Video Classification Overview

Videos can be understood as a series of individual images; and therefore, many deep learning practitioners would be quick to treat video classification as performing image classification a total of N times, where N is the total number of frames in a video.

There’s a problem with that approach though.

Video classification is more than just simple image classification - with video we can typically make the assumption that subsequent frames in a video are correlated with respect to their semantic contents.

If we are able to take advantage of the temporal nature of videos, we can improve our actual video classification results.

Neural network architectures such as Long short-term memory (LSTMs) and Recurrent Neural Networks (RNNs) are suited for time series data but in some cases, they may be overkill. They are also resource-hungry and time-consuming when it comes to training over thousands of video files as you can imagine.

How is image classification different from video classification?

When performing image classification, we:

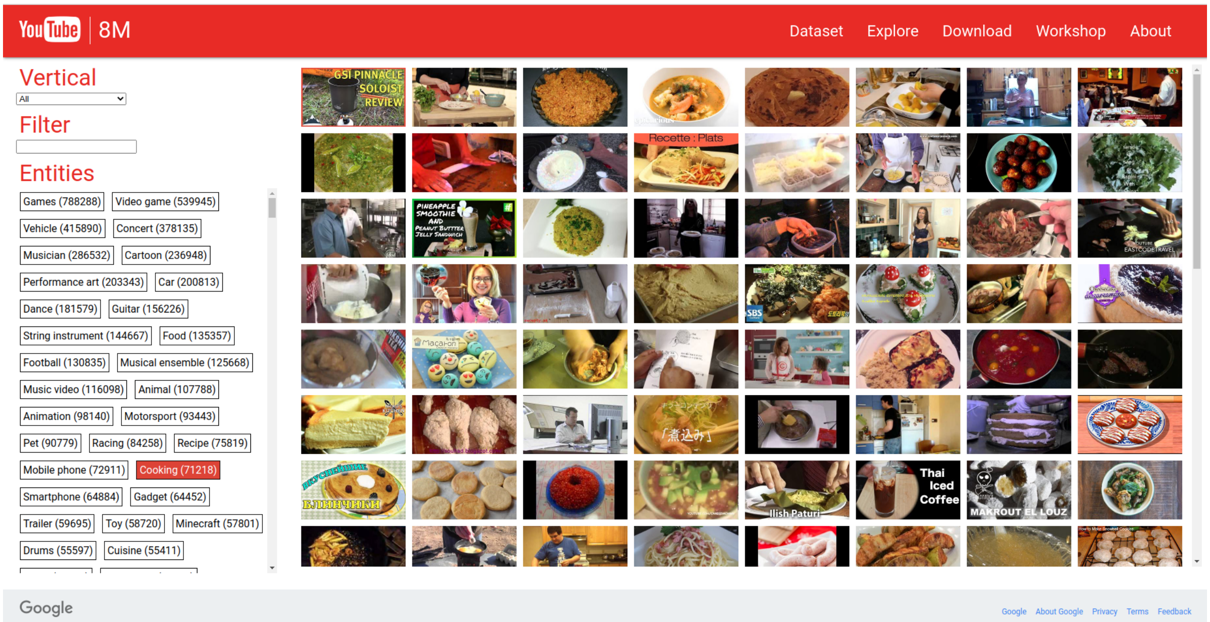
- Input an image to our CNN
- Obtain the predictions from the CNN
- Choose the label with the largest corresponding probability

Since a video is just a series of frames, a naive video classification method would be to:

- Loop over all frames in the video file
- For each frame, pass the frame through the CNN
- Obtain the predictions from the CNN
- Maintain a list of the last K predictions
- Compute the average of the last K predictions and choose the label with the largest corresponding probability
- Label the frame and write the output frame to disk

