Google Cloud & YouTube-8M Video Understanding

Video Group 2 Final Report

Yang Yu   Zhuoshu Yang

Abstract: In this paper, we introduce the system we developed for the Youtube-8M Video Understanding Challenge, in which a large-scale benchmark dataset was used for multilabel video classification. In the video-level classification, three methods are proposed to increase the classification performance, i.e. Logistic Regression, Mixture-of-Experts(MoE) and Hierarchical-Mixture-of-Experts (HMoE). In terms of the official evaluation Global Average Precision (GAP) at 20, our best submission to Kaggle achieves 0.79719 on the private test dataset and 0.79679 on the public test dataset.

Keywords—video classification; cloud computing; machine learning; deep learning

1. Introduction

Video captures a cross-section of our society, and major advances in analyzing and understanding video have the potential to touch all aspects of life from learning and communication to entertainment and play. Large-scale datasets such as ImageNet have been key enablers of recent progress in image understanding. By supporting the learning process of deep networks with millions of parameters, such datasets have played a crucial role for the rapid progress of image understanding to near-human level accuracy.

The YouTube-8M dataset (Fig. 1) contains over 7 million videos, and is split into 3 parts: training data, evaluation data, and testing data. All videos have ids that can be used to retrieve the original videos on YouTube. Videos in the training and evaluation data contain labels that correspond to video classes. The YouTube-8M dataset comes with frame-level features (1.7TB) including the RGB and audio features, and video-level. features (30GB) including the mean RGB and mean audio features. Each video is decoded at 1 frame per second, and is capped at the first 300 frames. The RGB feature matrices are created by feeding raw pixels through an Inception network trained on ImageNet, and taking the ReLU activation of the last hidden layer. The video-level data is created by taking the mean (or the standard deviation) of the features across frames.

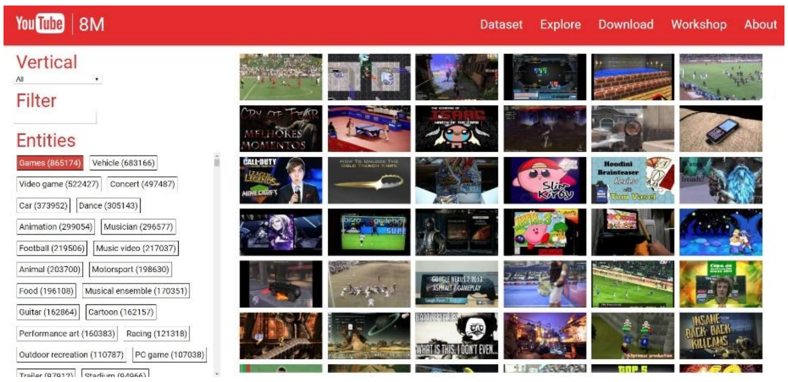
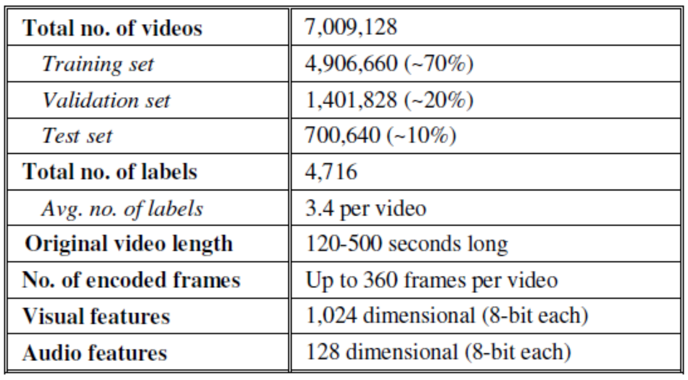


Fig1. Youtube-8M dataset exploration



Total no. of videos	7,009,128
Training set	4,906,660 (~70%)
Validation set	1,401,828 (~20%)
Test set	700,640 (~10%)
Total no. of labels	4,716
Avg. no. of labels	3.4 per video
Original video length	120-500 seconds long
No. of encoded frames	Up to 360 frames per video
Visual features	1,024 dimensional (8-bit each)
Audio features	128 dimensional (8-bit each)

Table1. YT-8M V2 dataset statistics

In this project, submissions are evaluated using Global Average Precision (GAP) at 20. For each video, we should submit a list of predicted labels and their corresponding confidence scores and the evaluation server takes the predicted labels that have the highest 20 confidence scores for each video, then treats each prediction and the confidence score as an individual data point in a long list of global predictions, to compute the Average Precision across all of the predictions and all the videos. In detail, the evaluation metric (GAP) is calculated as:

$$GAP = \sum_{i=1}^{N} p(i)\Delta r(i).$$

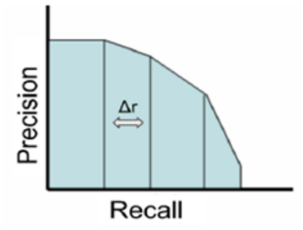


Fig.2 Precision Recall curve

where N is the number of final predictions with up to 20 label/confidence pair per video (Kaggle only considers the first 20 label/confidence pairs for each video), p(i) is the precision of the ith pair, and Δr(i) is the recall. Precision is the fraction of correct labels that we predict over the number of labels that we predict. Recall is the fraction of correct labels that we predict over the number of ground truth labels. This corresponds to the area under the curve of Fig.2, a precision-recall plot.

2. Method

In this section, we mainly demonstrate all the methods that we utilize to get the final submission in details. Our code is written in Python with TensorFlow. The reasons are:

(1) TensorFlow is an open source software library that makes it to perform complex machine learning concepts, especially deep learning;

(2) Kaggle provides starter code that performs training and inference, and that code uses TensorFlow. We stored and processed the data on Google Cloud Platform. It allows users to store large amount of data and easily create and train machine learning models.

There is a large amount of existing framework available on Google's platform using TensorFlow. This framework gave us a basis to build off and significantly reduced the amount of time required to build our models.

