feature vector is provided for each video.

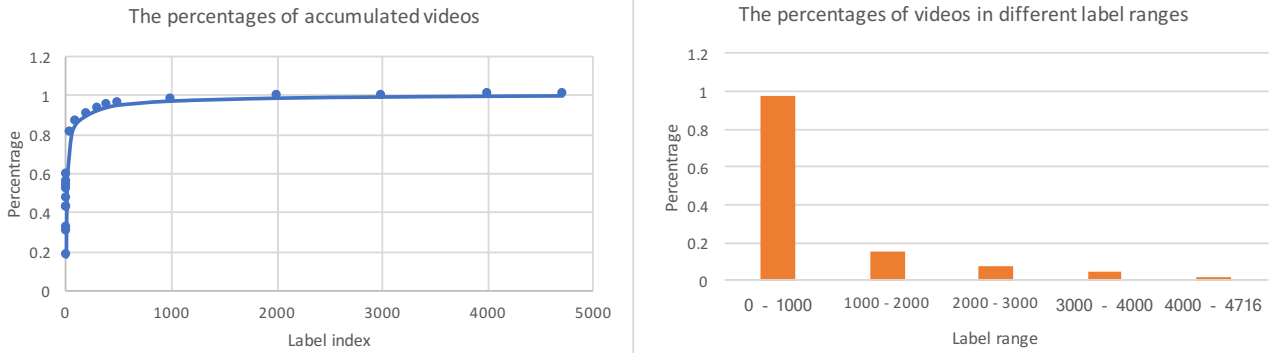

Figure 1. The vocabulary is basically ordered with regards to the number of positive instances in the class, *i.e.*, smaller label indices have more positive instances. We can observe that most videos (97%) are in label range [0, 1000). In label range [0, 50), there are 80% of the videos, while less than 1% videos are in [3000, 4716).

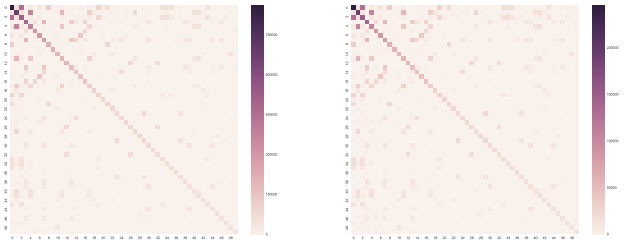

Figure 2. Label co-occurrence matrix of top 50 labels on the training set (left) and the validation set (right).

2.2.1 Mixture of Experts (MoE)

Given input x, the mixture of experts layer is directly applied on the input, which is calculated by,

$$f_{moe}(x) = \sum_{i=1}^{k} G(x)_i E_i(x), \tag{1}$$

where $G(x)_i$ is the gating weight for expert i, and $E_i(x)$ is the prediction of the ith expert. We adopted the MoE layer used in [1], where sigmoid activation is used on the expert output and the gating weights are soft assigned with a softmax function.

2.2.2 Parallel MoE (PMoE)

In our preliminary experiments, we found that increasing the number of mixtures from 2, 4 to 8 will increase the classification performance. We aim to train MoE with hundreds of mixtures. [11] used a sparsely-gated MoE with thousands of mixtures where only a small number of mixtures are updated in the training. Another way is to use hierarchical MoE [11]. In the multi-label classification setting, we can simply use model parallelism that the vocabulary is divided into small label groups, where one MoE layer only predicts a subset of the whole vocabulary. There are different ways to divide the vocabulary. One way is to divide the labels in vocabulary order. For example, in the 200-mixture setting, we can divide the labels into ranges $\{[0, 500), [500, 1000), \ldots, [4500, 4716)\}$. We can also randomly sample non-overlap labels from the vocabulary.

2.3. Frame-level Classification

In frame-level classification, the inputs are in variable lengths, which are zero-padded to sequence length 300. We use RNNs to model temporal transitions, while attention or VLAD are for aggregation. 1D ResNet is also used.

