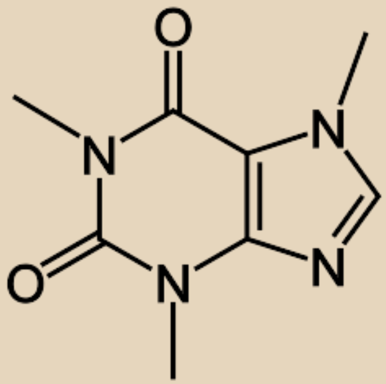


Caffe:

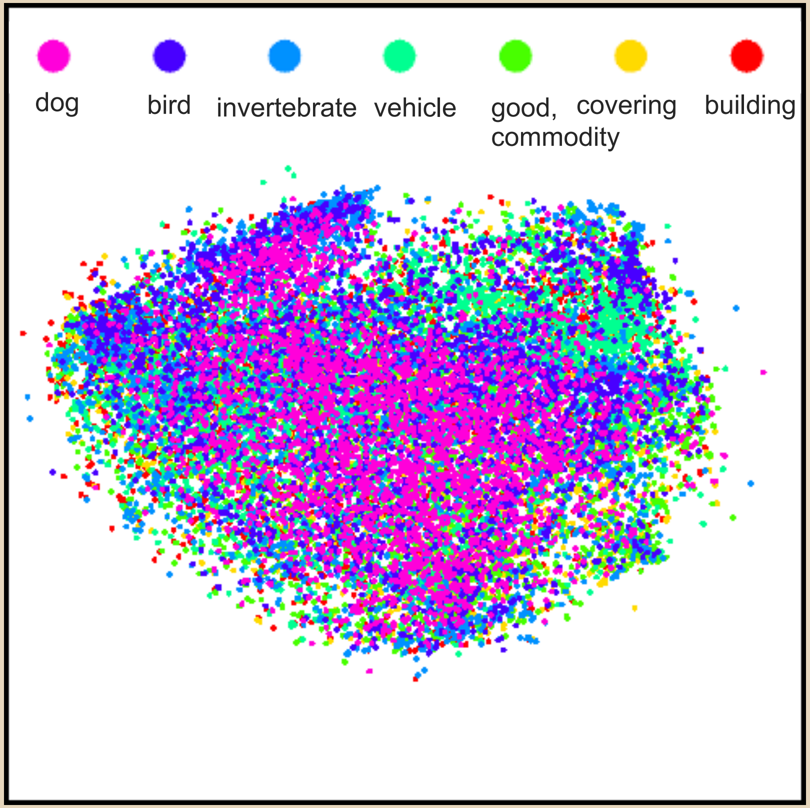
Convolutional Architecture for Fast Feature Embedding

Created by Yangqing Jia
Developed by BVLC

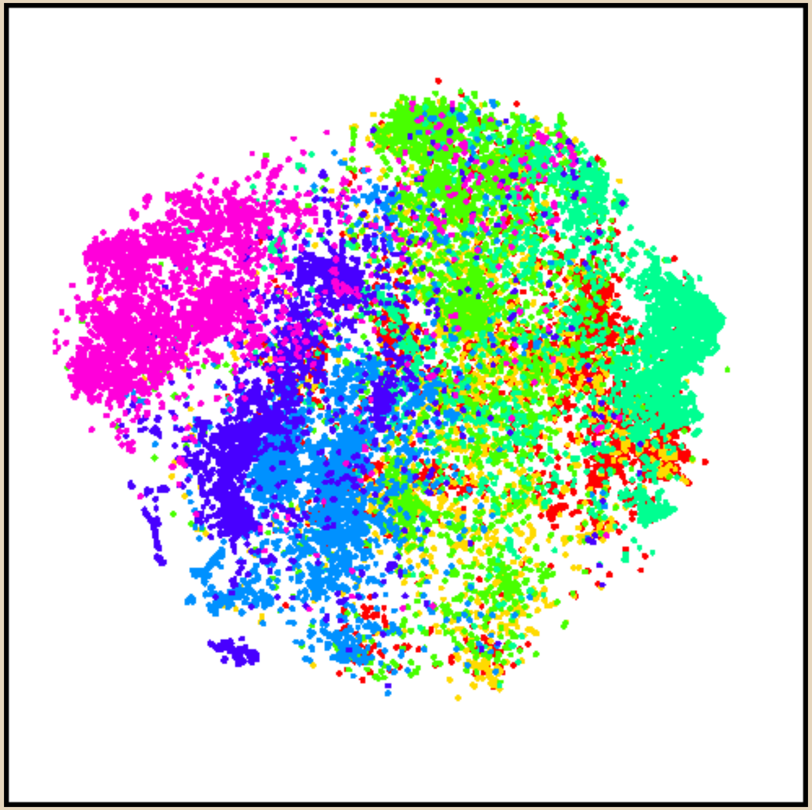
caffe.berkeleyvision.org
bvlc.eecs.berkeley.edu