Three models were trained on video-level features and empirically tested. Models include:

- ***Logistic Regression***: comprises of 4,716 one-vs-all binary logistic regression classifiers for each label.
- ***Mixture of experts (MoE)***: model comprising of k experts, each being a version of the above Logistic Regression classifier model.
- ***Hierarchical Mixture of Experts(HMoE)***: a primary gating network chooses a sparse weighted combination of "experts", each of which is itself a secondary mixture-of-experts with its own gating network.

2.1 Logistic Regression with L2 Regularization

The model comprises of 4,716 one-vs-all binary logistic regression classifiers for each label trained on video-level features. Fig.3 below details its architecture. Logistic Regression classifiers was selected as the baseline model given its relative ease of implementation and its simple intuitive strategy of resolving a multi-label classification problem into multiple single-label classification problems. During training, the classifiers were independently trained on video-level features with L2 regularization penalty of 1e-8. During inference, each video is scored by each classifier. Video-level label scores are then obtained by a simple concatenation of all scores across classifiers.

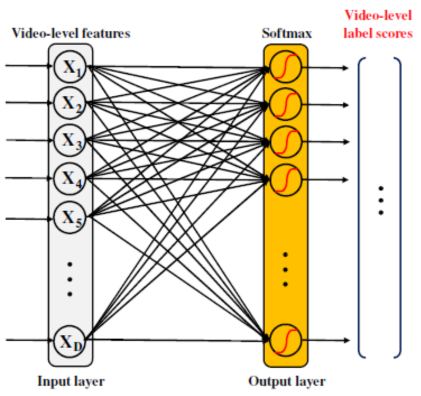
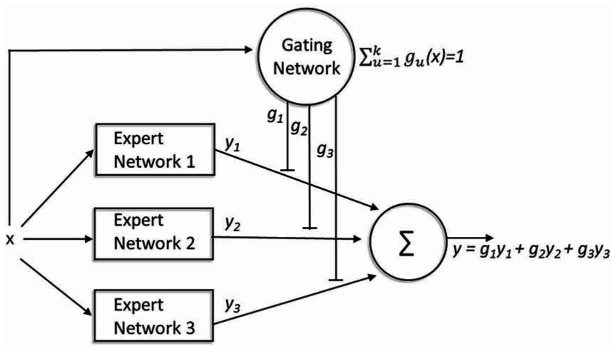
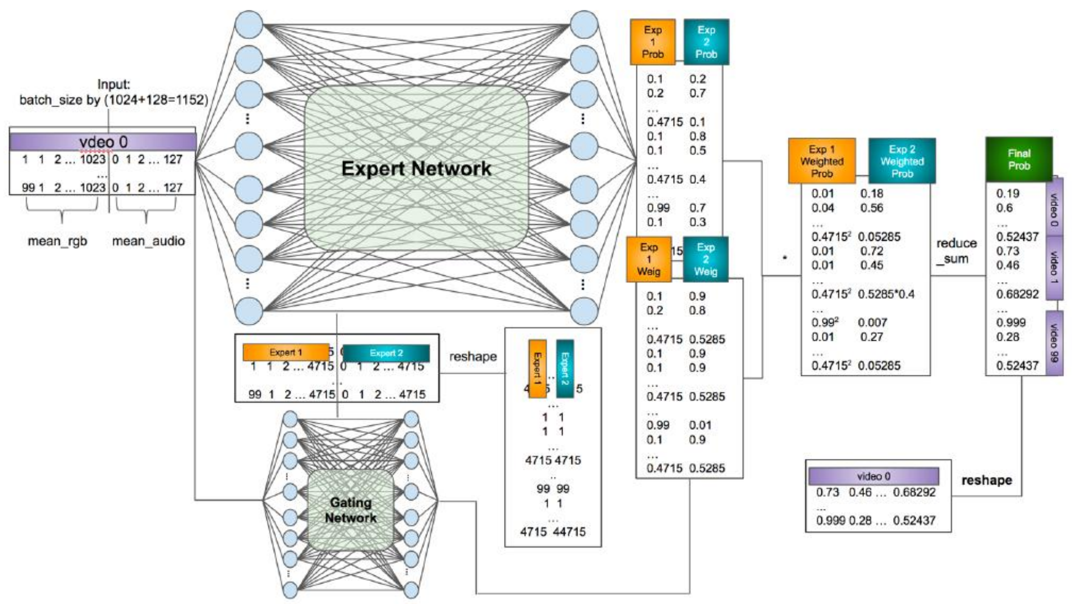


Fig.3 Logistic Regression Classifier

2.2 Mixture of experts (MoE)

The MoE model comprises of k experts where the final video-level prediction is a mix (weighted sum) of the predictions from each expert. Fig.4 above details the architecture.





The diagram above shows the how an input matrix is transformed into an output matrix by a MOE model with two experts. The entire process starts with the input matrix. Each row represents a video, so in this example, 100 videos are processed as a batch. The first 1024 elements of the video are its mean RGB features. They are appended by is 128 mean audio features. The actual matrix calculation starts in the expert network. The first layer of the expert network has 4716*2=9432 neurons, numbered as 0, 1, 2, …, 9432 from the top to the bottom. Each neuron conceptually corresponds to a video class and each expert looks at the input matrix independently, so if neuron 4716 corresponds to the same class as neuron 0, 4717 the same class 1, etc.

