Starting Caffe for Computer Vision Task

June 1 2017

Jungwon Kang

Presentation Schedule

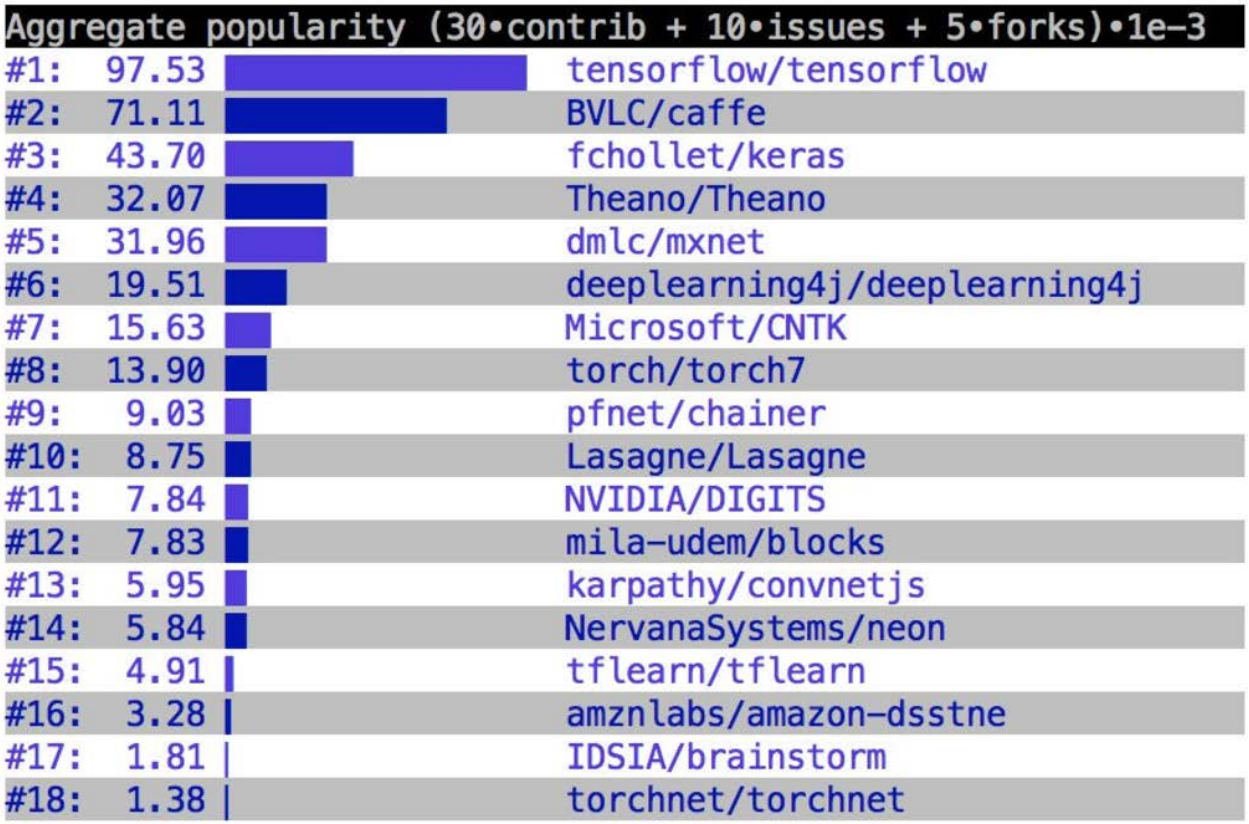
Date	Topic
6/1	**Starting Caffe** • Caffe Installation • Caffe Programming Overview
6/15 (Temporary)	**Caffe Programming Guide**
TBD	**Object Detection Using Caffe**

Contents

- **1. Introduction**
- 2. Caffe Installation
- 3. Caffe Programming Overview
- 4. Conclusion

Flood of Deep Learning Framework

- Caffe
- Tensorflow
- Torch
- PyTorch
- Theano
- Keras
- MatConvNet
- ……



The deep learning frameworks landscape, August 2016.
*source: pic.twitter.com/FLNKPpw88n

I Chose Caffe. Why?

