ECE6504 – Deep Learning for Perception

Introduction to CAFFE

Ashwin Kalyan V

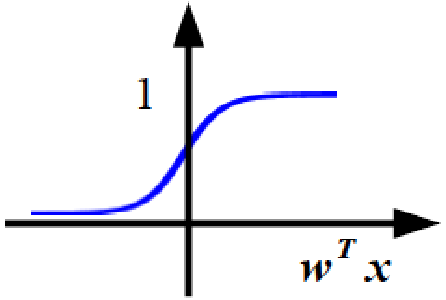
Linear Classifier: Logistic Regression

Input: $\boldsymbol{x} \in R^D$

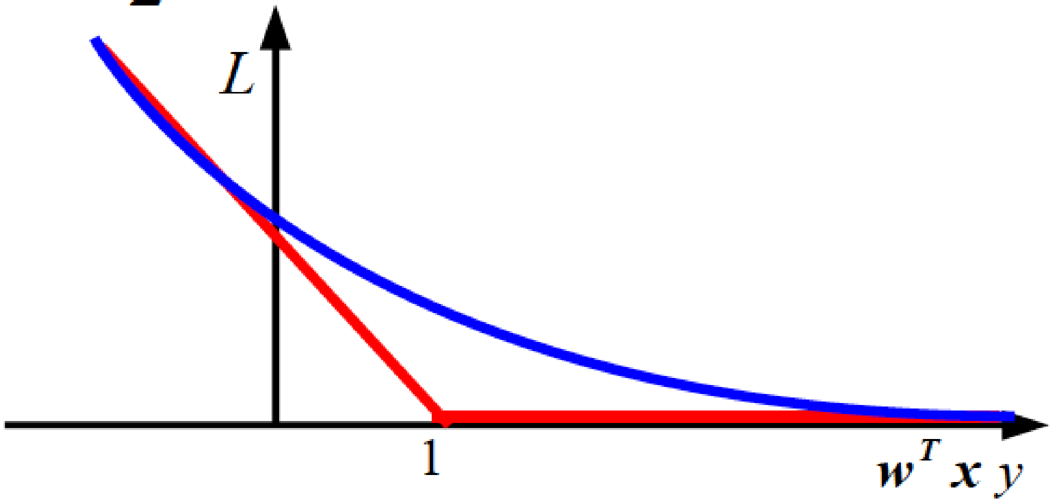
Binary label: $y \in \{-1, +1\}$

Parameters: $\boldsymbol{w} \in R^D$

Output prediction: $p(y=1|\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$



