YouTube-8M
Video Understanding Challenge

Can you produce the best video tag predictions?

Mikhail Trofimov

Dataset

[YouTube-8M (v.2)]

7 Million	450,000	3.2 Billion	4716	3.4
Video URLs	Hours of Video	Audio/Visual Features	Classes	Avg. Labels / Video

The videos are sampled uniformly to preserve the diverse distribution of popular content on YouTube, subject to a few constraints selected to ensure dataset quality and stability:

- Each video must be public and have at least 1000 views
- Each video must be between 120 and 500 seconds long
- Each video must be associated with at least one entity from our target vocabulary
- Adult & sensitive content is removed (as determined by automated classifiers)

YouTube | 8M

Dataset Explore Download Workshop About

Vertical

Science

Filter

Entities

Robotics (595) Comet (579) Skull (575)

Shrub (559) Eclipse (533) Kidney (516)

Chemical reaction (512) Tongue (499)

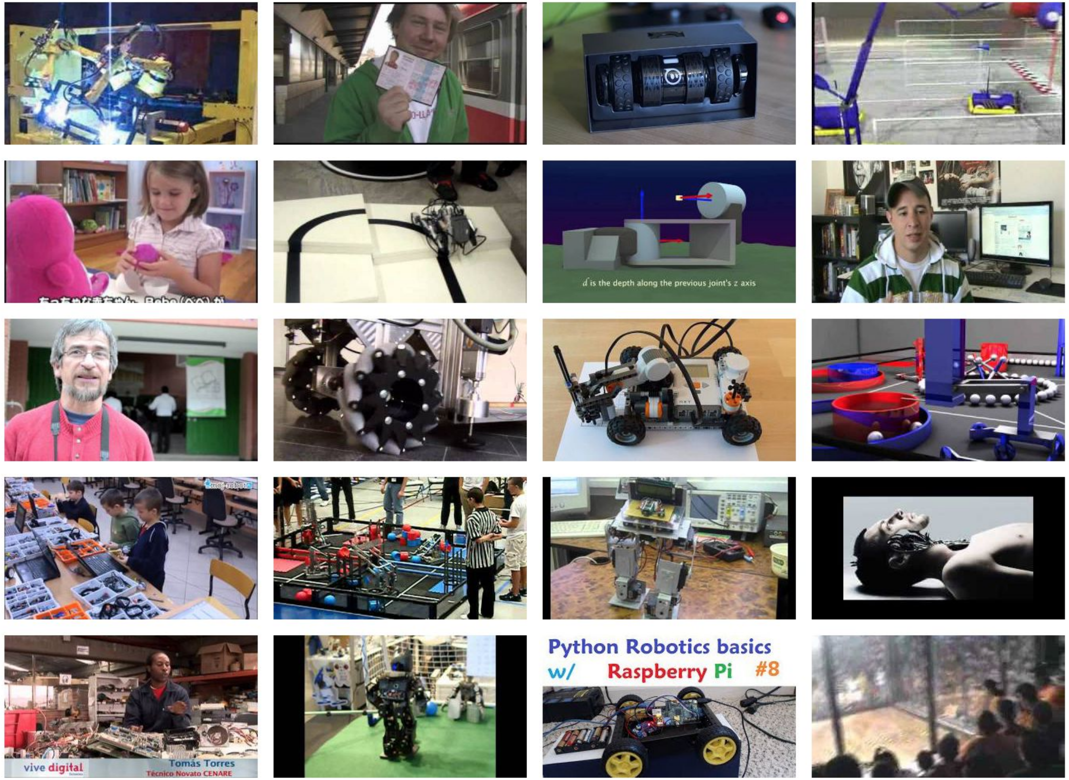
Tears (499) Emerald (465) Snail (463)

Bamboo (451) Human tooth (446) Melting (441)