- I would like to deal with deep learning-based computer vision problems such as **object detection** in 2D images.

- I chose Caffe because **numerous Caffe-based open source codes** for computer vision are available.

What is Caffe?

- **Caffe**: Convolutional Architecture for Fast Feature Embedding
  - Developed by BVLC (Berkeley Vision & Learning Center).
  - Caffe 1.0 (stable version released on Apr 2017), Now Caffe2 is available.

- Deep learning framework mainly for **computer vision tasks**
  - Not intended for other applications such as text, sound or time series data.

- Supports **c++** and **python** for Caffe-based programming.
  - **Caffe** : original caffe for c++
  - **PyCaffe**: caffe bindings for python

- However, Caffe is notorious for **tricky installation** and **poor documentation**.

Contents

- 1. Introduction
- **2. Caffe Installation**
- 3. Caffe Programming Overview
- 4. Conclusion

Prerequisites

- **Hardware**
  - NVIDIA Graphic Card (such as NVIDIA TITAN X, GeForce GTX 1080 Ti)

- **Software**
  - Linux: Ubuntu 14.04 or later
  - NVIDIA GPU libraries: CUDA, CUDNN(optional)
  - Python package, e.g. Anaconda
  - Python IDE: PyCharm / Jupyter (IPython Notebook)
  - Numerous miscellaneous libraries for supporting Caffe

Setting Up a System

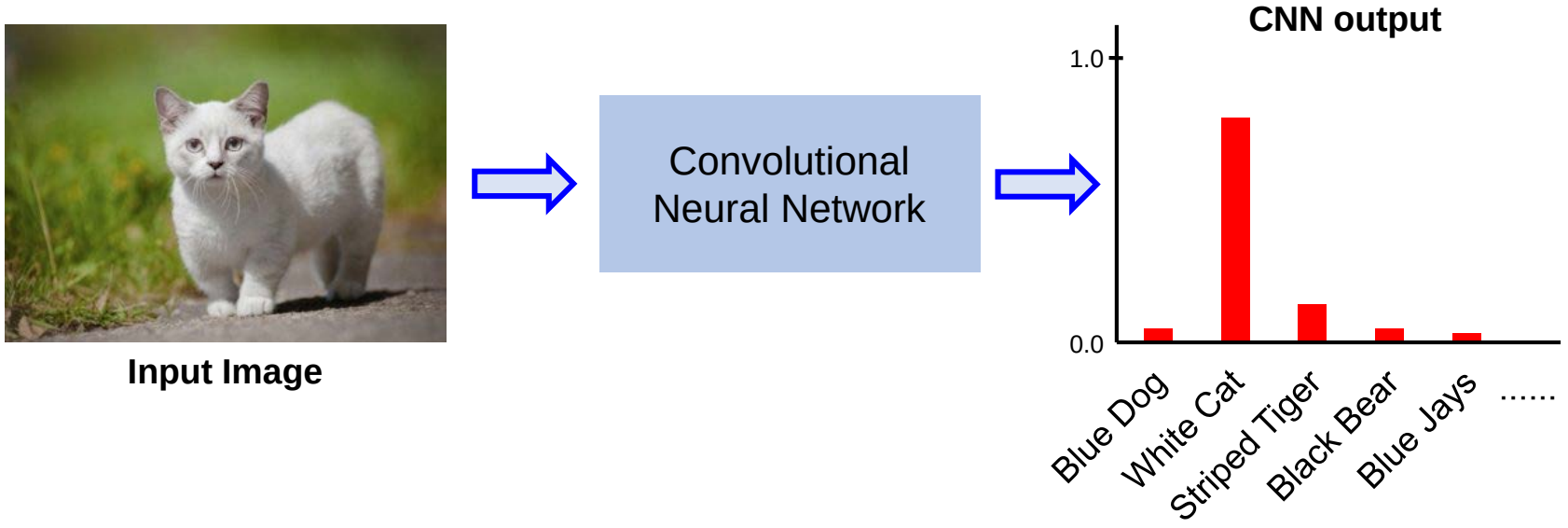
Computer Type	Price (USD)	Installation Type	Easiness of Installation	Usual programming environment
Your own computer	2,000 ~ 3,000	One-by-One	Difficult	GUI supported
		Using Docker	Easy (if familiar with Docker)	CUI e.g. Jupyter
AWS (Amazon EC2) cloud computer	600 for 1 GPU, 2,000 for 4 GPU, 5,000 for 8 GPU, 11,000 for 16 GPU (*running for 1 month)	One-by-One	Difficult	CUI e.g. Jupyter
		Using Docker	Easy (if familiar with Docker)	CUI e.g. Jupyter
		Using AMI	Easy (if familiar with AMI)	CUI e.g. Jupyter