Loss: $L = \frac{1}{2}||\boldsymbol{w}||^2 - \lambda \log(p(y|\boldsymbol{x}))$

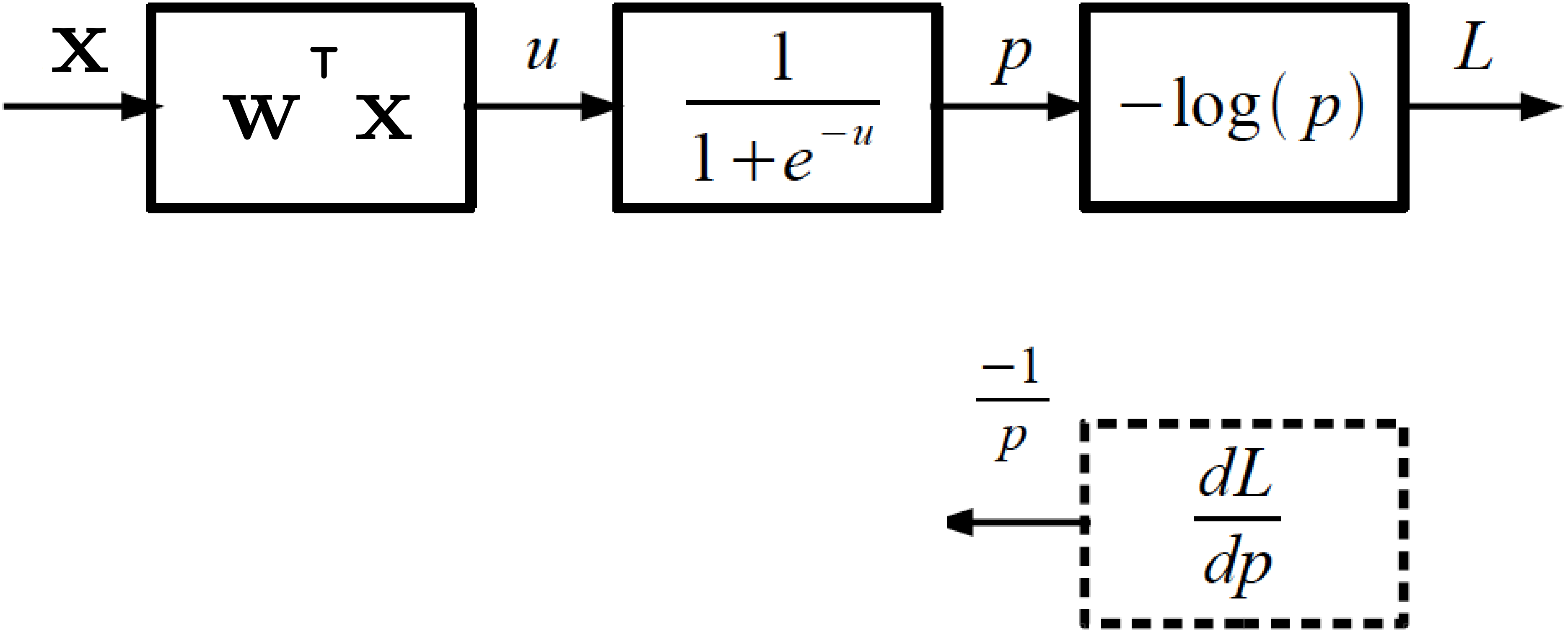


Log Loss

46

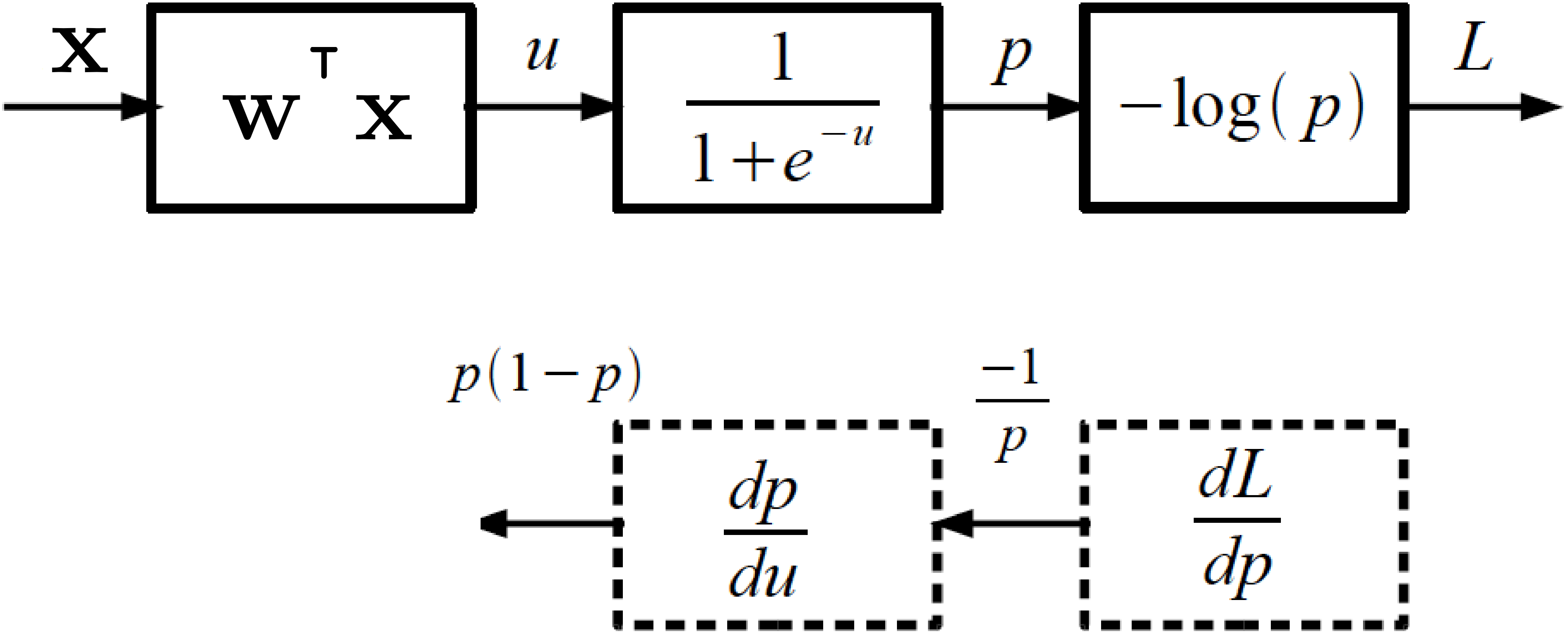
Ranzato

Logistic Regression as a Cascade



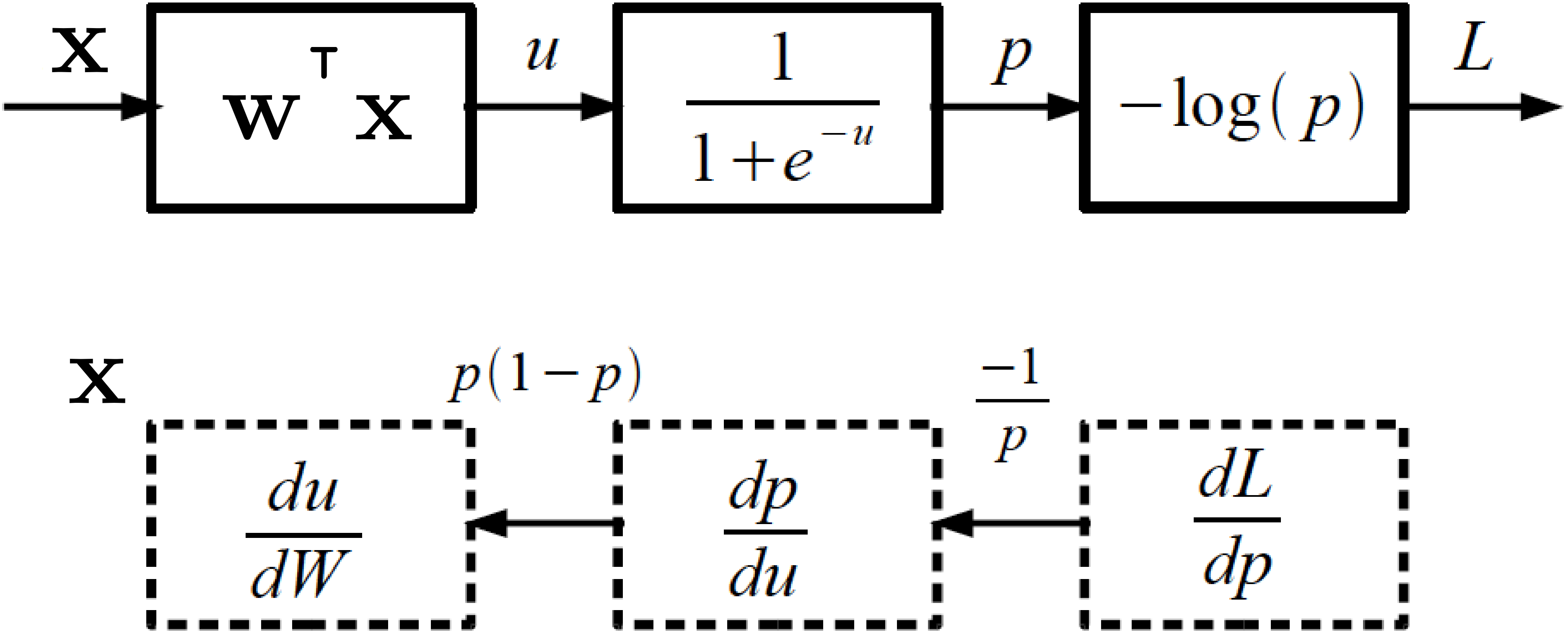
Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Logistic Regression as a Cascade



Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

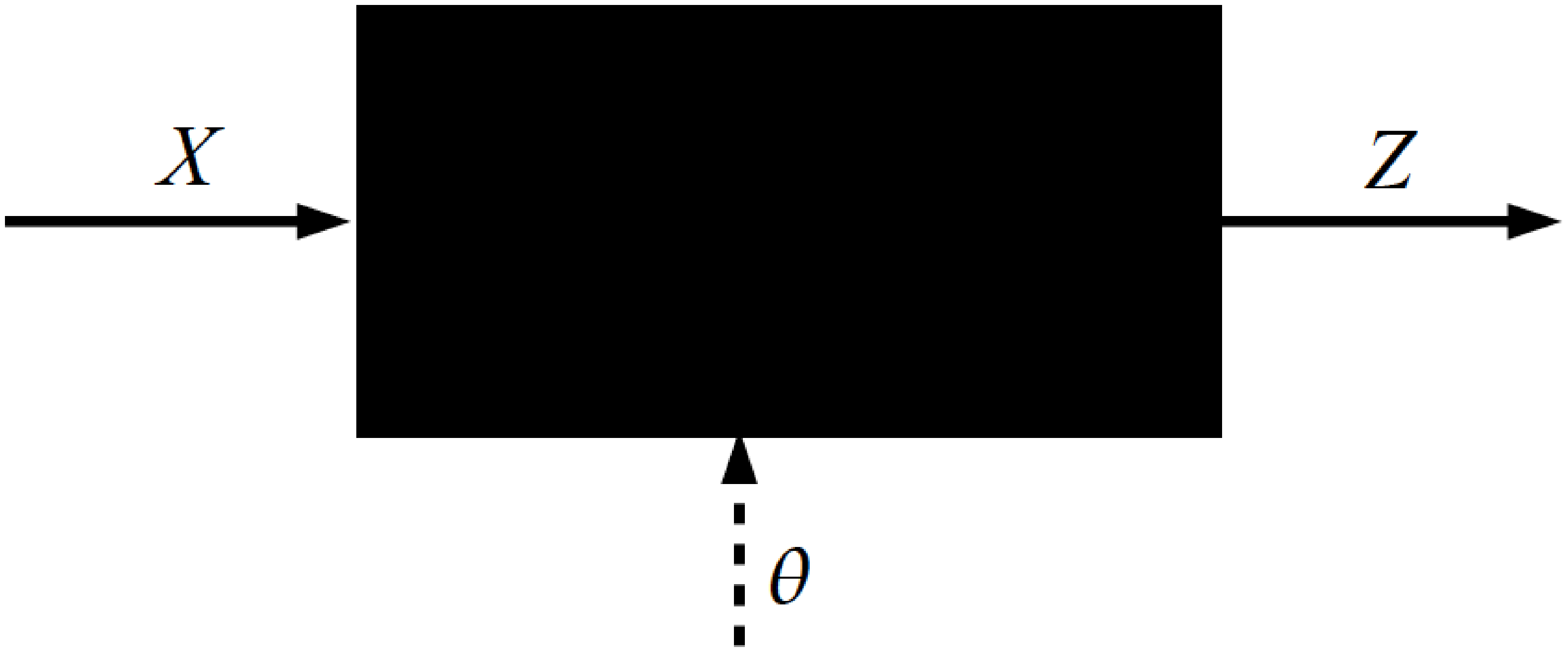
Logistic Regression as a Cascade



$$\frac{dL}{dW} = \frac{dL}{dp} \cdot \frac{dp}{du} \cdot \frac{du}{dW} = (p-1)\mathbf{x}$$

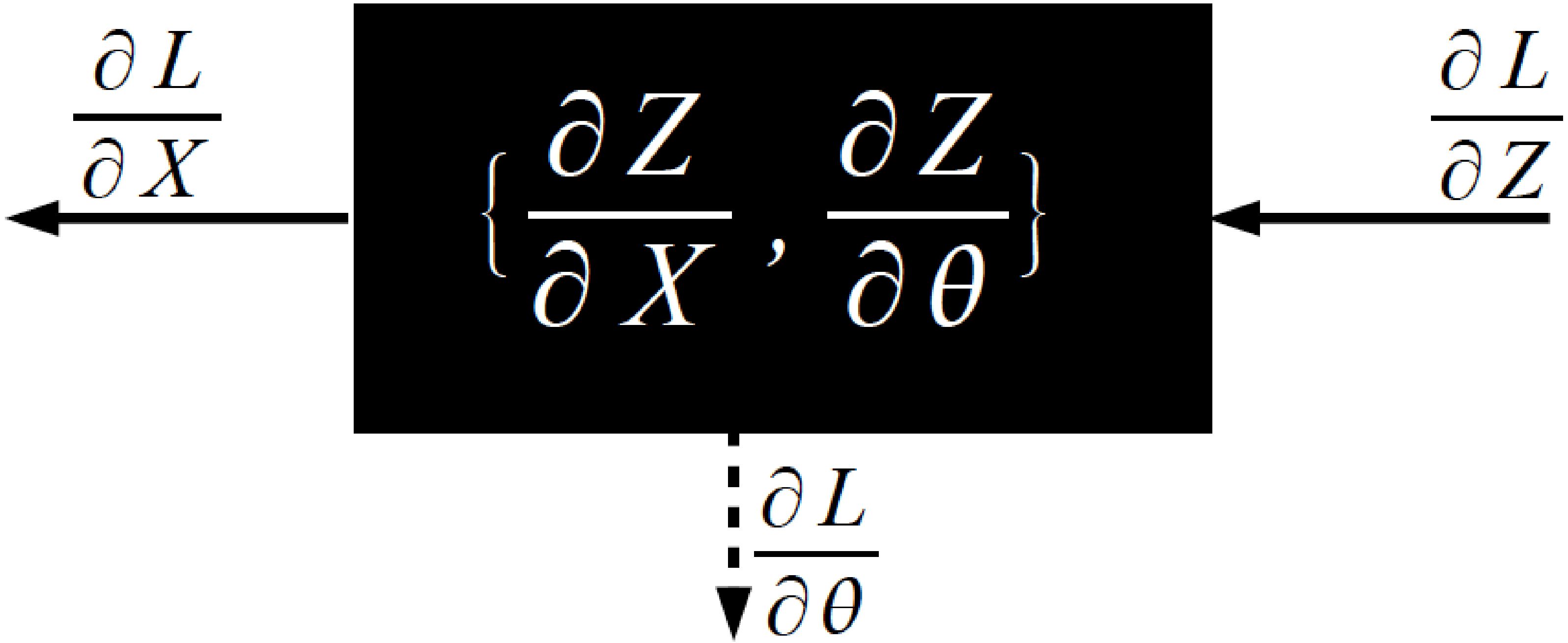
Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Key Computation: Forward-Prop



Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Key Computation: Back-Prop



$\frac{\partial L}{\partial X}$

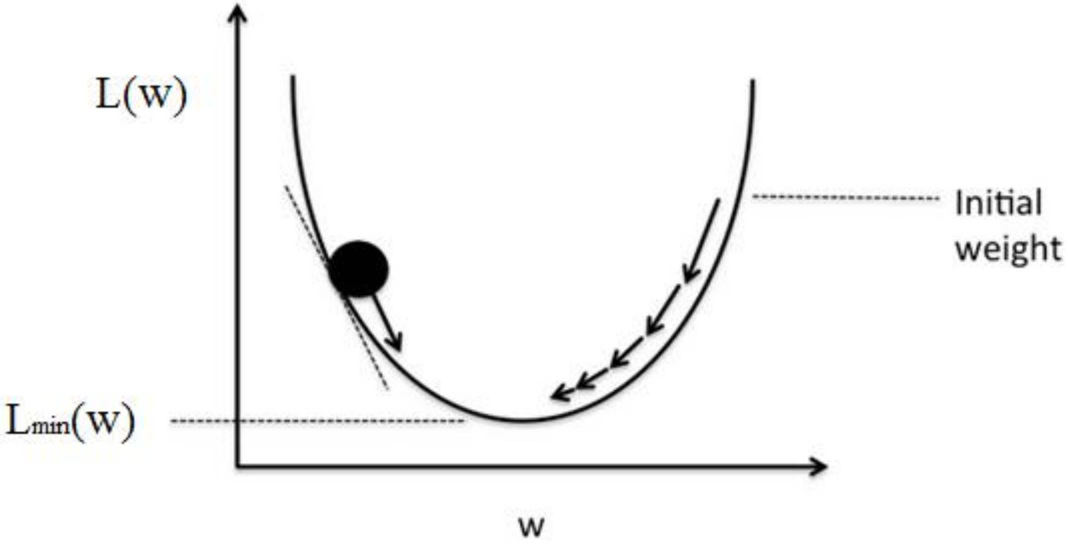
$\{\frac{\partial Z}{\partial X}, \frac{\partial Z}{\partial \theta}\}$