Why Caffe?
The Unreasonable Effectiveness of Deep Features

Feature Embedding: learning to map visual features into a discriminative space.



Low-level: Pool$_1$



High-level: FC$_6$

Note that classes separate at higher layers of the representation!

DeCAF: Deep Convolutional Activation Feature, Donahue & Jia* ICML14.*

So what is Caffe?

Convolutional Architecture for Fast Feature Embedding

- C++/CUDA framework for deep learning and vision
  - library of layers that compose into models
  - fast stochastic gradient descent (SGD) solver
  - tools, demos, and recipes
- Seamless switch between CPU and GPU
  - Caffe::set_mode(Caffe::CPU);
- BSD-2 licensed

Train Models

Experiment/Prototype

Inference at Scale

All with essentially the same code!

So what is Caffe?

Convolutional Architecture for Fast Feature Embedding

- Model schemas
  - Define the model and solving strategy and let Caffe take care of the rest
  - Adapt already learned models to new problems in one step

State-of-the-art solving in 14 lines.

```
train_net: "imagenet_train.prototxt"
test_net: "imagenet_val.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: "caffe_imagenet_train"
```

Models are schema, not code.

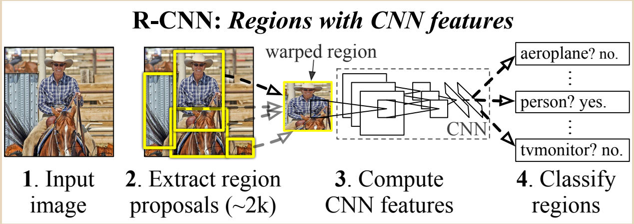
```
name: "CaffeNet"
input: "data"
input_dim: 10
input_dim: 3
input_dim: 227
input_dim: 227
# convolution 1: 96 filters
layers {
  layer {
    name: "conv1"
    type: "conv"
    num_output: 96
    kernelsize: 11
    stride: 4
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0.
    }
    blobs_lr: 1.
    blobs_lr: 2.
    weight_decay: 1.
    weight_decay: 0.
  }
  bottom: "data"
  top: "conv1"
}
```

. . .

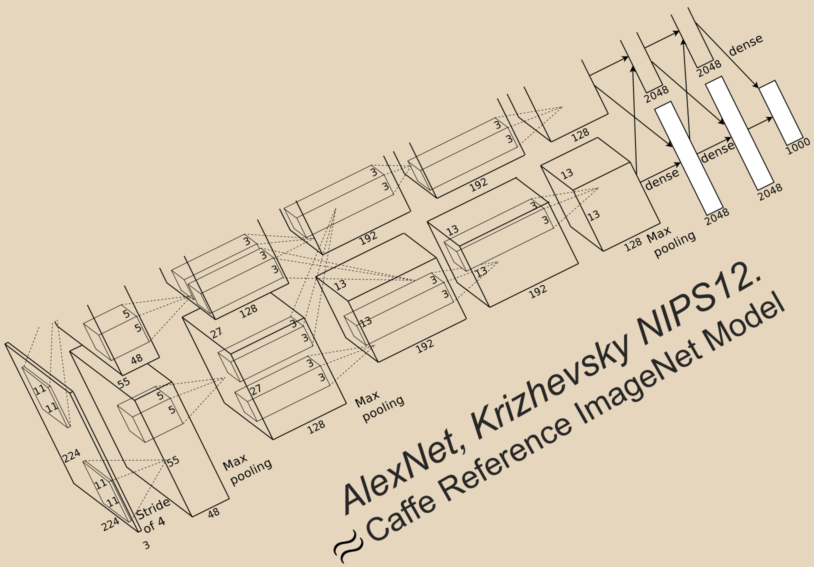
So what is Caffe?

