Wifi: IMPACT DEEP / deep2017

Scaling Deep Learning with MXNet

Mu Li, Alex Smola and Joseph Spisak

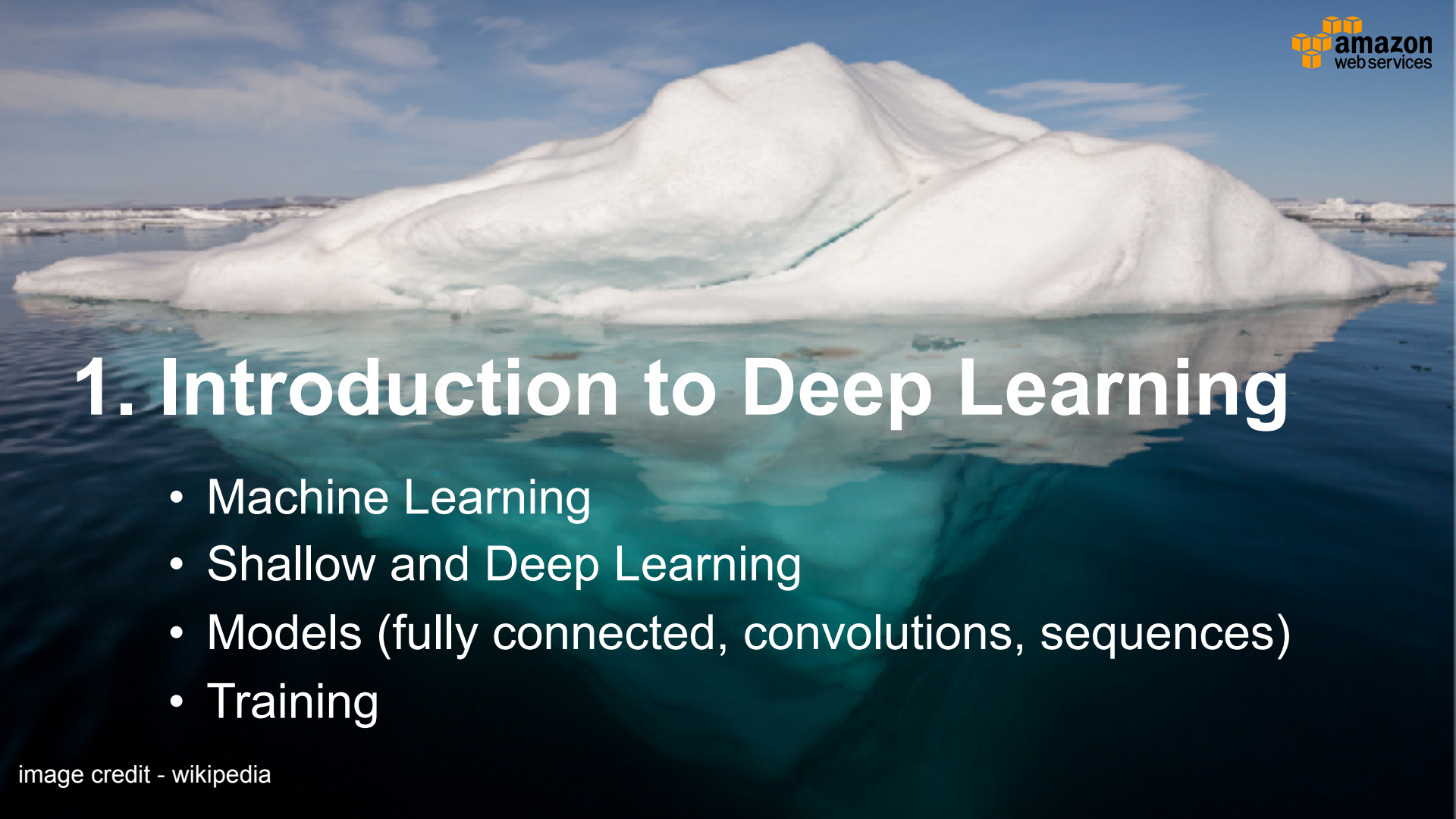
AWS Machine Learning

amazon web services

Modules

1. **Introduction to Deep Learning**
2. **Getting Started with MXNet**
   - Cloud - AMIs and CloudFormation Template
   - Laptop - build from source
3. **MXNet**
   - NDArrays and Symbols
   - Training Deep Networks
4. **Examples**
   - Computer Vision
   - Natural Language Processing
   - Recommender Systems

amazon web services



1. Introduction to Deep Learning

- Machine Learning
- Shallow and Deep Learning
- Models (fully connected, convolutions, sequences)
- Training

image credit - wikipedia

Programming FizzBuzz

Code

```
var o='';
for (var i=1; i<=100; i++) {
    i%3 || (o+='Fizz ');
    i%5 || (o+='Buzz ');
    !(i%3 && i%5) || (o+=(i+' '));
}
console.log(o);
```

Programming with Data

- **Generate training data**
  (9,'Fizz'),(10,'Buzz'),(2,'2'),
  (15,'Fizz Buzz') …
- **Extract input features**
  x -> (x%3, x%5, x%15)
- **Train a classifier**
  mapping input x to output y
  using training data

Inspired by joelgrus.com

That was silly.
Why would you ever do this?

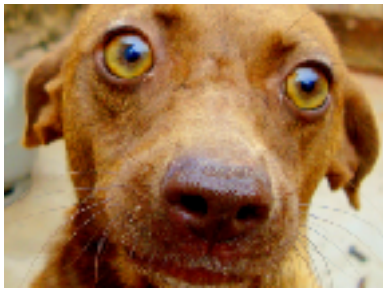
Programming with Data

- Given some input x we want to estimate output y
  - If it's a simple deterministic rule, probably don't bother
  - Often quite complicated, vague but lots of examples
  - Impossible to write code but can train estimator

cat

cat

dog



dog

image credit - wikipedia

Machine Learning 101

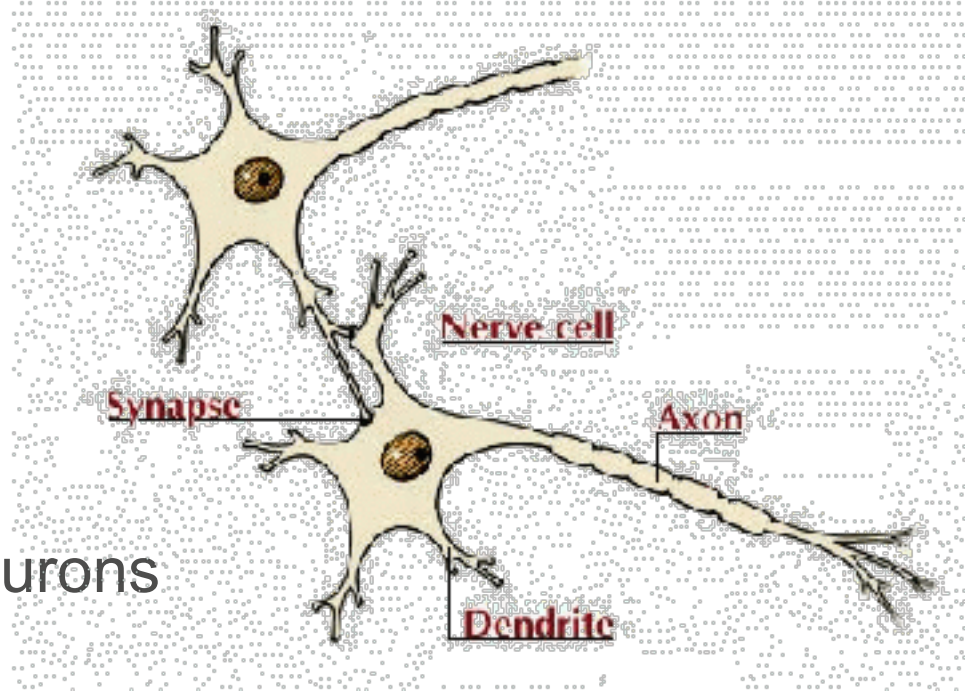
- **Training data** (patterns, labels)
  - Images (pictures / cat, dog, dinosaur, chicken, human)
  - Text (email / spam, ham)
  - Sound (recording / recognized speech)
  - Video (sequence / break-in, OK)
  - Images (X-ray image / size of cancer)
- **Loss function**
- **Estimator**
  Learns mapping data into outputs

	Edible	Poisonous
Truffle	0	10
Death cap	1,000,000	0

Machine Learning 101

- **Training Error**
  Error on the dataset that estimator uses to find **f**
- **Test Error**
  Error on a new, unseen test set. **This can be way off.**
  Due to overfitting. E.g. see predictions for election.
- **Training**
  Find parameters for **f**, e.g. by loss minimization
- **Model Selection**
  Adjust tuning parameters, network layers, trees, kernels …

Real Neural Networks



Soma (CPU)
Cell body - combines signals

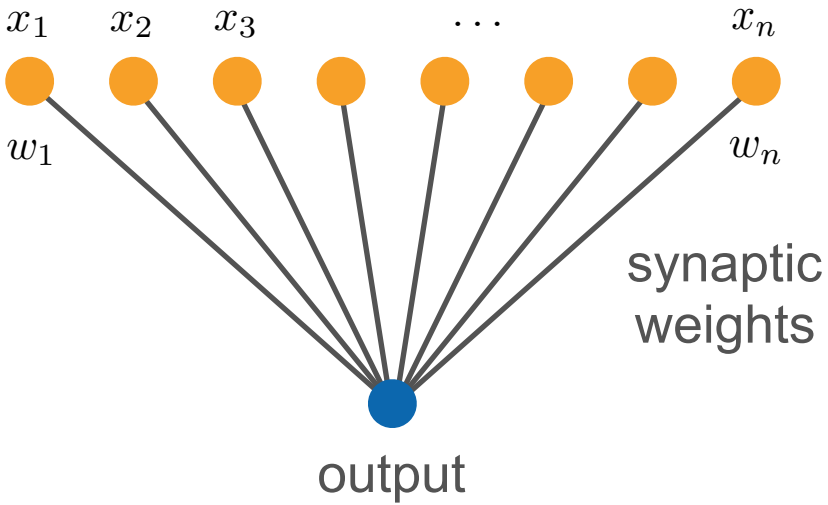
Dendrite (input bus)
Combines inputs from other neurons

Synapse (interface)
Interface and **parameter store** between neurons

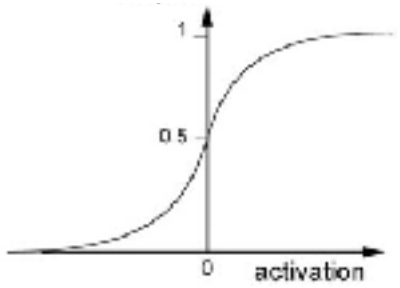
Axon (cable)
Up to 1m long. Send activation signal other neurons

image credit - wikipedia

Artificial Neuron

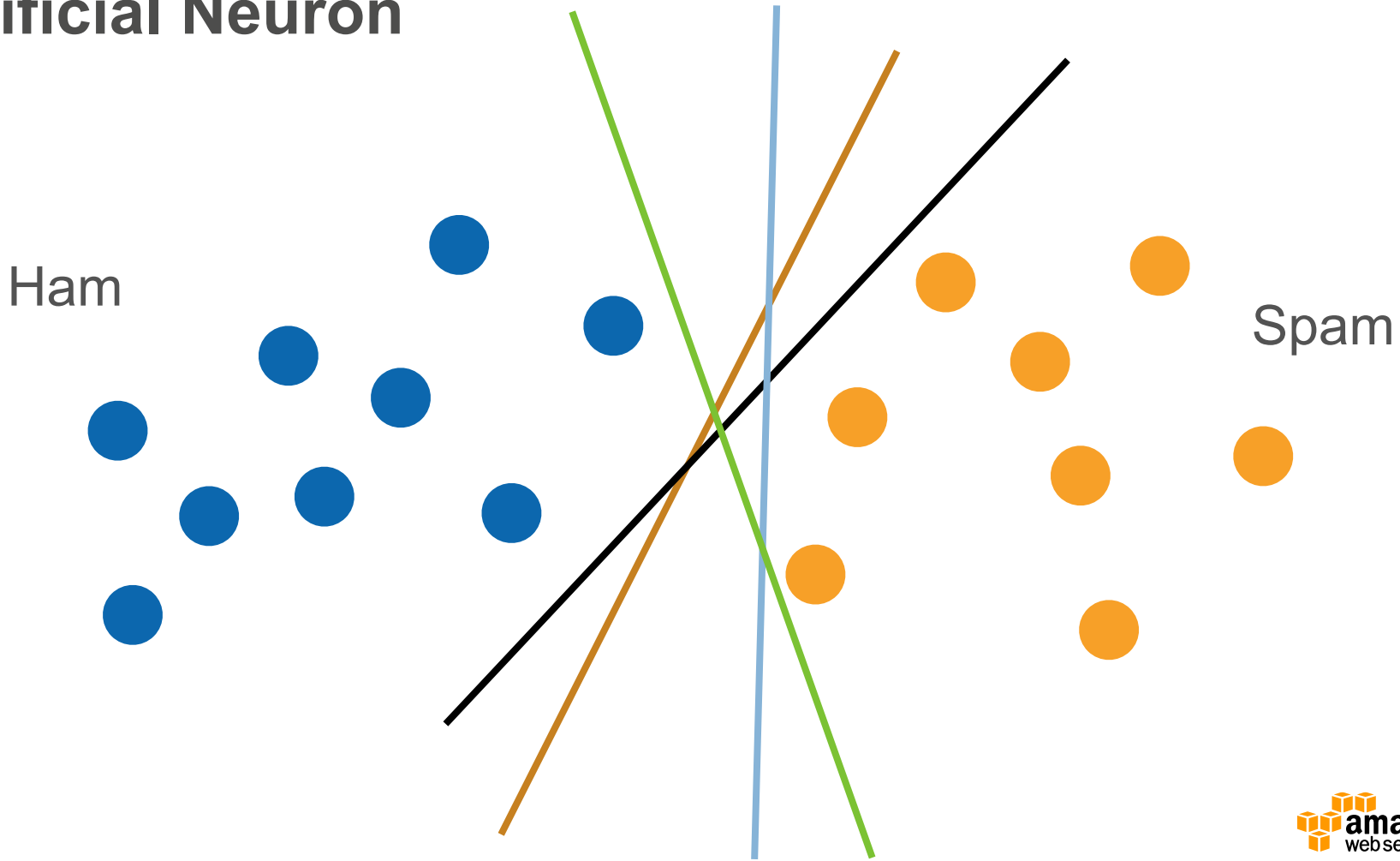


- **Input**
  Vector of training data x
- **Output**
  Linear function of inputs
- **Nonlinearity**
  Transform output into desired range of values, e.g. for classification we need probabilities [0, 1]
- **Training**
  Learn the weights w and bias b



$$f(x) = \sigma\left(\langle w, x \rangle + b\right)$$

Artificial Neuron



Machine Learning 101

Shallow Learning

- **Extract clever features** (preprocessing)
- **Map into feature space** (kernel methods)
- **Set of rules** (decision tree)
- **Combine multiple estimates** (boosting)

usually simple to learn

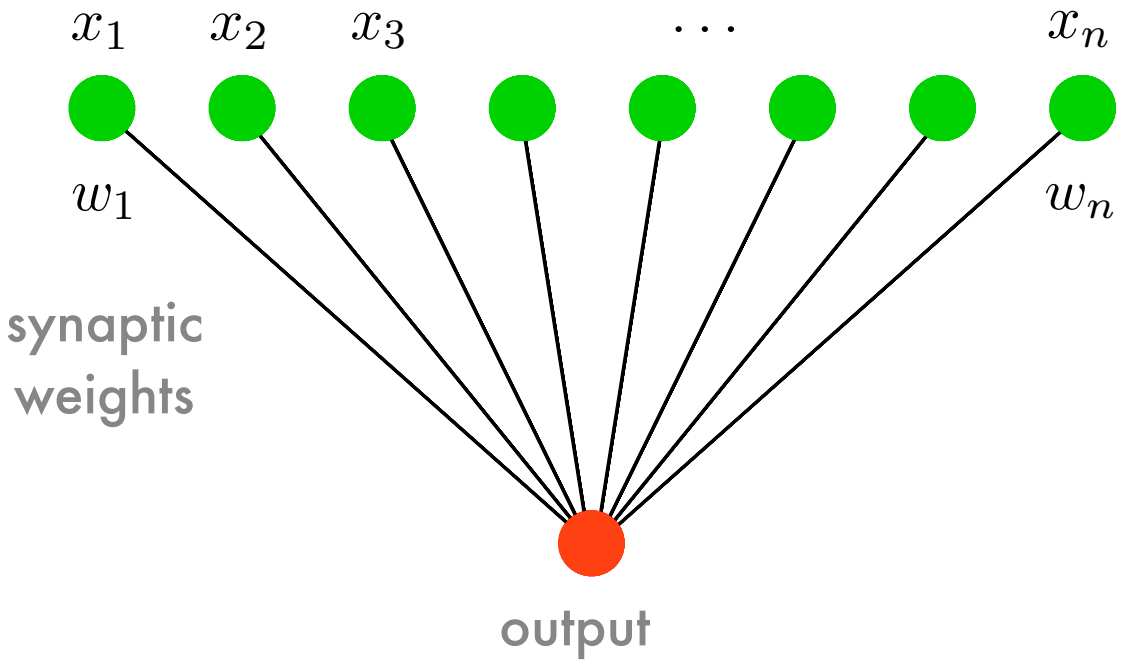
Deep Learning

- **Many simple neurons**
- **Specialized layers** (images, text, audio, …)
- **Stack layers** (hence deep learning)
- **Optimization is difficult**
  - Backpropagation
  - Stochastic gradient descent