Youtube 8M Dataset

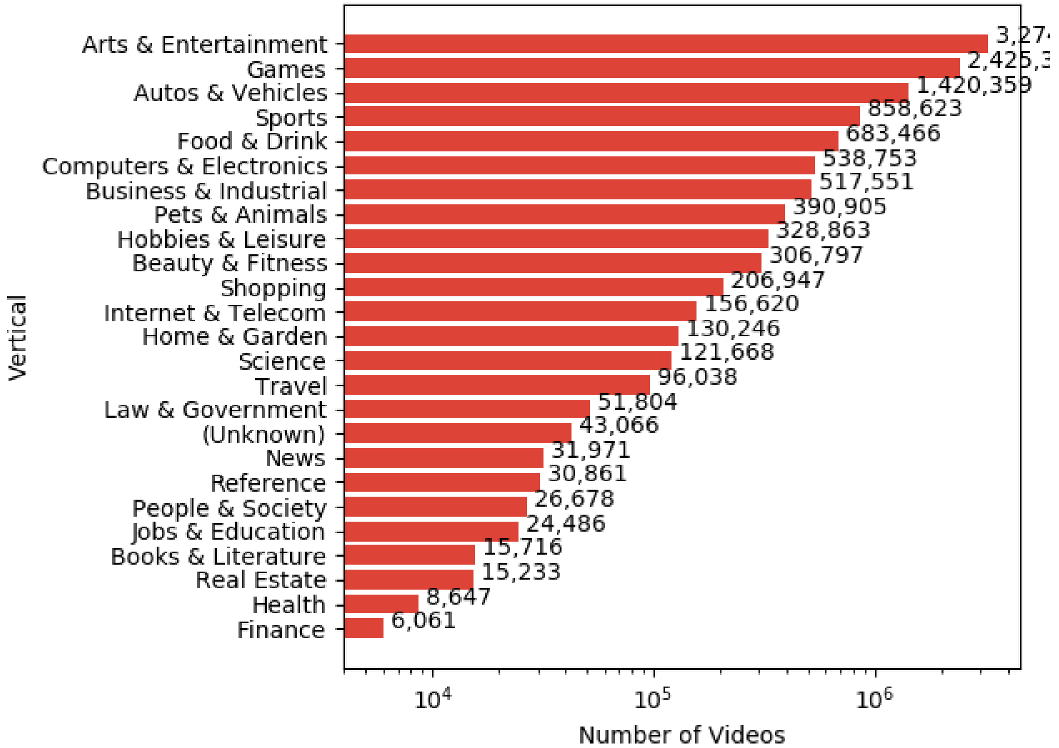
YouTube-8M is a large-scale labeled video dataset that consists of millions of YouTube video IDs, with high-quality machine-generated annotations from a diverse vocabulary of 3,800+ visual entities. It comes with precomputed audio-visual features from billions of frames and audio segments, designed to fit on a single hard disk. This makes it possible to train a strong baseline model on this dataset in less than a day on a single GPU! At the same time, the dataset's scale and diversity can enable deep exploration of complex audio-visual models that can take weeks to train even in a distributed fashion.



Reference:
https://research.google.com/youtube8m/explore.html

Dataset Vocabulary

The (multiple) labels per video are 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.



The YouTube-8M data has gone through a few different iterations. The dataset was built off of videos and labels publicly available on YouTube. The dataset has been adjusted and morphed over the last four years and the original 8 million video dataset has been deprecated.

The current available datasets are:

- Videos = 6.1M videos, 3862 classes, 3.0 labels / video, 2.6B audio-visual features
- Video segments = 230K human-verified segment labels, 1000 classes, average 5 segments/video

The dataset referenced on the YouTube-8M website is the latest and focuses on video segments. So it uses part of the video dataset and narrows the focus to 1000 classes for the segments in those videos.

Download Dataset

YouTube8M offers a dataset for download as TensorFlow Record files. It also provides a downloader script that fetches the dataset in shards and stores them in the current directory. It can be restarted if the connection drops. In which case, it only downloads shards that haven't been downloaded yet.

There are two versions of the features: frame-level and video-level features.

