Google Cloud

ASL Tech Talk:

YouTube8M

Video data

- With cheaper data storage and the proliferation of videos, there is an ever growing need to be able to analyze video type data.
- Use cases could be: recommendations, security, conservation, safety, autonomous driving, etc.

Video data

- Video data is pretty complex because it is a series of images and sound tied together through time.
- Therefore, to train a powerful model you will need a lot of video data.
- If only there was a place with tons of video data...

YouTube

YouTube

- YouTube was created in 2005 and acquired by Google in 2006.
- Over 500 hours of video are uploaded every minute.
- Over 1.9 billion users per month **signed in** back in 2018
- Over 5 billion videos watched per day back in 2018.

What is YouTube8M?

- Large-scale video dataset annotated with multiple machine-generated labels per video from millions of videos with a vocabulary of almost 4000.
- Originally was released Sep 2016 with 8.2M videos, 4800 classes, 1.8 labels/video, 1.9B visual-only features

Recent Updates

- Updated Feb 2017 to 7.0M videos, 4716 classes, 3.4 labels/video, 3.2B audio-visual features
- Currently there are 6.1M videos, 3862 classes, 3.0 labels/video, 2.6B audio-visual features
- June 27th 2019, YouTube8M-Segments dataset released
    - 237k human-verified segment labels
    - 1000 classes
    - Average 5.0 segments per video

YouTube8M Properties

6.1 Million Video IDs	350,000 Hours of Video	2.6 Billion Audio/Visual Features	3862 Classes	3.0 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)

YouTube8M Properties

6.1 Million	350,000	2.6 Billion	3862	3.0
Video IDs	Hours of Video	Audio/Visual Features	Classes	Avg. Labels / Video

The dataset represents over 350,000 hours of video, and would normally require hundreds of terabytes of storage. It would also take 50 CPU-years worth of computation to process this dataset (with real time video processing per CPU). To eliminate storage and computational bottlenecks, we are providing pre-computed and compressed features, which make it possible to train a starter model on this dataset in less than a day, on a single GPU!

YouTube8M Properties

6.1 Million	350,000	2.6 Billion	3862	3.0
Video IDs	Hours of Video	Audio/Visual Features	Classes	Avg. Labels / Video

Videos are pre-processed to extract state-of-the-art 1.3 Billion visual and 1.3 Billion audio features. We extract features at the video-level as well as features at frame- and segment-level granularity (at 1-second resolution). 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 described in Hershey et. al. on a preliminary version of YouTube-8M. Both the visual and audio features were PCA-ed and quantized to fit on a single hard disk. The combined set of all features are less than 2TB in size.

YouTube8M Properties

6.1 Million	350,000	2.6 Billion	3862	3.0
Video IDs	Hours of Video	Audio/Visual Features	Classes	Avg. Labels / Video

The target annotation vocabulary consists of 3862 Knowledge Graph entities, including both coarse and fine-grained entities, which have been semi-automatically curated and manually verified by 3 raters to be visually recognizable. Each entity has at least 200 corresponding video examples, with an average of 3552 training videos per entity. The three most popular entities are *Game*, *Video Game*, and *Vehicle*, respectively, with 788288, 539945, and 415890 training examples, respectively. The least frequent are Cylinder and Mortar, with 123 and 127 training videos, respectively. The entities are grouped into 24 high-level verticals, with the most frequent vertical being Arts & Entertainment (3.3M training videos) and the least frequent being Finance (6K training videos).