Convolutional Architecture for Fast Feature Embedding

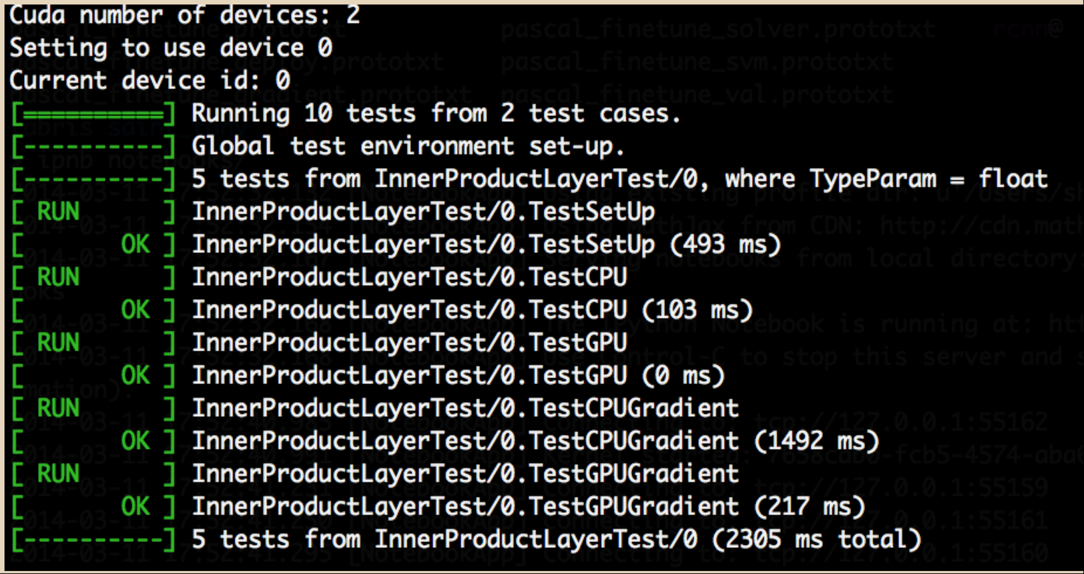
- Research & Engineering
  - Key part of our publication code
  - State-of-the-art models
  - Blazing fast, and it has unit tests!



R-CNN, Girshick CVPR14.



AlexNet, Krizhevsky NIPS12.
≈ Caffe Reference ImageNet Model



So what is Caffe?

Convolutional Architecture for Fast Feature Embedding

- An active research and development community
- See [our contributors]() and [recent activity]()

527 commits | Star 232 | Fork 113

February 12 2014 - March 12 2014 Period: **1 month**

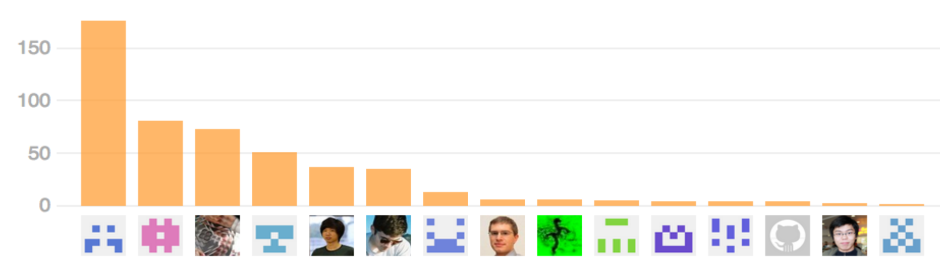
Overview

46 Active Pull Requests

62 Active Issues

23 Merged Pull Requests	**23** Proposed Pull Requests	**37** Closed Issues	**25** New Issues

23 authors have pushed **511 commits** to all branches, excluding merges. On master, **137 files** have changed and there have been **7,376 additions** and **1,436,167 deletions**.