$\frac{\partial L}{\partial Z}$

$\frac{\partial L}{\partial \theta}$

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

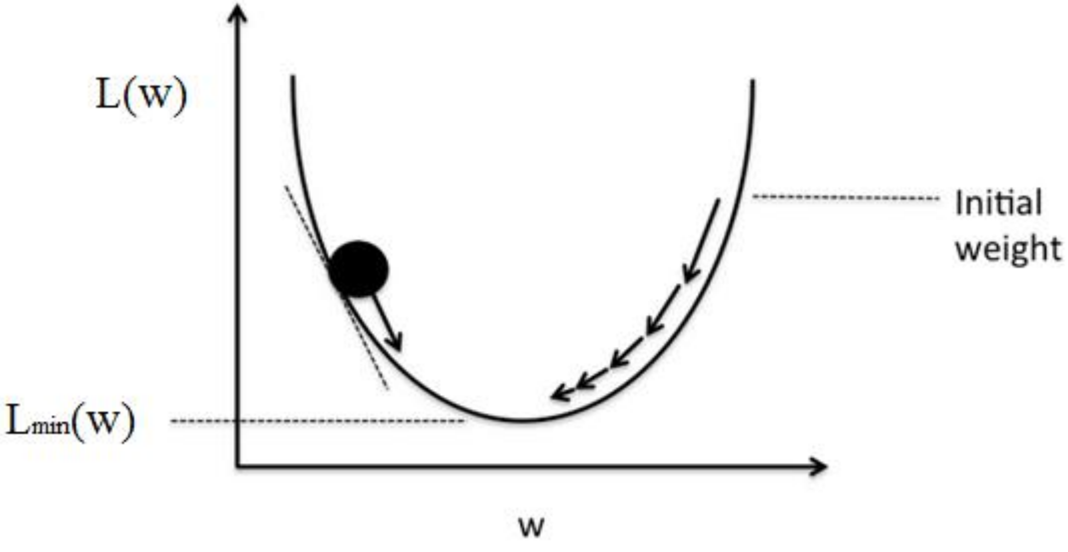
Training using Stochastic Gradient Descent



Schematic of gradient descent.

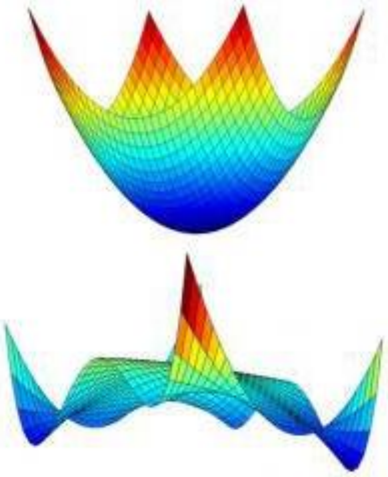
$$W \coloneqq W - \mu \nabla L$$

Training using Stochastic Gradient Descent



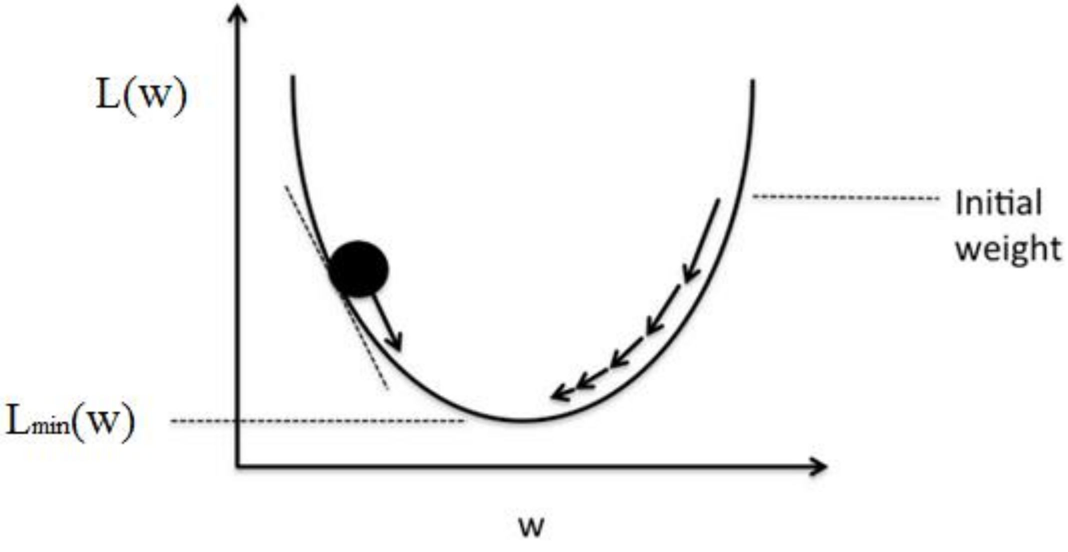
Schematic of gradient descent.

$$W := W - \mu \nabla \mathrm{L}$$



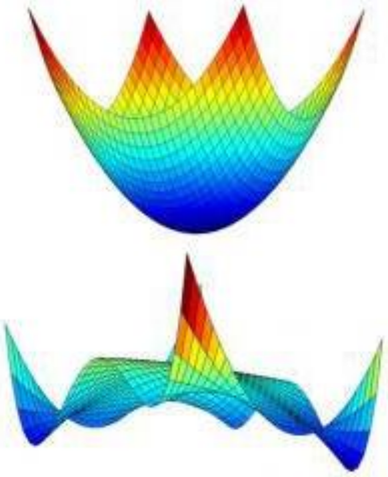
Loss functions of NN are almost always non-convex

Training using Stochastic Gradient Descent



Schematic of gradient descent.

$$W := W - \mu \nabla L$$

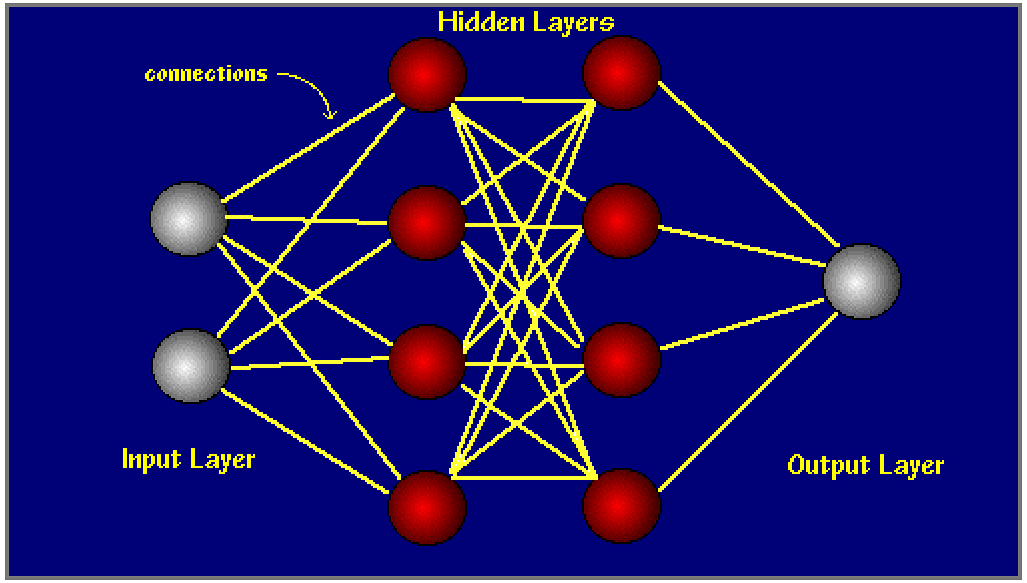


Loss functions of NN are almost always non-convex which makes training a little tricky.

Many methods to find the optimum, like momentum update, Nesterov momentum update, Adagrad, RMSPRop, etc

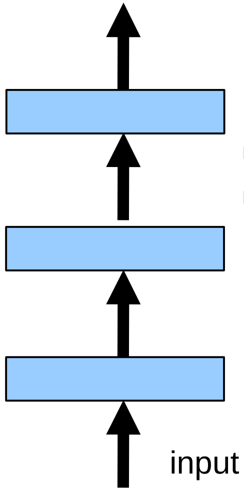
Network

- A network is a set of layers and its connections.
- Data and gradients move along the connections.
- Feed forward networks are Directed Acyclic graphs (DAG) i.e. they do not have any recurrent connections.