better accuracy

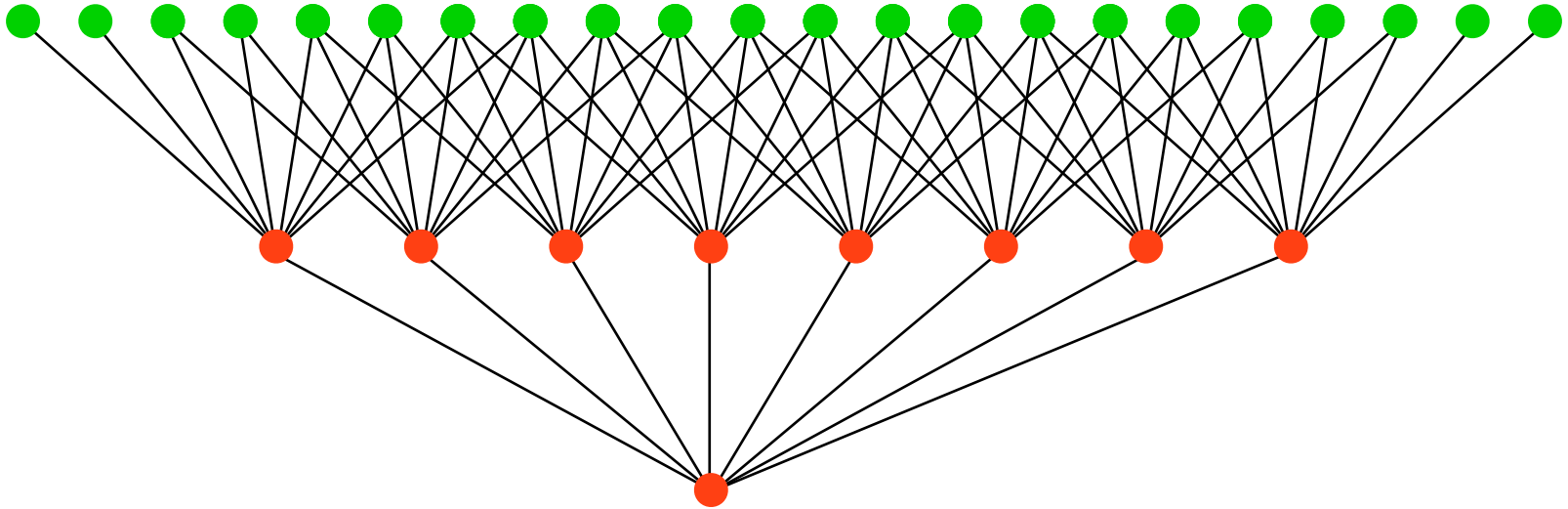
Multilayer Perceptron and Optimization

Perceptron



$$y(x) = \sigma(\langle w, x \rangle)$$

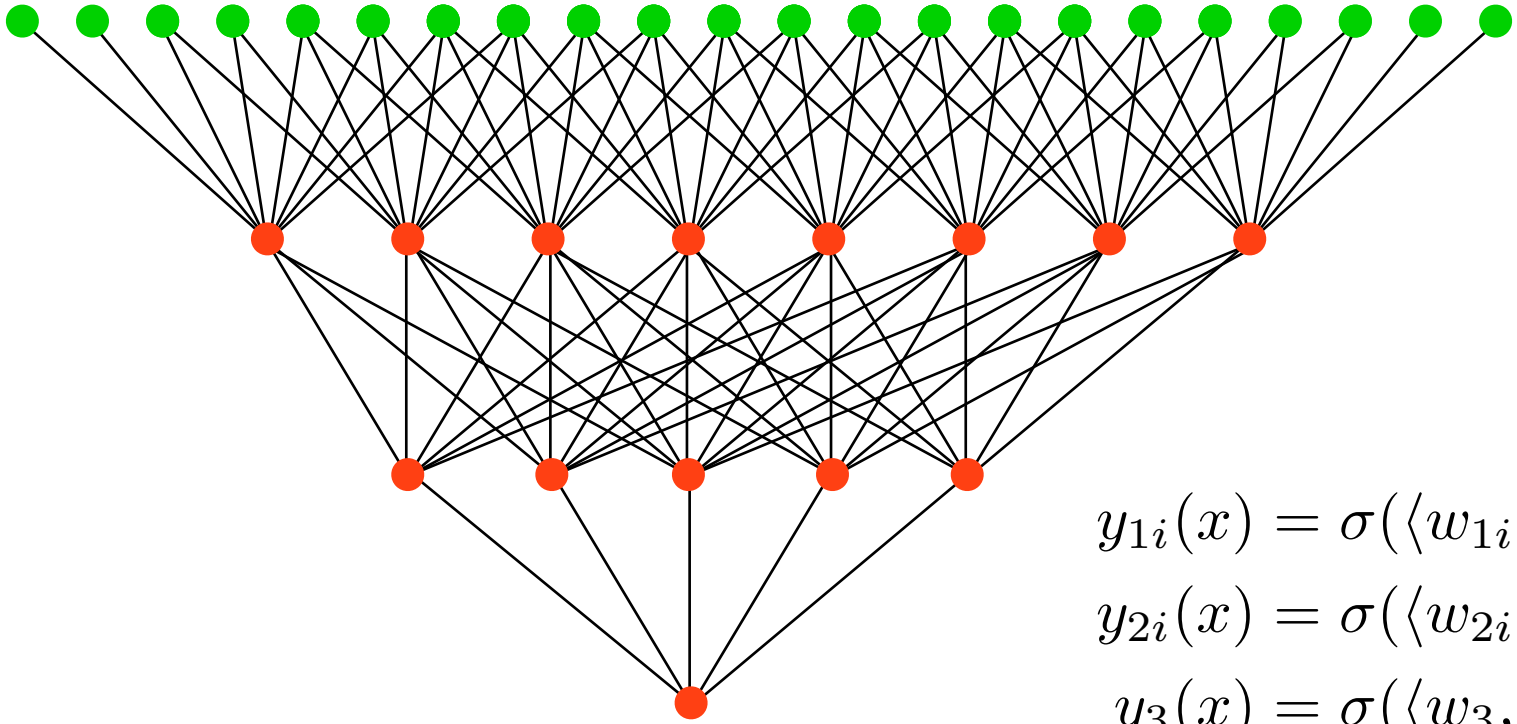
Multilayer Perceptron



$$y_{1i}(x) = \sigma(\langle w_{1i}, x \rangle)$$

$$y_2(x) = \sigma(\langle w_2, y_1 \rangle)$$

Multilayer Perceptron



$$y_{1i}(x) = \sigma(\langle w_{1i}, x \rangle)$$
$$y_{2i}(x) = \sigma(\langle w_{2i}, y_1 \rangle)$$
$$y_3(x) = \sigma(\langle w_3, y_2 \rangle)$$

Multilayer Perceptron

Layer Representation

$$y_i = W_i x_i$$

$$x_{i+1} = \sigma(y_i)$$

Alternate between linear function Wx and nonlinear function σ(x)

Loss function

to measure quality of estimate

$$l(y, y_i)$$

Multilayer Perceptron Training

Layer Representation

$$y_i = W_i x_i$$

$$x_{i+1} = \sigma(y_i)$$

Gradient descent

$$W_i \leftarrow W_i - \eta \partial_{W_i} l(y, y_n)$$

- **Stochastic gradient descent** (use only one sample)
- **Minibatch** (small subset)

Multilayer Perceptron Training

Layer Representation

$$y_i = W_i x_i$$

$$x_{i+1} = \sigma(y_i)$$

Change in Objective

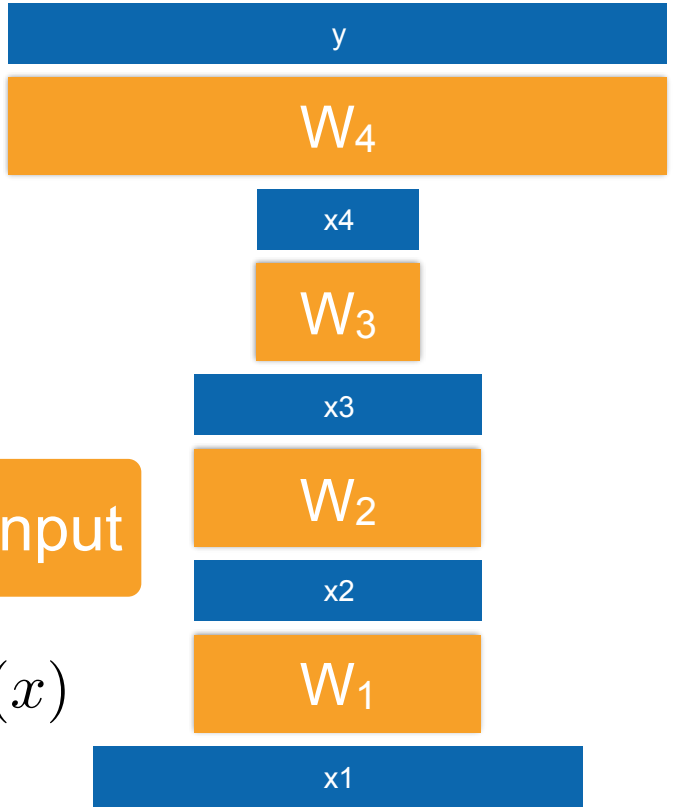
$$g_j = \partial_{W_j} l(y, y_i)$$

Chain Rule

$$\partial_x [f_2 \circ f_1](x) = [\partial_{f_1} f_2 \circ f_1](x) [\partial_x f_1](x)$$

effect of input

second layer



Backpropagation

Layers & Gradients

$$y_i = W_i x_i$$
$$x_{i+1} = \sigma(y_i)$$

$$\partial_{x_i} y_i = W_i$$
$$\partial_{W_i} y_i = x_i$$
$$\partial_{y_i} x_{i+1} = \sigma'(y_i)$$
$$\implies \partial_{x_i} x_{i+1} = \sigma'(y_i) W_i^\top$$

The chain rule (aka BackProp)

$$\begin{aligned}\partial_{x_i} l(y, y_n) &= \partial_{x_{i+1}} l(y, y_n) \partial_{x_i} x_{i+1} \\ &= \partial_{x_{i+1}} l(y, y_n) \sigma'(y_i) W_i^\top \\ \partial_{W_i} l(y, y_n) &= \partial_{y_i} l(y, y_n) \partial_{W_i} y_i \\ &= \partial_{x_{i+1}} l(y, y_n) \sigma'(y_i) x_i^\top\end{aligned}$$

matrix-vector multiply

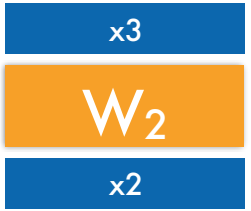
Rectified Linear Unit (ReLu)

Forward mapping

$$y_i = W_i x_i$$
$$x_{i+1} = \sigma(y_i)$$

Nonlinearity between layers

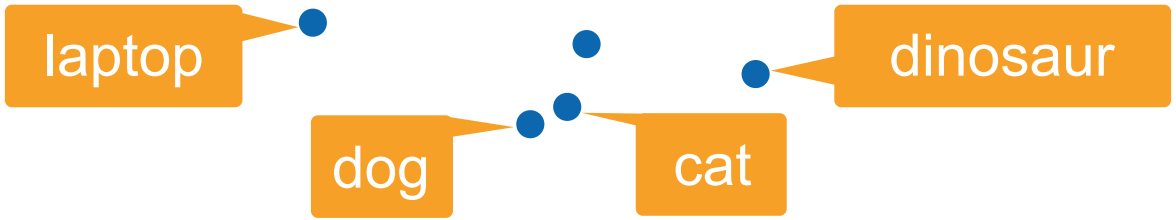
- Gradients vanish at tails for sigmoid
- Replace by max(0,x)
  Derivative is in {0; 1}, sparse signal



Example - Word Vectors

- Full vocabulary (e.g. 30k words)
- One-hot encoding for words
  (0,0,0,0,**1**,0,0,0) (0,0,0,0,0,0,**1**,0)

- Map words into vector space via y = Wx



- Postprocess as normal

That was complicated!
MXNet takes care of this for you …

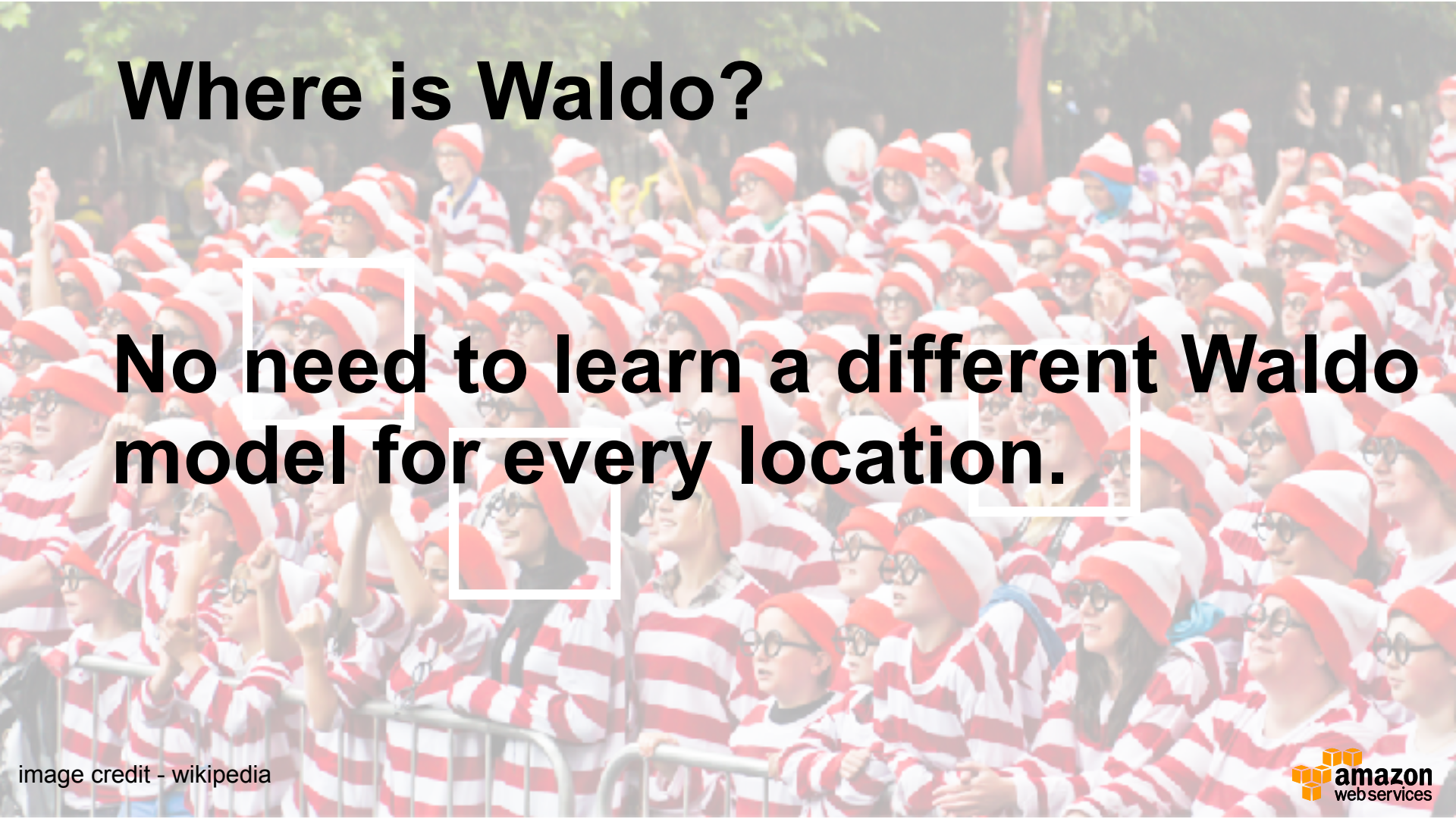
```
fc = mx.symbol.FullyConnected(data=input, num_hidden=4096)
relu = mx.symbol.Activation(data=fc, act_type="relu")
dropout = mx.symbol.Dropout(data=relu, p=0.5)
```

Where is Waldo?

image credit - wikipedia

amazon web services

Where is Waldo?



No need to learn a different Waldo model for every location.

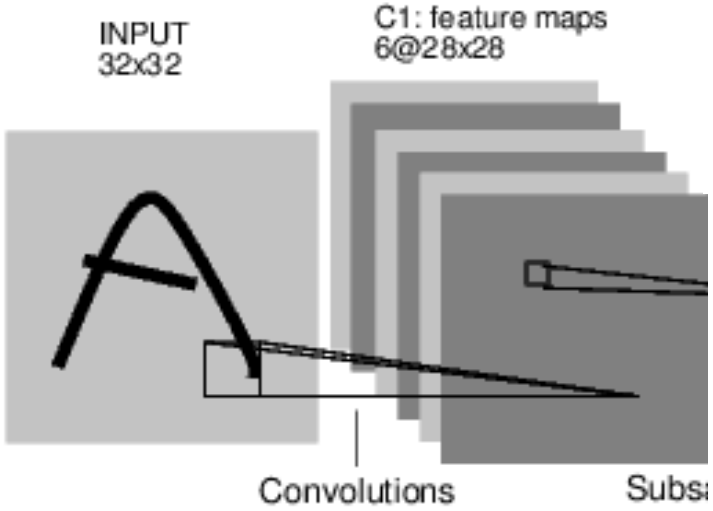
image credit - wikipedia

Convolutional Layers

- Images have shift invariance
- Low level mostly edge and feature detectors
- **Key Idea**
  Replace matrix multiplication with local computation

$$y_{ij} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} W_{ab} x_{i+a,j+b}$$

- Backprop automatic in MXNet



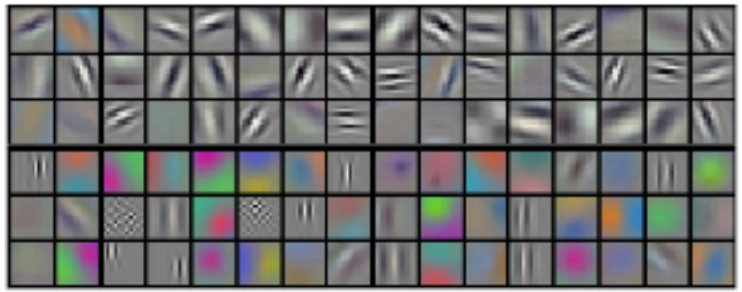
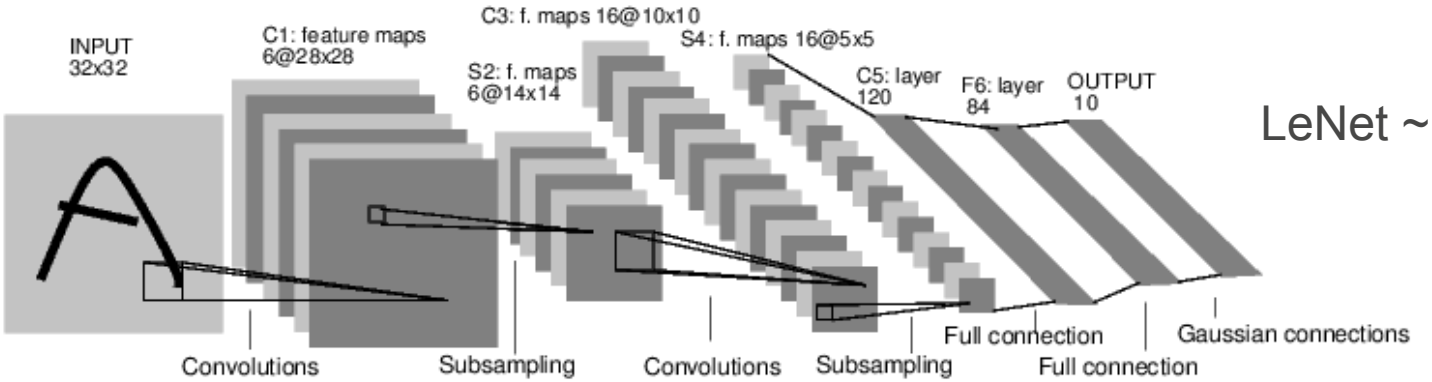


image credit - Le Cun et al, 1998, Olshausen and Field, 1998

Subsampling & MaxPooling

- Multiple convolutions blow up dimensionality



LeNet ~1995

- Subsampling - average over patches (works OK)
- MaxPooling - pick the maximum over patches (much better)

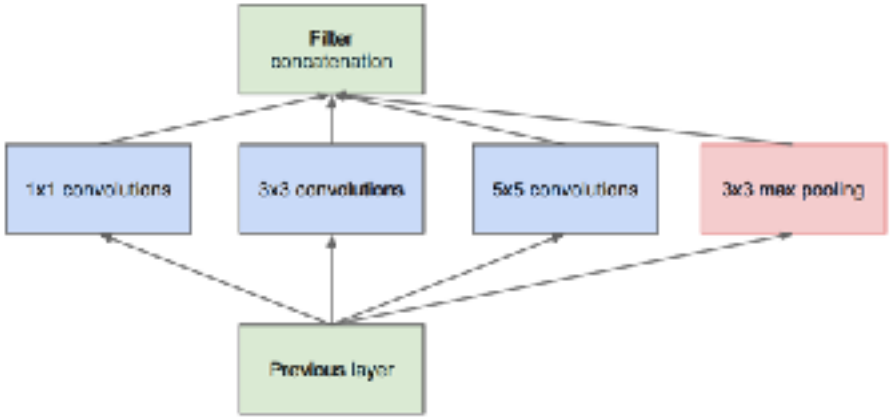
image credit - Le Cun et al, 1998

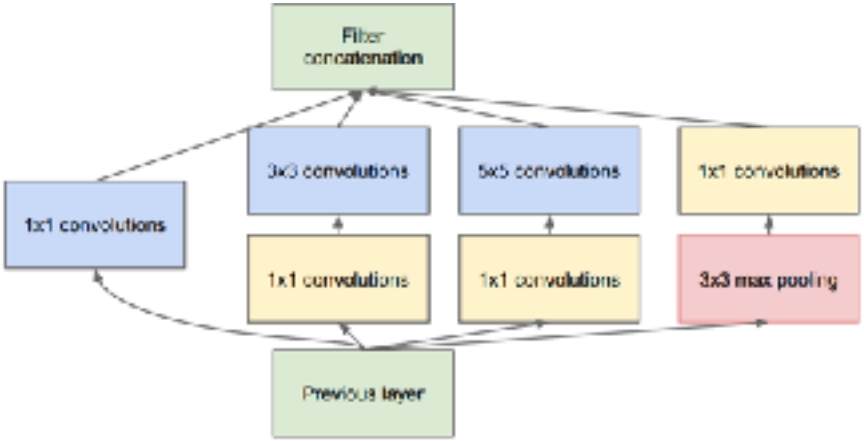
LeNet in R (using MXNet)

```
get_symbol <- function(num_classes = 1000) {
  data <- mx.symbol.Variable('data')
  # first conv
  conv1 <- mx.symbol.Convolution(data = data, kernel = c(5, 5), num_filter = 20)

  tanh1 <- mx.symbol.Activation(data = conv1, act_type = "tanh")
  pool1 <- mx.symbol.Pooling(data = tanh1, pool_type = "max", kernel = c(2, 2), stride = c(2, 2))

  # second conv
  conv2 <- mx.symbol.Convolution(data = pool1, kernel = c(5, 5), num_filter = 50)
  tanh2 <- mx.symbol.Activation(data = conv2, act_type = "tanh")
  pool2 <- mx.symbol.Pooling(data = tanh2, pool_type = "max", kernel = c(2, 2), stride = c(2, 2))
  # first fullc
  flatten <- mx.symbol.Flatten(data = pool2)
  fc1 <- mx.symbol.FullyConnected(data = flatten, num_hidden = 500)
  tanh3 <- mx.symbol.Activation(data = fc1, act_type = "tanh")
  # second fullc
  fc2 <- mx.symbol.FullyConnected(data = tanh3, num_hidden = num_classes)
  # loss
  lenet <- mx.symbol.SoftmaxOutput(data = fc2, name = 'softmax')
  return(lenet)
}
```