However, packages like TensorFlow process the entire input matrix rather than one video at a time for optimization opportunity. In other words, the actual calculation is performed at the matrix level. In this example, the 9432 neurons together produce a matrix with shape (batch size, 9432), which is shown below the first layer. A reshape operation is performed before the data is sent to a second layer, so that each row represents a particular class for a video. It has only two numbers, and no longer (and cannot) represent an entire video. The first video becomes the yellow vector plus the blue one, a rectangle block. Now this matrix is sent to the second layer, which is essentially applying a sigmoid function to each element as described on equation below.

$$\sigma(x) = \frac{1}{1+e^{x}}$$

The dimension does not change, but each element is "squashed" to a number between 0 and 1. In machine learning, this is essentially logistic regression. However, a single probability is needed both for human understanding and for the final submission purpose: it is not acceptable to say that video 0 belongs to class "Cat" with probability 0.1 and 0.2. A naive approach to combine them is give each expert 50% of weight, so the final submission is 0.15. However, the MOE model is more than that: it introduces another neural network, called the gating network, to give each expert a weight on each class for each video. That is the bottom neural network in the diagram. The only difference between the expert network and the gating network is the second layer; instead of a sigmoid function, the gating network uses a softmax function

$$\rho(x_{ij}) = \frac{e^{x_{ij}}}{\sum_{j=0}^{n} e^{x_{ij}}}$$

In this case, n=2 because there are two experts. Notice that each element of the resulting matrix also depends on the other elements in the same row. Specifically, each row of the resulting matrix adds up to 1. This nice property of the softmax function gives the resulting matrix the following interpretation: each element is a weight assigned to an expert for a class and a video. Because it has the same dimension as the expert matrix on the top, there is a one-to-one mapping between the expert matrix and the gating matrix. The first row of the gating matrix can be interpreted this way: expert 1 has only 10% weight on the question, while expert2 has 90% weight on the question. "Is video 0 a video about Cat?" By element-wise matrix multiplication and row summation (called "reduced_sum" in TensorFlow), the final probability that video 0 belongs to "Cat" is 0.1*0.1+0.2*0.9=0.19. Each video and class combination has a probability, which is exactly what isneeded. Finally, it is reshaped to the original dimension.

2.3 Hierarchical Mixture of Experts(HMoE)

The hierarchical mixture of experts (HME) is a parametric probabilistic model for solving regression and classification problems (Jordan and Jacobs 1994). The HME can be viewed as a conditional mixture model in which the distribution of the target variables is given by a mixture of component distributions in which the components, as well as the mixing coefficients, are conditioned on the input variables. The component distributions are referred to as experts, while mixing coefficients are controlled by gating distributions. Fig.5 below shows a two-layer Hierarchical Mixture of Experts architecture.

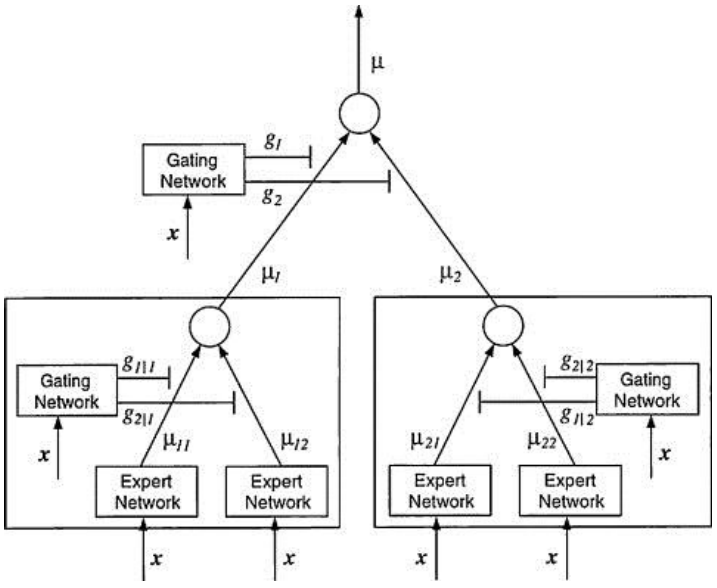


Fig5. Two-Layer Hierarchical Mixture of Experts

If the hierarchical MoE consists of a groups of b experts each, we denote the primary gating network by Gprimary, the secondary gating networks by (G1, G2..Ga), and the expert networks by (E0,0, E0,1..Ea,b). The output of the MoE is given by:

$$y_H = \sum_{i=1}^{a}\sum_{j=1}^{b} G_{primary}(x)_i \cdot G_i(x)_j \cdot E_{i,j}(x)$$

Our utilizations on Hierarchical Mixture of Experts is that the primary level of MoE model offers a coarse classified result on the 4716 video labels and the secondary level of MoE gives conditional probabilities on the previous coarse results while gating network(Softmax) gives weights on each of the probabilities.

3. Result and Submission to Kaggle

We have achieved results that range from a GAP score of about 72% to almost 80% using our different methods. From all the methods, we have mostly focused our efforts on the MoE model and its advanced model HMoE. In order to figure out whether to increase the numbers of the experts will influence the performance of GAP, we set a parallel experiment of training different experts{2,4,6,8} on the MoE model. And the best scores achieved by different models are shown in Table 2.

Methods	GAP
Logistic Regression	0.7164
Mixture of k Experts {k=2}	0.7548
Mixture of k Experts {k=4}	0.7696
Mixture of k Experts {k=6}	0.7796
Mixture of k Experts {k=8}	0.7878
Hierarchical Mixture of Experts	0.7992

Table2. final results

As we can see, apparently, the increase of number of experts leads to a higher performance of GAP. Further, a two-layer Hierarchical Mixture of Experts has an even better performance than a single MoE model that have eight experts, which means our idea of optimizations on the HMoE model do work in the experiment.

Here’s the GAP graphs that produced by Tensorboard for each models.