The dataset is made available by Google LLC. under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

For our use case, we've downloaded a **video-level** features dataset.

The details for downloading data can be found here
https://research.google.com/youtube8m/download.html

The video-level dataset that provides video-level features is stored as a tensorflow.Example object grouped into a total of 7,689 TFRecords. The total size is around 31GB. It has the following structure:

- **id**: unique YouTube video id. Train includes unique actual values and test/validation are anonymized
- **labels**: list of labels for that video
- **mean_rgb**: average of video rgb features as float array of length 1024
- **mean_audio**: average of audio features as float array of length 128

Filtering Dataset with Only Cooking Category

We downloaded the video-level dataset. Now let's filter only cookie category videos.

Import required libraries:

```
In [94]: import tensorflow as tf
         import glob
         import pandas as pd
         from IPython.display import YouTubeVideo
```

Read vocabulary.csv file (containing information about video labels)

```
In [85]: vocabulary_data = pd.read_csv('./vocabulary.csv')

In [93]: vocabulary_data.head(25)
Out[93]:
```

	Index	TrainVideoCount	KnowledgeGraphId	Name
0	0	788288	/m/03bt1gh	Game
1	1	539945	/m/01mw1	Video game
2	2	415890	/m/07yv9	Vehicle
3	3	378135	/m/01jddz	Concert
4	4	286532	/m/09jwl	Musician
5	5	236948	/m/0215n	Cartoon
6	6	203343	/m/01350r	Performance art
7	7	200813	/m/0k4j	Car
8	8	181579	/m/026bk	Dance
9	9	156226	/m/0342h	Guitar
10	10	144667	/m/0d8lm	String instrument
11	11	135357	/m/02wbm	Food
12	12	130835	/m/02vx4	Association football
13	13	125668	/m/05229	Musical ensemble
14	14	116098	/m/0mdxd	Music video
15	15	107788	/m/0jbk	Animal
16	16	98140	/m/0hcr	Animation
17	17	93443	/m/0410tth	Motorsport
18	18	90779	/m/068hy	Pet
19	19	84258	/m/0dfbw	Racing
20	20	75819	/m/0p57p	Recipe
21	21	72911	/m/050k8	Mobile phone
22	22	71218	/m/01mtb	Cooking
23	23	64884	/m/0169zh	Smartphone
24	24	64452	/m/02mf1n	Gadget

Understand vocabulary data

```
In [95]: vocabulary_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3862 entries, 0 to 3861
Data columns (total 4 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Index             3862 non-null   int64
 1   TrainVideoCount   3862 non-null   int64
 2   KnowledgeGraphId  3862 non-null   object
 3   Name              3806 non-null   object
dtypes: int64(2), object(2)
memory usage: 120.8+ KB
```

```
In [96]: vocabulary_data.describe()
Out[96]:
```

	Index	TrainVideoCount
count	3862.000000	3862.000000
mean	1930.500000	3032.527188
std	1115.007698	21182.048375
min	0.000000	123.000000
25%	965.250000	234.000000
50%	1930.500000	440.500000
75%	2895.750000	1199.750000
max	3861.000000	788288.000000

Reading one of the .tfrecord files from downloaded dataset and filter only “Cooking” category videos.

```
In [102]: record = "./train3815.tfrecord"

In [103]: vid_ids = []
          labels = []
          rgb = []
          audio = []

In [104]: for example in tf.compat.v1.python_io.tf_record_iterator(record):
              seq_example = tf.train.Example.FromString(example)
              # filter videos containing "Cooking" label
              if 22 in seq_example.features.feature['labels'].int64_list.value:
                  vid_ids.append(seq_example.features.feature['id'].bytes_list.value[0].decode(encoding='UTF-8'))
                  labels.append(seq_example.features.feature['labels'].int64_list.value)
                  rgb.append(seq_example.features.feature['mean_rgb'].float_list.value)
                  audio.append(seq_example.features.feature['mean_audio'].float_list.value)

In [105]: print('Number of videos in this tfrecord: ',len(vid_ids))
          print ('Number of labels in this tfrecord: ', len (labels))

          Number of videos in this tfrecord:  21
          Number of labels in this tfrecord:  21

In [106]: vid_ids
Out[106]: ['ZU9H',
           'Xz9H',
           'yU9H',
           '499H',
           '9u9H',
           'GC9H',
           'ij9H',
           'MJ9H',
           '7b9H',
           'Kp9H',
           'uc9H',
           'x29H',
           'hp9H',
           'ho9H',
           'Vh9H',
           'SI9H',
           'Wr9H',
           'IB9H',
           'ea9H',
           'jI9H',
           '6f9H']
```