2.3.1 Recurrent Neural Networks

RNNs have been successfully applied to video classification [9, 12], where Long Short-Term Memory (LSTM) [8] and Gated Recurrent Unit (GRU) [4] are commonly used. We use the following variants in our submission (Figure 3):

a) **Two-layer stacked RNN**. We stack two layer RNNs and evaluate the performance of different RNN architectures, *e.g.*, GRU and LSTM, on this setting. A MoE layer is added on the RNN output for classification.

b) **Stacked RNN with context injection**. Following [15], we use a seq2seq model to reconstruct video contexts, where an encoder encodes a clip to reconstruct its previous and next clips. The encoder thus encodes information beyond the seen clip. We design the stacked RNN model which takes x and the outputs of the context encoder as inputs, which is

$$x = \text{stop\_gradient}(\text{ContextRNN}(x)) + x. \tag{2}$$

Note that we do not backpropagate the gradient through the context encoder.

c) **Hierarchical RNN with hidden MoE layer**. One problem with the stacked RNN is that it is computation expensive and the gradient may still vanish when the sequence length grows. Following [10], we use a hierarchical RNN where the first RNN encodes video segments information

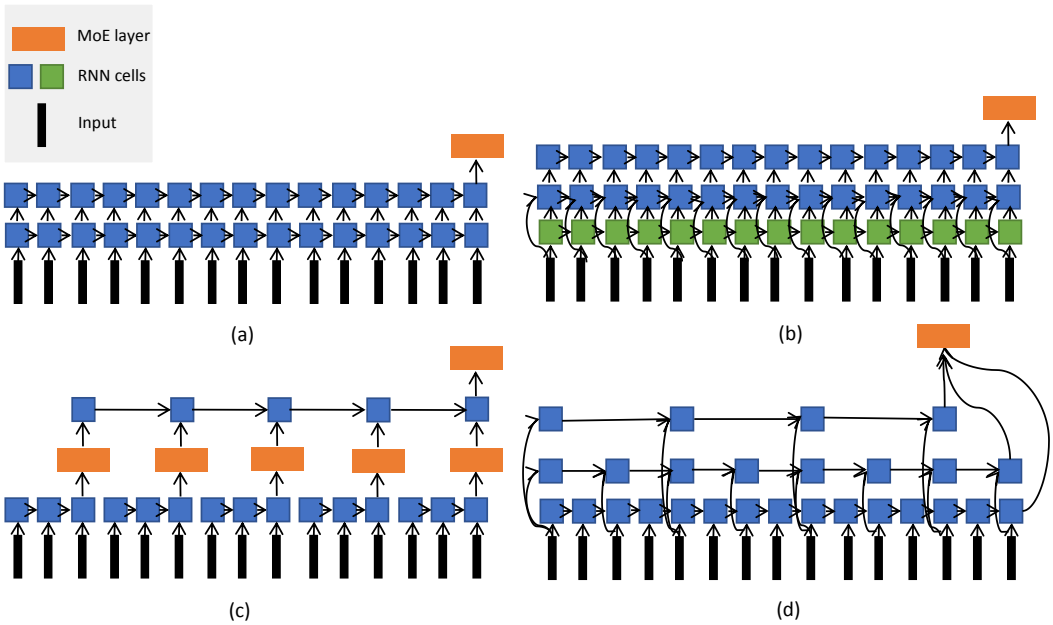

Figure 3. RNNs variants used in our submission. (a) Stacked RNN; (b) Stacked RNN with context injection; (c) HRNN; (d) Multi-scale RNN.

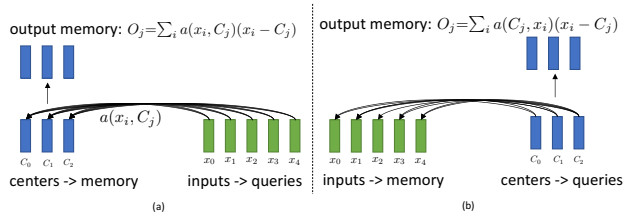

Figure 4. Two schemes to calculate the kernel a. The difference is that in (a), softmax is calculated over centers while in (b) it is calculated over the inputs.

and the second RNN further aggregates each segment. We plug a MoE layer between the layers but no activation funcion is used on the expert outputs.

d) **Multi-scale RNN**. In this variants, we divide the inputs into several groups with different intervals. The original frames are sampled at 1FPS, we further subsample the frames with lower frame rate. States of different rates are then concatenated for classification.

2.3.2 Vector of Locally Aggregated Descriptors

Instead of using the final state/output for classification, we can also use a weighted sum over the outputs of all steps. Linear annealing weights [9] and attention with multiple hops can be used. We modified the attention mechanism as follows,

$$output = flatten(softmax(\mathbf{W}_a tanh(\mathbf{W_i}\mathbf{X}^T))\mathbf{X}), \tag{3}$$

where $\mathbf{X}$ is the input with shape [seq_length, input_size], $\mathbf{W_i}$ is the projection matrix with shape [proj_size, input_size], $\mathbf{W_a}$ is the attention matrix with shape [num_hops, proj_size].