- **Installing one-by-one in your own computer**
  : installing Caffe 1.0 on Ubuntu 16.04 with Anaconda2, Cuda 8.0, Cudnn 6.0
  → http://blog.daum.net/jungwonkang

Contents

- 1. Introduction
- 2. Caffe Installation
- **3. Caffe Programming Overview**
- 4. Conclusion

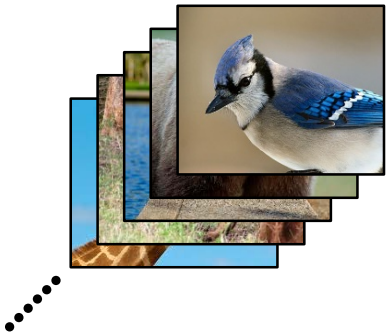
Image Classification



Whole Process for Image Classification

Training Database

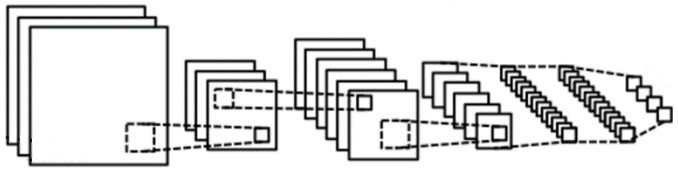
Images

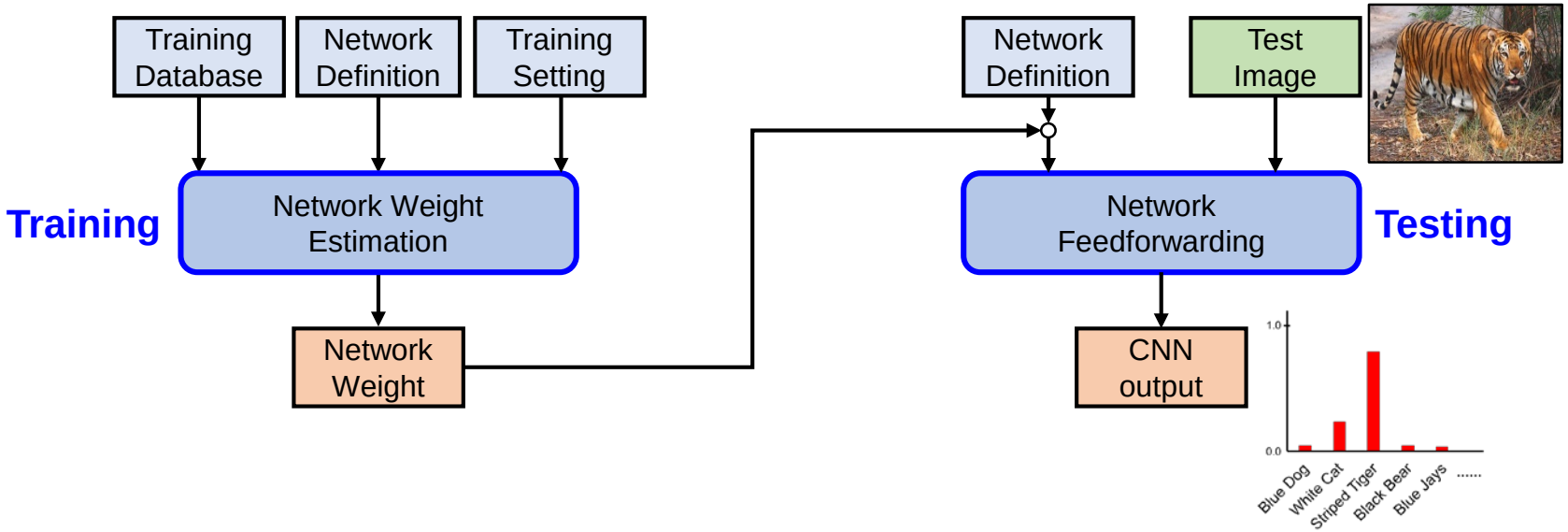


Labels

Blue Jays
Black Bear
Yellow Dog
Striped Tiger
Giraffe
...
...