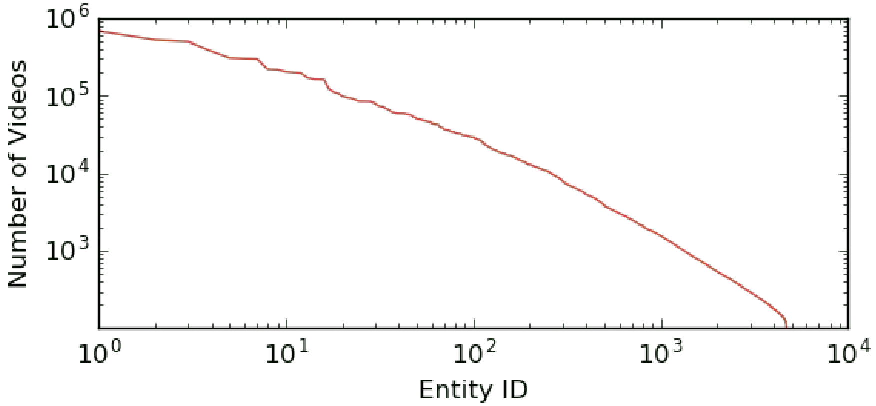
Caridean Shrimp (437) Square (428) Fly (428)

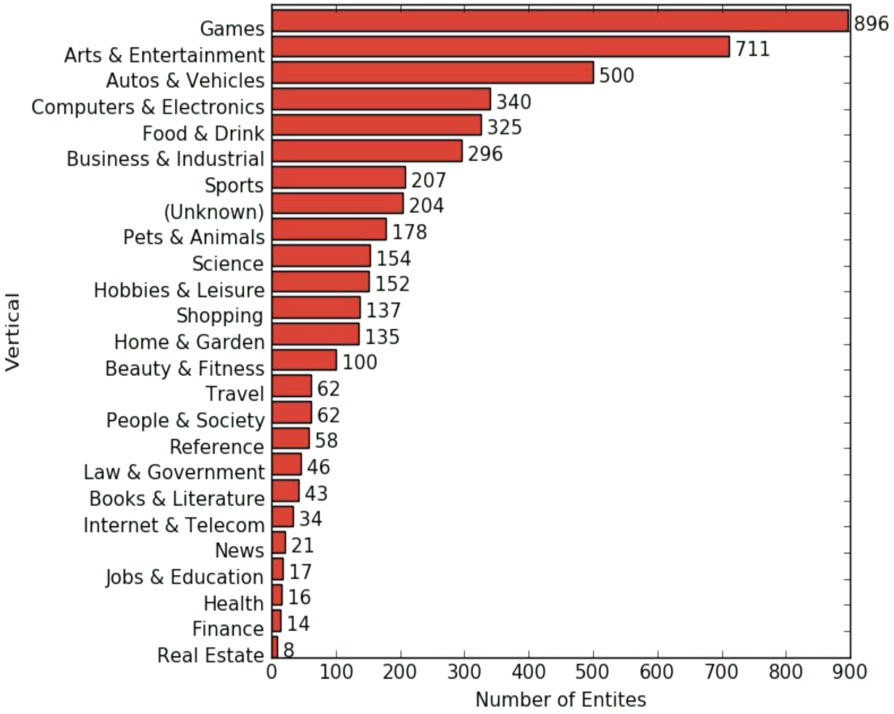
Tail (404) Organism (403) Jupiter (387)

Scorpion (383) Glacier (378) Fog (370)



Dataset





See details in [tech report]

Dataset/Features

Full dataset (frame-level):

- 1fps -> internal CNN -> PCA -> quantization (8-bit) -> 1024 features
- up to 6 minutes (300 frames)
- 128 audio features
- ~2 TB of data (uint8)
- train / validation / test -- 70 / 20 / 10, iid!
- videoIDs are available for train/validation only

There is a small version of this dataset (video-level):

- just means -> (1024 + 128) features for single video
- ~30 GB of data (float32)

Metric

If a submission has N predictions (label/confidence pairs) sorted by its confidence score, then the Global Average Precision is computed as:

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

where N is the number of final predictions (if there are 20 predictions for each video, then N = 20 * #Videos), p(i) is the precision, and r(i) is the recall.