YouTube8M Properties

6.1 Million	350,000	2.6 Billion	3862	3.0
Video IDs	Hours of Video	Audio/Visual Features	Classes	Avg. Labels / Video

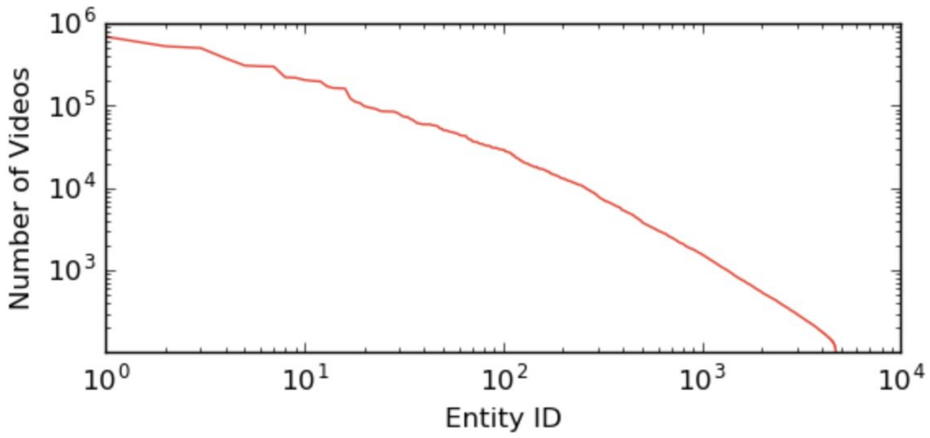
The ground-truth video labels are the main themes of each video, as determined by a YouTube video annotation system using content, metadata, contextual, and user signals. The number of ground truth labels per video varies from 1 to 23, with an average of 3.01 per video. The 60th and 80th percentiles of labels / video are 3.0 and 4.0, respectively.

YouTube8M Vocabulary

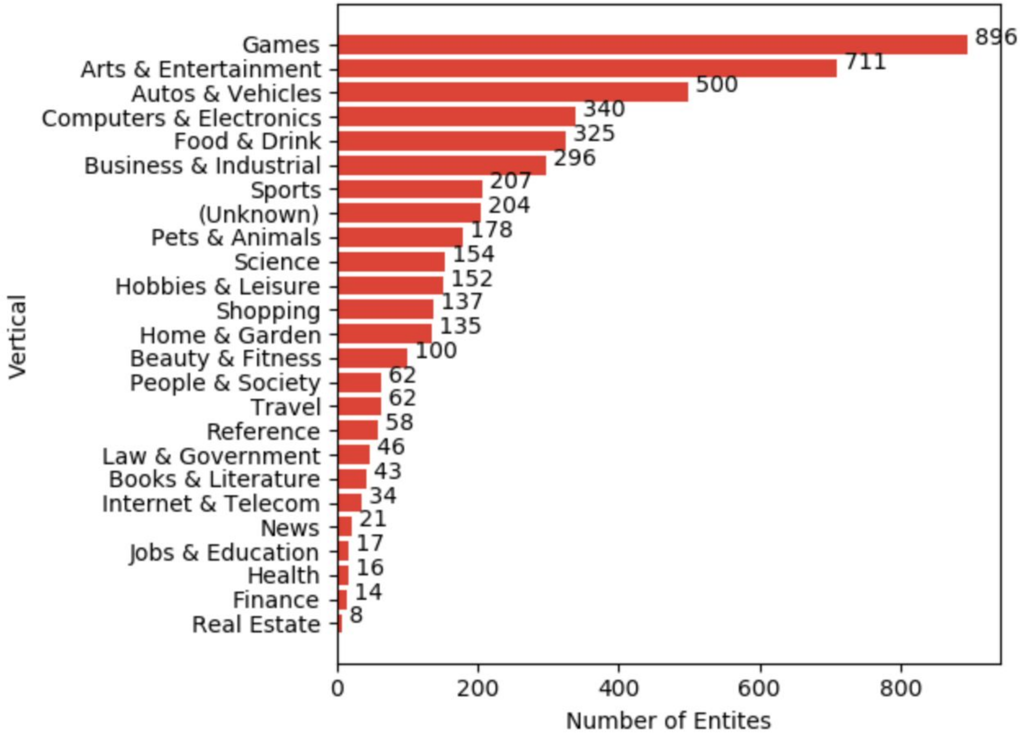
	Index	TrainVideoCount	KnowledgeGraphId	Name	WikiUrl	Vertical1	Vertical2	Vertical3	WikiDescription
1	0	788288	/m/03bt1gh	Game	n.wikipedia.org/wiki/Game	Games			f the oldest known games.
2	1	539945	/m/01mw1	Video game	pedia.org/wiki/Video_game	Games			er, varies across platforms.
3	2	415890	/m/07yv9	Vehicle	.wikipedia.org/wiki/Vehicle	Autos & Vehicles			pes, terms and definitions.
4	3	378135	/m/01jddz	Concert	.wikipedia.org/wiki/Concert	Arts & Entertainment			nity to hear musicians play.
5	4	286532	/m/09jwl	Musician	wikipedia.org/wiki/Musician	Arts & Entertainment			the orchestration of music.
6	5	236948	/m/0215n	Cartoon	.wikipedia.org/wiki/Cartoon	Arts & Entertainment			published on the Internet.
7	6	203343	/m/01350r	Performance art	a.org/wiki/Performance_art	Arts & Entertainment			ar time constitute the work.
8	7	200813	/m/0k4j	Car	//en.wikipedia.org/wiki/Car	Autos & Vehicles			ogressively more complex.
9	8	181579	/m/026bk	Dance	n.wikipedia.org/wiki/Dance	Arts & Entertainment			ny other forms of athletics.
10	9	156226	/m/0342h	Guitar	n.wikipedia.org/wiki/Guitar	Arts & Entertainment			imes called a "jazz guitar".