Why not live caffeine-free?

- It's all about speed.

- cuda-convnet and DeCAF are awesome
  - but cuda-convnet is inflexible
  - and DeCAF is too slow

- Caffe is fast
  - with CPU: 2x speedup over DeCAF
  - with GPU: 10x speedup (under C++)

- Forward pass of a single image takes 2.5ms
  - Caffe reference ImageNet model with ~60 million parameters
  - (when in batch mode)
  - (~20ms in CPU mode)

Caffe and cuda-convnet

Caffe

- C++/CUDA deep learning and vision
  - library of layers
  - **fast, general-purpose for** ILSVRC, PASCAL, **your data**
- An active research and development community: public on GitHub
- Seamless GPU/CPU switch
- Model schemas
  - Define the model
  - Configure solver
  - **Finetuning**
- **Wrappers** for Python and MATLAB

cuda-convnet

- C++/CUDA deep learning and vision
  - library of layers
  - **highly-optimized for given data and GPU**
- Static codebase, no community contributions: last update Jul 17, 2012
- GPU only
- Model schemas
  - Define the model
  - Write and run solver command
  - **No finetuning**
- **No wrappers**: monolithic

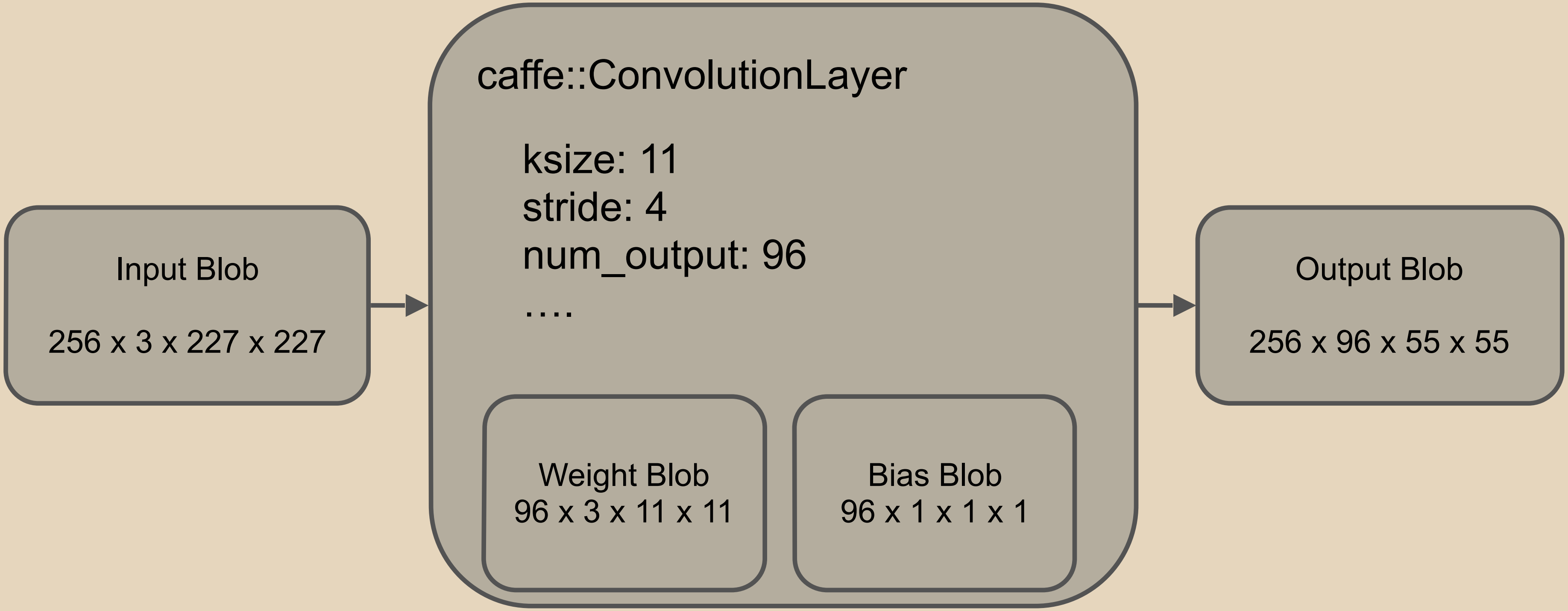
A Caffe Net

Blob: all your data, derivatives, and parameters.

- example input blob (256 images, RGB, height, width)
  - ImageNet training batches: 256 x 3 x 227 x 227
- example convolutional parameter blob
  - 96 filters with 3 input channels: 96 x 3 x 11 x 11

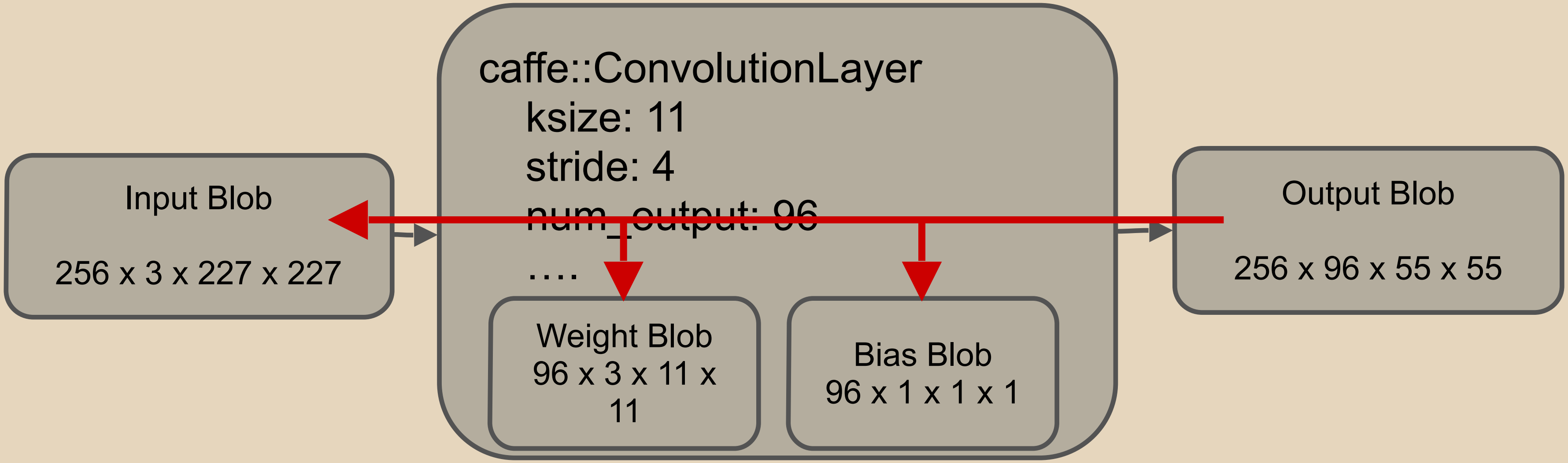
A Layer

- The layer is the fundamental unit of computation.
- Caffe nets are composed of layers as defined in model schema.



A Layer defines...

- Forward: given input, computes the output.
- Backward: given the gradient w.r.t. the output, compute the gradient w.r.t. the input and its internal parameters.
- Setup: how to initialize the layer.