Fancy structures

- Compute different filters
- Compose one big vector from all of them
- Layer them iteratively

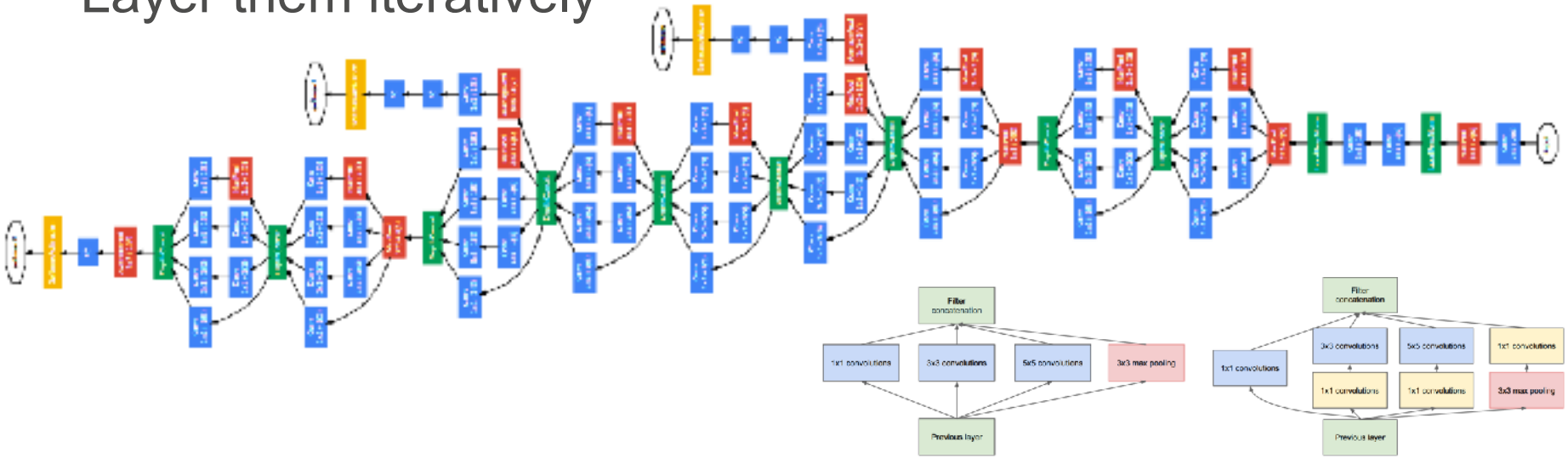




Szegedy et al. arxiv.org/pdf/1409.4842v1.pdf

Fancy structures

- Compute different filters
- Compose one big vector from all of them
- Layer them iteratively

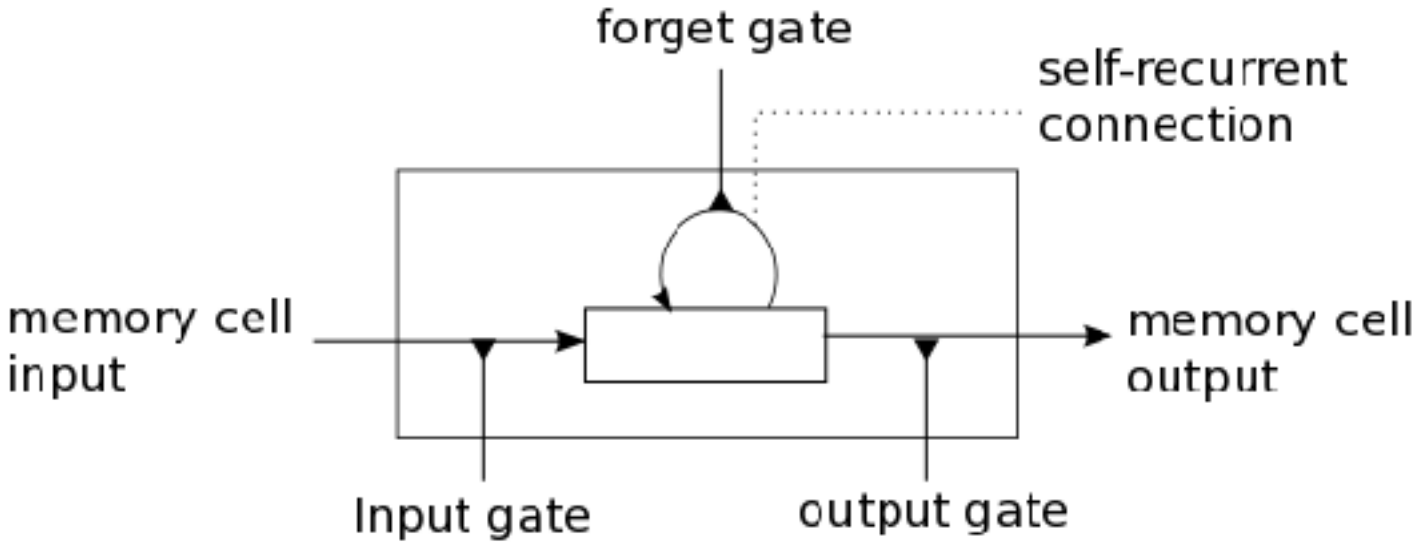


Szegedy et al. arxiv.org/pdf/1409.4842v1.pdf

Inception v3 in Python (using MXNet)

```
def get_symbol(num_classes=1000):
    # data
    data = mx.symbol.Variable(name="data")
    # stage 1
    conv1 = ConvFactory(data=data, num_filter=64, kernel=(7, 7), stride=(2, 2), pad=(3, 3), name='1')
    pool1 = mx.symbol.Pooling(data=conv1, kernel=(3, 3), stride=(2, 2), name='pool_1', pool_type='max')
    # stage 2
    conv2red = ConvFactory(data=pool1, num_filter=64, kernel=(1, 1), stride=(1, 1), name='2_red')
    conv2 = ConvFactory(data=conv2red, num_filter=192, kernel=(3, 3), stride=(1, 1), pad=(1, 1), name='2')
    pool2 = mx.symbol.Pooling(data=conv2, kernel=(3, 3), stride=(2, 2), name='pool_2', pool_type='max')
    # stage 2
    in3a = InceptionFactoryA(pool2, 64, 64, 64, 64, 96, "avg", 32, '3a')
    in3b = InceptionFactoryA(in3a, 64, 64, 96, 64, 96, "avg", 64, '3b')
    in3c = InceptionFactoryB(in3b, 128, 160, 64, 96, '3c')
    # stage 3
    in4a = InceptionFactoryA(in3c, 224, 64, 96, 96, 128, "avg", 128, '4a')
    in4b = InceptionFactoryA(in4a, 192, 96, 128, 96, 128, "avg", 128, '4b')
    in4c = InceptionFactoryA(in4b, 160, 128, 160, 128, 160, "avg", 128, '4c')
    in4d = InceptionFactoryA(in4c, 96, 128, 192, 160, 192, "avg", 128, '4d')
    in4e = InceptionFactoryB(in4d, 128, 192, 192, 256, '4e')
    # stage 4
    in5a = InceptionFactoryA(in4e, 352, 192, 320, 160, 224, "avg", 128, '5a')
    in5b = InceptionFactoryA(in5a, 352, 192, 320, 192, 224, "max", 128, '5b')
    # global avg pooling
    avg = mx.symbol.Pooling(data=in5b, kernel=(7, 7), stride=(1, 1), name="global_pool", pool_type='avg')
    # linear classifier
    flatten = mx.symbol.Flatten(data=avg, name='flatten')
    fc1 = mx.symbol.FullyConnected(data=flatten, num_hidden=num_classes, name='fc1')
    softmax = mx.symbol.SoftmaxOutput(data=fc1, name='softmax')
    return softmax
```

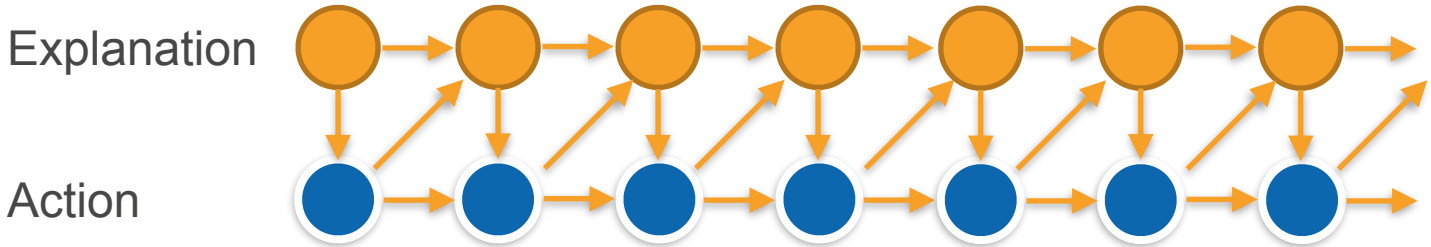
Sequence Models



Schmidhuber & Hochreiter, 1998

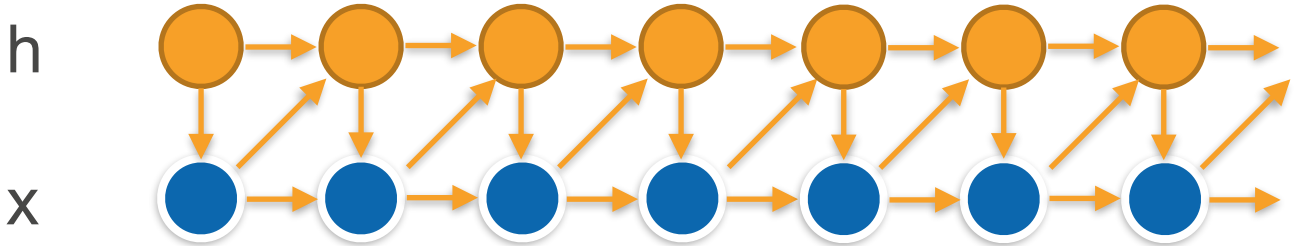
Sequence Models

- **Sequence of observations**
  Purchases, likes, app use, e-mails, queries, ratings, words, dialogue
- **Hidden state to explain behavior**
  - Navigational, informational queries in search
  - Interest distributions for users over time
  - Trajectory and location modeling
  - Content and meaning in text



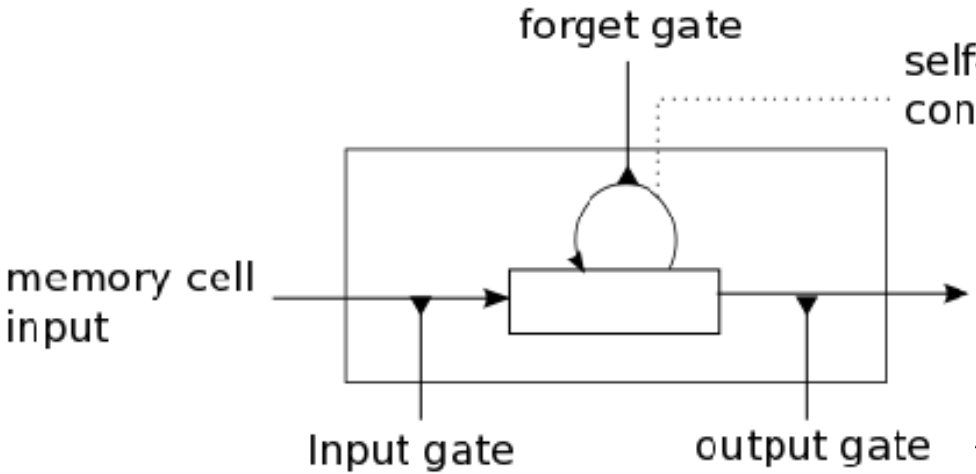
Latent Variable Models

- **Temporal sequence of observations**
  Purchases, likes, app use, e-mails, queries, ratings, words, dialogue
- **Latent state to explain behavior**
  - Plain deep network = RNN
  - Deep network with attention = LSTM / GRU …
    (learn when to update state, how to read out)



Long Short Term Memory

Schmidhuber and Hochreiter, 1998



input

$$i_t = \sigma(W_i(x_t, h_t) + b_i)$$

forgetting

$$f_t = \sigma(W_f(x_t, h_t) + b_f)$$

$$z_{t+1} = f_t \cdot z_t + i_t \cdot \tanh(W_z(x_t, h_t) + b_z)$$

output

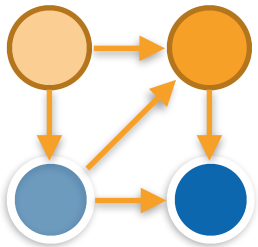
$$o_t = \sigma(W_o(x_t, h_t, z_{t+1}) + b_o)$$

$$h_{t+1} = o_t \cdot \tanh z_{t+1}$$

state

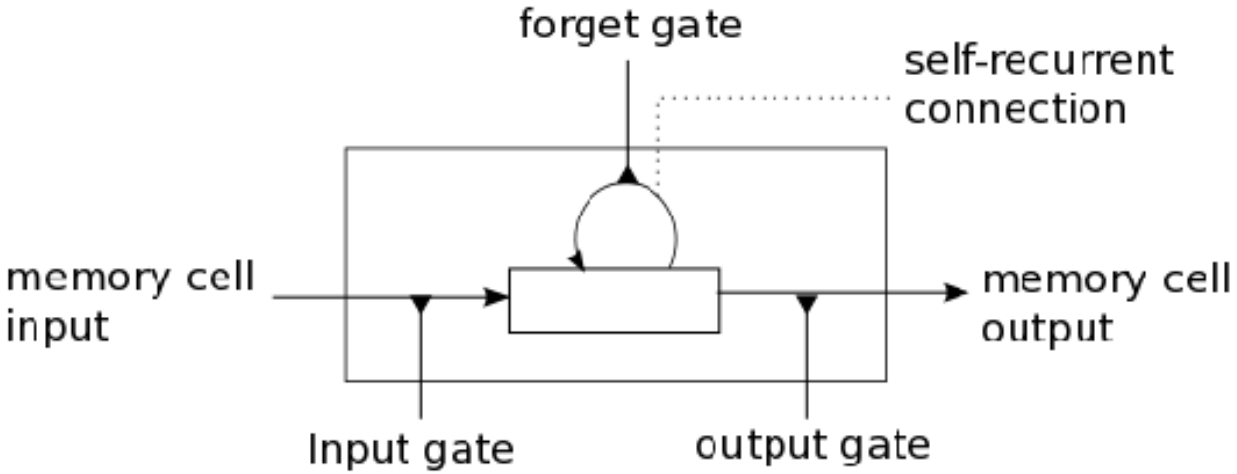
h

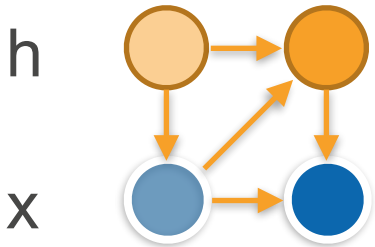
x



Long Short Term Memory

Schmidhuber and Hochreiter, 1998



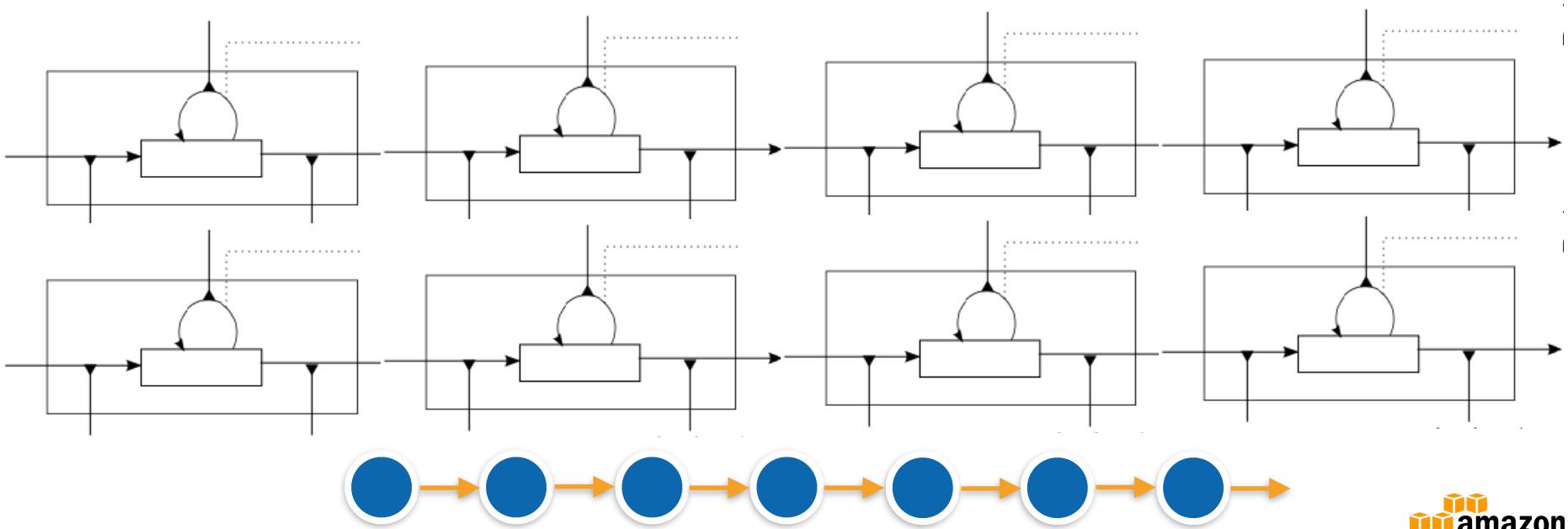


$$(z_{t+1}, h_{t+1}, o_t) = \mathrm{LSTM}(z_t, h_t, x_t)$$

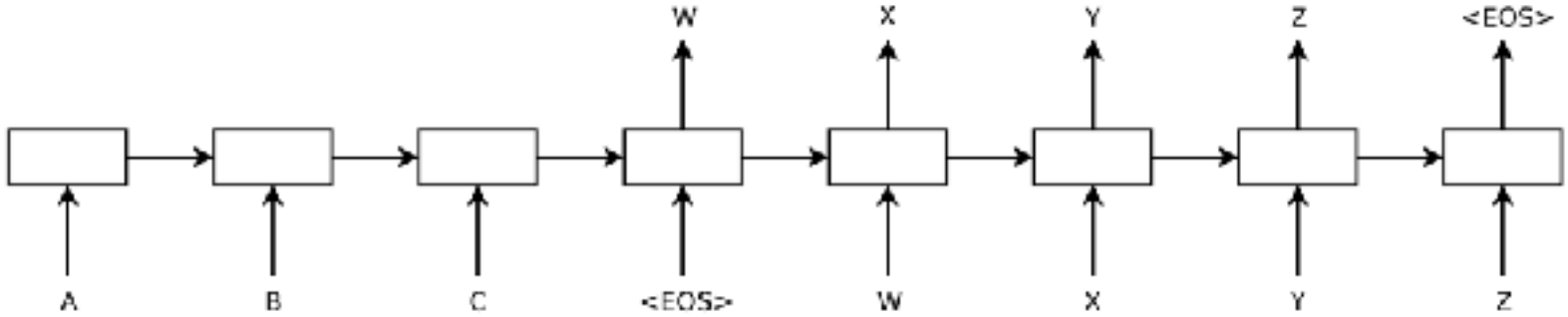
Treat it as a black box

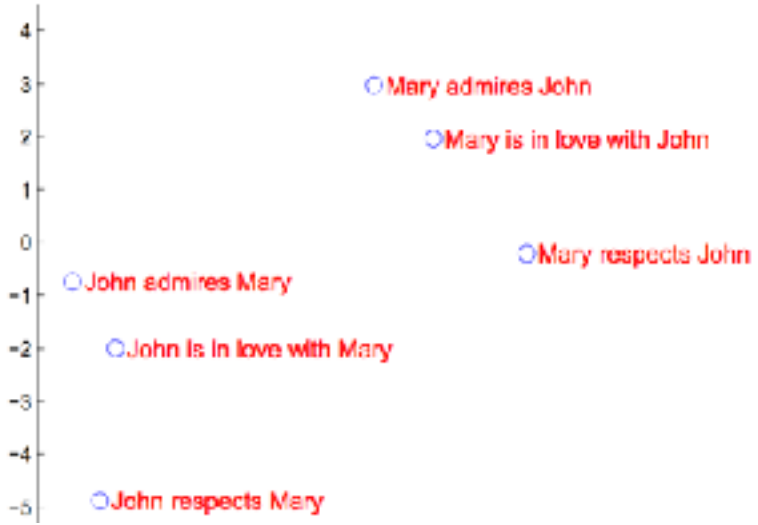
LSTM (Long Short Term Memory)