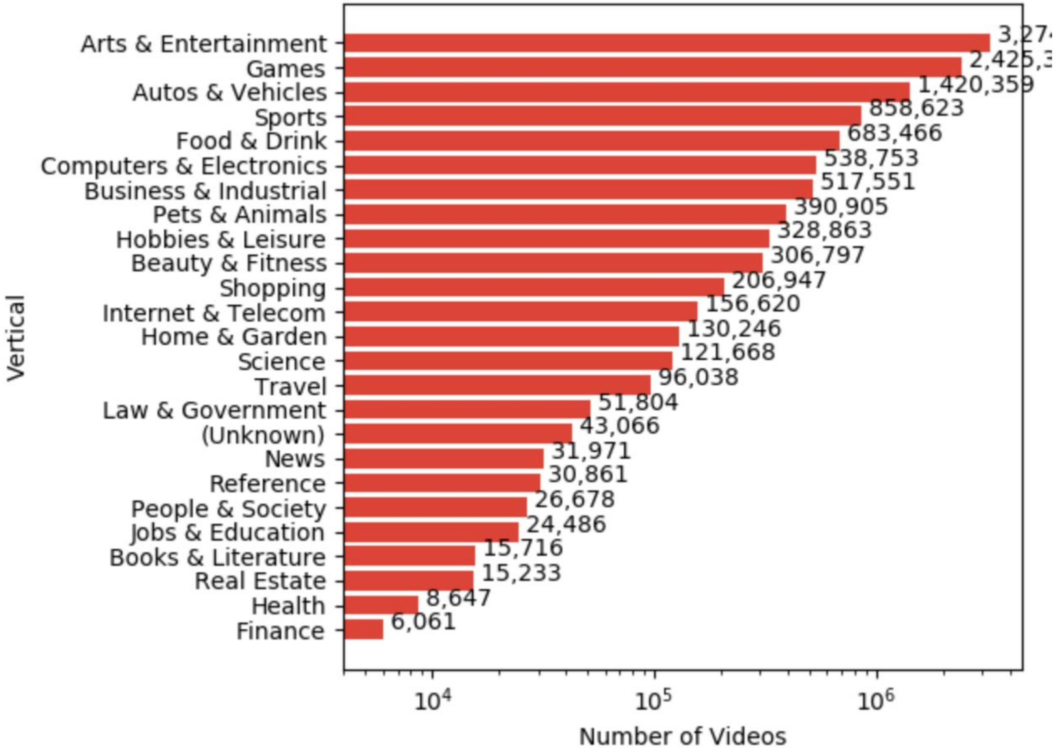
Knowledge Graph entities organized into 24 top-level verticals. Each entity represents a semantic topic that is visually recognizable in video, and the video labels reflect the main topics of each video.

Entity frequencies in log-log scale, Zipf-like distribution.



