Caffe

Deep learning framework by the BVLC

Created by
Yangqing Jia
Lead Developer
Evan Shelhamer

View On GitHub

Data: Ins and Outs

Data flows through Caffe as Blobs. Data layers load input and save output by converting to and from Blob to other formats. Common transformations like mean-subtraction and feature-scaling are done by data layer configuration. New input types are supported by developing a new data layer – the rest of the Net follows by the modularity of the Caffe layer catalogue.

This data layer definition

```
layer {
  name: "mnist"
  # Data layer loads leveldb or lmdb storage DBs for high-throughput.
  type: "Data"
  # the 1st top is the data itself: the name is only convention
  top: "data"
  # the 2nd top is the ground truth: the name is only convention
  top: "label"
  # the Data layer configuration
  data_param {
    # path to the DB
    source: "examples/mnist/mnist_train_lmdb"
    # type of DB: LEVELDB or LMDB (LMDB supports concurrent reads)
    backend: LMDB
    # batch processing improves efficiency.
    batch_size: 64
  }
  # common data transformations
  transform_param {
    # feature scaling coefficient: this maps the [0, 255] MNIST data to
[0, 1]
    scale: 0.00390625
  }
}
```

loads the MNIST digits.

Tops and Bottoms: A data layer makes **top** blobs to output data to the model. It does not have **bottom** blobs since it takes no input.

Data and Label: a data layer has at least one top canonically named **data**. For ground truth a second top can be defined that is canonically named **label**. Both tops simply produce blobs and there is nothing inherently special about these names. The (data, label) pairing is a convenience for classification models.

Transformations: data preprocessing is parametrized by transformation messages within the data layer definition.

```
layer {
  name: "data"
  type: "Data"
  [...]
  transform_param {
    scale: 0.1
    mean_file_size: mean.binaryproto
    # for images in particular horizontal mirroring and random cropping
    # can be done as simple data augmentations.
    mirror: 1  # 1 = on, 0 = off
    # crop a `crop_size` x `crop_size` patch:
    # - at random during training
    # - from the center during testing
    crop_size: 227
  }
}
```

Prefetching: for throughput data layers fetch the next batch of data and prepare it in the background while the Net computes the current batch.

Multiple Inputs: a Net can have multiple inputs of any number and type. Define as many data layers as needed giving each a unique name and top. Multiple inputs are useful for non-trivial ground truth: one data layer loads the actual data and the other data layer loads the ground truth in lock-step. In this arrangement both data and label can be any 4D array. Further applications of multiple inputs are found in multi-modal and sequence models. In these cases you may need to implement your own data preparation routines or a special data layer.

Improvements to data processing to add formats, generality, or helper utilities are welcome!

Formats

Refer to the layer catalogue of data layers for close-ups on each type of data Caffe understands.

Deployment Input

For on-the-fly computation deployment Nets define their inputs by `input` fields: these Nets then accept direct assignment of data for online or interactive computation.