3.1 Logistic Regression

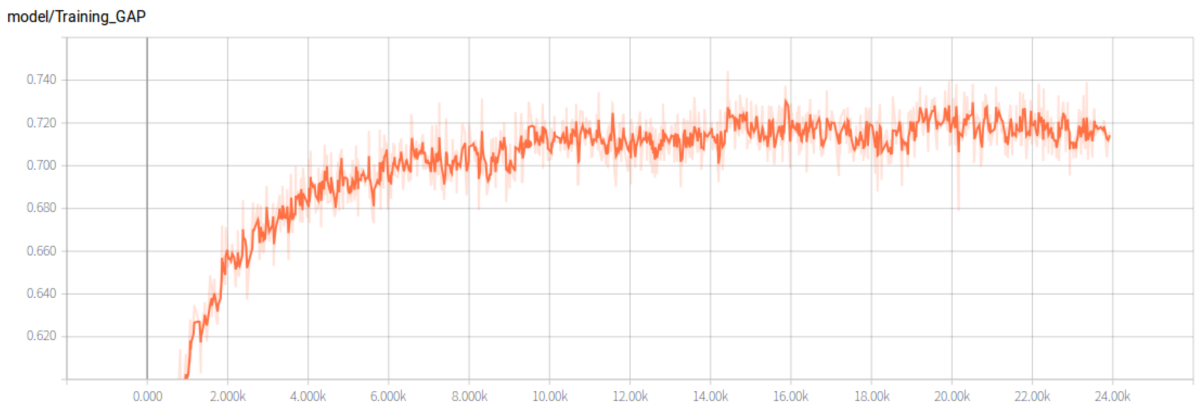


Fig.6 GAP by training step for the Logistic Regression model

3.2 Mixture of Experts

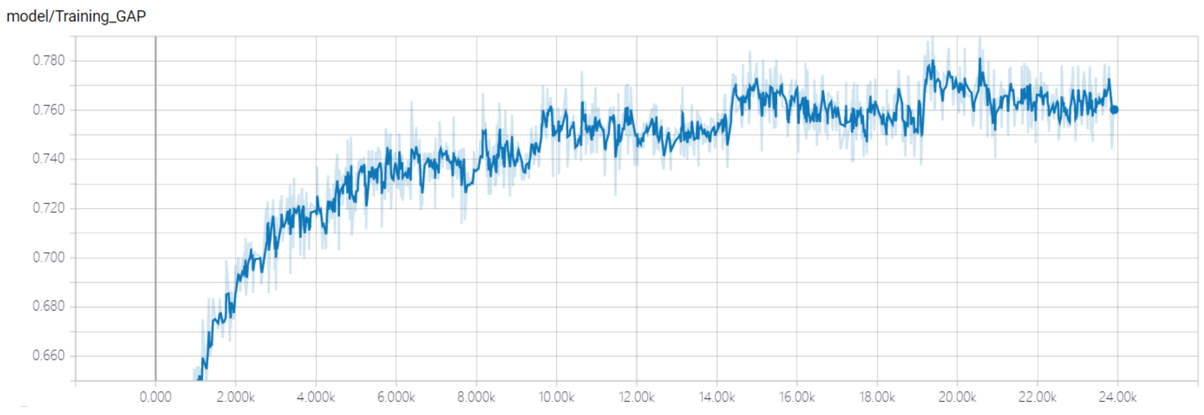
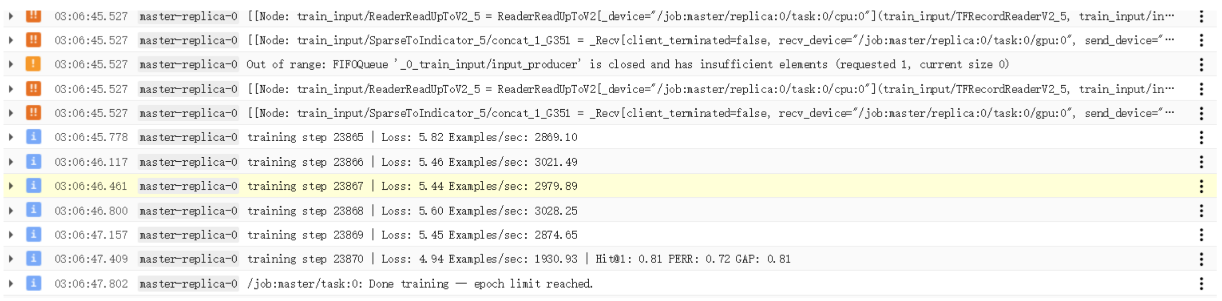


Fig.7 GAP by training step for the Mixture of Experts {k=8} model

3.3 Hierarchical Mixture of Experts



```
03:06:45.527  master-replica-0  [[Node: train_input/ReaderReadUpToV2_5 = ReaderReadUpToV2[_device="/job:master/replica:0/task:0/cpu:0"](train_input/TFRecordReaderV2_5, train_input/in…
03:06:45.527  master-replica-0  [[Node: train_input/SparseToIndicator_5/concat_1_G351 = _Recv[client_terminated=false, recv_device="/job:master/replica:0/task:0/gpu:0", send_device="…
03:06:45.527  master-replica-0  Out of range: FIFOQueue '_0_train_input/input_producer' is closed and has insufficient elements (requested 1, current size 0)
03:06:45.527  master-replica-0  [[Node: train_input/ReaderReadUpToV2_5 = ReaderReadUpToV2[_device="/job:master/replica:0/task:0/cpu:0"](train_input/TFRecordReaderV2_5, train_input/in…
03:06:45.527  master-replica-0  [[Node: train_input/SparseToIndicator_5/concat_1_G351 = _Recv[client_terminated=false, recv_device="/job:master/replica:0/task:0/gpu:0", send_device="…
03:06:45.778  master-replica-0  training step 23865 | Loss: 5.82 Examples/sec: 2869.10
03:06:46.117  master-replica-0  training step 23866 | Loss: 5.46 Examples/sec: 3021.49
03:06:46.461  master-replica-0  training step 23867 | Loss: 5.44 Examples/sec: 2979.89
03:06:46.800  master-replica-0  training step 23868 | Loss: 5.60 Examples/sec: 3028.25
03:06:47.157  master-replica-0  training step 23869 | Loss: 5.45 Examples/sec: 2874.65
03:06:47.409  master-replica-0  training step 23870 | Loss: 4.94 Examples/sec: 1930.93 | Hit@1: 0.81 PERR: 0.72 GAP: 0.81
03:06:47.802  master-replica-0  /job:master/task:0: Done training — epoch limit reached.
```