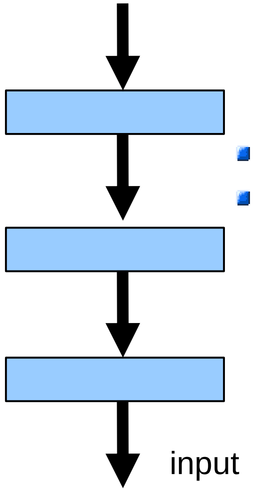
Main types of deep architectures

feed-forward



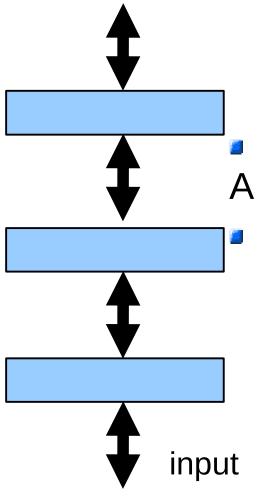
- Neural nets
- Conv Nets

Feed-back



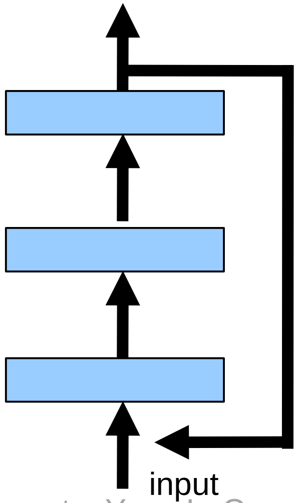
- Hierar. Sparse Coding
- Deconv Nets

Bi-directional



- Stacked Auto-encoders
- DBM

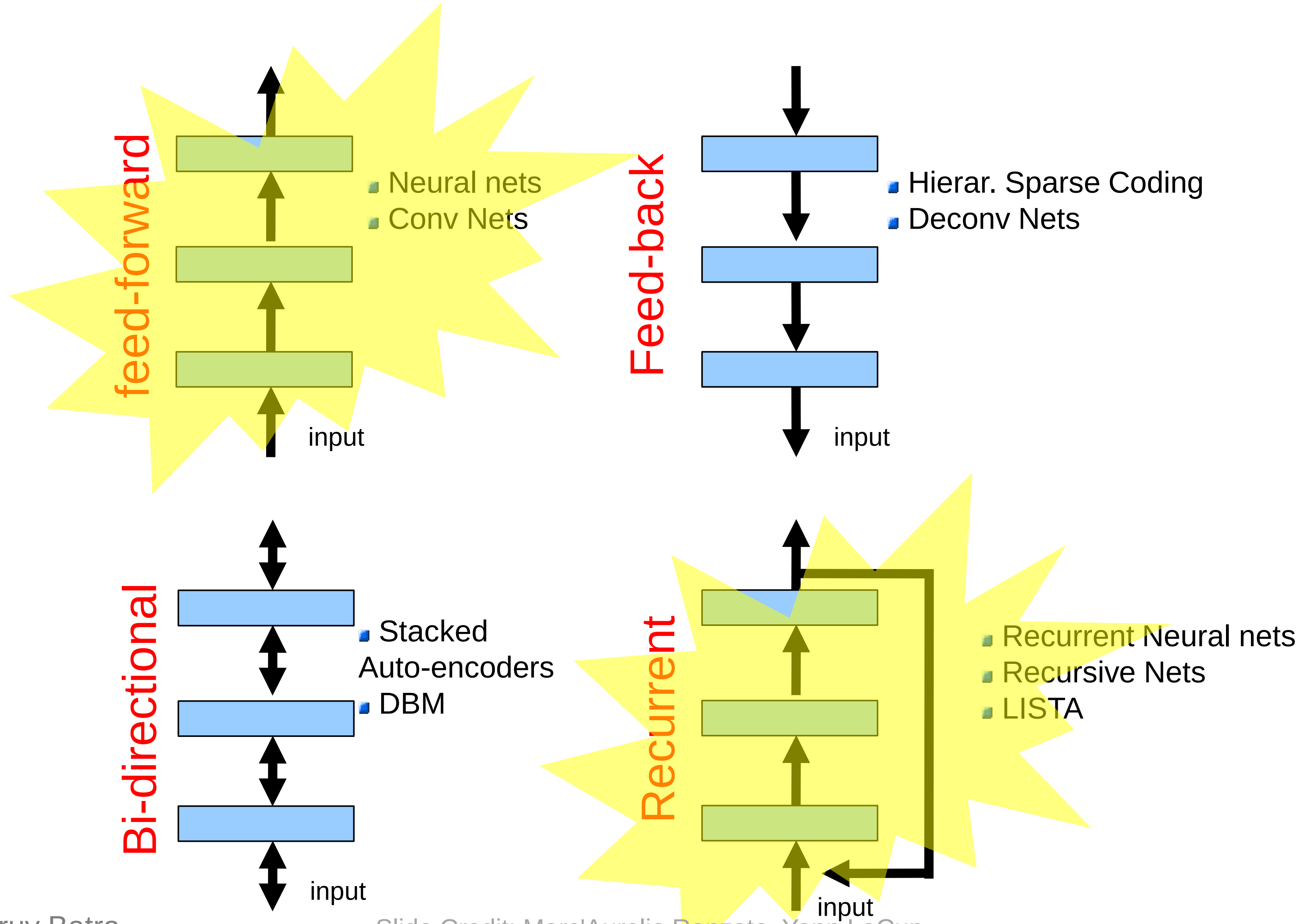
Recurrent



- Recurrent Neural nets
- Recursive Nets
- LISTA

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Focus of this course



feed-forward

- Neural nets
- Conv Nets

input

Feed-back

- Hierar. Sparse Coding
- Deconv Nets

input

Bi-directional

- Stacked Auto-encoders
- DBM

input

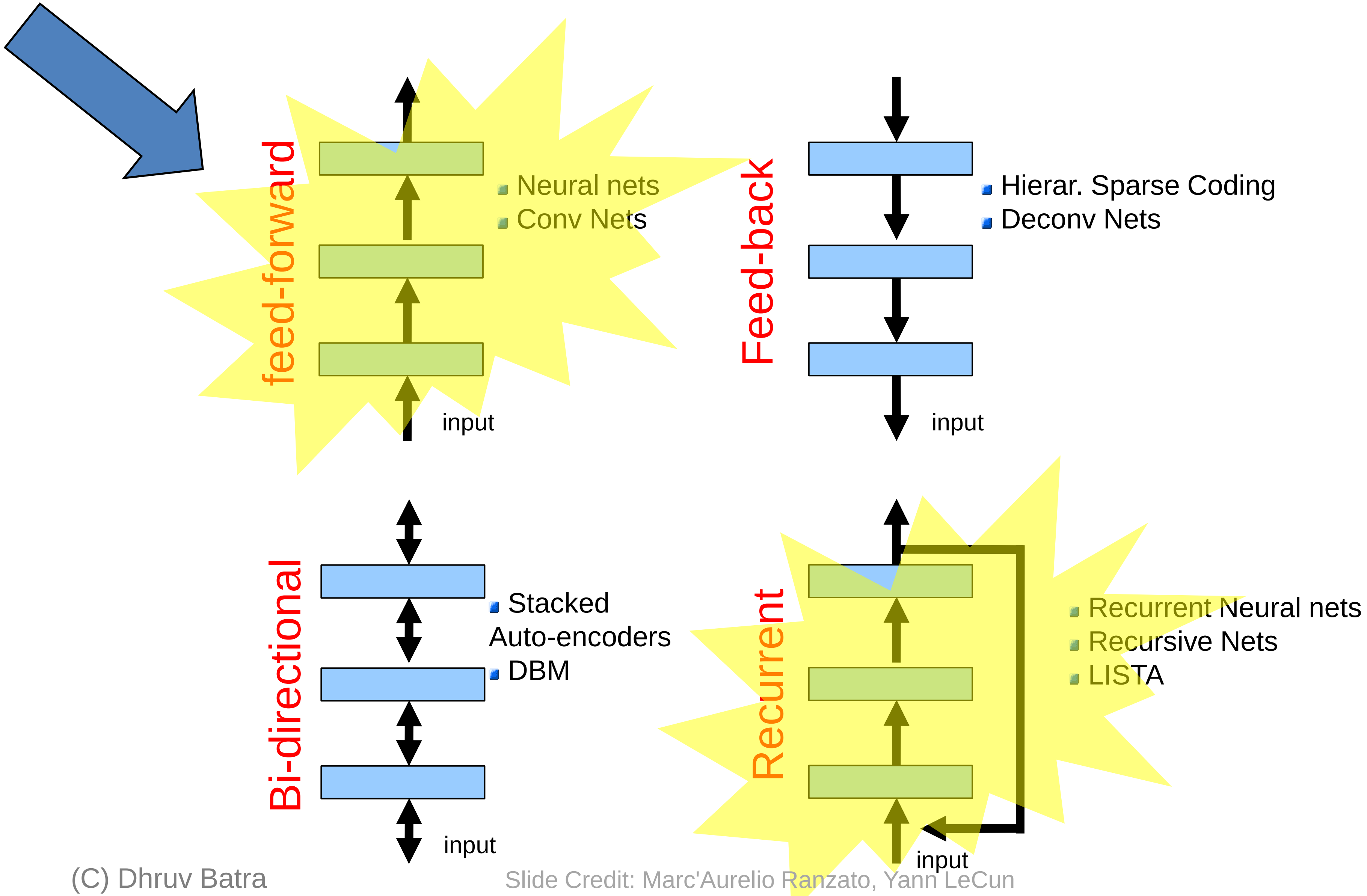
Recurrent

- Recurrent Neural nets
- Recursive Nets
- LISTA

input

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Focus of this class



- Neural nets
- Conv Nets

- Hierar. Sparse Coding
- Deconv Nets

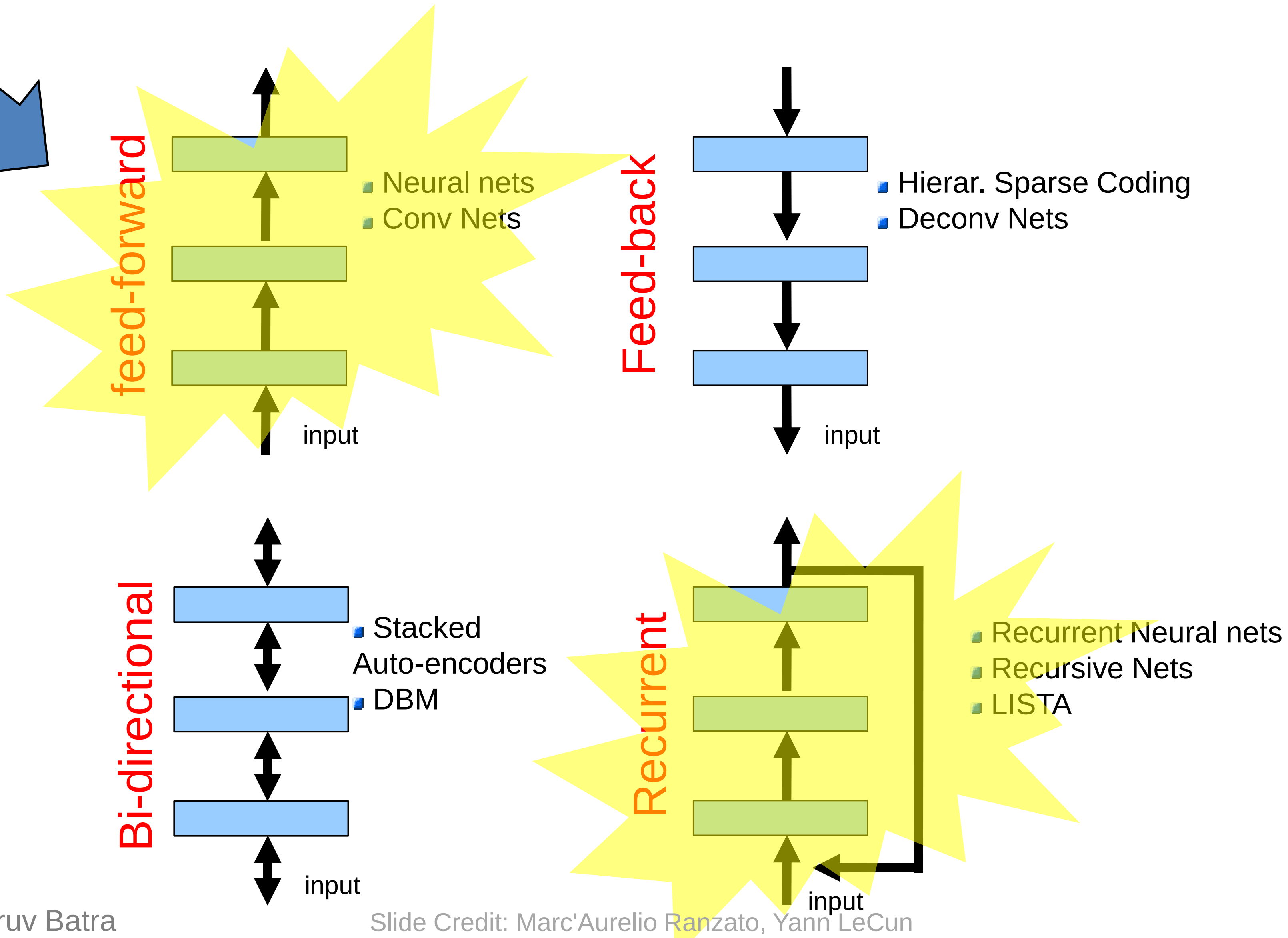
- Stacked Auto-encoders
- DBM

- Recurrent Neural nets
- Recursive Nets
- LISTA

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Focus of this class

Why?
Because official CAFFE release supports DAG



- Neural nets
- Conv Nets

- Hierar. Sparse Coding
- Deconv Nets

- Stacked Auto-encoders
- DBM

- Recurrent Neural nets
- Recursive Nets
- LISTA

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Outline

- Caffe?
- Installation
- Key Ingredients
- Example: Softmax Classifier
- Pycaffe
- Roasting
- Resources
- References

What is Caffe?