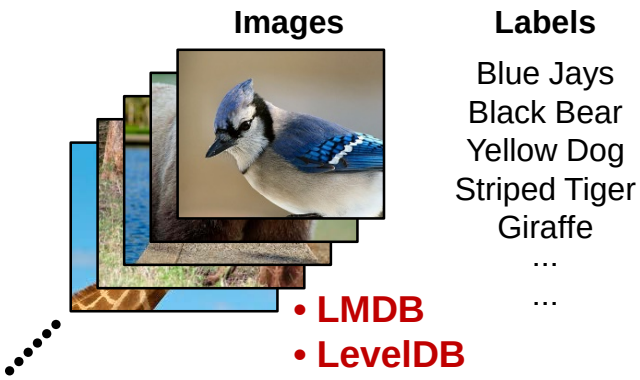
Network Definition





Available Implementation

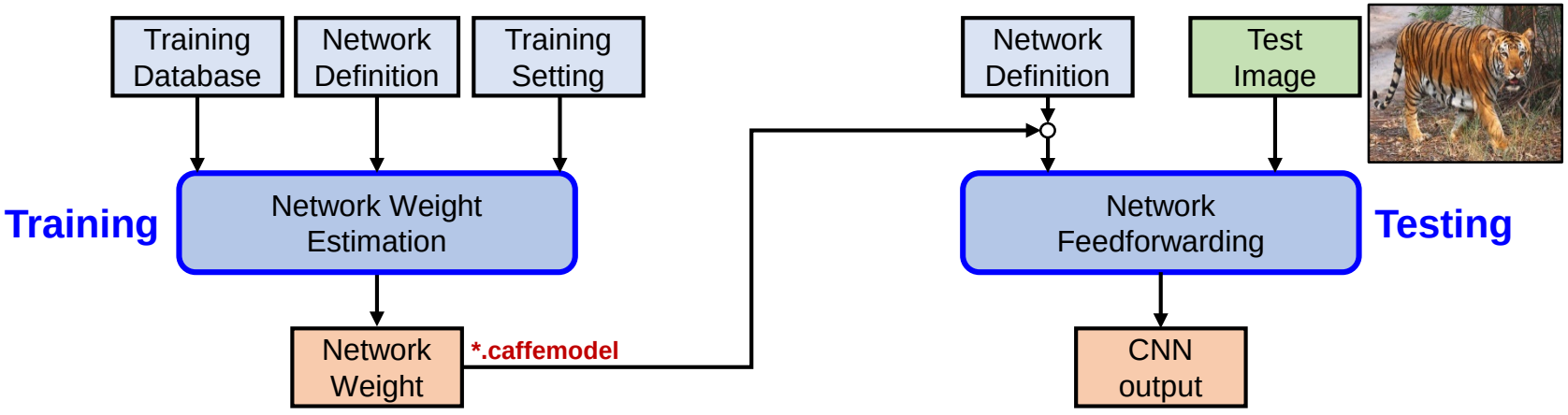
Training Database



- **LMDB**
- **LevelDB**
- **Raw images & labels**

Network Definition

- **protobuf**
- **python**

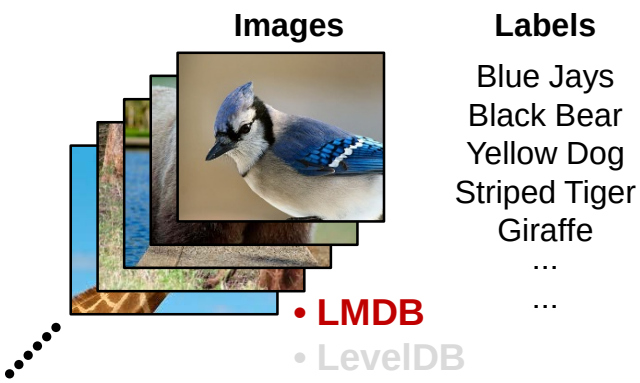


- **caffe training binary + protobuf**
  (only if network definition is done using protobuf)
- **python**