Fig.8 training result from a two-layer Hierarchical Mixture of Experts model

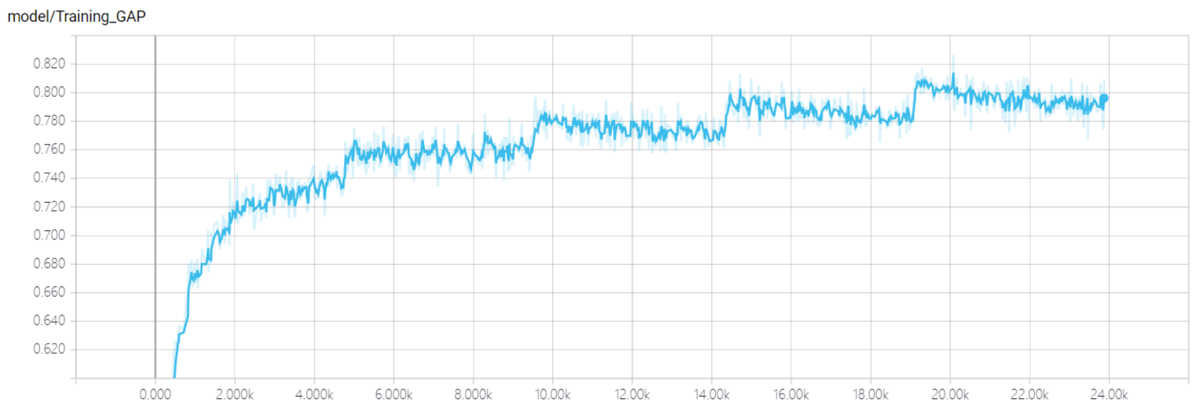


Fig.9 GAP by training step for Hierarchical Mixture of Experts model

3.4 Overall Models Comparison

Fig. 10 shows the GAP comparison between all the models.
The sky blue curve represents for Hierarchical Mixture of Experts model.
The navy blue curve represents for Mixture of Experts model where experts equals 8.
The orange curve represents for Logistic Regression with L2 Regularization.

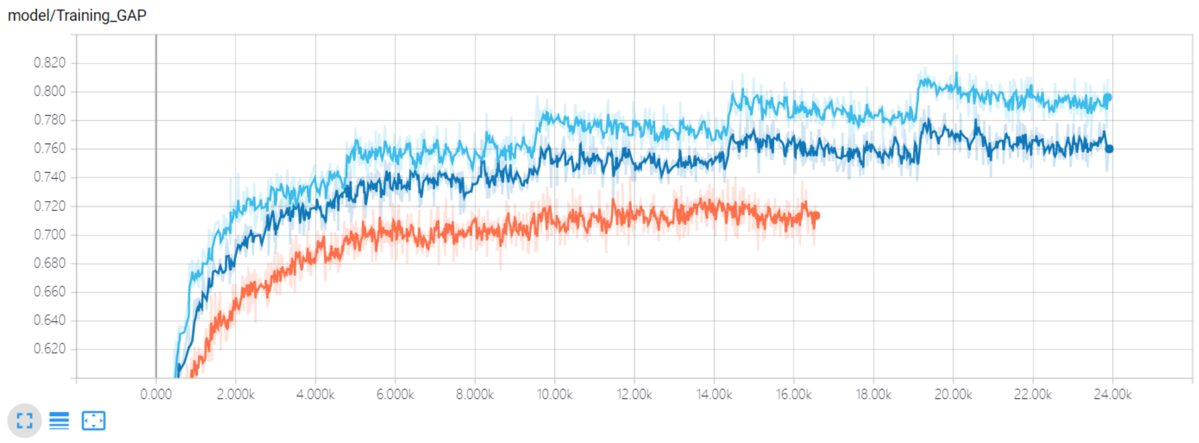
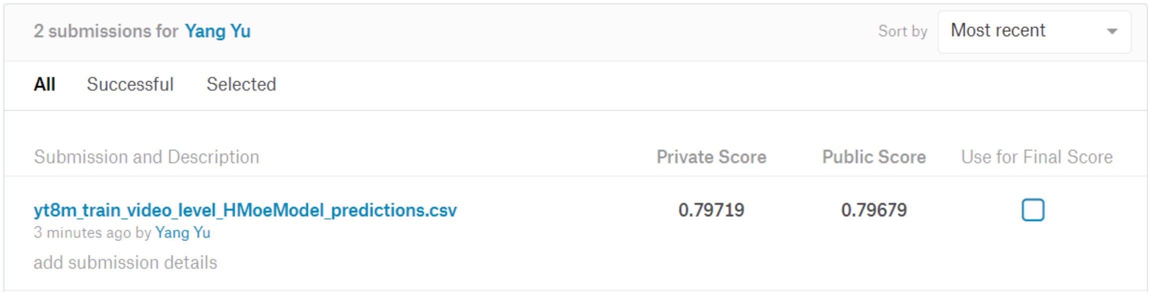


Fig. 10 Overall Models Comparison

3.5 Submission to Kaggle

We uploaded our best prediction csv file from Hierarchical Mixture of Experts model. From the scoreboard(Fig.11), Hierarchical Mixture of Experts model achieves 0.79719 on the private test dataset and 0.79679 on the public test dataset which is a bit lower than the local-trained GAP.