Open framework, models, and worked examples
for deep learning

- 1.5 years
- 450+ citations, 100+ contributors
- 2,500+ forks, >1 pull request / day average
- focus has been vision, but branching out:
  sequences, reinforcement learning, speech + text

Prototype

Train

Deploy

What is Caffe?

Open framework, models, and worked examples

for deep learning

- Pure C++ / CUDA architecture for deep learning
- Command line, Python, MATLAB interfaces
- Fast, well-tested code
- Tools, reference models, demos, and recipes
- Seamless switch between CPU and GPU

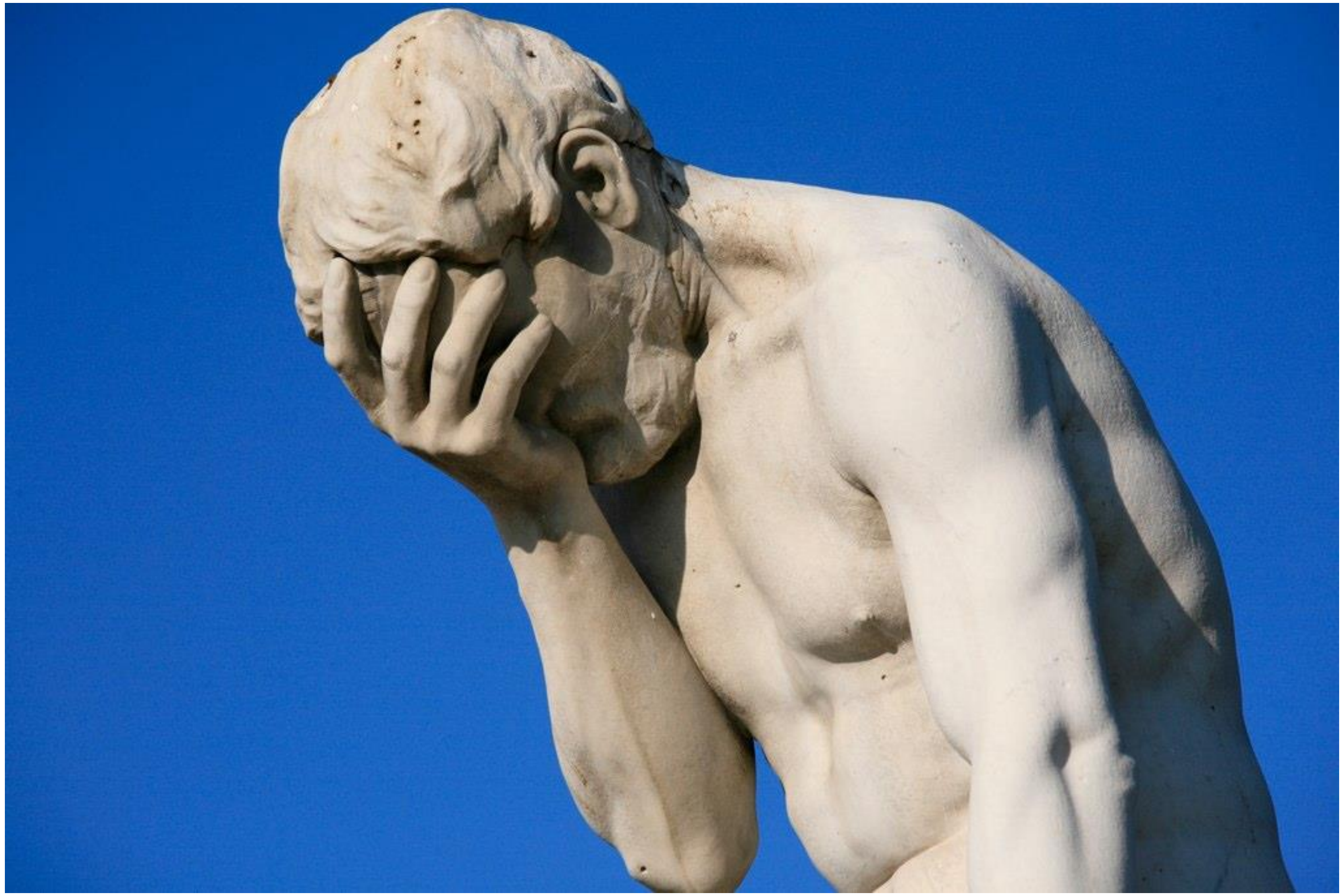
Prototype

Train

Deploy

Installation

Installation



Installation

- Strongly recommended that you use Linux (Ubuntu)/ OS X. Windows has some unofficial support though.
- Prior to installing look at the installation page and the wiki

  - the wiki has more info. But all support needs to be taken with a pinch of salt

  - lots of dependencies
- Suggested that you back up your data!

Installation

- **CUDA** (Compute Unified Device Architecture) is a parallel computing platform and application programming interface (API) model created by NVIDIA
- Installing CUDA

  – check if you have a cuda supported Graphics Processing Unit (GPU).

  If not, go for a cpu only installation of CAFFE.

  - Do not install the nvidia driver if you do not have a supported GPU

Installation

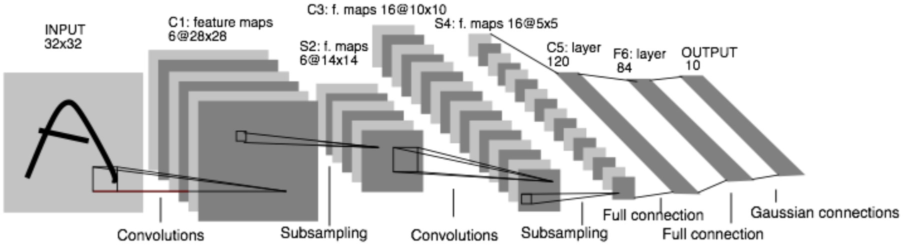
- Clone the repo from here
- Depending on the system configuration, make modifications to the **Makefile.config** file and proceed with the installation instructions.
- We suggest that you use Anaconda python for the installation as it comes with the necessary python packages.

Quick Questions?

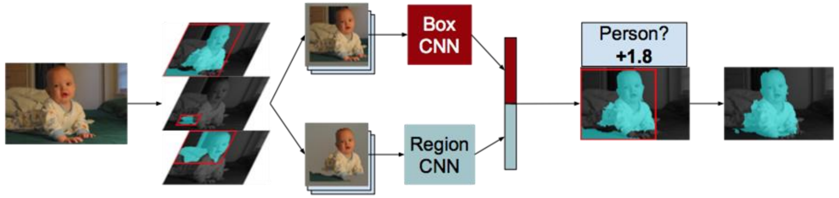
Key Ingredients

DAG

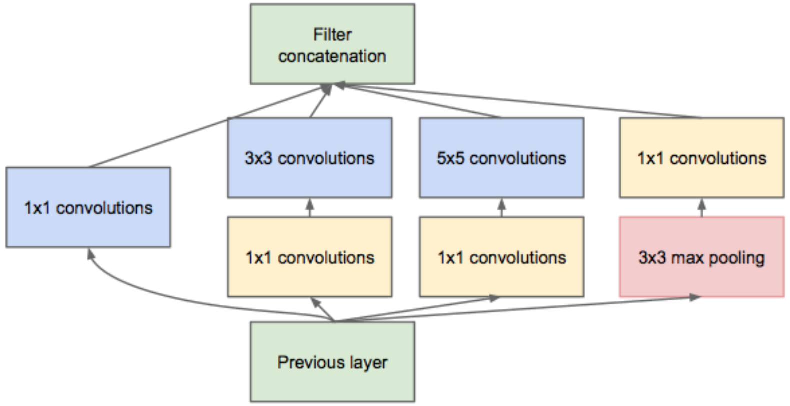
Many current deep models have linear structure



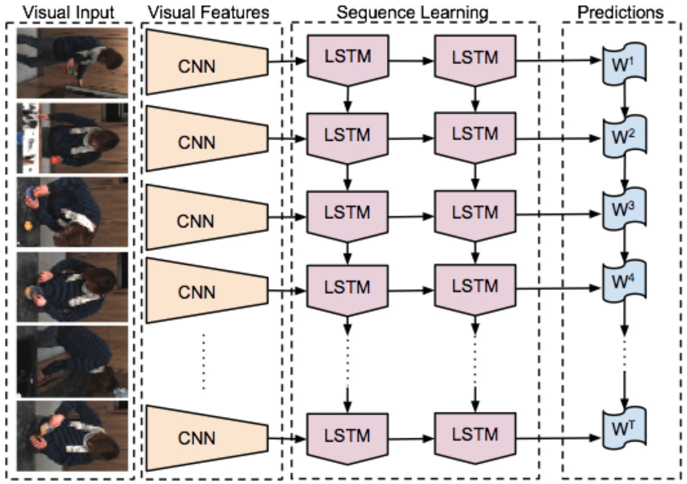
Caffe nets can have any directed acyclic graph (DAG) structure.



SDS two-stream net



GoogLeNet Inception Module

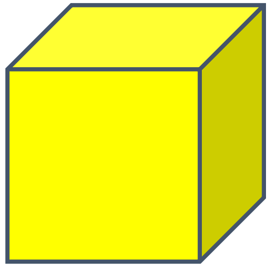


LRCN joint vision-sequence model

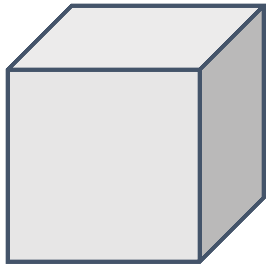
Blob

Blobs are N-D arrays for storing and communicating information.