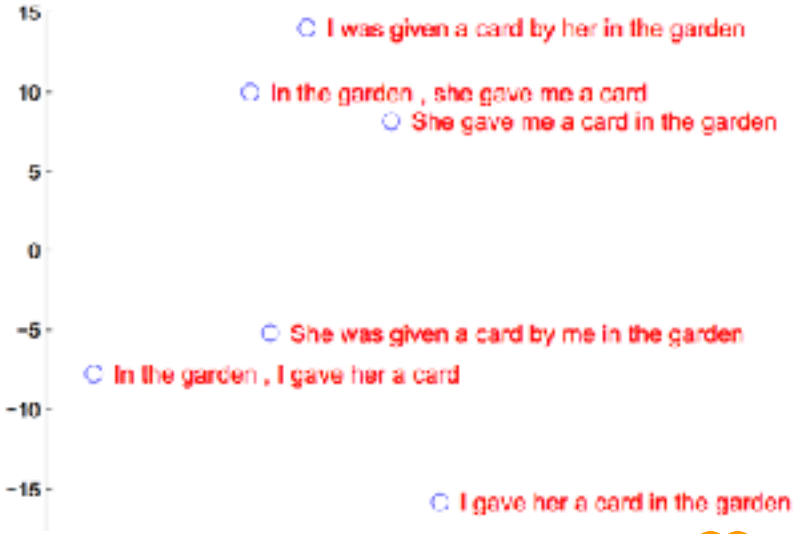
- Group LSTM cells into layers
- Can model different scales of dynamics



Natural Language Translation

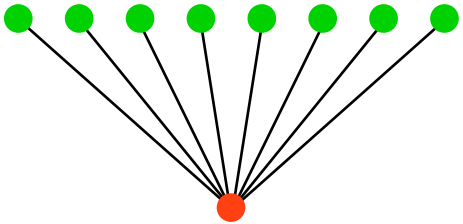




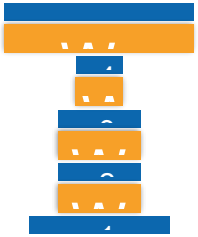


Le, Sutskever and Vinyals, 2014

Summary



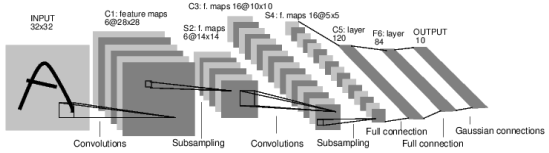
Perceptron



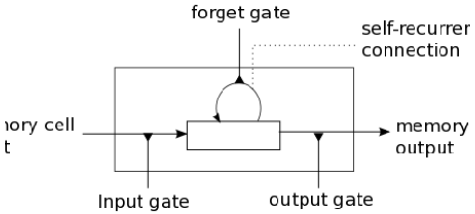
Multilayer Perceptron

$$\begin{aligned}\partial_{x_i} l(y, y_n) &= \partial_{x_{i+1}} l(y, y_n) \partial_{x_i} x_{i+1} \\ &= \partial_{x_{i+1}} l(y, y_n) \sigma'(y_i) W_i^\top \\ \partial_{W_i} l(y, y_n) &= \partial_{y_i} l(y, y_n) \partial_{W_i} y_i \\ &= \partial_{x_{i+1}} l(y, y_n) \sigma'(y_i) x_i^\top\end{aligned}$$

Backpropagation



Convolutional Nets



LSTM

MXNet

Modules

1. **Introduction to Deep Learning**
2. **Getting Started with MXNet**
   - Cloud - AMIs and CloudFormation Template
   - Laptop - build from source
3. **MXNet**
   - NDArrays and Symbols
   - Training Deep Networks
4. **Examples**
   - Computer Vision
   - Natural Language Processing
   - Recommender Systems

amazon web services


mxnet
2. Running MXNet
• Amazon Machine Images (AMI)
• Cloud Formation Templates
• Local install & resources
amazon
web services

Amazon Machine Image for Deep Learning

http://bit.ly/deepami

Getting started with Deep Learning

- For data scientists and developers
- Setting up a DL system takes (install) time & skill
  - Keep packages up to date and compile
  - Install all dependencies (licenses, compilers, drivers, etc.)
  - **NVIDIA** Drivers for G2 and P2 servers
  - **Intel** MKL linked for everything else (C5 coming soon)

Deep Learning AMI

Sold by: Amazon Web Services

The Deep Learning AMI is a supported and maintained Amazon Linux image provided by Amazon Web Services for use on Amazon Elastic Compute Cloud (Amazon EC2). It is designed to provide a stable, secure, and high performance execution environment for deep learning applications running on Amazon EC2. It includes popular deep learning frameworks, including MXNet, Caffe, Tensorflow, Theano, Torch, and CNTK as well as packages that enable easy integration with AWS, including launch configuration tools and many popular AWS libraries and tools. It also includes the Anaconda Data Science Platform... Read more

http://bit.ly/deepami

Getting started with Deep Learning

- **Drivers**
  CUDA / CUDNN / CUFFT / CUSPARSE
- **Development tools**
  Python 2 and 3, Anaconda, Jupyter notebooks, Graphviz
- **Deep Learning Platforms (compiled & tested)**
  - **MXNet, Tensorflow, CNTK**
    multi-GPU, multi-machine (MXNet recommended)
  - **Caffe, Theano, Torch, Keras**

Always up to date (less than 1 month), optimized & tested on AWS

Getting started

```
acbc32cf4de3:image-classification smola$ ssh ec2-user@54.210.246.140
Last login: Fri Nov 11 05:58:58 2016 from 72-21-196-69.amazon.com
================================================================================
       __|  __|_  )
       _|  (     /   Deep Learning AMI for Amazon Linux
      ___|\___|___|

This is beta version of the Deep Learning AMI for Amazon Linux.

The README file for the AMI →→→→→→→→→→→→→→→→→→→  /home/ec2-user/src/README.md
Tests for deep learning frameworks →→→→→→→→→→→→   /home/ec2-user/src/bin
================================================================================

7 package(s) needed for security, out of 75 available
Run "sudo yum update" to apply all updates.
Amazon Linux version 2016.09 is available.
[ec2-user@ip-172-31-55-21 ~]$ cd src/
[ec2-user@ip-172-31-55-21 src]$ ls
anaconda2  bazel  caffe   cntk   keras  mxnet                 OpenBLAS  README.md   Theano
anaconda3  bin    caffe3  demos  logs   Nvidia_Cloud_EULA.pdf  opencv    tensorflow  torch
```

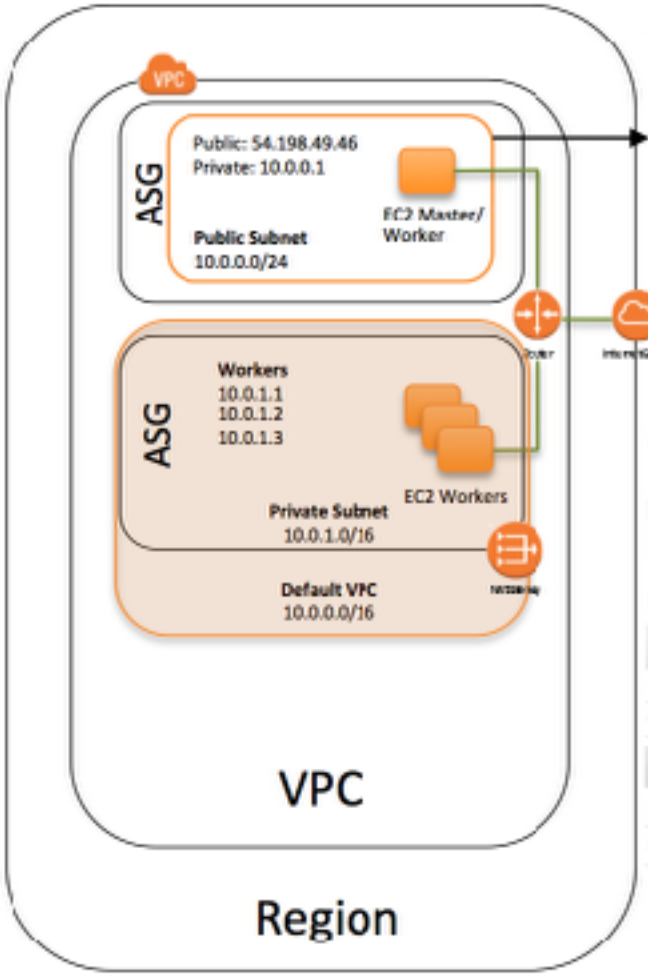
AWS CloudFormation Template for Deep Learning

http://bit.ly/deepcfn

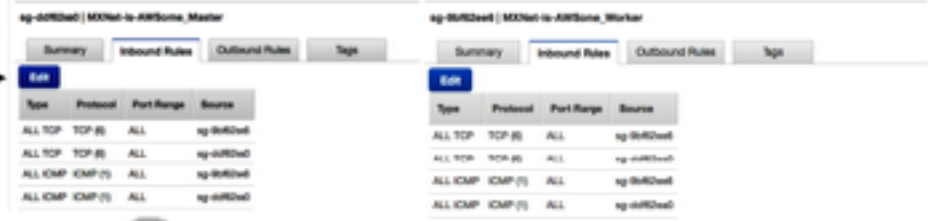
amazon web services

AWS CloudFormation Templates

- **Hardware as Code**
  - Define compute resources
  - Define network
  - Define OS installation and configuration
- **Deep Learning**
  - Many toolkits barely deal with multiple GPUs (DGX-1)
  - BigData needs massive compute resources
  - Hardware availability and software support



Security group to open ports on private Interface from Master to workers and vis-a-vis



Security group to enable external SSH Access to Master

Route Table

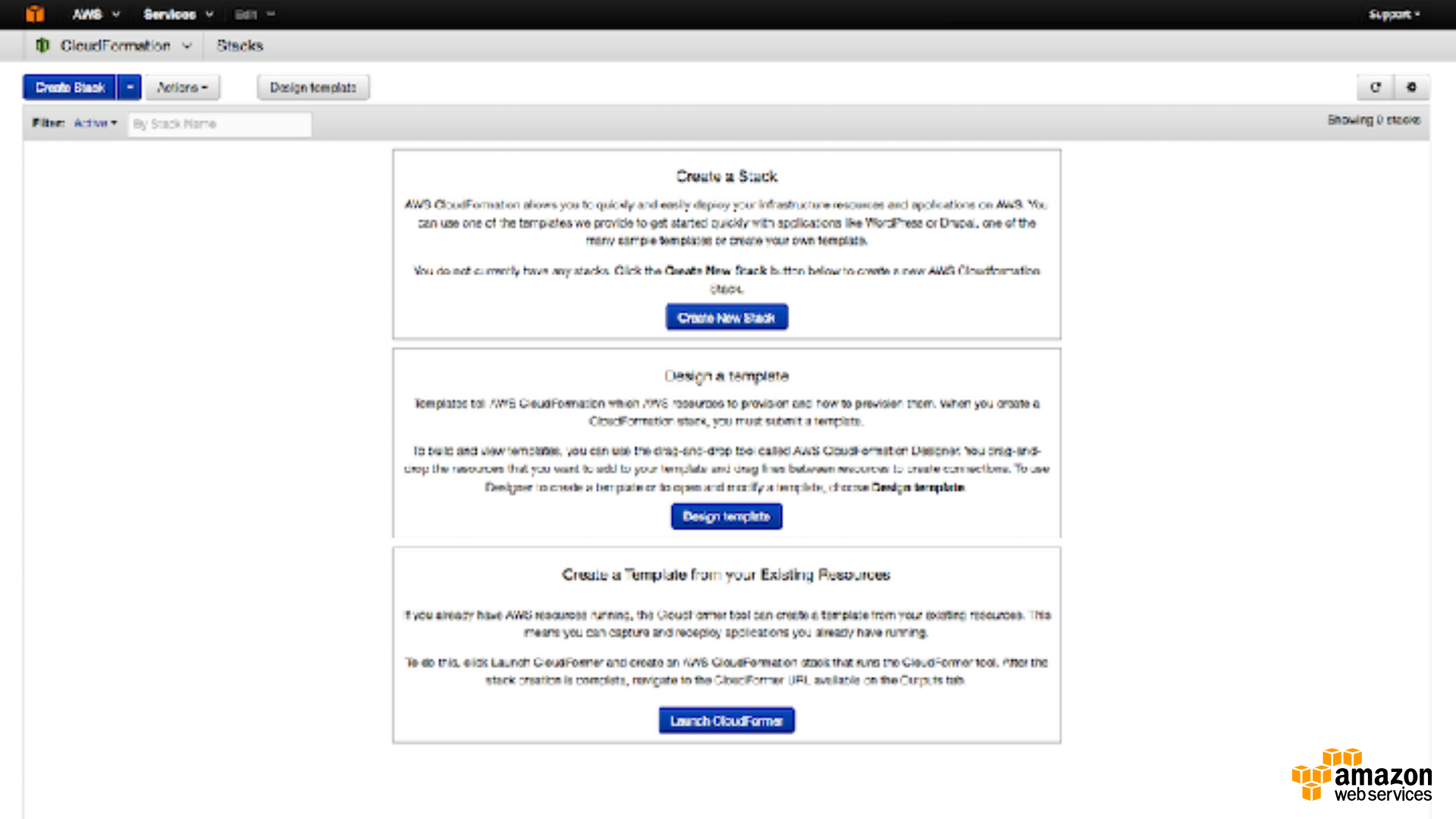
Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	nat-0617c31f9b64a13b0	Active	No

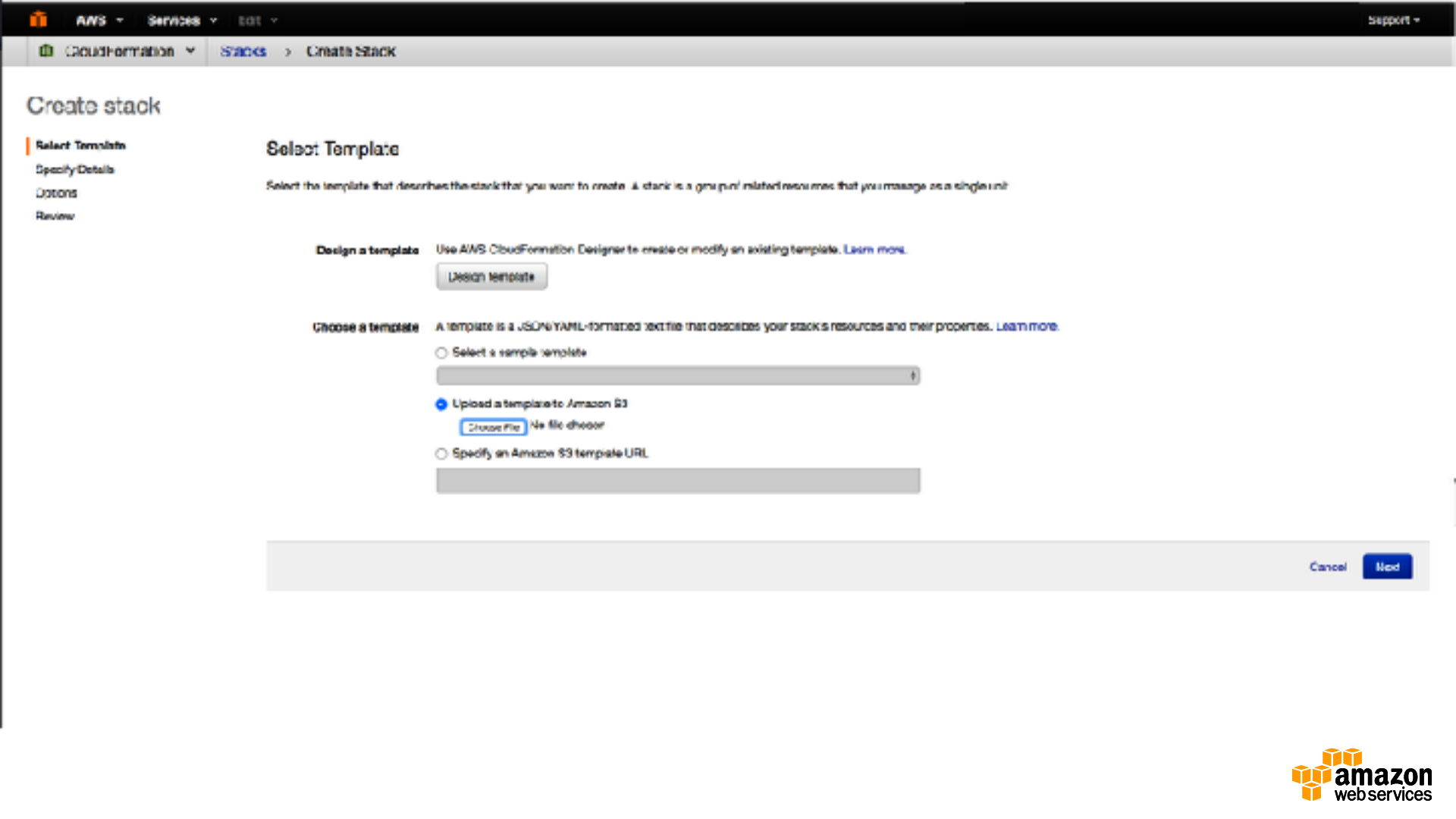
Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	igw-7d48b11a	Active	No