Definition of a Net

Model schema are defined as Protocol Buffers:

```
message NetParameter {
  optional string name = 1;
  repeated LayerConnection layers = 2;
  repeated string input = 3;
  repeated int32 input_dim = 4;
}
```

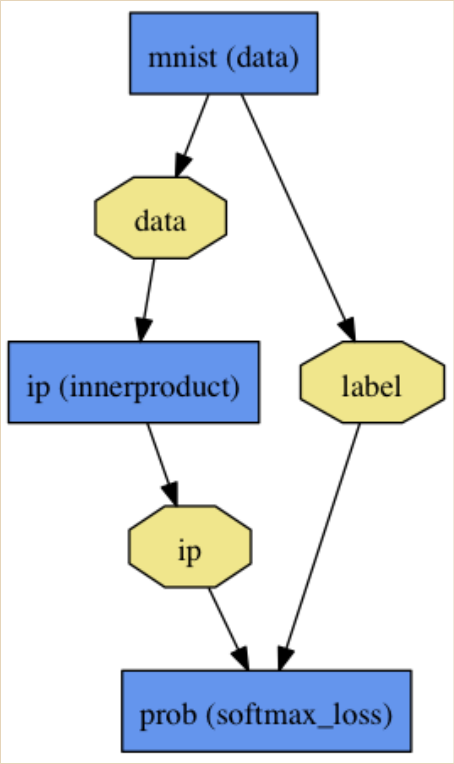
```
name: “linear_regressor”
input: “data”
input_dim: 1
input_dim: 3
input_dim: 28
input_dim: 28
layers {
   layer {
      name: “ip”
      type: “innerproduct”
      num_output: 10
   }
   bottom: “data”
   top: “prediction”
}
```

schema definition at /src/caffe/proto/caffe.proto
Protocol Buffer documentation:
https://developers.google.com/protocol-buffers/docs/overview

Definition of a Net

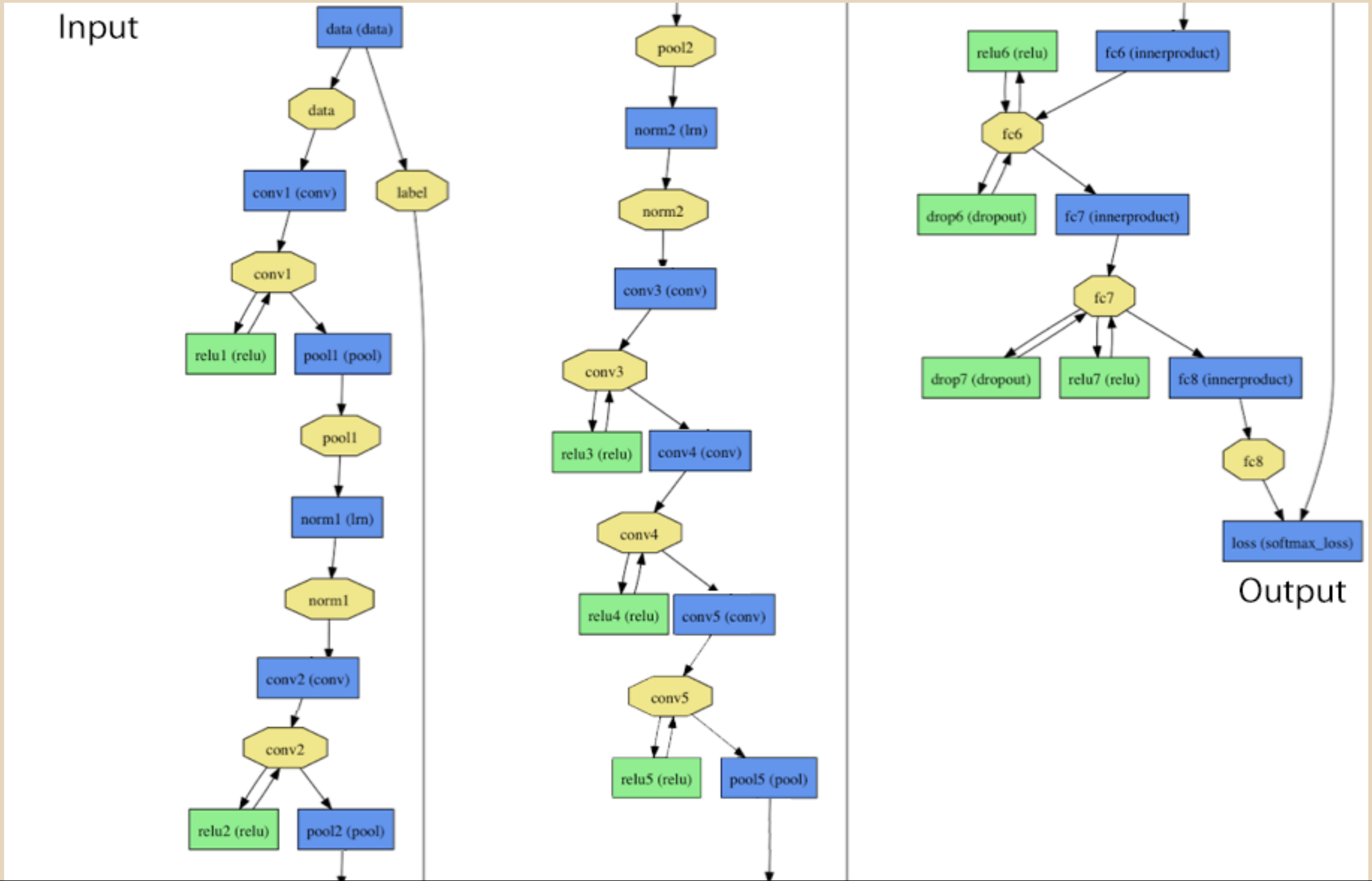
```
name: "mnist-small"
# data layer for input
layers {
  layer {
    name: "mnist"
    type: "data"
    source: "data/mnist-train-leveldb"
    batchsize: 64
    scale: 0.00390625
  }
  top: "data"
  top: "label"
}
#  linear classifier by inner product
layers {
  layer {
    name: "ip"
    type: "innerproduct"
    num_output: 10
    weight_filler {
      type: "xavier"
    }
  }
  bottom: "data"
  top: "ip"
}
```

```
#  softmax loss for training
# takes classifier output and labels
layers {
  layer {
    name: "prob"
    type: "softmax_loss"
  }
  bottom: "ip"
  bottom: "label"
}
```