2 submissions for Yang Yu

Sort by Most recent

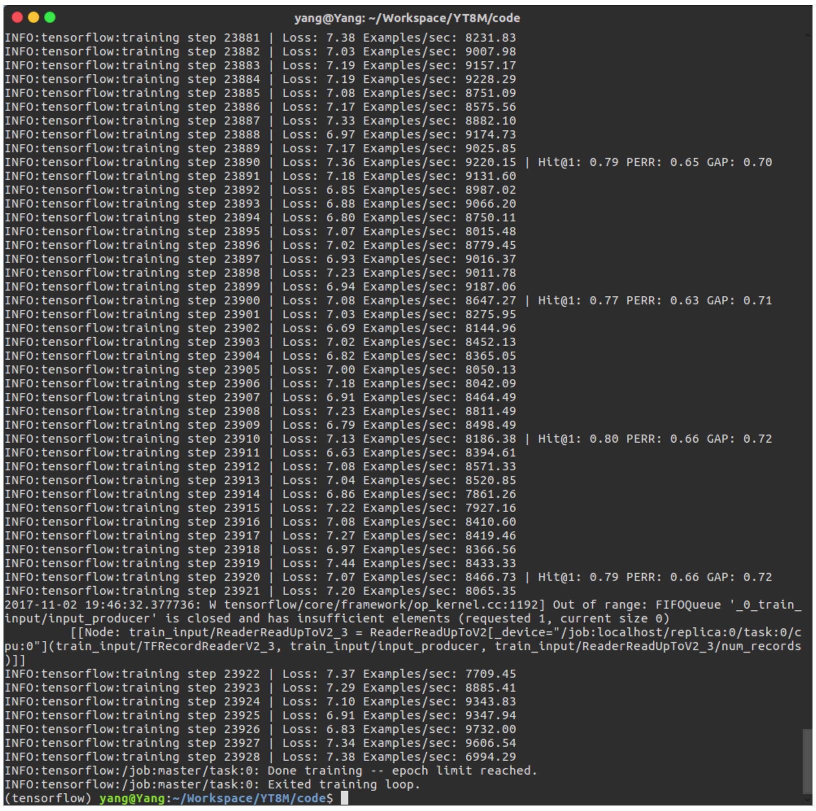
All Successful Selected

Submission and Description	Private Score	Public Score	Use for Final Score
yt8m_train_video_level_HMoeModel_predictions.csv 3 minutes ago by Yang Yu add submission details	0.79719	0.79679	☐

Fig.11 Kaggle Scoreboard

4. Challenges and Solutions

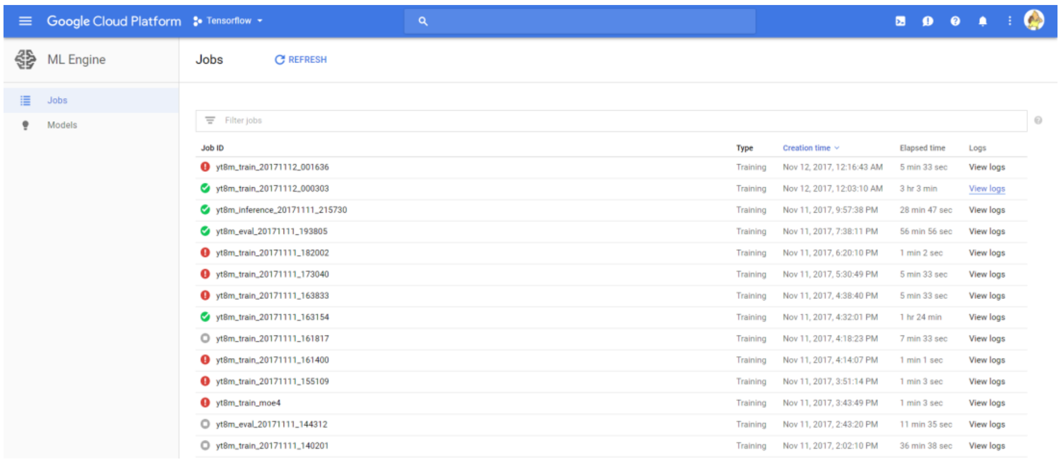
At the early stage of our project, considering that it is quite hard to connect to Google Cloud Platform in China, even using a VPN do not make the connection to be stable. So we decided to train Youtube-8M dataset locally on a Linux laptop with CUDA's GPU accelerations. It all worked well when we utilized Logistic Regression model to do the training and evaluation. The total process of training LR models only cost us half of a day. Fig.12 shows our training result on a Linux laptop with CUDA8.0 and CUDNN6.0.



```
yang@Yang: ~/Workspace/YT8M/code
INFO:tensorflow:training step 23881 | Loss: 7.38 Examples/sec: 8231.83
INFO:tensorflow:training step 23882 | Loss: 7.03 Examples/sec: 9007.98
INFO:tensorflow:training step 23883 | Loss: 7.19 Examples/sec: 9157.17
INFO:tensorflow:training step 23884 | Loss: 7.19 Examples/sec: 9228.29
INFO:tensorflow:training step 23885 | Loss: 7.08 Examples/sec: 8751.09
INFO:tensorflow:training step 23886 | Loss: 7.17 Examples/sec: 8575.56
INFO:tensorflow:training step 23887 | Loss: 7.33 Examples/sec: 8882.10
INFO:tensorflow:training step 23888 | Loss: 6.97 Examples/sec: 9174.73
INFO:tensorflow:training step 23889 | Loss: 7.17 Examples/sec: 9025.85
INFO:tensorflow:training step 23890 | Loss: 7.36 Examples/sec: 9220.15 | Hit@1: 0.79 PERR: 0.65 GAP: 0.70
INFO:tensorflow:training step 23891 | Loss: 7.18 Examples/sec: 9131.60
INFO:tensorflow:training step 23892 | Loss: 6.85 Examples/sec: 8987.02
INFO:tensorflow:training step 23893 | Loss: 6.88 Examples/sec: 9066.20
INFO:tensorflow:training step 23894 | Loss: 6.80 Examples/sec: 8750.11
INFO:tensorflow:training step 23895 | Loss: 7.07 Examples/sec: 8015.48
INFO:tensorflow:training step 23896 | Loss: 7.02 Examples/sec: 8779.45
INFO:tensorflow:training step 23897 | Loss: 6.93 Examples/sec: 9016.37
INFO:tensorflow:training step 23898 | Loss: 7.23 Examples/sec: 9011.78
INFO:tensorflow:training step 23899 | Loss: 6.94 Examples/sec: 9187.06
INFO:tensorflow:training step 23900 | Loss: 7.08 Examples/sec: 8647.27 | Hit@1: 0.77 PERR: 0.63 GAP: 0.71
INFO:tensorflow:training step 23901 | Loss: 7.03 Examples/sec: 8275.95
INFO:tensorflow:training step 23902 | Loss: 6.69 Examples/sec: 8144.96
INFO:tensorflow:training step 23903 | Loss: 7.02 Examples/sec: 8452.13
INFO:tensorflow:training step 23904 | Loss: 6.82 Examples/sec: 8365.05
INFO:tensorflow:training step 23905 | Loss: 7.00 Examples/sec: 8050.13
INFO:tensorflow:training step 23906 | Loss: 7.18 Examples/sec: 8042.09
INFO:tensorflow:training step 23907 | Loss: 6.91 Examples/sec: 8464.49
INFO:tensorflow:training step 23908 | Loss: 7.23 Examples/sec: 8811.49
INFO:tensorflow:training step 23909 | Loss: 6.79 Examples/sec: 8498.49
INFO:tensorflow:training step 23910 | Loss: 7.13 Examples/sec: 8186.38 | Hit@1: 0.80 PERR: 0.66 GAP: 0.72
INFO:tensorflow:training step 23911 | Loss: 6.63 Examples/sec: 8394.61
INFO:tensorflow:training step 23912 | Loss: 7.08 Examples/sec: 8571.33
INFO:tensorflow:training step 23913 | Loss: 7.04 Examples/sec: 8520.85
INFO:tensorflow:training step 23914 | Loss: 6.86 Examples/sec: 7861.26
INFO:tensorflow:training step 23915 | Loss: 7.22 Examples/sec: 7927.16
INFO:tensorflow:training step 23916 | Loss: 7.08 Examples/sec: 8410.60
INFO:tensorflow:training step 23917 | Loss: 7.27 Examples/sec: 8419.46
INFO:tensorflow:training step 23918 | Loss: 6.97 Examples/sec: 8366.56
INFO:tensorflow:training step 23919 | Loss: 7.44 Examples/sec: 8433.33
INFO:tensorflow:training step 23920 | Loss: 7.07 Examples/sec: 8466.73 | Hit@1: 0.79 PERR: 0.66 GAP: 0.72
INFO:tensorflow:training step 23921 | Loss: 7.20 Examples/sec: 8065.35
2017-11-02 19:46:32.377736: W tensorflow/core/framework/op_kernel.cc:1192] Out of range: FIFOQueue '_0_train_
input/input_producer' is closed and has insufficient elements (requested 1, current size 0)
	 [[Node: train_input/ReaderReadUpToV2_3 = ReaderReadUpToV2[_device="/job:localhost/replica:0/task:0/c
pu:0"](train_input/TFRecordReaderV2_3, train_input/input_producer, train_input/ReaderReadUpToV2_3/num_records
)]]
INFO:tensorflow:training step 23922 | Loss: 7.37 Examples/sec: 7709.45
INFO:tensorflow:training step 23923 | Loss: 7.29 Examples/sec: 8885.41
INFO:tensorflow:training step 23924 | Loss: 7.10 Examples/sec: 9343.83
INFO:tensorflow:training step 23925 | Loss: 6.91 Examples/sec: 9347.94
INFO:tensorflow:training step 23926 | Loss: 6.83 Examples/sec: 9732.00
INFO:tensorflow:training step 23927 | Loss: 7.34 Examples/sec: 9606.54
INFO:tensorflow:training step 23928 | Loss: 7.38 Examples/sec: 6994.29
INFO:tensorflow:/job:master/task:0: Done training -- epoch limit reached.
INFO:tensorflow:/job:master/task:0: Exited training loop.
(tensorflow) yang@Yang:~/Workspace/YT8M/code$
```

