ETL, Data Augmentation

Preprocessing

Datasets usually need to be transformed for DL training

- selection, sorting, shuffling, sharding
- transcoding
- size/color normalization, feature extraction
- augmentation
- inference (classification-selection, upscaling, segmentation, feature extraction)

Online vs Ofline Preprocessing

online, streaming preprocessing

- read data from storage
- pass through a streaming processor
- take input from stream

offline preprocessing

- apply transformations to data set offline and store result
- may greatly expand dataset for augmentation
- usually carried out using map-reduce

No single best solution; depends on many factors.

Fundamental Concerns in Preprocessing

- preprocessing costs are amortized over hundreds of epochs of training
- even seemingly small costs are often better carried out offline
- storage is cheap for offline preprocessing
- offline augmentation reduces variability of results
- preprocessing often carried out on CPUs
- often need extra CPU servers for efficient use of GPUs

Frameworks

- offline preprocessing usually uses map-reduce like frameworks
  - common: Hadoop, find+xargs
  - simpler: tarproc
- online preprocessing requires streaming processing
  - common: Apache Kafka
  - simpler: ZMQ pipes, tensorcom

File Based: `find|xargs`

Common operations over file systems:

```
$ find . -name '*.jpg' -print | xargs -I {} 'convert {} {}.png'
```

Parallel version:

```
$ find . -name '*.jpg' -print | xargs -P 8 -I {} 'convert {} {}.png'
```

Using GNU Parallel:

```
$ find . -name '*.jpg' -print | parallel 'convert {} {.}.jpg'
```

File Based Sequential: `tarproc`

- similar to `find|xargs` but operates over `.tar` files
- operates on groups of files with the same basename

WebDataset Input:

```
$ tar -tvf imagenet.tar
-rw-rw-rw- bigdata/bigdata     3 2019-06-08 12:12 n03788365_17158.cls
-rw-rw-rw- bigdata/bigdata 75884 2019-06-08 12:12 n03788365_17158.jpg
-rw-rw-rw- bigdata/bigdata   382 2019-06-08 12:12 n03788365_17158.json
-rw-rw-rw- bigdata/bigdata     3 2019-06-08 12:12 n03000247_49831.cls
-rw-rw-rw- bigdata/bigdata 57068 2019-06-08 12:12 n03000247_49831.jpg
-rw-rw-rw- bigdata/bigdata   104 2019-06-08 12:12 n03000247_49831.json
...
```

File Based Sequential: `tarproc` (output)

Processing:

$ tarproc -p 8 -c 'convert sample.jpg sample.png && rm sample.jpg sample.json' < imagenet.tar -o imagenet-png.tar

$ tar -tvf imagenet-png.tar

-r--r--r-- bigdata/bigdata 3 2019-11-19 13:38 n03788365_17158.cls

-r--r--r-- bigdata/bigdata 246551 2019-11-19 13:38 n03788365_17158.png

-r--r--r-- bigdata/bigdata 3 2019-11-19 13:38 n03000247_49831.cls

-r--r--r-- bigdata/bigdata 177289 2019-11-19 13:38 n03000247_49831.png

-r--r--r-- bigdata/bigdata 3 2019-11-19 13:38 n03000247_22907.cls

...

Writing Transforms in Python:

```
src = WebDataset("imagenet.tar")
sink = TarWriter("imagenet-png.tar)

for sample in src:
    sample["png"] = zoom(sample["jpg"], (0.5, 0.5, 1))
    del sample["jpg"]
    sink.write(sample)

sink.close()
```

Parallel Mapping of Many Shards

```
for shard in {0000..0147}; do
    kubetpl -c "

        gsutil stat gs://output-bucket/imagenet-$shard.tgz ||
        gsutil cat gs://bucket/imagenet-$shard.tgz |
        tarproc -c 'convert sample.jpg sample.ppm; rm sample.jpg' |
        gsutil cp - gs://output-bucket/imagenet-$shard.tgz

    " | kubectl apply -f -
done
```

- here: cloud bucket to cloud bucket
- process 148 shards in parallel on K8s
- familiar file system commands

Map-Reduce as $(MS)^2$

Hadoop-style map-reduce can be written as:

- sequential processing
- sorting
- sequential processing
- sorting

Note:

- sequential processing can use tarproc, WebDataset, or any other framework
- AIStore has highly optimized server-side support for the sort phase

ETL Example

(notebook)