How about ImageNet?

- It's another network definition… only this time a state-of-the-art model
- See caffe/models/imagenet.prototxt



Finetuning

- Once you have a model–like caffe_reference_imagenet_model–you can solve many problems.
- Where training from scratch can fail for lack of sufficient data, finetuning can succeed.

- Rename the layers you need to change…
- …and continue training.
- No coding needed.

```
layer {
        name: "fc8"
        type: "innerproduct"
        num_output: 1000
        …
}
```

```
layer {
        name: "fc8-t"
        type: "innerproduct"
        num_output: 397
        …
}
```

A Few Practical Questions

- What's the shortest path to features?
  - Swap deep features into your pipeline without tears via the Caffe Reference ImageNet model.
  - Any layer can be extracted.
  - Prototype with Python and MATLAB wrappers.

- Do I have to train from scratch for every problem?
  - Not at all! Finetune learned models to new data and tasks.
  - Define a new model and solver.
  - Call ./finetune_net new_solver old_model # then get a cup of coffee

- What do I do with my own loss, special operation, or data format?
  - Well, this is trickier but doable.
  - Code the layers needed.
  - Define the model and carry on.

Questions!

Check out caffe.berkeleyvision.org,
the Github repository https://github.com/BVLC/caffe,
and our issue tracker https://github.com/BVLC/caffe/issues (but search before posting).

Try our examples and tutorials!

References

[SuperVision/AlexNet] A. Krizhevsky, I. Sutskever, and G. Hinton. *Imagenet classification with deep convolutional neural networks*. NIPS, 2012.

[DeCAF] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. *Decaf: A deep convolutional activation feature for generic visual recognition*. ICML, 2014.

[R-CNN] R. Girshick, J. Donahue, T. Darrell, and J. Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. CVPR, 2014.