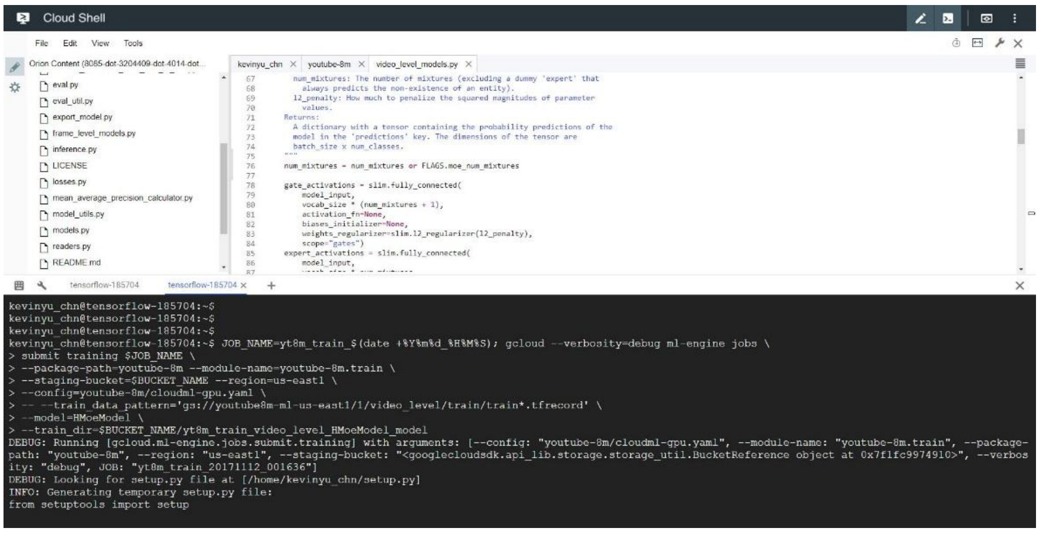
Fig 12. Training LR model locally

However, the problem came when we switched our training model from Logistic Regression to Mixture of Experts. With the increasing number of experts we put in the MoE model, the training time jumps from half of a day to two days. It is so time-consuming even to train a MoE of 6 experts. So after our group discussion, we quickly turned to the proposal of using Google Cloud for training. In order to fix the connection problem, we used the Google cloud computing engine to create a VM environment and build a personal VPS server. With the help of VPS server, we were able to migrate our codes and train the models more stable. And thanks to Google Machine Learning Engine, its powerful NVIDIA TESLA K40(12G) enables us to train seven times faster than our laptop graphic cards which frees us from waiting for a long time of training models. Fig.13&14 shows the interface of Google Machine Learning Engine.