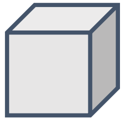
- hold data, derivatives, and parameters
- lazily allocate memory
- shuttle between CPU and GPU



Data
*N*umber x *K* Channel x *H*eight x *W*idth
256 x 3 x 227 x 227 for ImageNet train input

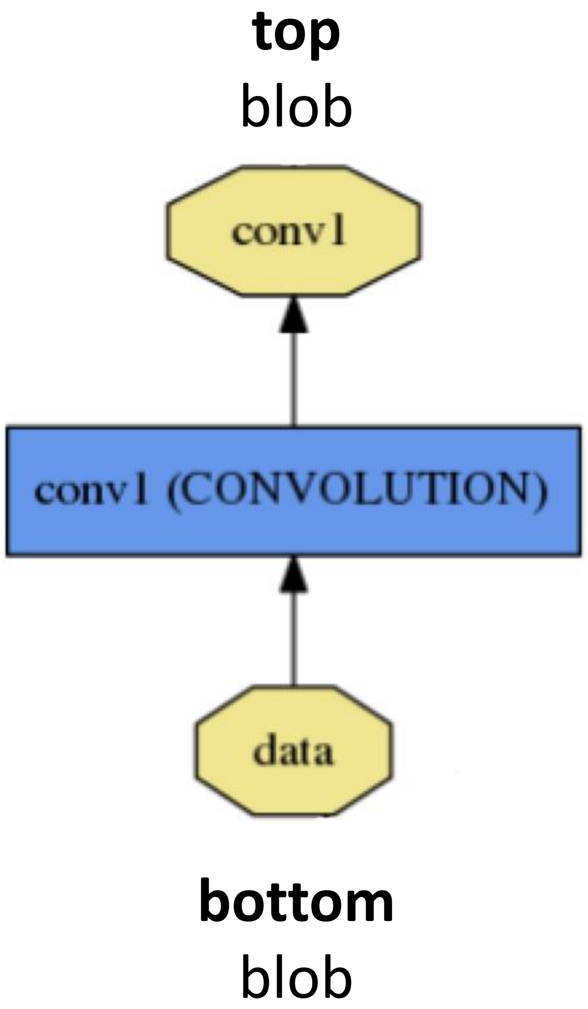


Parameter: Convolution Weight
N Output x *K* Input x *H*eight x *W*idth
96 x 3 x 11 x 11 for CaffeNet conv1



Parameter: Convolution Bias
96 x 1 x 1 x 1 for CaffeNet conv1

```
name: "conv1"
type: CONVOLUTION
bottom: "data"
top: "conv1"
… definition …
```



Layer Protocol

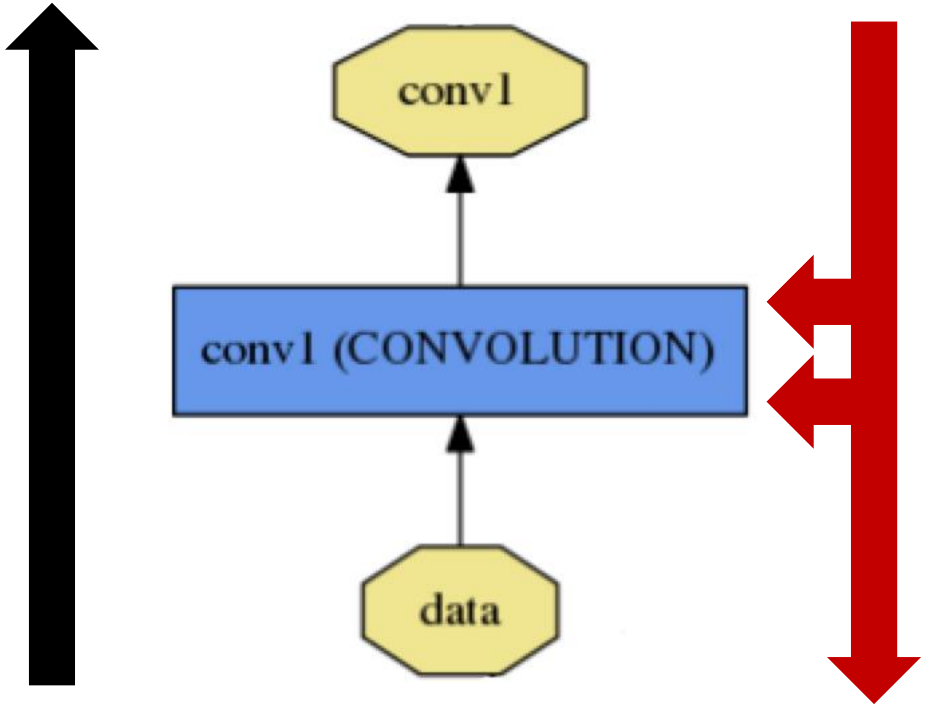
Setup: run once for initialization.

Forward: make output given input.

Backward: make gradient of output
- w.r.t. bottom
- w.r.t. parameters (if needed)

Reshape: set dimensions.

Compositional Modeling
The Net’s forward and backward passes are composed of the layers’ steps.



[Layer Development Checklist](#)

Layers

- Caffe divides layers into

  - neuron layers (eg: Inner product),

  - Vision layers (Convolutional, pooling,etc)

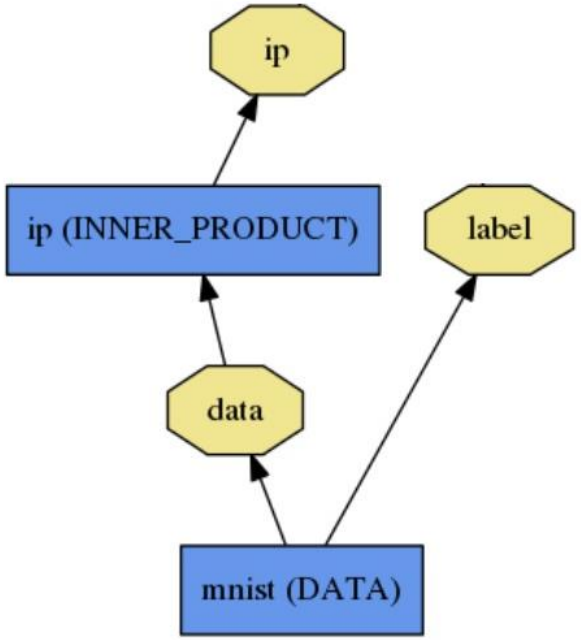
  - Data layers (to read in input)

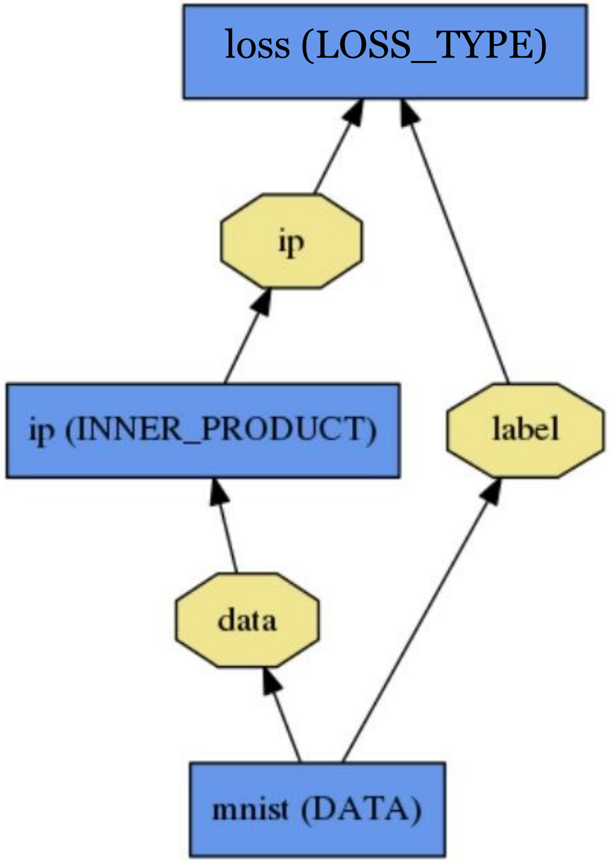
  - Loss layers

- You can write your own layers. More development guidelines are here

Loss

What kind of model is this?





Define the task by the **loss**.

Classification
`SoftmaxWithLoss`
`HingeLoss`

Linear Regression
`EuclideanLoss`

Attributes / Multiclassificat
`SigmoidCrossEntropyLoss`

Others…

New Task
`NewLoss`

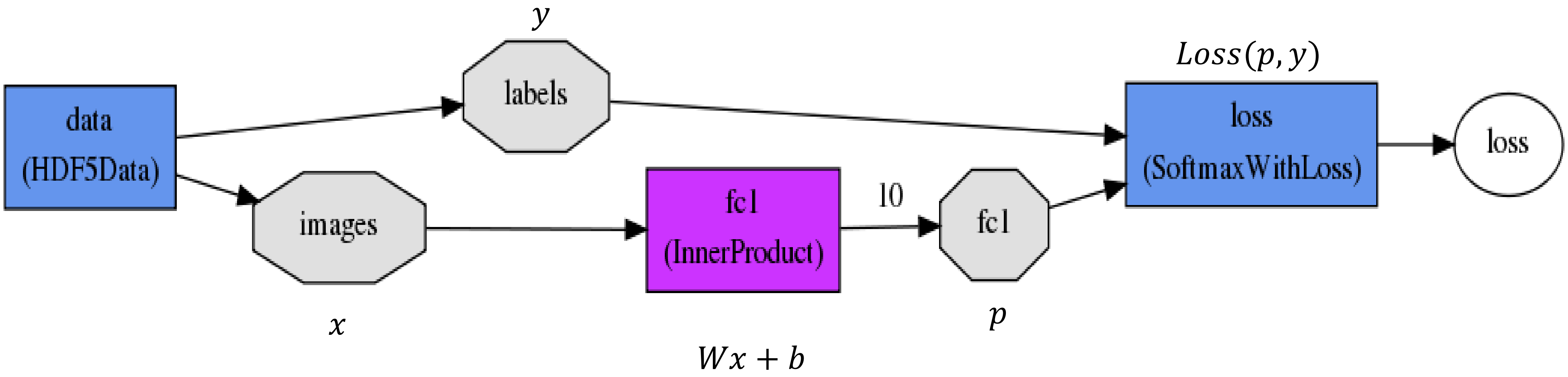
Protobuf Model Format

- Strongly typed format
- Auto-generates code
- Developed by Google
- Defines Net / Layer / Solver schemas in **caffe.proto**