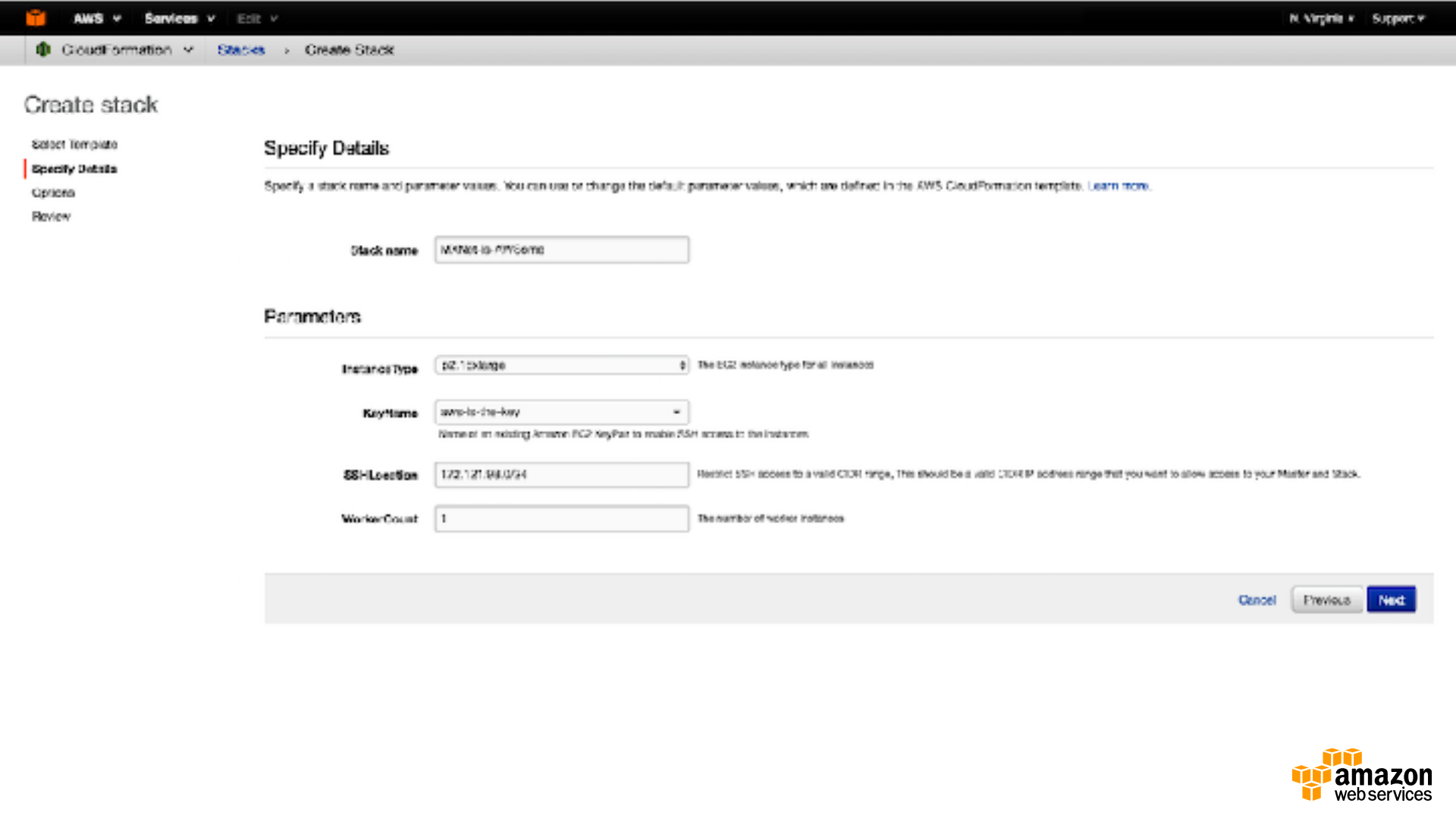
amazon web services

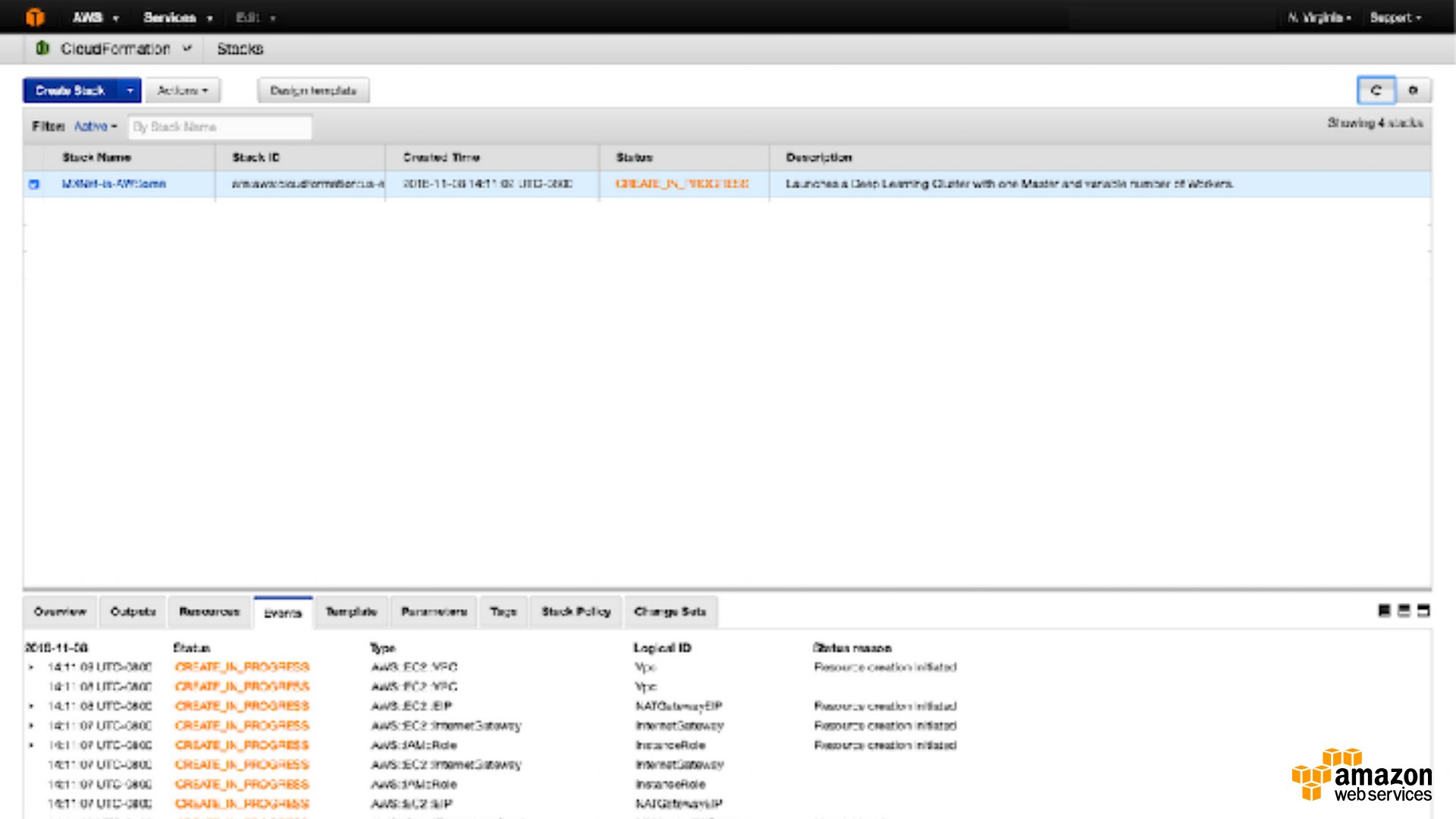
AWS CloudFormation Components

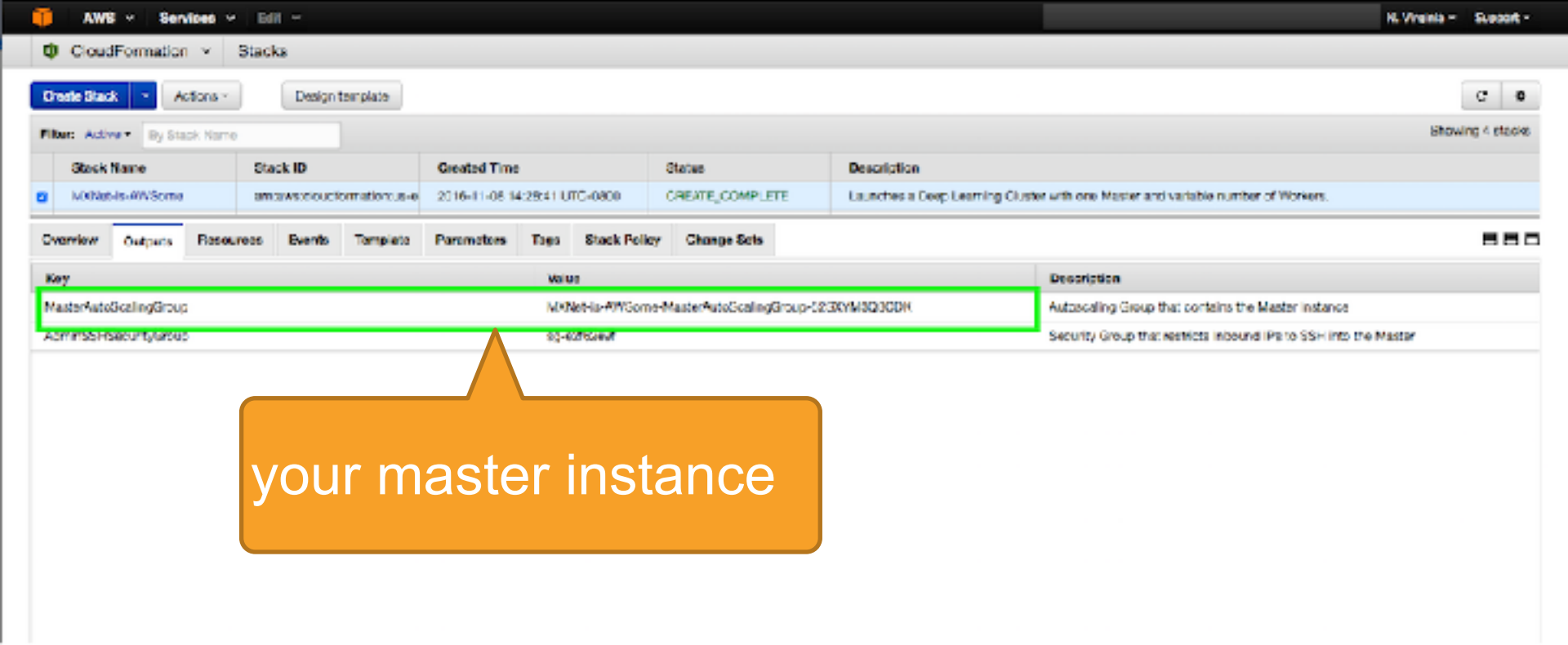
- **VPC** in the customer account.
- The requested number of **worker instances** in an Auto Scaling group within the VPC. Workers are launched in a **private subnet**.
- **Master instance** in a separate Auto Scaling group that acts as a proxy to enable connectivity to the cluster via **SSH**.
- Two security groups that open ports on the **private subnet** for communication between the master and workers.
- **IAM role** that allows users to access and query Auto Scaling groups and the private IP addresses of the EC2 instances.
- **NAT gateway** used by instances within the VPC to talk to the outside.


AWS
Services
Edit
Support
CloudFormation
Stacks
Create Stack
Actions
Design template
Filter: Active
By Stack Name
Showing 0 stacks
Create a Stack
AWS CloudFormation allows you to quickly and easily deploy your infrastructure resources and applications on AWS. You can use one of the templates we provide to get started quickly with applications like WordPress or Drupal, one of the many sample templates or create your own template.
You do not currently have any stacks. Click the Create New Stack button below to create a new AWS CloudFormation stack.
Create New Stack
Design a template
templates tell AWS CloudFormation which AWS resources to provision and how to provision them. When you create a CloudFormation stack, you must submit a template.
To build and view templates, you can use the drag-and-drop tool called AWS CloudFormation Designer. You drag-and-drop the resources that you want to add to your template and drag lines between resources to create connections. To use Designer to create a template or to open and modify a template, choose Design template.
Design template
Create a Template from your Existing Resources
If you already have AWS resources running, the CloudFormer tool can create a template from your existing resources. This means you can capture and redeploy applications you already have running.
To do this, click Launch CloudFormer and create an AWS CloudFormation stack that runs the CloudFormer tool. After the stack creation is complete, navigate to the CloudFormer URL available on the Outputs tab.
Launch CloudFormer
amazon
web services


AWS
Services
Edit
Support
CloudFormation
Stacks
Create Stack
Create stack
Select Template
Specify Details
Options
Review
Select Template
Design a template
Use AWS CloudFormation Designer to create or modify an existing template. Learn more.
Design template
Choose a template
A template is a JSON/YAML-formatted text file that describes your stack's resources and their properties. Learn more.
Select a sample template
Upload a template to Amazon S3
Choose File
No file chosen
Specify an Amazon S3 template URL
Cancel
Next


AWS
Services
Edit
N. Virginia
Support
CloudFormation
Stacks
Create Stack
Create stack
Select template
Specify Details
Options
Review
Specify Details
Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. Learn more.
Stack name
Parameters
InstanceType
The EC2 instance type for all instances
KeyName
aws-is-the-key
Name of an existing Amazon EC2 KeyPair to enable SSH access to the instances
SSHLocation
Restrict SSH access to a valid CIDR range. This should be a valid CIDR IP address range that you want to allow access to your Master and Slave.
WorkerCount
1
The number of worker instances
Cancel
Previous
Next


AWS
Services
Edit
N. Virginia
Support
CloudFormation
Stacks
Create Stack
Actions
Design template
Filter: Active
By Stack Name
Showing 4 stacks
Stack Name
Stack ID
Created Time
Status
Description
Launches a Deep Learning Cluster with one Master and variable number of Workers.
Overview
Outputs
Resources
Events
Template
Parameters
Tags
Stack Policy
Change Sets
2018-11-08
Status
Type
Logical ID
Status reason
CREATE_IN_PROGRESS
AWS::EC2::VPC
Vpc
Resource creation initiated
AWS::EC2::EIP
NATGatewayEIP
AWS::EC2::InternetGateway
InternetGateway
AWS::IAM::Role
InstanceRole
amazon
web services



Launching an MXNet job

MNIST digit classification demo

```
../../tools/launch.py -n
    $DEEPLEARNING_WORKERS_COUNT -H
    $DEEPLEARNING_WORKERS_PATH
    python train_mnist.py
    --gpus $(seq -s , 0 1 $
    (($DEEPLEARNING_WORKER_GPU_COUNT - 1)))
    --network lenet --kv-store dist_sync
```

network choice

parameter server

update policy

Local Installation

Installation Pointers

http://mxnet.io/get_started/setup.html

(many packages - Python, graphviz, OpenCV, OpenMP, CUDA)

- MXNet with **Docker**
- Installing MXNet on the **Cloud** (AWS AMI)
- Installing MXNet on **Ubuntu**
- Installing MXNet on **Amazon Linux**
- Installing MXNet on **OS X (Mac)**
- Installing MXNet on **Windows**

Ubuntu & Python

```
# download and install CUDA 8 and CUDNN 8
git clone https://github.com/dmlc/mxnet.git ~/mxnet --recursive

# If building with GPU, add configurations to config.mk file:
cd ~/mxnet
cp make/config.mk .
echo "USE_CUDA=1" >> config.mk
echo "USE_CUDA_PATH=/usr/local/cuda" >> config.mk
echo "USE_CUDNN=1" >> config.mk

# Install MXNet for Python with all required dependencies
cd ~/mxnet/setup-utils
bash install-mxnet-ubuntu-python.sh

# We have added MXNet Python package path in your ~/.bashrc.
# Run the following command to refresh environment variables.
source ~/.bashrc
```

Mac and Scala

- Install prerequisites
  Graphviz, Python, OpenCV, Jupyter
- Build general MXNet package, then

```
make scalapkg
make scalainstall
```

… enjoy …

Windows and R

- Prebuilt binary **libmxnet.dll** for download (CUDA / Intel)
  https://github.com/dmlc/mxnet/releases
- Build from source in VisualStudio
  (needs lots dependences - OpenCV, GraphViz, …)
- Install prebuilt binary for R

```
install.packages("drat", repos="https://cran.rstudio.com")
drat:::addRepo("dmlc")
install.packages("mxnet")
```

- Install from source …

Any language. Anywhere. Native Performance

Linux

Windows

OS X

Docker

Android, iOS, IoT …

NVIDIA

Intel

ARM …

Python

R

Scala

Julia

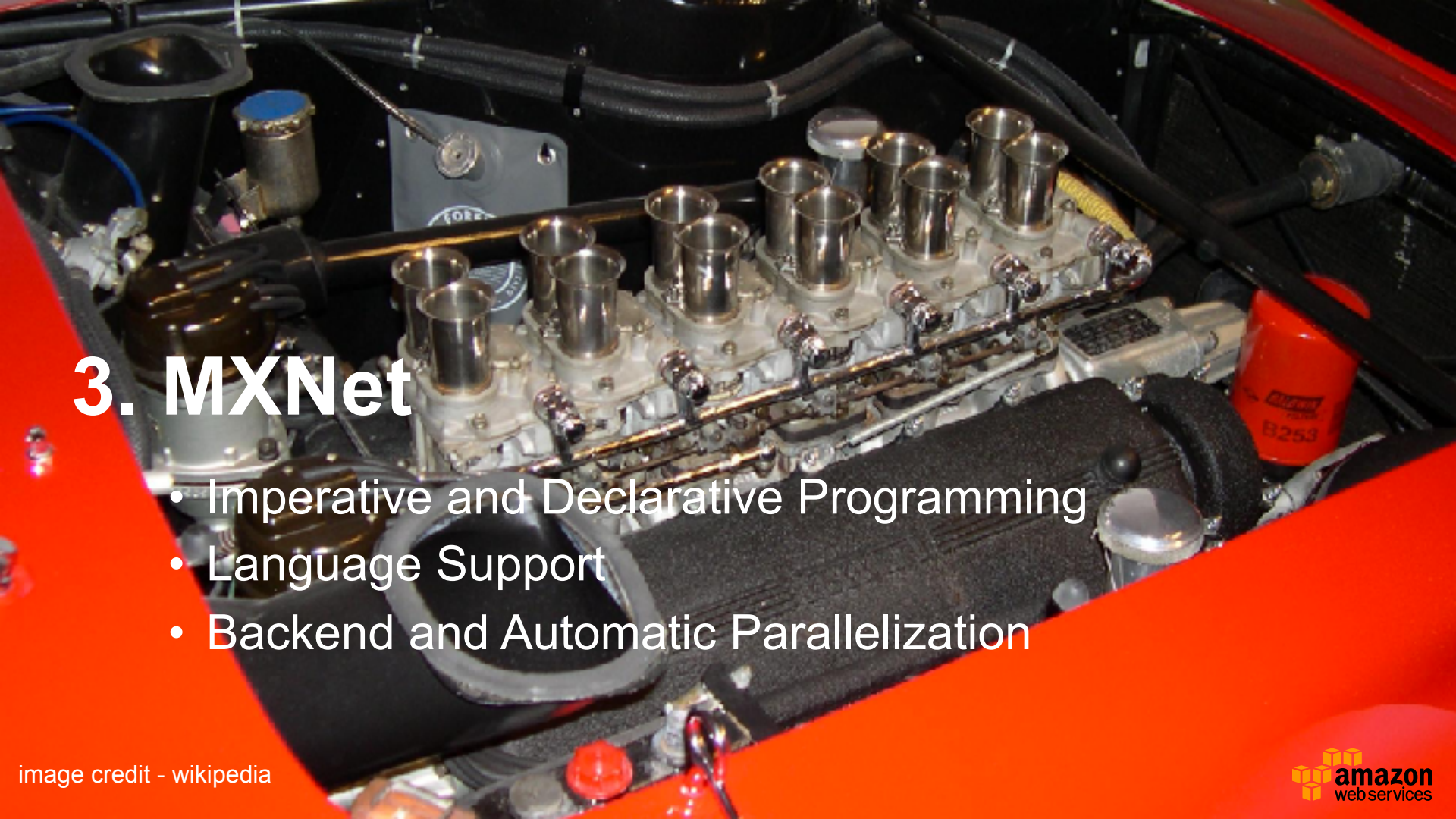
JavaScript

Torch, Caffe …

Modules

1. **Introduction to Deep Learning**
2. **Getting Started with MXNet**
   - Cloud - AMIs and CloudFormation Template
   - Laptop - build from source
3. **MXNet**
   - NDArrays and Symbols
   - Training Deep Networks
4. **Examples**
   - Computer Vision
   - Natural Language Processing
   - Recommender Systems

amazon web services



3. MXNet

- Imperative and Declarative Programming
- Language Support
- Backend and Automatic Parallelization

image credit - wikipedia

amazon web services

Why yet another deep networks tool?

Caffe
Torch
Theano
Tensorflow
CNTK
Keras
Paddle
Chainer
SINGA
DL4J

image credit - Banksy/wikipedia

amazon web services

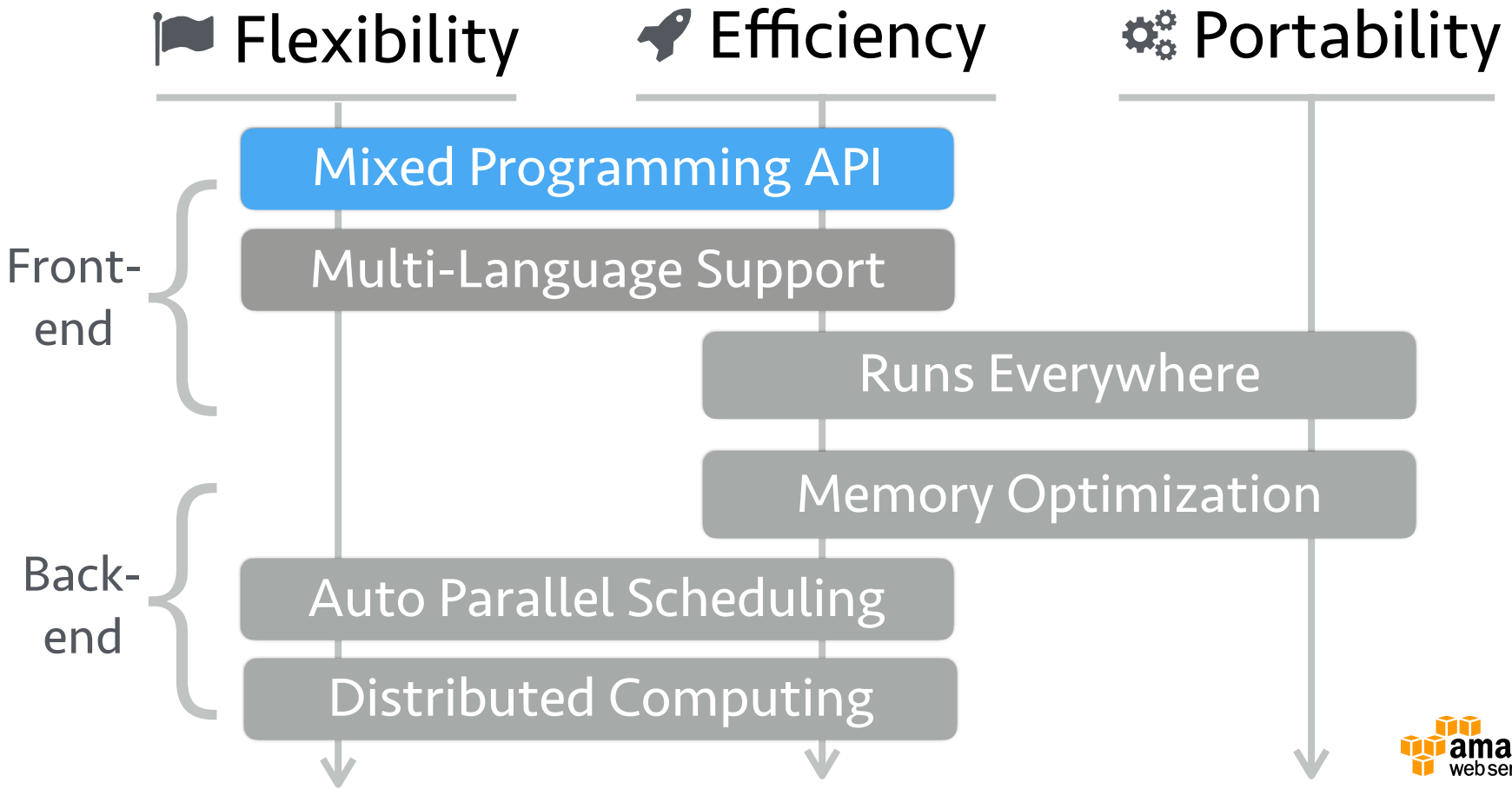
Why yet another deep networks tool?

