Caffe Deep Learning Framework Cheatsheet

A quick reference guide for Caffe, covering its architecture, layers, tools, and usage for deep learning tasks.

Core Concepts & Architecture

Caffe Components

Blob	The standard array/structure to store, communicate, and manipulate the actual data. It hides the computation and memory overhead of synchronization and communication from the user.
Net	A Net is composed of interconnected layers which describe a complete model. The net defines the data flow from layer to layer and stores the intermediate results.
Layer	The basic building block of a model. Layers take blobs as input and output blobs. They encapsulate all model parameters (weights) and computation.
Solver	Caffe drives learning and prediction by the `Solver` which orchestrates the optimization of the network.

Data Layer

The data layer sits at the bottom of a Caffe model. It is responsible for efficiently reading in data from disk, transforming it, and loading it into the network.
Common types include:

- Data
- ImageData
- HDF5Data
- MemoryData

Common Layers

Convolution Layer	Applies a convolution filter to the input. Parameters: `num_output`, `kernel_size`, `stride`, `pad`.
Pooling Layer	Performs pooling (e.g., max or average) over the input. Parameters: `kernel_size`, `stride`, `pool` (MAX or AVE).
ReLU Layer	Applies the rectified linear unit activation function (max(0, x)).
InnerProduct Layer	Also known as a fully connected layer. Parameters: `num_output`.
Softmax Layer	Applies the softmax function to produce a probability distribution over classes.

Defining Models with Protobuf

Net Definition (prototxt)

Caffe models are defined using protocol buffer (protobuf) text files (`.prototxt`). These files specify the network architecture (layers), data sources, and solver parameters.

Example Net Definition:

```
name: "MyAwesomeNet"
layer {
  name: "data"
  type: "Data"
  top: "data"
  top: "label"
  data_param {
    source: "/path/to/data.txt"
    batch_size: 64
    backend: LMDB
  }
}
```

Layer Definition

Each layer in the network is defined by a `layer` block. Key parameters include:

- `name`: The name of the layer.
- `type`: The type of layer (e.g., Convolution, ReLU, Pooling).
- `bottom`: The input blob(s) to the layer.
- `top`: The output blob(s) from the layer.
- Layer-specific parameters (e.g., `convolution_param`, `pooling_param`).

Solver Definition (prototxt)

The solver prototxt file defines the optimization parameters for training the network.

Example Solver Definition:

```
net: "/path/to/net.prototxt"
test_iter: 100
test_interval: 1000
base_lr: 0.01
lr_policy: "step"
gamma: 0.1
stepsize: 10000
max_iter: 100000
momentum: 0.9
weight_decay: 0.005
snapshot: 5000
snapshot_prefix:
"/path/to/snapshots/model"
solver_mode: GPU
```

Caffe Command Line Tools

Training a Model

Use the `caffe train` command to train a model.

```
caffe train --solver solver.prototxt
```

Options:

- `--solver`: Path to the solver prototxt file.
- `--gpu`: Specify GPU(s) to use (e.g., `--gpu 0,1`). Omit to use CPU.
- `--weights`: Path to pretrained model weights to initialize from.

Testing a Model

You can test a trained model using `caffe test` (though testing is often integrated into the training prototxt).

```
caffe test --model deploy.prototxt --
weights model.caffemodel --gpu 0
```

Options:

- `--model`: Path to the deploy prototxt file.
- `--weights`: Path to the trained model weights (`.caffemodel`).
- `--gpu`: Specify GPU(s) to use.

Converting Data

Caffe often uses LMDB or LevelDB databases for efficient data storage. The `convert_imageset` tool converts a directory of images into one of these formats.

```
convert_imageset --resize_height 256 --
resize_width 256 /path/to/images/
/path/to/labels.txt lmdb/data_lmdb
```

Python Interface

Basic Usage

Caffe provides a Python interface for model definition, training, and inference.

```
import caffe

# Set the mode
caffe.set_mode_gpu()
caffe.set_device(0)

# Load the model
net = caffe.Net('deploy.prototxt',
'model.caffemodel', caffe.TEST)

# Load input data
# ... (preprocessing steps)
net.blobs['data'].data[...] = input_data

# Run forward pass
output = net.forward()

# Get output
predictions = output['prob'][0].argmax()
```

Working with Blobs

Blobs are accessed via `net.blobs` (for input and intermediate data) and `net.params` (for learned parameters).

```
# Access data blob
data = net.blobs['data'].data

# Access weights of a layer
weights = net.params['conv1'][0].data

# Access biases of a layer
biases = net.params['conv1'][1].data
```

Custom Layers (Python)

Caffe allows you to define custom layers in Python. You need to define `setup()`, `reshape()`, `forward()`, and `backward()` methods.

Example:

```
import caffe
import numpy as np

class MyPythonLayer(caffe.Layer):
    def setup(self, bottom, top):
        # ... initialization

    def reshape(self, bottom, top):
        # ... reshape outputs

    def forward(self, bottom, top):
        # ... compute outputs

    def backward(self, top,
propagate_down, bottom):
        # ... compute gradients
```