Vector of Locally Aggregated Descriptors (VLAD) can aggregate frame-level ConvNets activations for video classification [14]. The length of the feature vector is usually over 60,000 which is also very sparse. Thus training classifiers, *e.g.*, logistic regression, on VLAD can easily lead to overfitting. Training VLAD end-to-end has been first attempted in [2] and later Girdhar *et al*. [5] applied NetVLAD to action recognition. The original VLAD is calculated by,

$$O(j) = \sum_{i=1}^{N} a(x_i, C_j)(x_i - C_j), \tag{4}$$

where x_i is the ith input, C_j is the jth center, $a(x_i, C_j)$ denotes the membership of the input x_i to center C_j, *i.e.*, $a(x_i, C_j) = 1$ if center C_j is closest to input x_i and 0 otherwise. Instead of hard assignment, [2, 5] used a soft assigment with

$$a(x_i, C_j) = \frac{\exp\{-\alpha||x_i - C_j||^2\}}{\sum_{k'} \exp\{-\alpha||x_i - C_{k'}||^2\}}, \tag{5}$$

which is further decoupled into,

$$a'(x_i, C_j) = \frac{\exp\{w_j^T x_i + b_j\}}{\sum_{k'} \exp\{w_{k'}^T x_i + b_{k'}\}}. \tag{6}$$

We use another kernel $a(x_i, C_j)$ that is commonly used in attention,

$$a''(x_i, C_j) = \frac{\exp\{W_a tanh(W_c C_j + W_i x_i + b)\}}{\sum_{k'} \exp\{W_a tanh(W_c C_{k'} + W_i x_i + b)\}}. \quad (7)$$

Another constraint is added to minimize $\sum_{i=1}^{N} a(x_i, C_j)||x_i - C_j||^2$.

Equation 4 is very similar to the attention machnism with two differences. First, Equation 4 has an additional residual connection. Second, the weighted sum is applied on the memory in attention, while in the VLAD case, the weighted residuals are concatenated. We would investage more in the future. We show two different schemes to calculate the kernel a in Figure 4.

2.3.3 1D Convolution

By replacing the 2D kernel with 1D kernel and keeping other setups unchanged, we train a 1D ResNet-50 [6] on the features provided. The input "1D image" has shape [300, 1]. As the length of the input channel is 1,536, we increase the first convolutional channel to 512, and the following channels are 512, 1,024, 2,048, 4,096. Global average pooling is applied and softmax activation is replaced by a sigmoid activation.

2.4. Fusion

In our submission, for each class, we fuse the predictions weighted by the Average Precision (AP) score on the validation set. We normalize the per-class APs of all models as the class-level fusion weights. Given M predictions, and the APs on the validation set for ith prediction are $\text{ap}_{i,0}, \text{ap}_{i,1}, \ldots, \text{ap}_{i,4715}$, we can calculate the weights for class c over all models by

$$W_c = \text{norm}(\text{ap}_{0,c}, \text{ap}_{1,c}, \ldots, \text{ap}_{M-1,c}). \quad (8)$$

The normalization function can be ℓ_1-norm, ℓ_2-norm or other norms.

3. Experiments

Two features are provided for each frame sampled at 1FPS. We did not investigate late fusion and the two features were concatenated as inputs to all the models. We trained the models with the following settings unless otherwise stated. We optimized the models with ADAM optimizer and the learning rate decays 0.9 every 4,000,000 examples. The initial learning rate for MoE models is 0.01, and we use learning rate 5e-4 when training RNNs. GAP is the evaluation metric and we report both the GAP and the mAP scores on the validation set. The metrics are reported using single checkpoint for each model.

3.1. Video-level Classification

We show the results of the MoE models in Table 1. The performance increases when the number of mixtures increases. For 2, 4, 8, 50, 100 mixtures, we used a single MoE layer and we trained on CPU if OOM occurred on GPU. For the 200-mixture model, we divided the vocabulary in order with window size 500, while the labels are randomly selected in "200random". We can see the performance difference between "200" and "200random" is small. For the 1000-mixture model, we only trained on the first 1,000 labels. We averaged the prediction of 200-mixture and 1000-mixture which achieves GAP 0.8141 on the validation set.