- **Frugality & resource efficiency**
  Engineered for cheap GPUs with smaller memory, slow networks
- **Speed**
  - Linear scaling with #machines and #GPUs
  - High efficiency on single machine, too (C++ backend)
- **Simplicity**
  Mix declarative and imperative code

single implementation of backend system and common operators

performance guarantee regardless which frontend language is used

MXNet Highlights



Imperative Programs


```
import numpy as np
a = np.ones(10)
b = np.ones(10) * 2
c = b * a
print c
d = c + 1
```

Easy to tweak with python codes

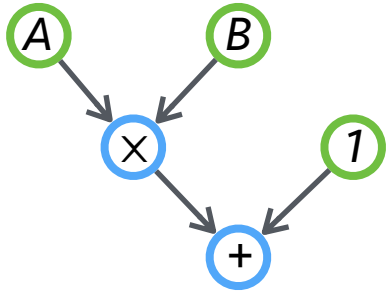
Pro

- Straightforward and flexible.
- Take advantage of language native features (loop, condition, debugger)

Con

- Hard to optimize

Declarative Programs




```
A = Variable('A')
B = Variable('B')
C = B * A
D = C + 1
f = compile(D)
d = f(A=np.ones(10),
      B=np.ones(10)*2)
```

C can share memory with *D*, because *C* is deleted later

Pro

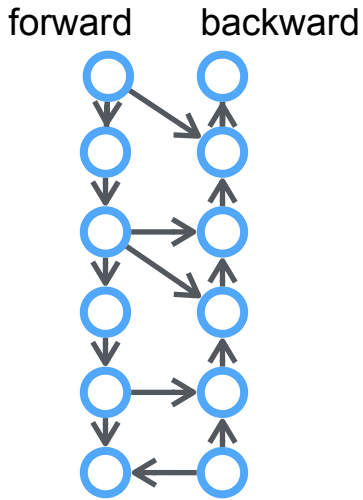
- More chances for optimization
- Cross different languages

Con

- Less flexible

Imperative vs. Declarative for Deep Learning

Computational Graph
of the Deep Architecture



Needs heavy optimization,
fits **declarative** programs

Updates and Interactions
with the graph

- Iteration loops
- Parameter update

$$w \leftarrow w - \eta \partial_w f(w)$$

- Beam search
- Feature extraction …

Needs mutation and more
language native features, good for
imperative programs

MXNet: Mix the Flavors Together

Imperative NDArray API

```
>>> import mxnet as mx
>>> a = mx.nd.zeros((100, 50))
>>> b = mx.nd.ones((100, 50))
>>> c = a + b
>>> c += 1
>>> print(c)
```

Declarative Symbolic Executor

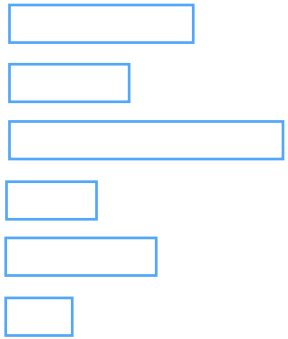
```
>>> import mxnet as mx
>>> net = mx.symbol.Variable('data')
>>> net = mx.symbol.FullyConnected(data=net, num_hidden=128)
>>> net = mx.symbol.SoftmaxOutput(data=net)
>>> texec = mx.module.Module(net)
>>> texec.forward(data=c)
>>> texec.backward()
```

Imperative NDArray can be set as input to the graph

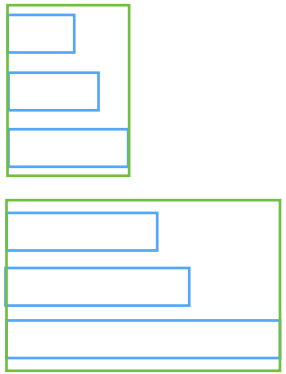
amazon web services

Mixed API for Quick Extensions

Variable length sentences



Bucketing



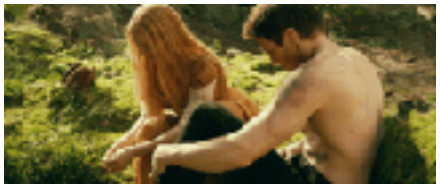
- Runtime switching between different graphs depending on input
- Useful for sequence modeling and image size reshaping
- Use of imperative code in Python, 10 lines of additional Python code

3D Image Construction

https://github.com/piiswrong/deep3d

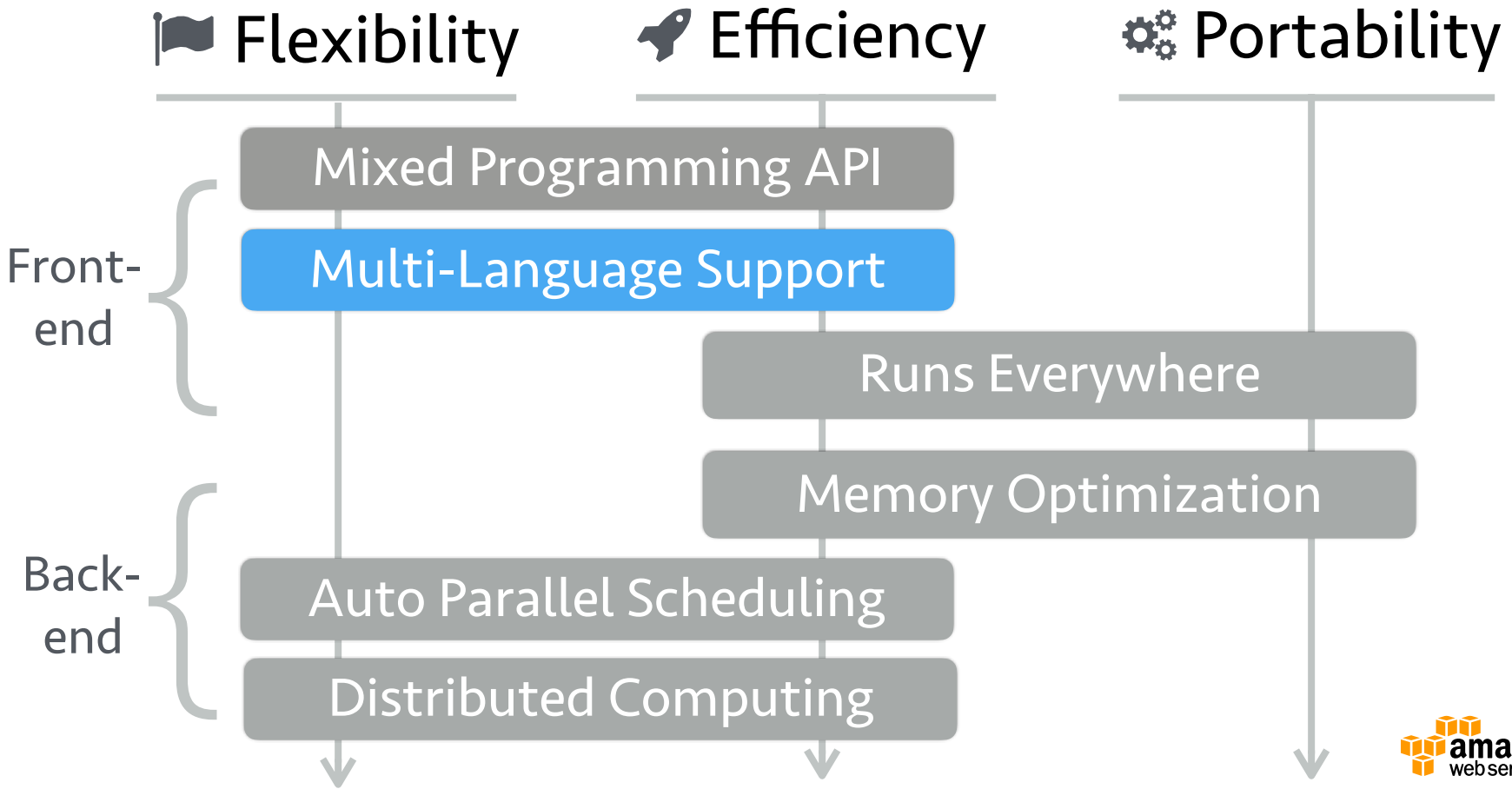
Deep3D

100 lines of Python code



amazon web services

MXNet Highlights



What We Heard from Users

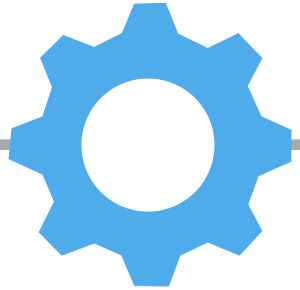
Programming Languages:

- Python is nice, but I like R/Julia/ Matlab more
- I want Scala to work with the Spark pipeline
- I need C++ interface to run on embedded systems
- I prefer Javascript to run on user browsers

Frameworks:

- I used Torch for 7 years
- All my codes are in Caffe
- I started deep learning with Tensorflow
- I only used Numpy before, how should I start?

Multiple Programming Languages



frontend

backend

single implementation of backend system and common operators

performance guarantee regardless of which frontend language is used

Bringing Caffe to MXNet

Caffe is widely used in computer vision

Call Caffe Operators in MXNet

```
import mxnet as mx
data = mx.symbol.Variable('data')
fc1  = mx.symbol.CaffeOp(data_0=data, num_weight=2, prototxt=
       "layer{type:\"InnerProduct\" inner_product_param{num_output: 128} }")
act1 = mx.symbol.CaffeOp(data_0=fc1, prototxt="layer{type:\"TanH\"}")
fc2  = mx.symbol.CaffeOp(data_0=act1, num_weight=2, prototxt=
       "layer{type:\"InnerProduct\" inner_product_param{num_output: 10}}")
mlp  = mx.symbol.SoftmaxOutput(data=fc3)
```

Bringing Torch to MXNet

Torch is a popular Lua framework for
both scientific computing and deep learning

Tensor Computation

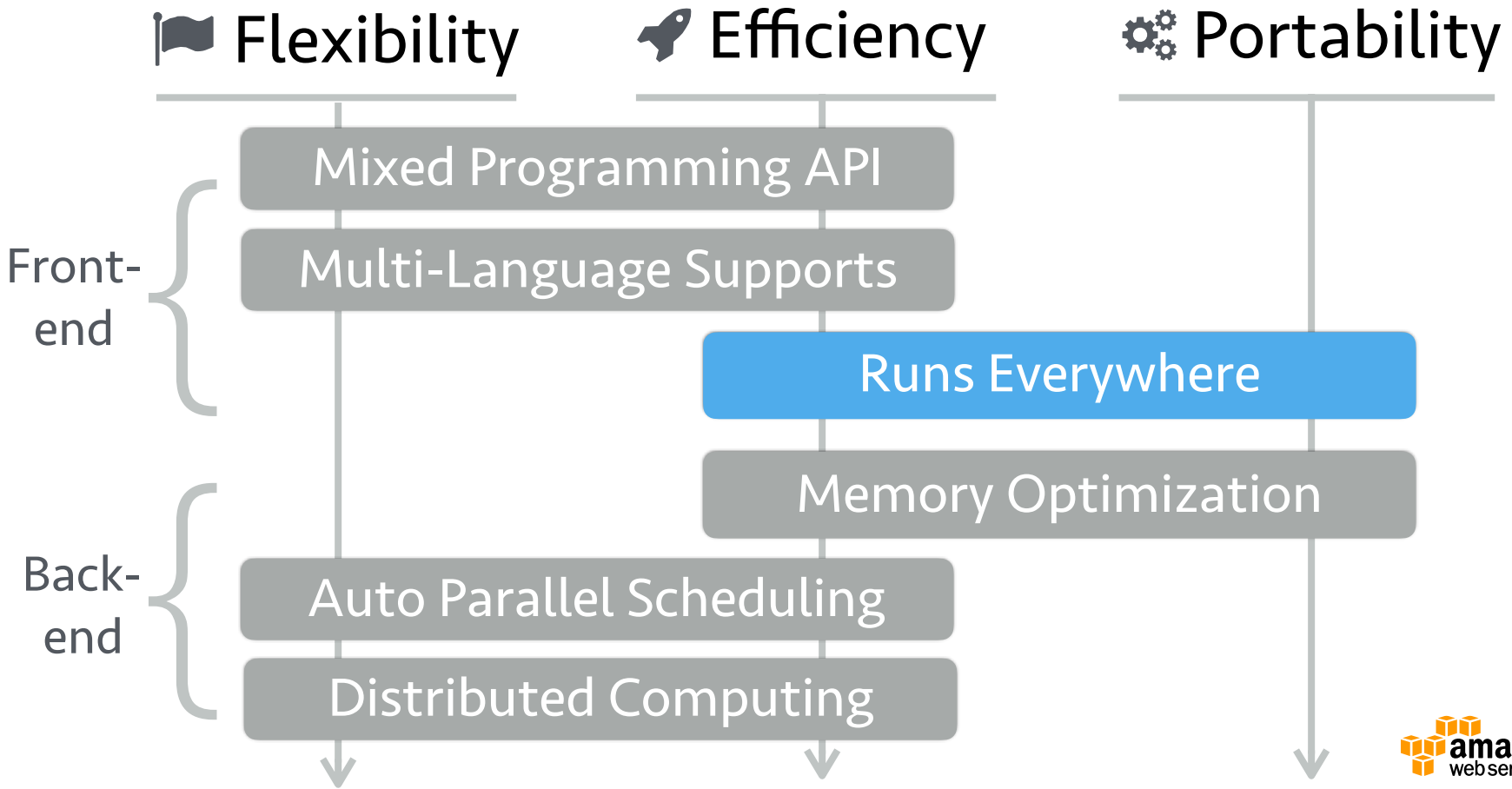
```
import mxnet as mx
x = mx.th.randn(2, 2, ctx=mx.gpu(0))
y = mx.th.abs(x)
print y.asnumpy()
```

Modules (Layers)

```
import mxnet as mx
data = mx.symbol.Variable('data')
fc   = mx.symbol.TorchModule(data_0=data, lua_string='nn.Linear(784, 128)',…
mlp  = mx.symbol.TorchModule(data_0=fc, lua_string='nn.LogSoftMax()',…
```

amazon web services

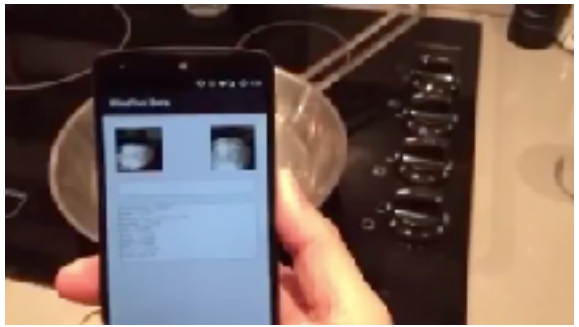
MXNet Highlights



Deploy Everywhere

Amalgamation

- Fit the core library with all dependencies into a single C++ source file
- Easy to compile on ...



BlindTool by Joseph Paul Cohen, demo on Nexus 4

Beyond

Runs in browser with Javascript

The first image for search “dog” at images.google.com

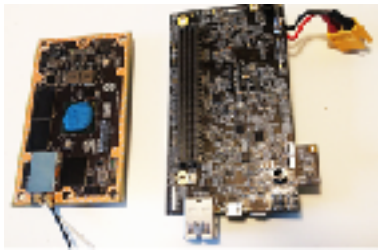
Outputs “beagle” with prob = 73% within 1 sec

TX1 on Flying Drone

AEVENA

图森 tuSimple

TX1 with customized board



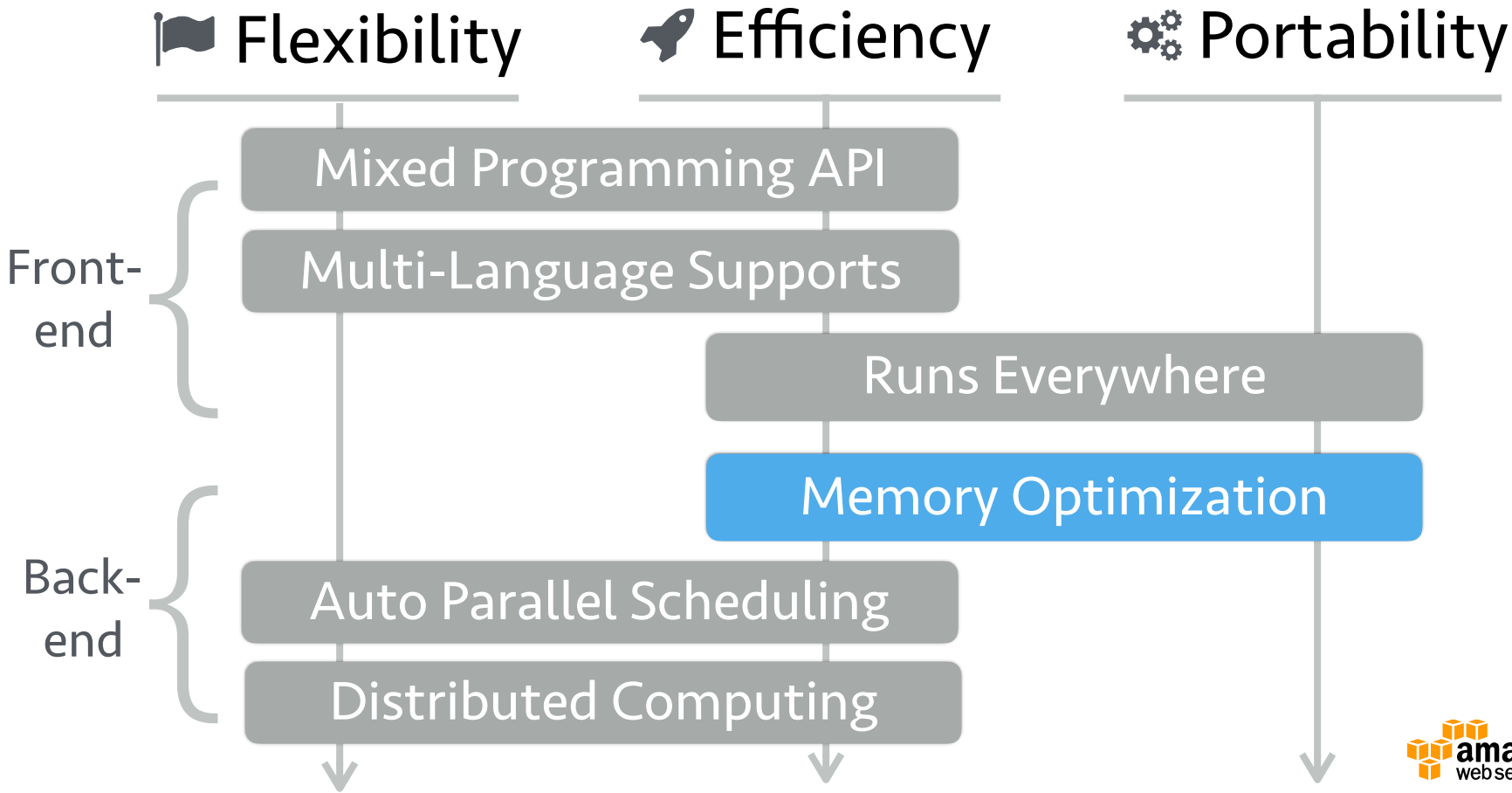
Drone

Realtime detection and tracking on TX1

~10 frame/sec with 640x480 resolution

amazon web services

MXNet Highlights



Backend

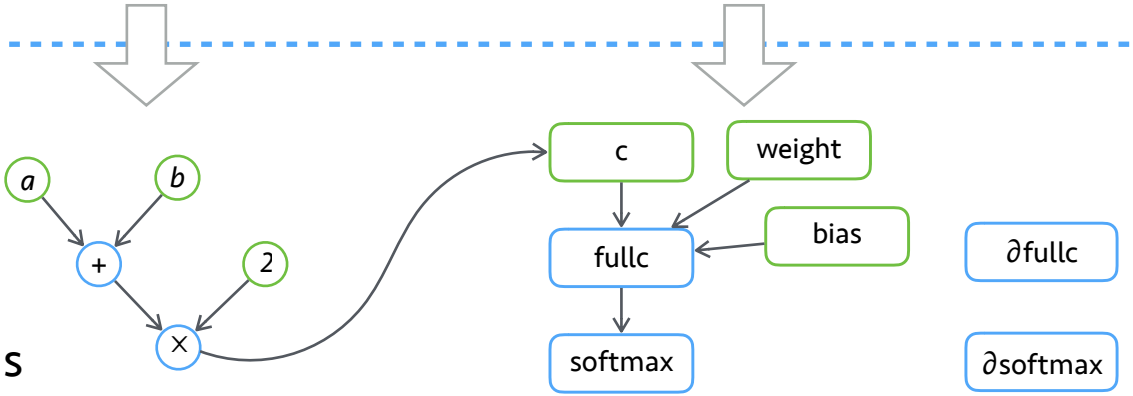
```
import mxnet as mx
a = mx.nd.ones((100, 50))
b = mx.nd.ones((100, 50))
c = a + b
c *= 2
```

```
import mxnet as mx
net = mx.symbol.Variable('data')
net = mx.symbol.FullyConnected(
        data=net, num_hidden=128)
net = mx.symbol.SoftmaxOutput(data=net)
texec = mx.module.Module(net)
texec.forward(data=c)
texec.backward()
```