Submission File

For each VideoId in the test set, you must predict a list of Labels and their corresponding confidence scores. The file should contain a header and have the following format:

```
VideoId,LabelConfidencePairs
100000001,1 0.5 2 0.3 3 0.1 4 0.05 5 0.05
etc.
```

Starter Kit

https://github.com/google/youtube-8m

- Code for calculate metric
- TensorFlow pipeline
- Frame\video-level models
- TensorBoard included
- Multi-GPU support

Starter Kit

https://github.com/google/youtube-8m

- Code for calculate metric
- TensorFlow pipeline
- Frame\video-level models
- TensorBoard included
- Multi-GPU support
- Quite slow
- A lot of TF-related noodles
- Hard to do feature engineering

My choice

- Video-level features
- Convert dataset: tfrecord -> npz
  - 30 GB -> 16 GB (float16)
  - lower IO
- Use pytorch's Dataset class
  - simple custom preprocessing
  - parallel batching
- TensorFlow as DL engine
  - sparse tensor support (for labels)
  - easy to build very custom models
  - codebase from starter kit
  - TensorBoard

Models

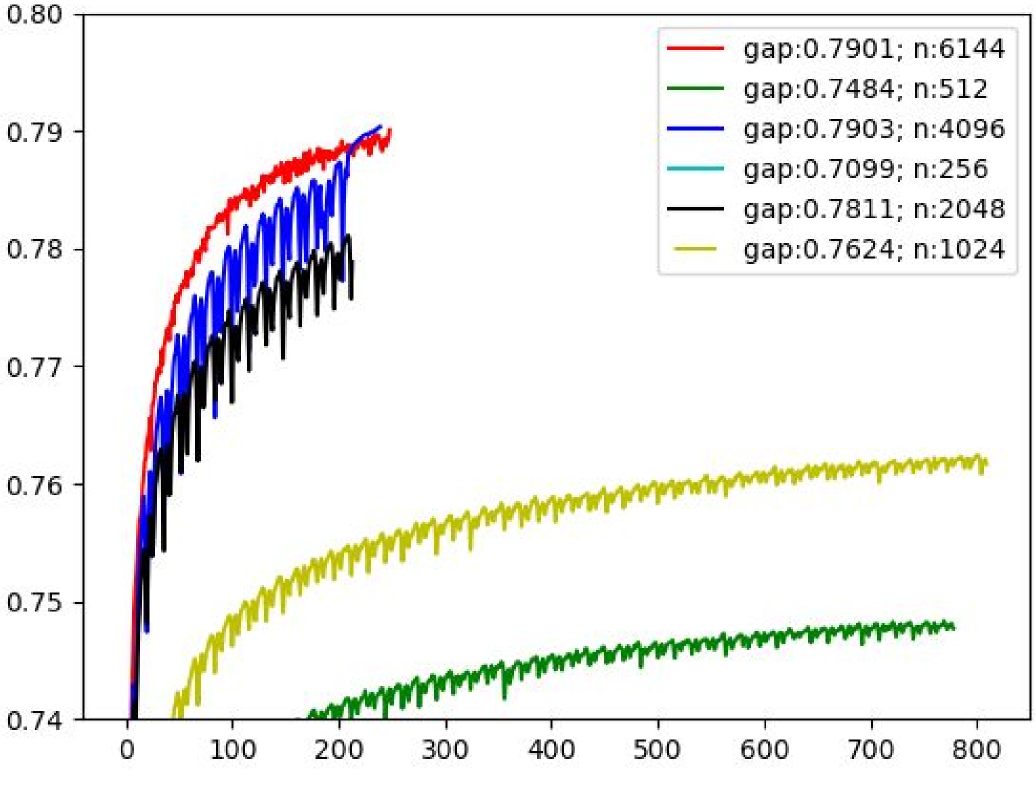
Baseline:

- ~5k logregs, trained jointly on mean_rgb + mean_audio
- GAP: 0.747

Simple tricks #1:

- normalize mean_rgb and mean_audio separately
- drop Adam LR
- GAP: 0.762

LR drop



Картинка @n01z3

Models

Simple tricks #2:

- piecewise linear model
- X_piecewise = tf.concat(tf.nn.relu(X), tf.nn.relu(-X))
- GAP: 0.775

Simple tricks #3:

- add traditional deep NN, (1024+128)->1024->1024->4716
- output = tf.sigmoid(deep + wide)
- GAP: 0.8034 (my best single model)

Models

Simple tricks #2:

- piecewise linear model
- X_piecewise = tf.concat(tf.nn.relu(X), tf.nn.relu(-X))
- GAP: 0.775

Simple tricks #3:

- add traditional deep NN, (1024+128)->1024->1024->4716
- output = tf.sigmoid(deep + wide)
- GAP: 0.8034 (my best single model) [but it can be better with bigger model]

Ensembling

- Just average of 4 NN models: GAP 0.816 (48th private)
- Other type of averaging were worse
- Discrete mixing: not implemented, upper bound 0.819

Frame-level features

- Download part of them
- Extract mean, std, top5, bottom5, num_frames
- Almost no gain from them

Re-ranking

- in >80% of samples top 20 predictions contain all true labels
- We can cut top2 with current model and re-rank
- Upper bound on GAP: 0.93
- In my case -- NN converges to same local minima and starts to overfit

link to paper

Scalable Learning of Non-Decomposable Objectives

Elad Eban
elade@google.com

Mariano Schain
marianos@google.com

Alan Mackey
mackeya@google.com

Ariel Gordon
gariel@google.com

Rif A. Saurous
rif@google.com

Gal Elidan
elidan@google.com

Google, Inc.

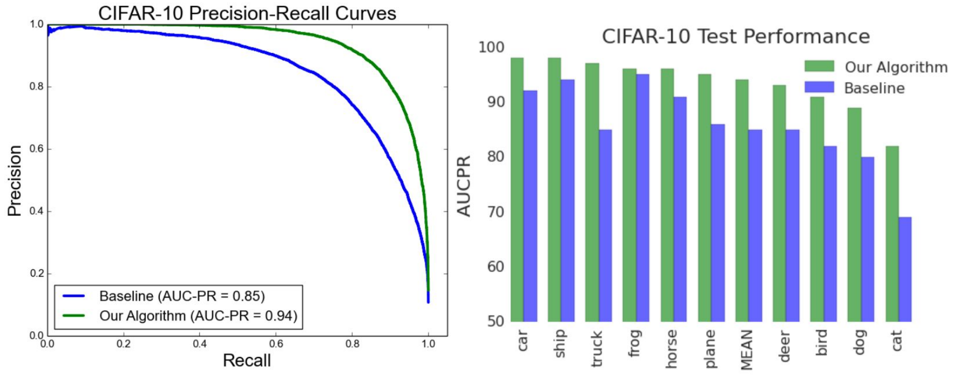


Figure 2: Comparison of a baseline model trained to optimize accuracy and a model trained to optimize **AUCPR** using our method on the CIFAR10 dataset. (left) Shows the aggregate precision-recall curve for all 10 classes. (right) Compares the **AUCPR** for each of the 10 classes.

Our contribution is thus threefold. First, we provide a unified approach that, using the same building blocks, allows for the optimization of a wide range of rank-based objectives that include **AUCROC**, **AUCPR**, **P@R**, **R@P**, and $\mathbf{F}_\beta$. Third, our unified framework also easily allows for novel objectives such as the area under the curve for a region of interest, i.e. when the precision or recall are in some desired range. Finally,

$$\min_{f, b_1, \ldots b_k} \max_{\lambda_1 \ldots \lambda_k} \sum_{t=1}^{k} \Delta_t \Big((1+\lambda_t) \mathscr{L}^+(f, b_t) + \lambda_t \frac{\alpha_t}{1-\alpha_t} \mathscr{L}^-(f, b_t) - \lambda_t |Y^+| \Big). \quad (12)$$

As before, we can solve this saddle point problem by SGD [17].

Secret sauce of top teams

- Data augmentation:
  - calc video-level features on part of frames,
  - use it as independent sample
  - +0.01 GAP
- Ensembles
  - a lot of different models
    - aggregating checkpoints of the same model
  - important to include frame-level models
    - LSTM: <0.8 LB GAP, bit +0.01 in ensemble
- Codes and details not released yet, but will be at CVPR Workshop