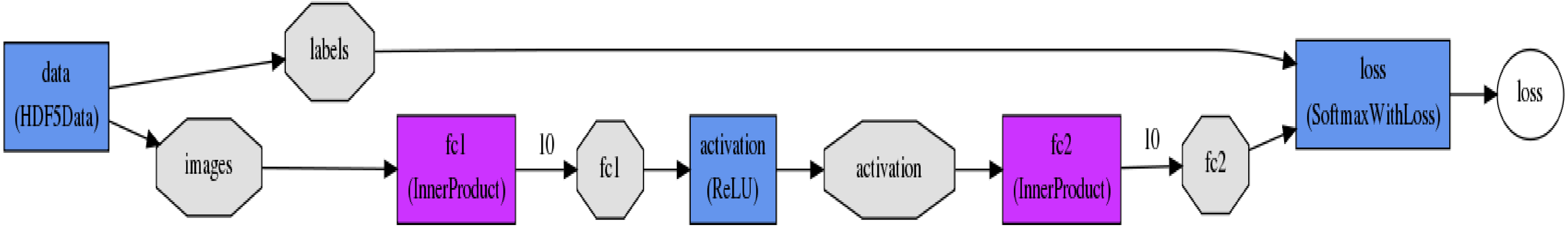
```
message ConvolutionParameter {
 // The number of outputs for the layer
 optional uint32 num_output = 1;
 // whether to have bias terms
 optional bool bias_term = 2 [default = true];
}
```

```
layer {
 name: "ip"
 type: "InnerProduct"
 bottom: "data"
 top: "ip"
 inner_product_param {
  num_output: 2
 }
}
```

Softmax Classifier

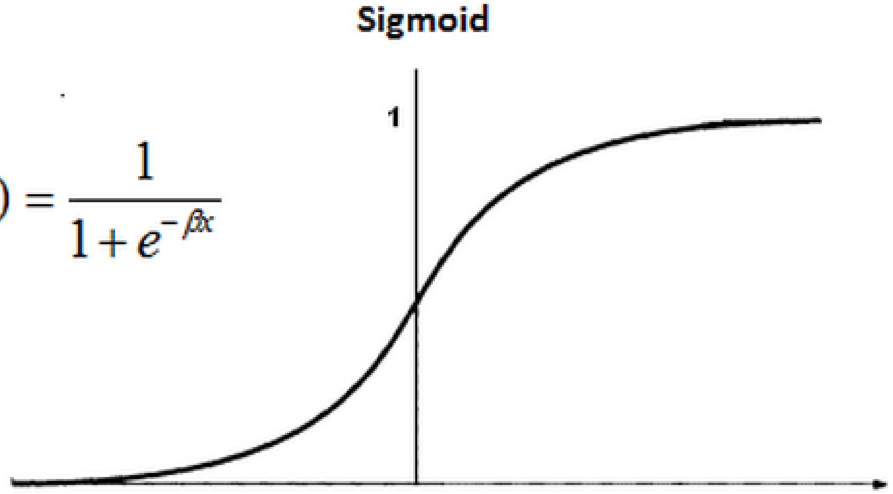


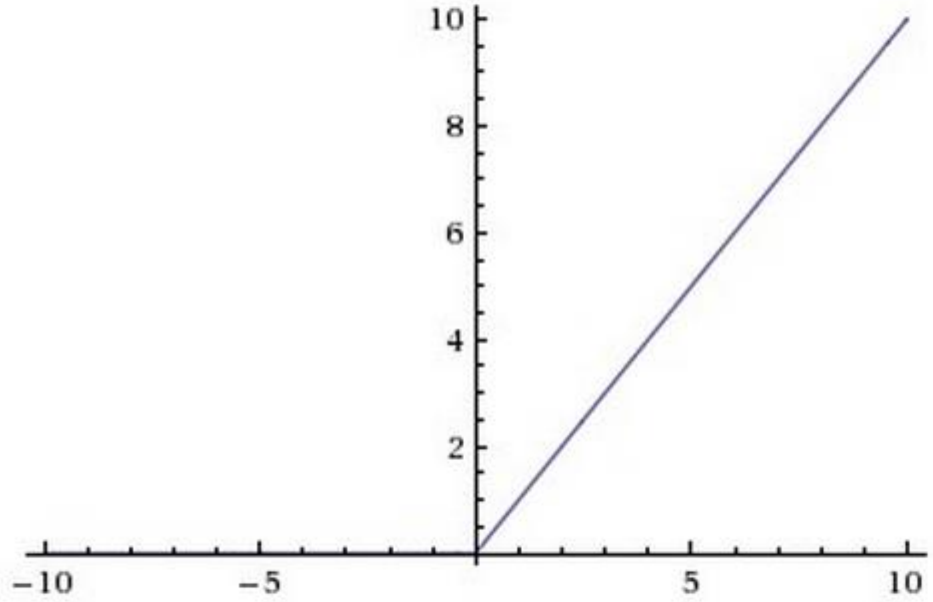
Neural Network



Activation function

$$f(x) = \frac{1}{1+e^{-\beta x}}$$





Rectified Linear Unit (ReLU) Activation

Recipe for brewing a net

- **Convert the data to caffe-supported format**
  LMDB, HDF5, list of images
- Define the net
- Configure the solver
- Start train from supported interface (command line, python, etc)

Layers – Data Layers

- Data Layers : gets data into the net

- Data: LMDB/LEVELDB
efficient way to input data, only for 1-of-k classification tasks

- HDF5Data: takes in HDF5 format
- easy to create custom non-image datasets but supports only float32/float64

- Data can be written easily in the above formats using python support. (using lmdb and h5py respectively). We will see how to write hdf5 data shortly

- Image Data: Reads in directly from images. Can be a little slow.

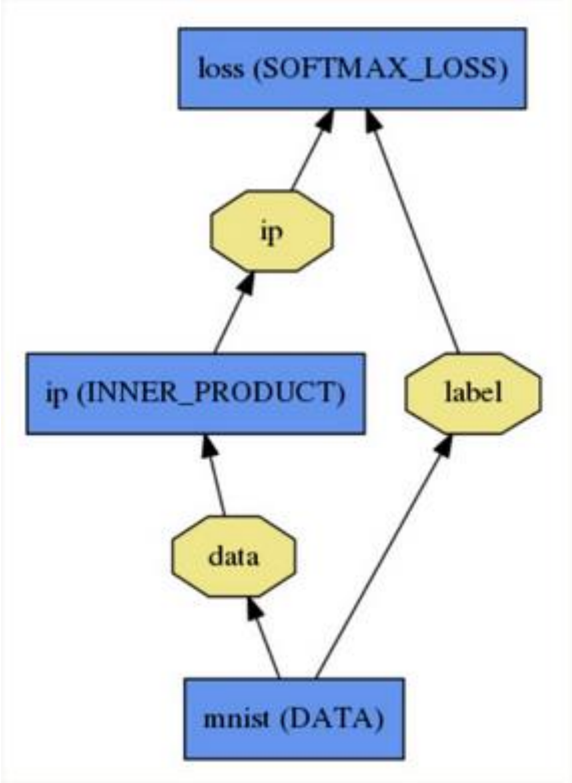
- All layers (except hdf5) support standard data augmentation tasks

Recipe for brewing a net

- Convert the data to caffe-supported format
  LMDB, HDF5, list of images
- **Define the network/architecture**
- Configure the solver
- Start train from supported interface (command line, python, etc)

Example: Softmax Classifier

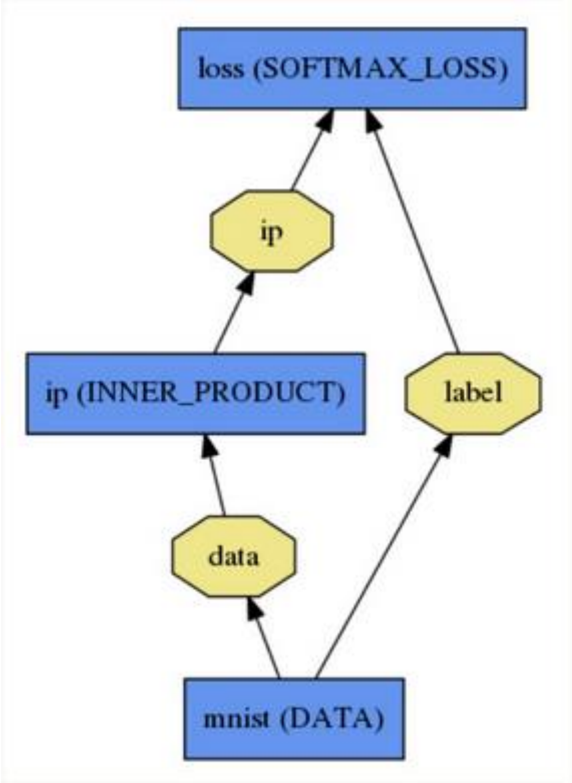
Architecture file



```
name: "LogReg"
layer {
  name: "mnist"
  type: "Data"
  top: "data"
  top: "label"
  data_param {
    source: "input_leveldb"
    batch_size: 64
  }
}
```

Example: Softmax Classifier

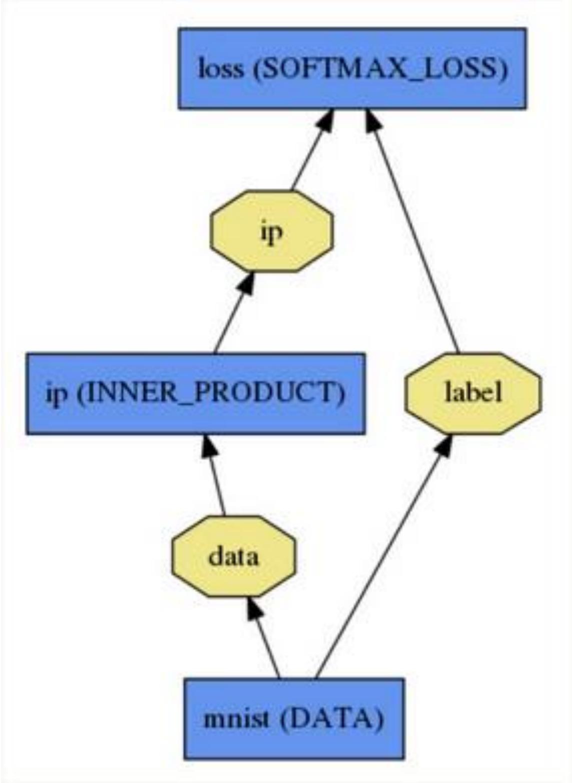
Architecture file



```
name: "LogReg"
layer {
  name: "mnist"
  type: "Data"
  top: "data"
  top: "label"
  data_param {
    source: "input_leveldb"
    batch_size: 64
  }
}
layer {
  name: "ip"
  type: "InnerProduct"
  bottom: "data"
  top: "ip"
  inner_product_param {
    num_output: 2
  }
}
```