Front-end

Back-end

NNVM

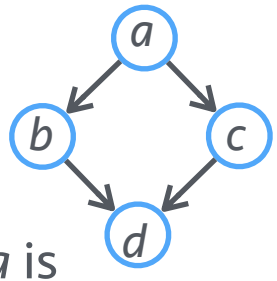


A lot of optimizations happen here

Memory Optimization

Traverse the computation graph to reduce the memory footprint with linear time complexity

aliveness analysis

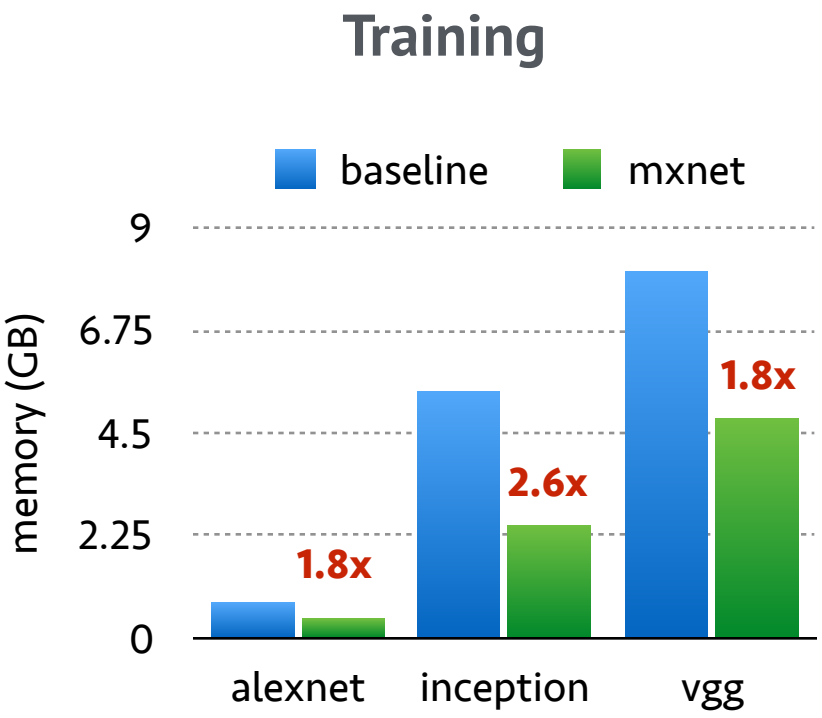


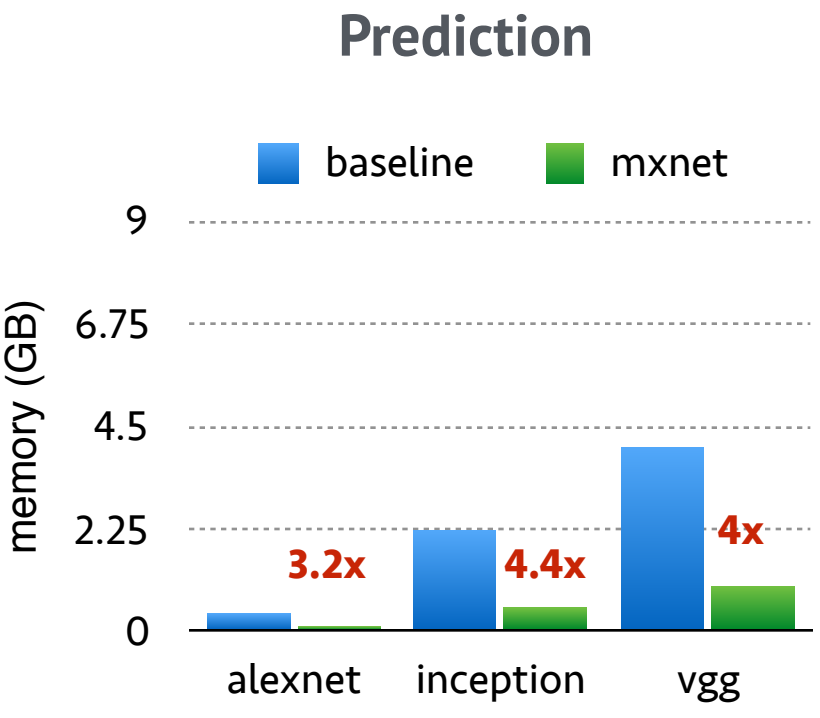
now *a* is deletable

shared space between variables

share *a* and *b*

Results for Deep CNNs on IMAGENET





amazon web services

Neural Art

1M pixels
GTX 980 TI 6G

(in 20x speed)

amazon web services

Trading Computation for Memory

ResNet

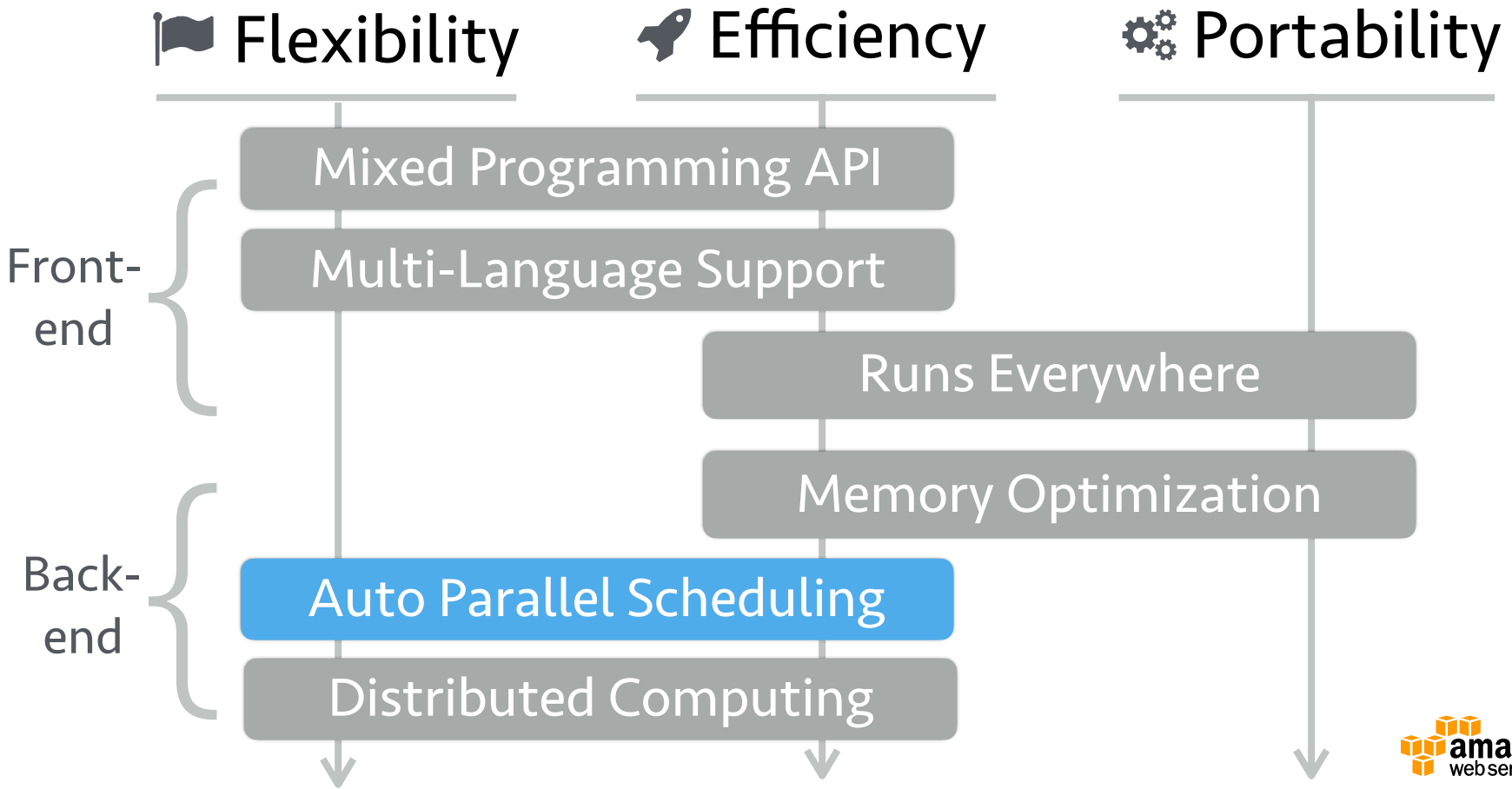
- 1000 layers
- batch size 32

LSTM

- 4 layers
- 1000 hidden size
- 1000 unroll
- batch size 32

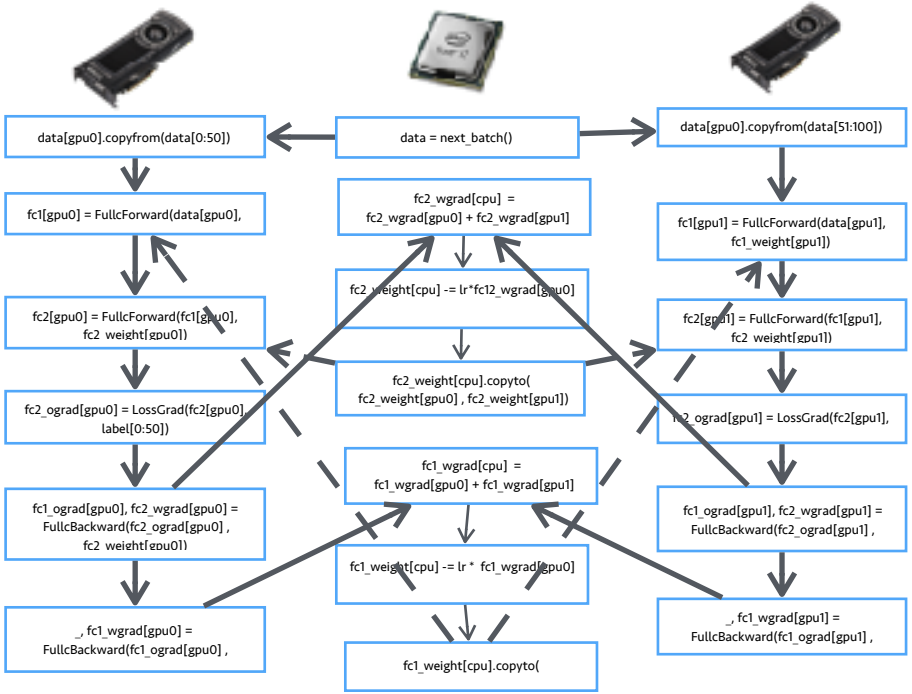
	Before	After
Resnet	130 GB	4 GB
LSTM	270 GB	2.5 GB

MXNet Highlights



Writing Parallel Programs is Painful

Dependency graph for 2-layer neural networks with 2 GPUs



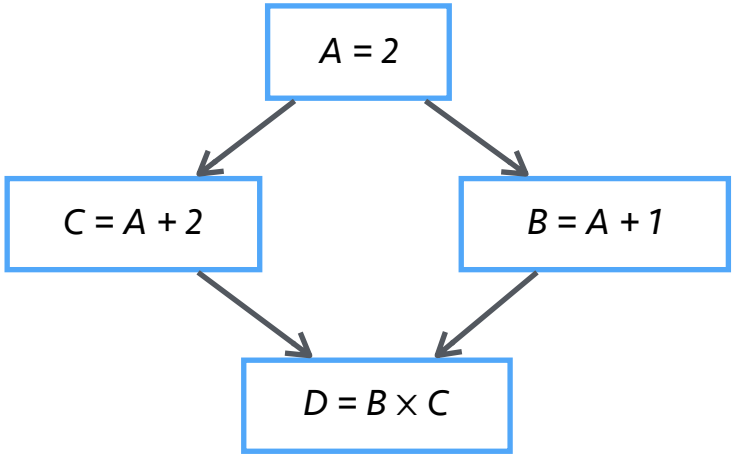
Each forward-backward-update involves O(num_layer), which is often 100—1,000, tensor computations and communications

Auto Parallelization

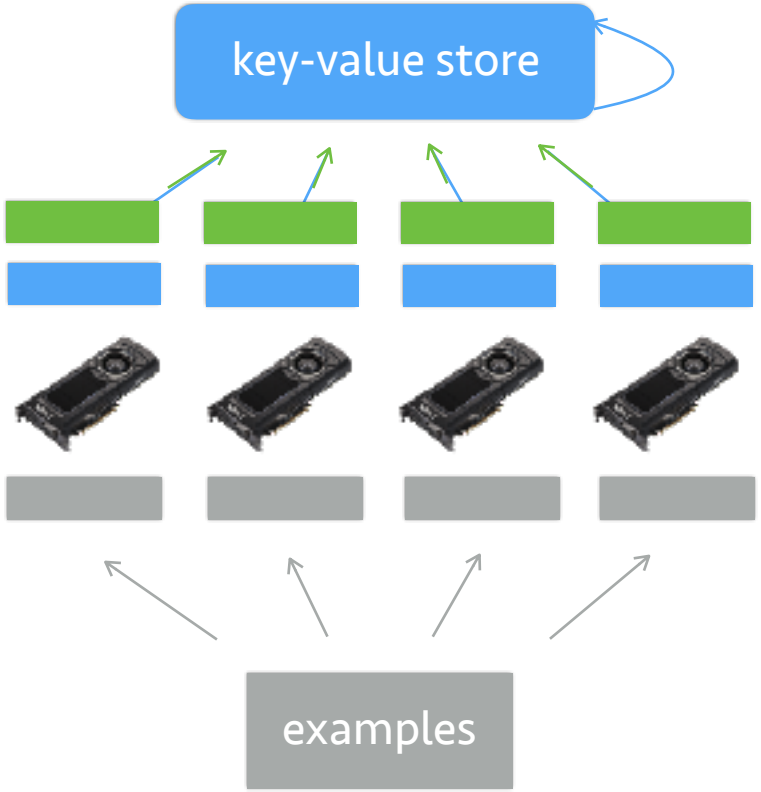
Write **serial** programs

```
import mxnet as mx
A = mx.nd.ones((2,2)) *2
C = A + 2
B = A + 1
D = B * C
```