- **caffe testing binary**
  (only if network definition is done using protobuf)
- **python**

Usual Implementation

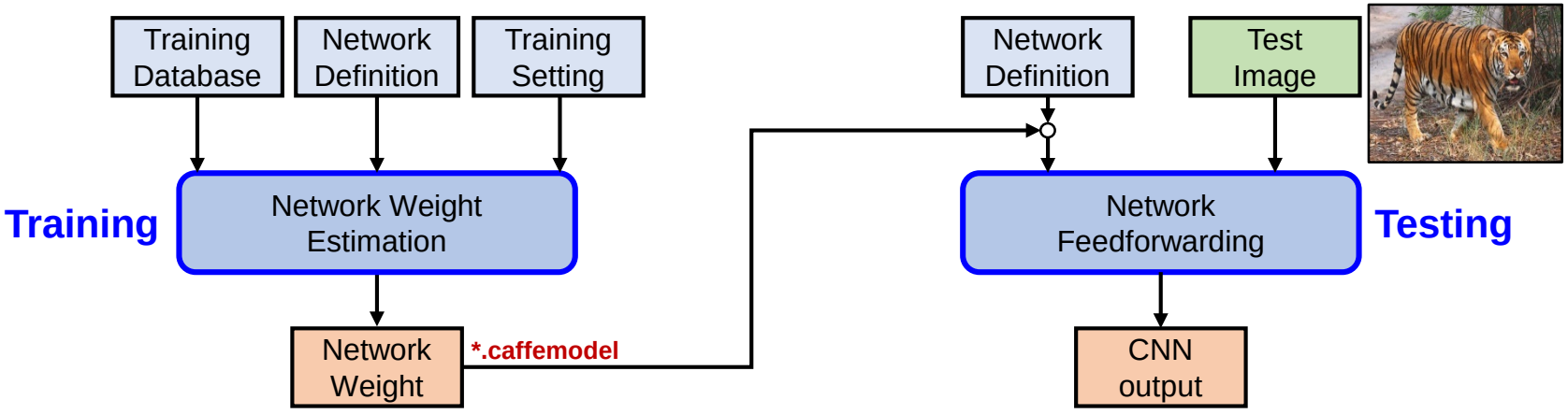
Training Database



- **LMDB**
- LevelDB
- Raw images & labels

Network Definition

- **protobuf**
- python



- **caffe training binary + protobuf**
  (only if network definition is done using protobuf)
- python

- caffe testing binary
  (only if network definition is done using protobuf)
- **python**

Step 1 in Usual Implementation

Training Database

Images

Labels

Blue Jays
Black Bear
Yellow Dog
Striped Tiger
Giraffe
...
...

- **LMDB**
- **LevelDB**
- **Raw images & labels**

Step 2 in Usual Implementation

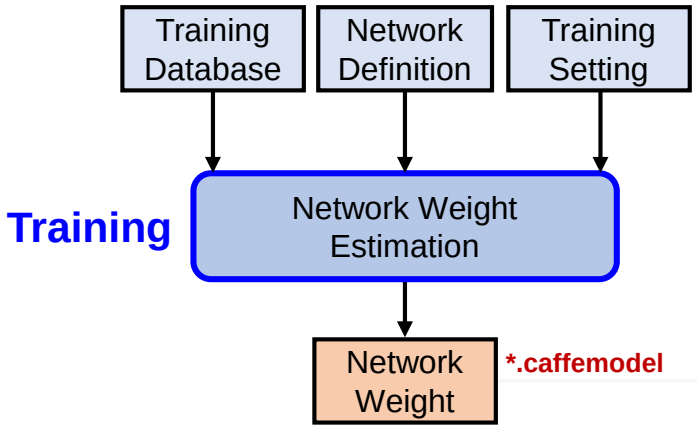
Protobuf for Network Definition

```
layer {
  name: "data"
  type: "Input"
  top: "data"
  input_param { shape: { dim: 10 dim: 3 dim: 227 dim: 227 } }
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 96
    kernel_size: 11
    stride: 4
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}
......
......
```