Histograms for top-level verticals





YouTube 8M

Dataset Explore Download Workshop About

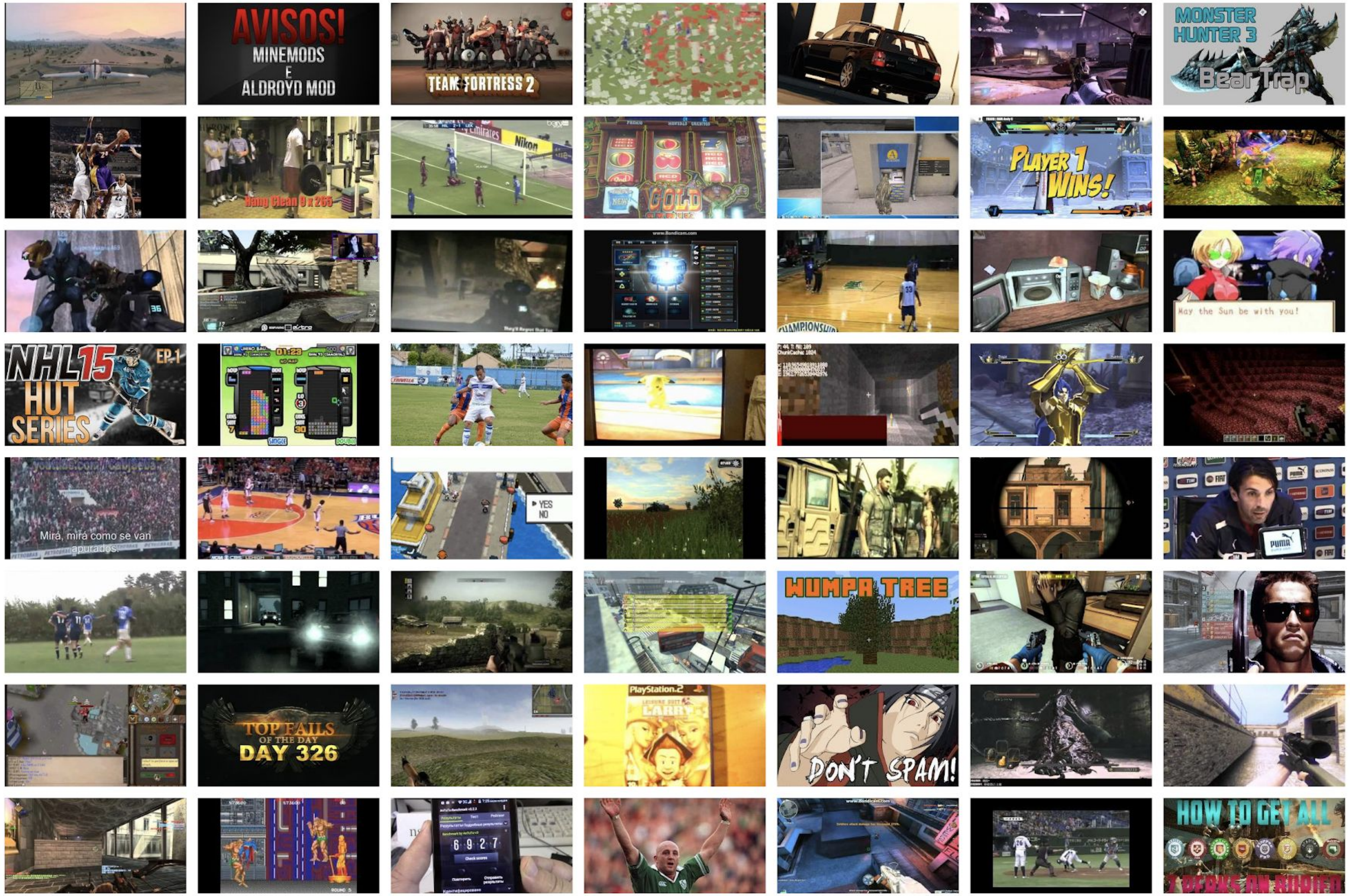
Vertical

All

Filter

Entities

Games (788288) Video game (539945)
Vehicle (415890) Concert (378135)
Musician (286532) Cartoon (236948)
Performance art (203343) Car (200813)
Dance (181579) Guitar (156226)
String instrument (144667) Food (135357)
Football (130835) Musical ensemble (125668)
Music video (116098) Animal (107788)
Animation (98140) Motorsport (93443)
Pet (90779) Racing (84258) Recipe (75819)
Mobile phone (72911) Cooking (71218)
Smartphone (64884) Gadget (64452)
Trailer (59695) Toy (58720) Minecraft (57801)
Drums (55597) Cuisine (55411) Piano (55201)
Motorcycle (54950) Dish (54730)