So the .tfrecord had around 138 videos, out of which 21 videos were from the “Cooking” label.

Let’s pick any random video id from 21 videos and convert it to equivalent Youtube-ID and display the video to confirm whether it is cooking video or not.

```
In [107]: # SI9H -> tknvPTMGJEg
          YouTubeVideo('tknvPTMGJEg')
Out[107]:
```

Let’s download the video to detect objects (to be covered later in detail)

```
In [15]: from pytube import YouTube

         yt = YouTube("https://www.youtube.com/watch?v=tknvPTMGJEg")
         yt.streams.filter(file_extension="mp4").get_by_resolution("360p").download("./downloaded-video.mp4")
```

Next Steps

- Train the model on downloaded dataset for cooking only
- Write script to get input from user for any youtube video
- Download the video and convert downloaded video into frames
- Run the trained model to identify labels for the video

Example of Object Detection in Image

Object Detection - Image

Install PixelLib and its dependencies

- *pip3 install pycocotools*
- *pip3 install pixellib*

Download the PointRend model. This is the code for image segmentation.

In [17]:
```
import pixellib
from pixellib.torchbackend.instance import instanceSegmentation
```

Read input image from `sample/input-image.jpg` then convert and save image with object identification at `sample/output-image.jpg`

In [18]:
```
ins = instanceSegmentation()
ins.load_model("./models/pointrend_resnet50.pkl")
info = ins.segmentImage("./sample/input-image.jpg", show_bboxes=True, output_image_name="./sample/output-image.jpg"
```

```
The checkpoint state_dict contains keys that are not used by the model:
  proposal_generator.anchor_generator.cell_anchors.{0, 1, 2, 3, 4}
```

Read what objects were detected in the image

In [19]:
```
print(", ".join(info[0]["class_names"]))
```

```
bowl, knife, spoon, apple, bottle, apple, sink, knife
```

Example of Object Detection in Video

Object Detection - Video ¶

Install PixelLib and its dependencies

- *pip3 install pycocotools*
- *pip3 install pixellib*

Download the PointRend model. This is the code for video segmentation.

In [2]:
```
import pixellib
from pixellib.torchbackend.instance import instanceSegmentation
```

Read input video from `sample/input-video.mp4` then convert and save image with object identification at `sample/output-video.mp4`

In [3]:
```
ins = instanceSegmentation()
ins.load_model("./models/pointrend_resnet50.pkl", detection_speed = "rapid")
info = ins.process_video("./sample/input-video.mp4", show_bboxes=True, frames_per_second=12, output_video_name="./sa
```

```
No. of frames: 698
No. of frames: 699
No. of frames: 700
No. of frames: 701
No. of frames: 702
No. of frames: 703
No. of frames: 704
No. of frames: 705
No. of frames: 706
No. of frames: 707
No. of frames: 708
No. of frames: 709
No. of frames: 710
No. of frames: 711
No. of frames: 712
No. of frames: 713
No. of frames: 714
No. of frames: 715
No. of frames: 716
No. of frames: 717
```

Read what objects were detected in the image

In [6]:
```
print(", ".join(info[0]["class_names"]))
```

```
bowl, cake, banana, dining table
```