Network Definition

- **protobuf**
- python

Step 3 in Usual Implementation



- **caffe training binary + protobuf**
  (only if network definition is done using protobuf)
- python

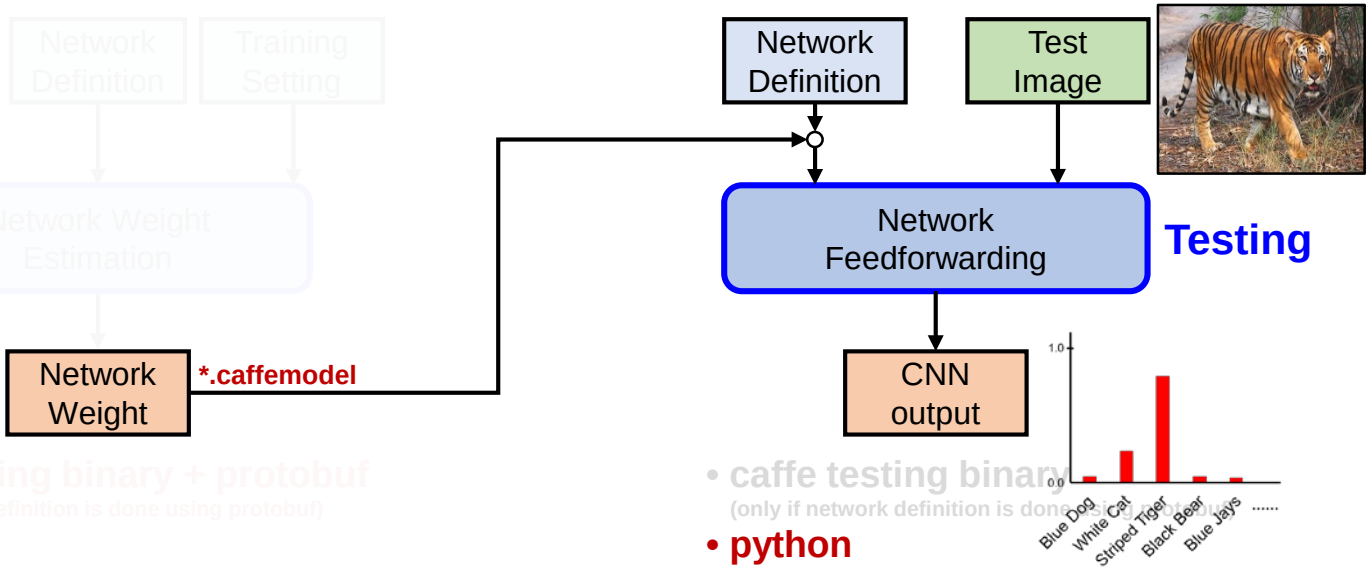
Protobuf for Training

```
net: "models/bvlc_alexnet/alexnet.prototxt"
test_iter: 1000
test_interval: 1000
base_lr: 0.01
lr_policy: "step"
gamma: 0.1
stepsize: 100000
display: 20
max_iter: 450000
momentum: 0.9
weight_decay: 0.0005
snapshot: 10000
snapshot_prefix: "models/bvlc_alexnet/caffe_alexnet_train"
solver_mode: GPU
```

Step 4 in Usual Implementation

Python Code for Testing

```
import caffe
...
model_def = caffe_root +'models/bvlc_alexnet/alexnet.prototxt'
model_weights = caffe_root + 'models/bvlc_alexnet/bvlc_alexnet.caffemodel'
...
net = caffe.Net(model_def, model_weights, caffe.TEST)
...
output = net.forward()
...
```



• python

Contents

- 1. Introduction
- 2. Caffe Installation
- 3. Caffe Programming Overview
- **4. Conclusion**

What We Need for Starting Caffe

- For Caffe installation, please be patient.

- For Caffe programming, we need to be familiar with
  - **Ubuntu linux**
  - **Python**
  - **Protobuf**
  - **LMDB**

- In addition, we need to closely look at
  - how to use **docker**
  - how to use **AWS**