Example: Softmax Classifier

Architecture file



```
name: "LogReg"
layer {
 name: "mnist"
 type: "Data"
 top: "data"
 top: "label"
 data_param {
  source: "input_leveldb"
  batch_size: 64
 }
}
layer {
 name: "ip"
 type: "InnerProduct"
 bottom: "data"
 top: "ip"
 inner_product_param {
  num_output: 2
 }
}
layer {
 name: "loss"
 type: "SoftmaxWithLoss"
 bottom: "ip"
 bottom: "label"
 top: "loss"
}
```

Recipe for brewing a net

- Convert the data to caffe-supported format
  LMDB, HDF5, list of images
- Define the net
- **Configure the solver**
- Start train from supported interface (command line, python, etc)

Example: Softmax Classifier

Solver file

```
net: "logreg_train_val.prototxt”
test_iter: 10
test_interval: 500
base_lr: 0.0000001
momentum: 0.0
weight_decay: 50000
lr_policy: "step”
stepsize: 2000
display: 100
max_iter: 2000
snapshot: 1000
snapshot_prefix: "logreg-snapshot/”
solver_mode: GPU
```

Example: Softmax Classifier

Solver file

```
net: "logreg_train_val.prototxt”
test_iter: 10
test_interval: 500
base_lr: 0.0000001
momentum: 0.0
weight_decay: 50000
lr_policy: "step”
stepsize: 2000
display: 100
max_iter: 2000
snapshot: 1000
snapshot_prefix: "logreg-snapshot/”
solver_mode: GPU
```

CAFFE has many common solver methods:

- SGD
- Adagrad
- RMSProp
- Nesterov Momentum, etc

More details in this page

Recipe for brewing a net

- Convert the data to caffe-supported format
  LMDB, HDF5, list of images
- Define the net
- Configure the solver
- **Train from supported interface (command line, python, etc)**

Softmax Classifier Demo

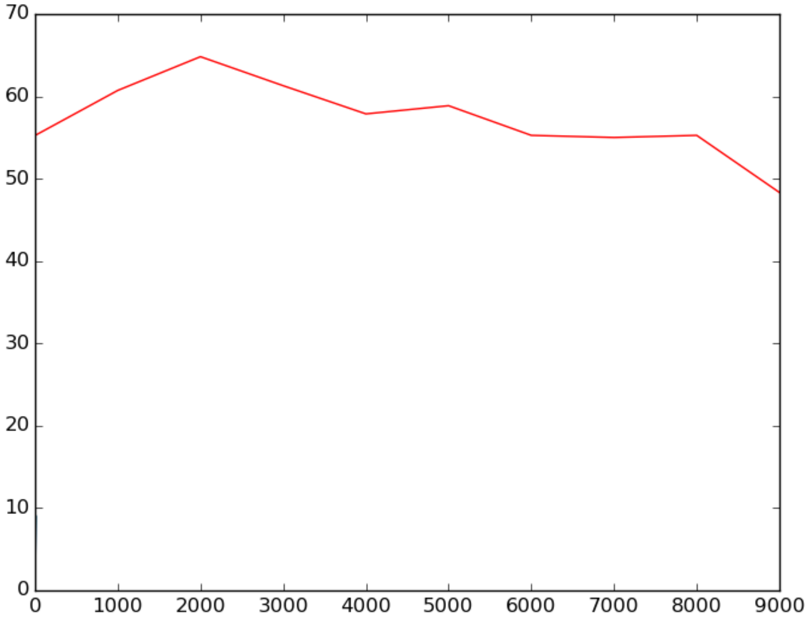
Command line interface

< Ipython notebook>

Pycaffe Demo

Softmax Classifier example on pycaffe

Need for tuning Hyper - parameters



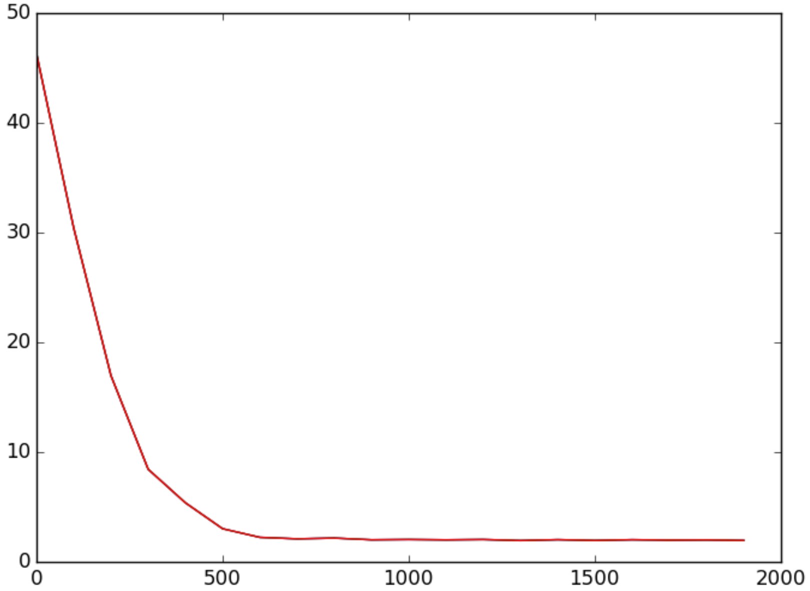


Figure on the left has a high learning rate and the loss on the training set does not converge. When hyper-parameters like learning rate and weight-decay are tuned, the loss decreases rapidly as shown in the figure on the right.

Logging

- It is use full to generate a log file where caffe dumps values like training loss, iteration number, norm of the weights of each blob, etc.
- Parse log file to obtain useful hints about training process
  - see caffe/tools/extra/parse_log.py
- The above is a generic function. Custom log parsing can be created by you keeping the above as an example.

Log Parse Demo

Pycaffe Demo

- pycaffe to visualize weights of a pre-trained model
- Model Zoo has pretrained models of deep learning architectures like alexnet
- Running a forward pass to

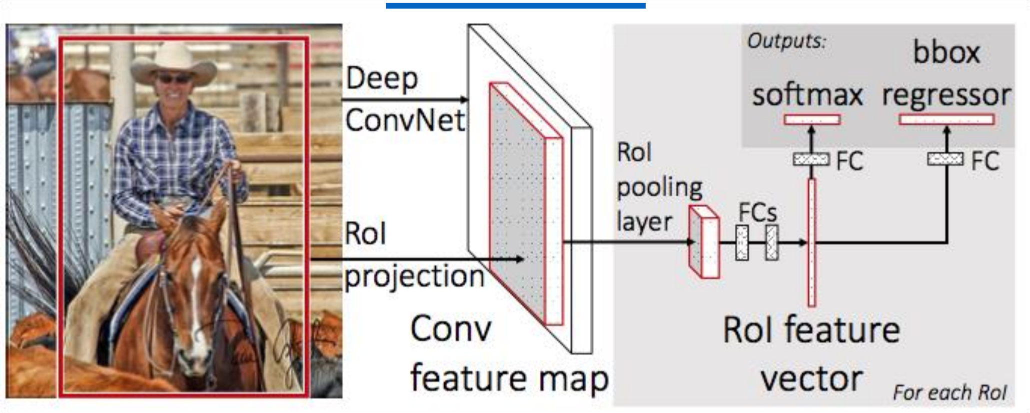
  - predict class

Pycaffe documentation is sparse!

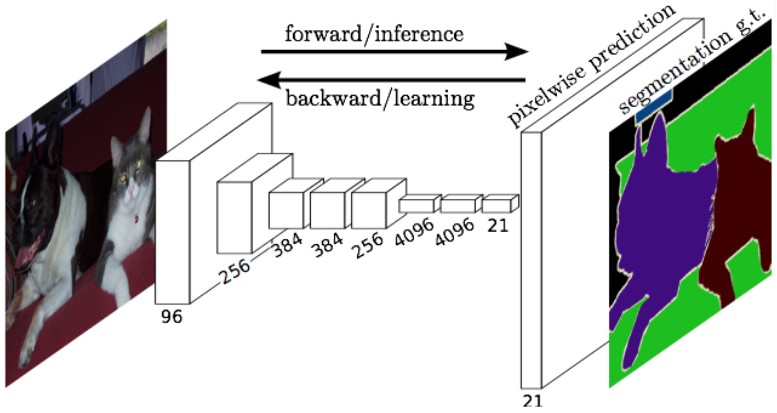
Looking at examples and reading code is inevitable if you want to make the best use of CAFFE!

Up Next The Latest Roast

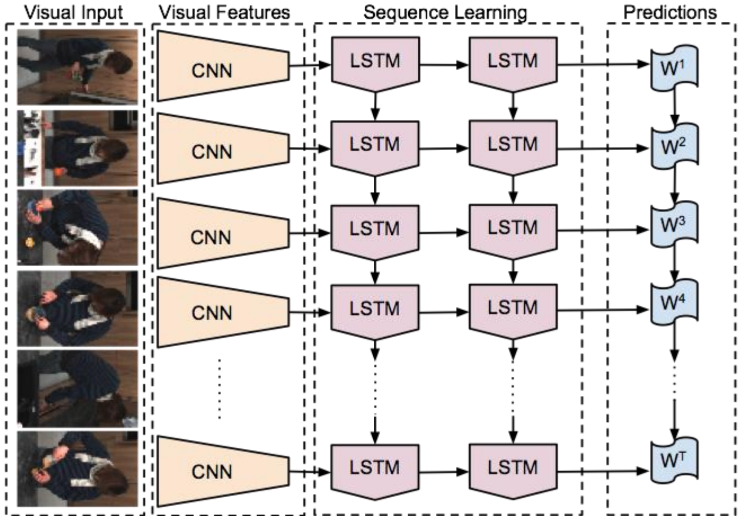
Detection



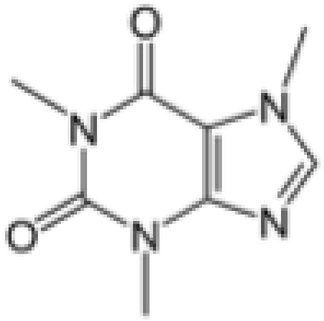
Pixelwise Prediction



Sequences



Framework Future



Resources

- Many examples are provided in the caffe-master/examples directory
- Ipython notebooks for common Neural network tasks like filter visualization, fine-tuning, etc
- Caffe-tutorials
- Caffe chat
- Caffe-users group
- Watch out for new features!

References

1. http://caffe.berkeleyvision.org/
2. DIY Deep Learning for Vision with Caffe

THANK YOU