Collecting Videos

- Got videos related to the 10K visual entities and have at least 1K views, using the YouTube video annotation system. Kept only videos of length between 120 and 500 secs.
- Randomly sampled 10M videos among them.
- Obtained all entities for the sampled 10M videos using the YouTube video annotation system.
- Filtered out entities with less than 200 videos, and videos with no remaining entities.
- Split videos into 3 partitions, Train : Validate : Test, with ratios 70% : 20% : 10%.

Features

- The raw dataset was 100s of TB and 500K hours of video.
- This is not feasible for the average person for storage or time to process.
- Therefore, a featurizer was created to compress the raw data into a much smaller and usable form while still maintaining similar evaluation metrics after training.

Featurizer

- Decode video at 1 FPS up to first 6 minutes
- Feed frames into pretrained Inception image network
- Extract 2048 dimensional vector from final ReLU layer per second of video
- Apply PCA and whitening to reduce to 1024 dimensions
- Apply quantization
- Results in compression by a factor of 8
- Evaluation metrics lose less than 1%

Frame-level Features

- The labels are at a video level, one-hot encoded per each entity.
- There is no information about where labels occur within a video or their prominence.
- For each video, we can sample n random frames between 1 <= n <= 120, since 120 seconds is the minimum video length.

Decoding Frame-level Features

```
# This function will decode frame examples from the frame level TF Records
def frame_decode_example(serialized_examples):
  # Create context and sequence feature map
  context_features = {
    "id": tf.FixedLenFeature(shape=[], dtype=tf.string),
    "labels": tf.VarLenFeature(dtype=tf.int64)
  }
  sequence_features = {
    "rgb": tf.FixedLenSequenceFeature(shape=[], dtype=tf.string),
    "audio": tf.FixedLenSequenceFeature(shape=[], dtype=tf.string)
  }

  # Parse TF Records into our features
  contexts, features = tf.parse_single_sequence_example(
      serialized=serialized_examples,
      context_features=context_features,
      sequence_features=sequence_features)
```

Video-level Features

- Creating fixed dimensional video-level features from frames has its advantages.
  - Standard classifiers can apply.
  - Compactness.
  - More suitable for domain adaptation.

Video-level Features

$$\varphi(\mathbf{x}^v_{1:F_v}) = \begin{bmatrix} \mu(\mathbf{x}^v_{1:F_v}) \\ \sigma(\mathbf{x}^v_{1:F_v}) \\ \mathrm{Top}_K(\mathbf{x}^v_{1:F_v}) \end{bmatrix}$$

Mean $\mathbb{R}^{1024}$

Standard Deviation $\mathbb{R}^{1024}$

Top K frame features $\mathbb{R}^{K}$

$$\mathrm{Top}_K(\mathbf{x}^v(j)_{1:F_v})$$

pth dimension contains pth highest value from jth frame-level dimension over the entire video.

Followed by centering, PCA, and whitening

Decoding Video-level Features

```
# This function will decode video examples from the video level TF Records
def video_decode_example(serialized_examples):
  # Create feature map
  feature_map = {
      "video_id": tf.FixedLenFeature(shape = [], dtype = tf.string),
      "labels": tf.VarLenFeature(dtype = tf.int64),
      "mean_rgb": tf.FixedLenFeature(shape = [1024], dtype = tf.float32),
      "mean_audio": tf.FixedLenFeature(shape = [128], dtype = tf.float32)
  }

  # Parse TF Records into our features
  features = tf.parse_single_example(serialized = serialized_examples, features = feature_map)
```

cloud.google.com