Job ID	Type	Creation time	Elapsed time	Logs
yt8m_train_20171112_001636	Training	Nov 12, 2017, 12:16:43 AM	5 min 33 sec	View logs
yt8m_train_20171112_000303	Training	Nov 12, 2017, 12:03:10 AM	3 hr 3 min	View logs
yt8m_inference_20171111_215730	Training	Nov 11, 2017, 9:57:38 PM	28 min 47 sec	View logs
yt8m_eval_20171111_193805	Training	Nov 11, 2017, 7:38:11 PM	56 min 56 sec	View logs
yt8m_train_20171111_182002	Training	Nov 11, 2017, 6:20:10 PM	1 min 2 sec	View logs
yt8m_train_20171111_173040	Training	Nov 11, 2017, 5:30:49 PM	5 min 33 sec	View logs
yt8m_train_20171111_163833	Training	Nov 11, 2017, 4:38:40 PM	5 min 33 sec	View logs
yt8m_train_20171111_163154	Training	Nov 11, 2017, 4:32:01 PM	1 hr 24 min	View logs
yt8m_train_20171111_161817	Training	Nov 11, 2017, 4:18:23 PM	7 min 33 sec	View logs
yt8m_train_20171111_161400	Training	Nov 11, 2017, 4:14:07 PM	1 min 1 sec	View logs
yt8m_train_20171111_155109	Training	Nov 11, 2017, 3:51:14 PM	1 min 3 sec	View logs
yt8m_train_moe4	Training	Nov 11, 2017, 3:43:49 PM	1 min 3 sec	View logs
yt8m_eval_20171111_144312	Training	Nov 11, 2017, 2:43:20 PM	11 min 35 sec	View logs
yt8m_train_20171111_140201	Training	Nov 11, 2017, 2:02:10 PM	36 min 38 sec	View logs

Fig.13 Training jobs in Google Machine Learning Engine



```
      num_mixtures: The number of mixtures (excluding a dummy 'expert' that
        always predicts the non-existence of an entity).
      l2_penalty: How much to penalize the squared magnitudes of parameter
        values.
    Returns:
      A dictionary with a tensor containing the probability predictions of the
      model in the 'predictions' key. The dimensions of the tensor are
      batch_size x num_classes.
    """

    num_mixtures = num_mixtures or FLAGS.moe_num_mixtures

    gate_activations = slim.fully_connected(
        model_input,
        vocab_size * (num_mixtures + 1),
        activation_fn=None,
        biases_initializer=None,
        weights_regularizer=slim.l2_regularizer(l2_penalty),
        scope="gates")
    expert_activations = slim.fully_connected(
        model_input,
```

```
kevinyu_chn@tensorflow-185704:~$
kevinyu_chn@tensorflow-185704:~$
kevinyu_chn@tensorflow-185704:~$
kevinyu_chn@tensorflow-185704:~$ JOB_NAME=yt8m_train_$(date +%Y%m%d_%H%M%S); gcloud --verbosity=debug ml-engine jobs \
> submit training $JOB_NAME \
> --package-path=youtube-8m --module-name=youtube-8m.train \
> --staging-bucket=$BUCKET_NAME --region=us-east1 \
> --config=youtube-8m/cloudml-gpu.yaml \
> -- --train_data_pattern='gs://youtube8m-ml-us-east1/1/video_level/train/train*.tfrecord' \
> --model=HMoeModel \
> --train_dir=$BUCKET_NAME/yt8m_train_video_level_HMoeModel_model
DEBUG: Running [gcloud.ml-engine.jobs.submit.training] with arguments: [--config: "youtube-8m/cloudml-gpu.yaml", --module-name: "youtube-8m.train", --package-
path: "youtube-8m", --region: "us-east1", --staging-bucket: "<googlecloudsdk.api_lib.storage.storage_util.BucketReference object at 0x7f1fc9974910>", --verbos
ity: "debug", JOB: "yt8m_train_20171112_001636"]
DEBUG: Looking for setup.py file at [/home/kevinyu_chn/setup.py]
INFO: Generating temporary setup.py file:
from setuptools import setup
```

Fig.14 Using cloud shell to training models on the cloud

5. Summery and Future Work

In conclusion, our project was an approach to a large-scale video classification. We use Google Cloud Platform and Tensorflow to store and process the data. Initial research indicated that Hierarchical Mixture of Experts may be the best approach, but we eventually determined that MOE was the best model for approaching this problem. Our results show that with the Hierarchical MOE model, we can approach about an almost 80% accuracy on labeling for the given videos, while Logistic Regression model and other variants of the MOE model give lower accuracy. These results show that even on a very large scale, video understanding using neural networks can reach very high levels of accuracy.

As for suggestions for future improvements on our design, we think that in order to further test the efficiency of Mixture of Experts, experts can be increase from using only 8 models to 30 models or even a hundred. Actually, we tried once using 30 experts on MoE models but it failed after two hours' training. After analyzing the logs, we surprisingly found that the 12G Graphic Cards from Google Cloud Platform ran out of memory. Therefore, for our further work, we are planning to change the configuration of GPU training setting in Google Cloud Buckets which enables us to use three TESLA K40 Graphic Card at one time. We believe this improvement on the graphic cards will solve the problem of running out of memory

6. Reference

[1] *Google Cloud & YouTube-8M Video Understanding Challenge. Accessed April 21, 2017. https://www.kaggle.com/c/youtube8m/*

[2] *Tensorflow: Image recognition. https://github.com/google/youtube-8m/.*

[3] *Youtube-8m tensorflow starter code. https://www.tensorflow.org/tutorials/image_recognition.*

[4] *Abu-El-Haija, Sami, 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).*

[5] *H. He and E. A. Garcia. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9):1263–1284, 2009*

[6] *Jordan, Michael I., and Robert A. Jacobs."Hierarchical mixtures of experts and the EM algorithm." Neural computation 6, no. 2 (1994):181-214.*

[7] *Noam Shazeer1, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis1, Quoc Le, Geoffrey Hinton and Jeff Dean. "Outrageously Large Neural Networks:The Sparsely-Gated Mixture-Of-Experts Layer"*

[8] *Wenxin Jiang, Martin A. Tanner. "Hierarchical Mixtures-Of-Experts For Exponential Family Regression Models: Approximation And Maximum Likelihood Estimation" The Annals of Statistics, 1999, Vol. 27, No. 3, 987–1011*