3.2. Frame-level classification

We first show the results of our stacked RNN variants on Table 2. By default, we used 2 stacked layers and 2-mixture MoE for classification. From the result, we can see that LSTM performs worse than GRU and bidirectional RNN performs slightly worse than the basic RNN. Besides, increasing the number of mixtures did not boost the performance. In "GRU+fc", we added an output projection on the GRU states.

In Table 3, we show the result of other RNNs. "HGRU" is the hierarchical GRU with the default window size 15. We used 2-mixture MoE and dropout keep ratio is set to 0.5. "LN HGRU" is the HGRU where the activations are layer normalized [3], which does not have improvements. We observed over-fitting in these models and thus shuffled the order of input frames ("HGRU (random order)"), but it leads to worse performance. "Multi-scale GRU" has similar performance to "HGRU". A slightly improvement can be obtained by changing the windows size from 15 to 20 ("HGRU@20").

The results of 1D ResNet and our modified end-to-end VLAD are shown in Table 4. Note that in these two models, the inputs are the original frame-level features rather than outputs of RNNs. The initial learning rate is set to 0.1 for 1D ResNet and it decays 0.1 every 10 million examples. In our preliminary experiments, we can obtain accuracy 82.28% (rgb only) on UCF-101 split 1 using our version of VLAD, where the average pooling result is 78.57%. Notably, only 10 center is used in UCF-101 and the dimension of the feature vector is 2,560. We used 10 centers on YouTube-8M and did not try other settings. Further investigation would be made in the future.

3.3. Fusion

In our final submission, we fused all the above models to obtain the prediction. Some models were not used in the final submission, and we did not report their results. The results on validation and test (private) can be seen in Table 5. ℓ_3-norm fusion obtained the highest results on vali-

# of Mixtures	mAP	GAP
2	0.4150	0.7820
4	0.4180	0.7890
8	0.4205	0.7932
50	0.4262	0.8000
100	0.4249	0.8011
200	0.4291	0.8023
200random	0.4291	0.8019
200+1000	**0.4430**	**0.8141**

Table 1. The results of different number of mixtures in MoE for video-level classification.

Models	mAP	GAP
LSTM	0.3884	0.8044
BiLSTM	0.4035	0.8042
GRU	**0.4302**	**0.8147**
BiGRU	0.4280	0.8092
GRU+4-mixture	0.4258	0.8127
GRU+fc	0.4187	0.8128

Table 2. The results of stacked RNN variants. In this case, we found increasing the number of mixtures in MoE did not improve the performance.

Model	mAP	GAP
HGRU	0.4429	0.8243
LN HGRU	0.4385	0.8220
HGRU (random order)	0.4078	0.8150
Multi-scale GRU	**0.4464**	0.8225
HGRU@20	0.4445	**0.8246**
StackGRU+context (*)	-	0.8210

Table 3. The results of other RNN variants. * denotes the model is not fused in the final submission.

Model	mAP	GAP
1D ResNet	0.4294	0.8176
AttnVLAD+2-mixture	0.4282	0.8004

Table 4. The results of 1D ResNet and AttnVLAD. The "AttnVLAD" model is our modified end-to-end VLAD.

dation set and this submission got GAP 0.84081 on the test set (private).

4. Conclusion

In this paper, we presented our solution to YouTube-8M Challenge. We found hierarchical GRU model with MoE has the best single model performance, and Multi-scale GRU performs slightly worse. Motion information is missed but we would like to evaluate how much the motion information contributes to the performance on the untrimmed noisy data in the future. Another future work is to evaluate the aggregation methods on the large-scale dataset through an ablation study.

Model	GAP (val)	GAP (private)
Average Fusion	0.840819	-
ℓ_1 APs	0.841035	-
ℓ_2 APs	0.841142	-
ℓ_3 APs	**0.841169**	**0.84081**

Table 5. The fusion results. By simply averaging all prediction, we can obtain GAP 0.840819. Slight improvement can be achieved with our fusion method.

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.

[2] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, 2016.

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2015.

[5] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[7] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, et al. Cnn architectures for large-scale audio classification. In *ICASSP*, 2017.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[9] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[10] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. In *CVPR*, 2016.

[11] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017.

[12] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[14] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative CNN video representation for event detection. In *CVPR*, 2015.

[15] L. Zhu, Z. Xu, and Y. Yang. Bidirectional multirate reconstruction for temporal modeling in videos. In *CVPR*, 2017.