Run in **parallel**



Data Parallelism



1. Read a data partition
2. Pull the parameters
3. Compute the gradient
4. Push the gradient
5. Update the parameters

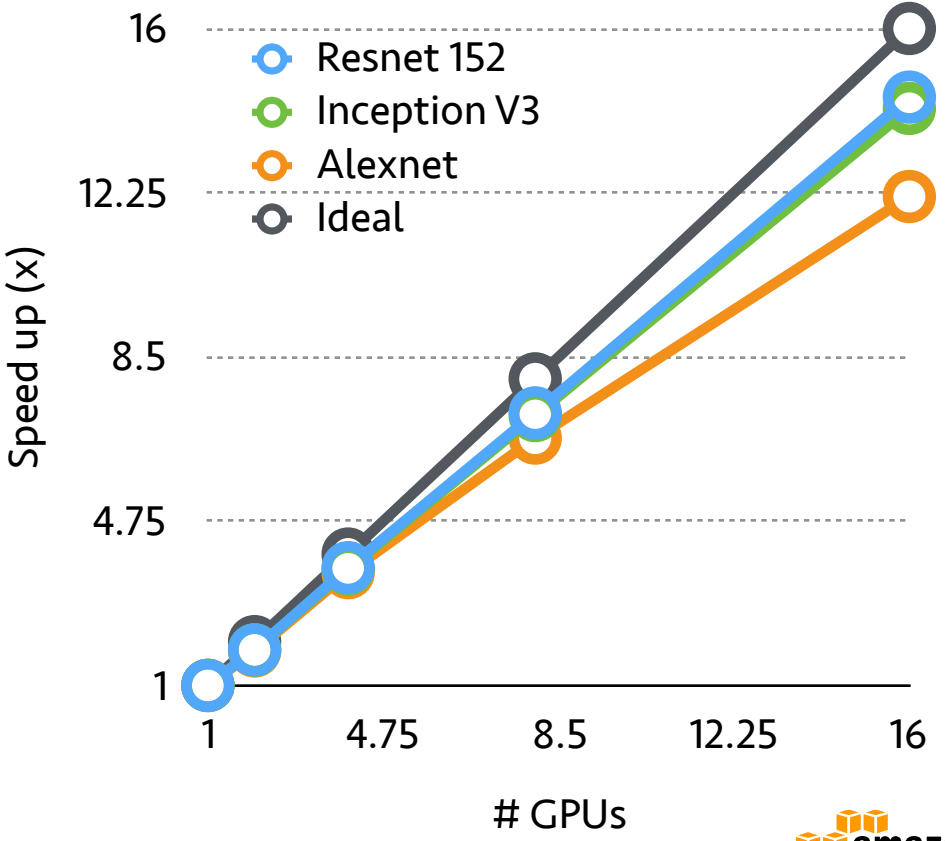
Scalability on Multiple GPUs

AWS EC2 P2.16xlarge

8 Nvidia Tesla K80 (16 GPUs)

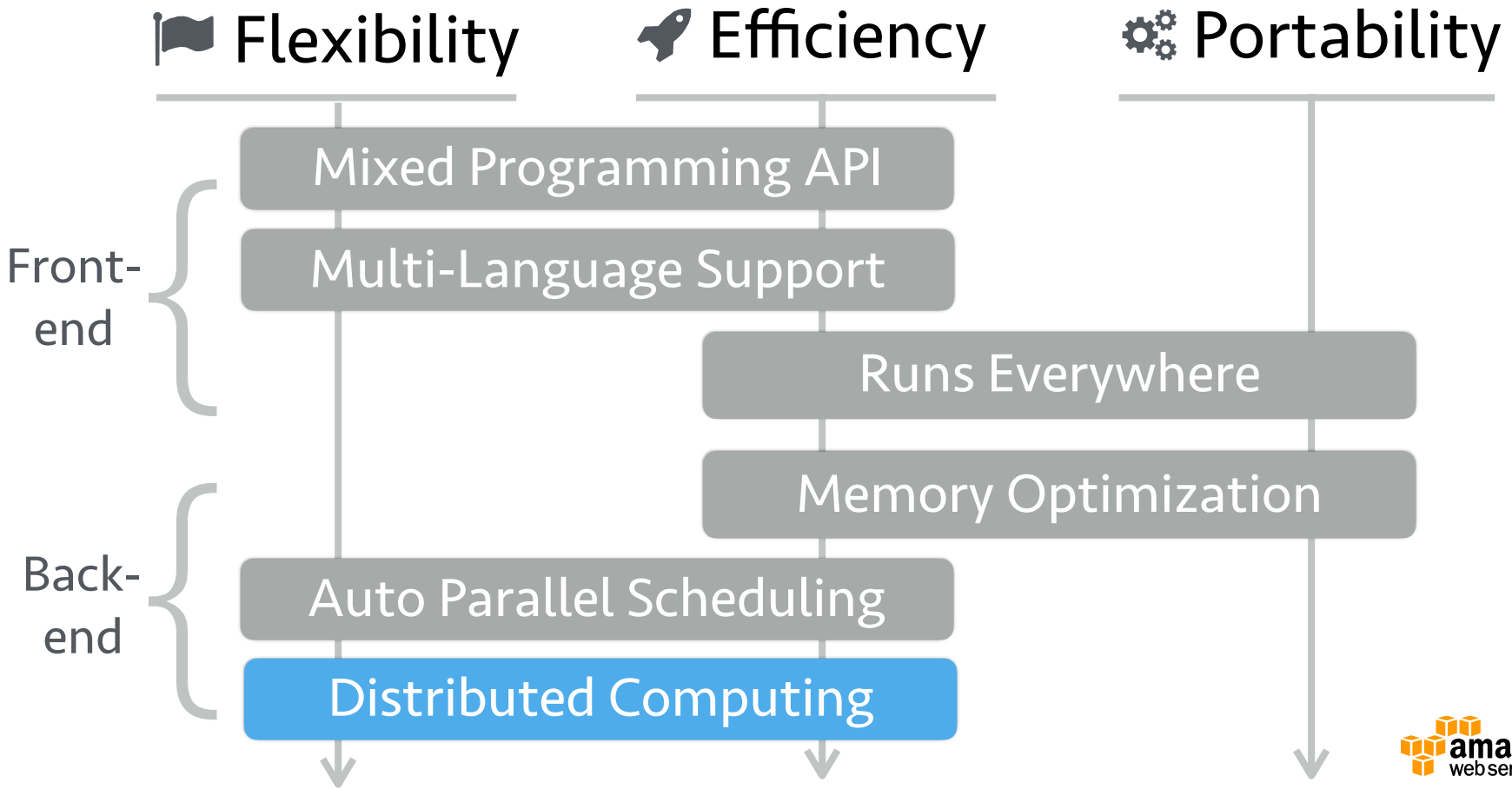
Synchronized SGD

IMAGENET

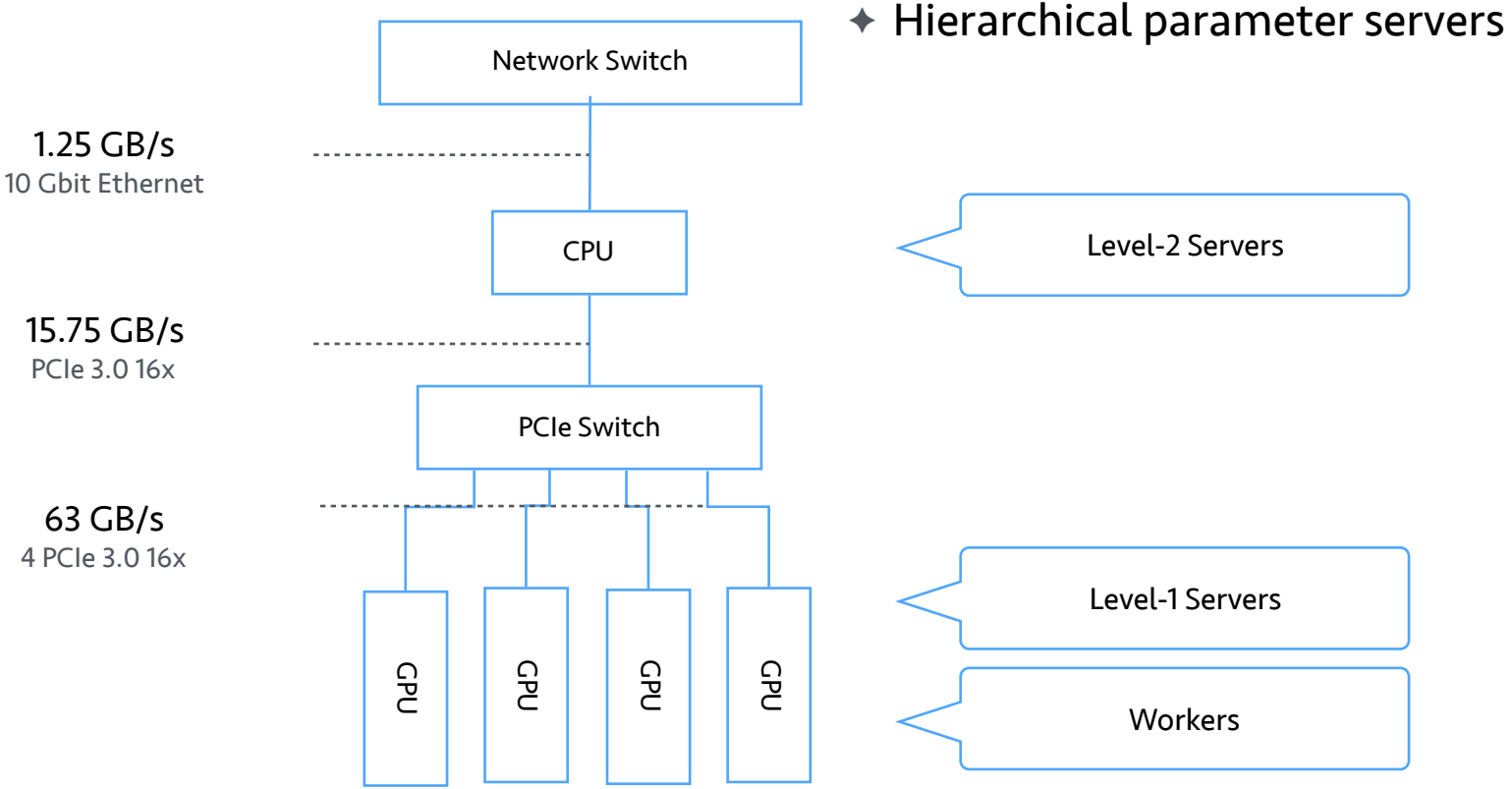


amazon web services

MXNet Highlights



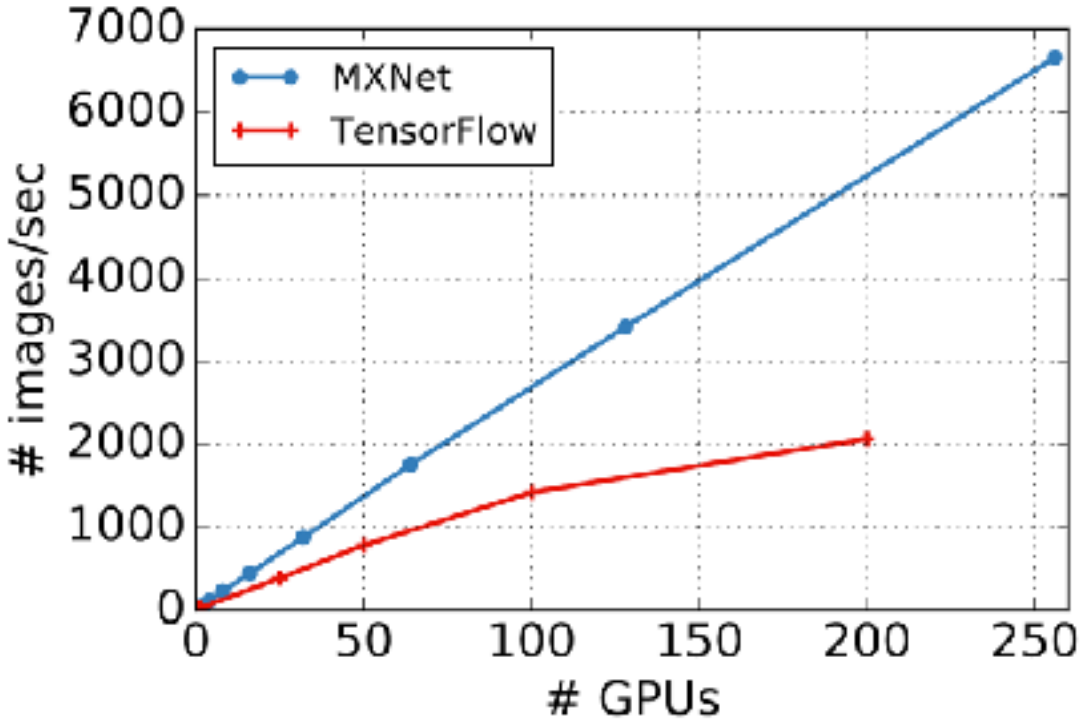
Implementation



- Hierarchical parameter servers

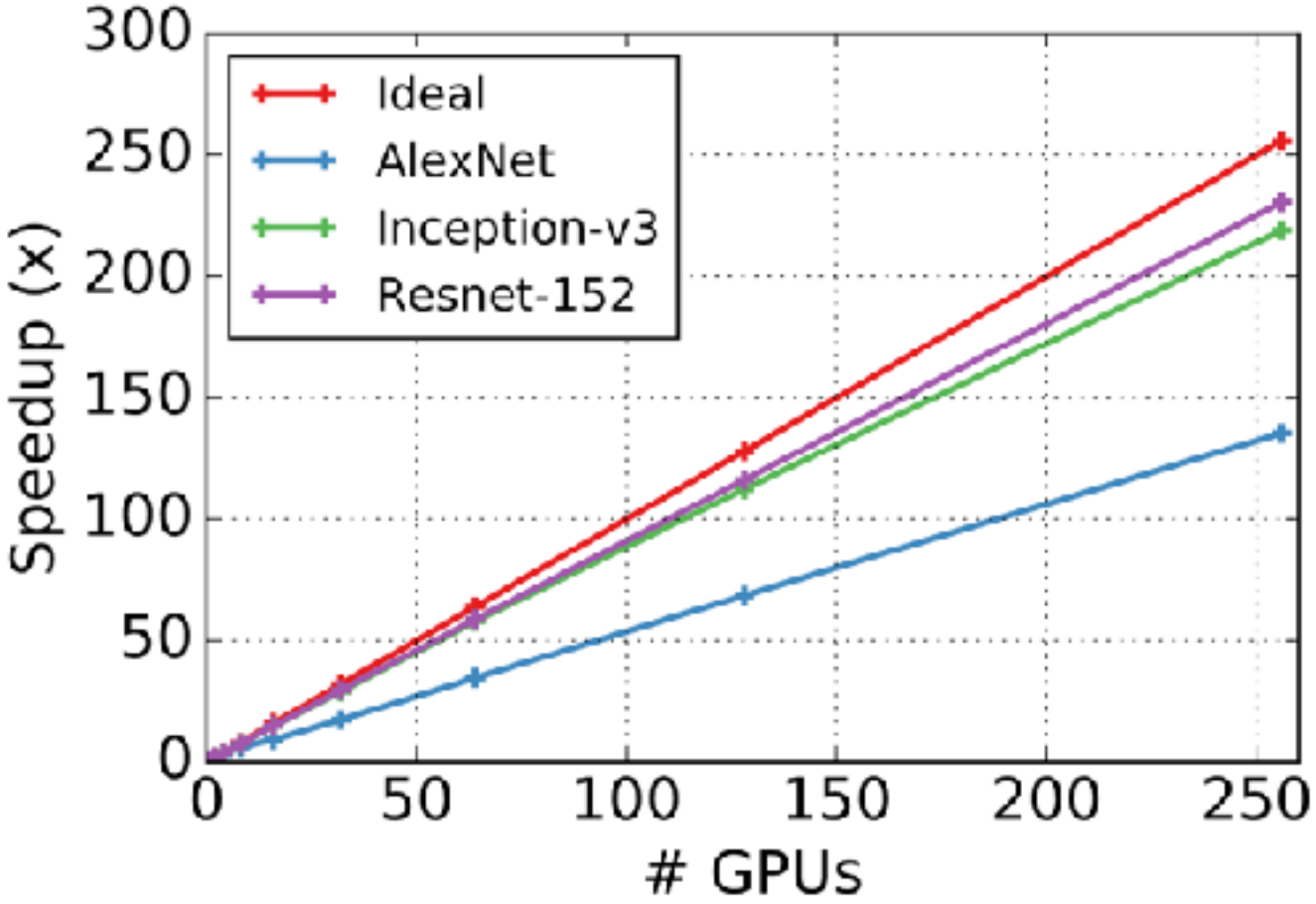
Distributed Experiments

- Google Inception v3
- Increasing machines from 1 to 47
- 2x faster than TensorFlow if using more than 10 machines



amazon web services

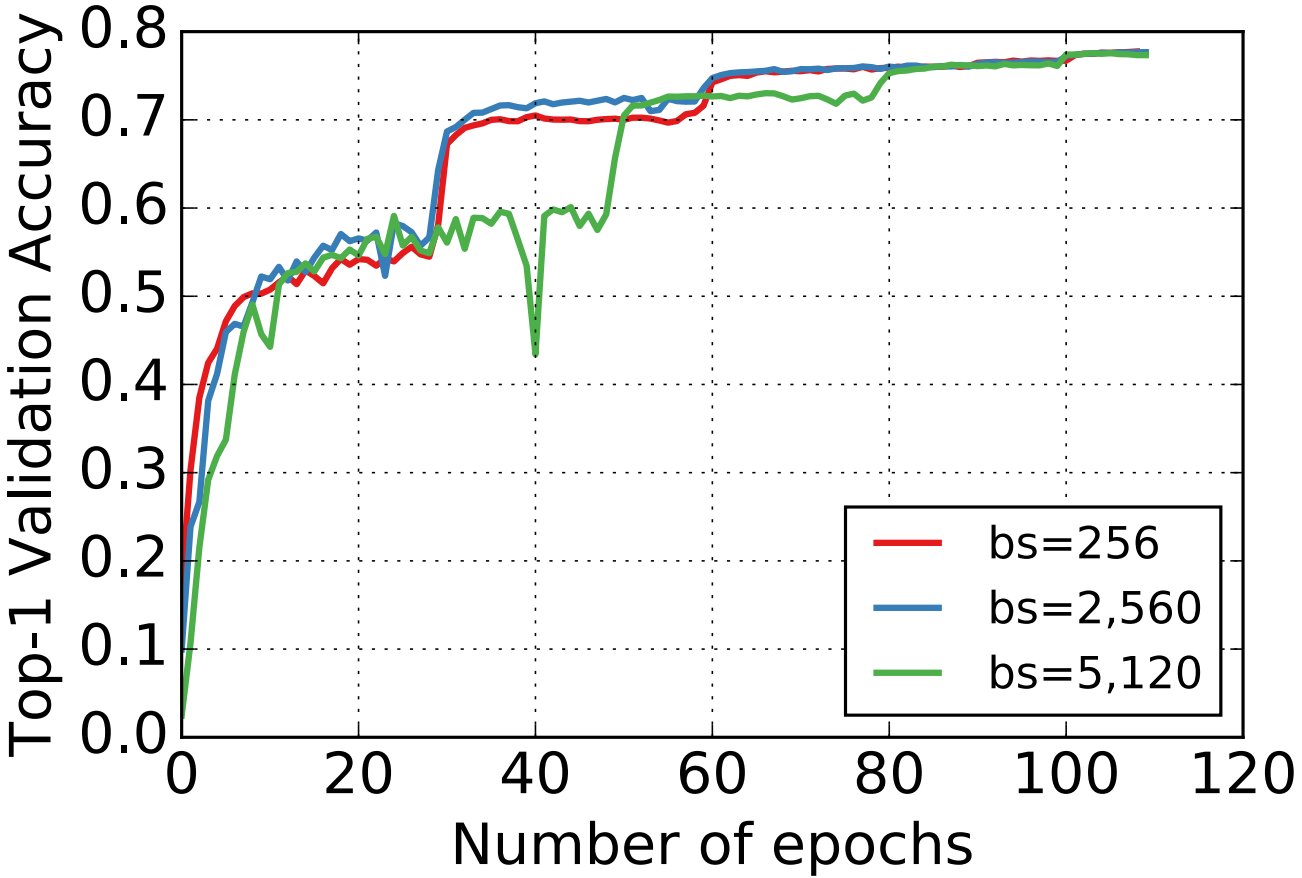
Distributed Training Speedup



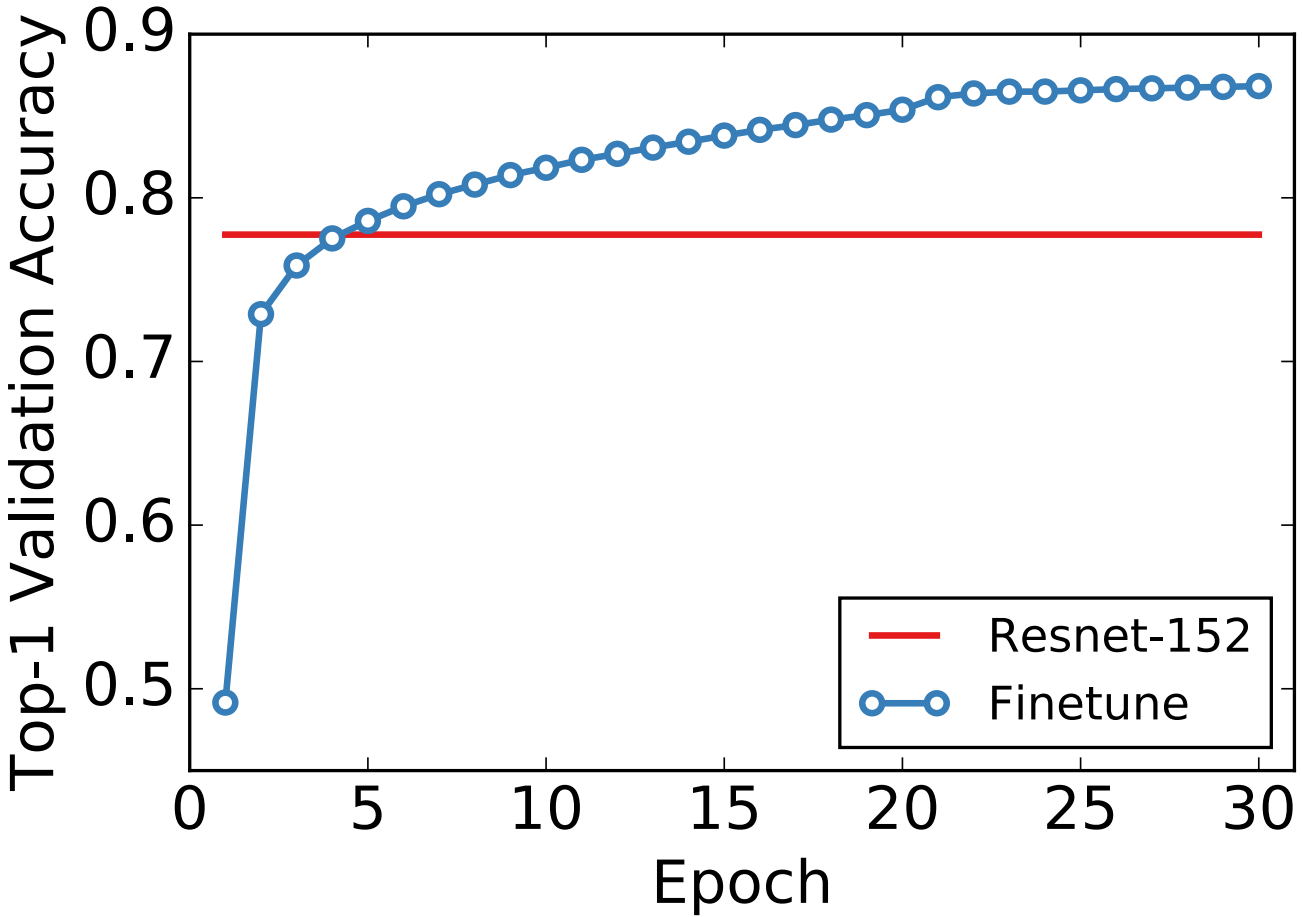
- Cloud formation with Deep Learning AMI
- 16x P2.16xlarge
- Mounted on EFS
- ImageNet
  1.2M images
  1K classes
- 152-layer ResNet
  5.4d on 4x K80s
  (1.2h per epoch)
  0.22 top-1 error

amazon web services

Distributed Training Convergence



Fine tuning for distributed inference



Modules

1. **Introduction to Deep Learning**
2. **Getting Started with MXNet**
    - Cloud - AMIs and CloudFormation Template
    - Laptop - build from source
3. **MXNet**
    - NDArrays and Symbols
    - Training Deep Networks
4. **Examples**
    - Computer Vision
    - Natural Language Processing
    - Recommender Systems

amazon web services

4. Examples

- NDArray, Symbols
- Parallel Training
- Computer Vision
- Natural Language Processing
- Recommender Systems

Python notebooks

https://github.com/dmlc/mxnet-notebooks

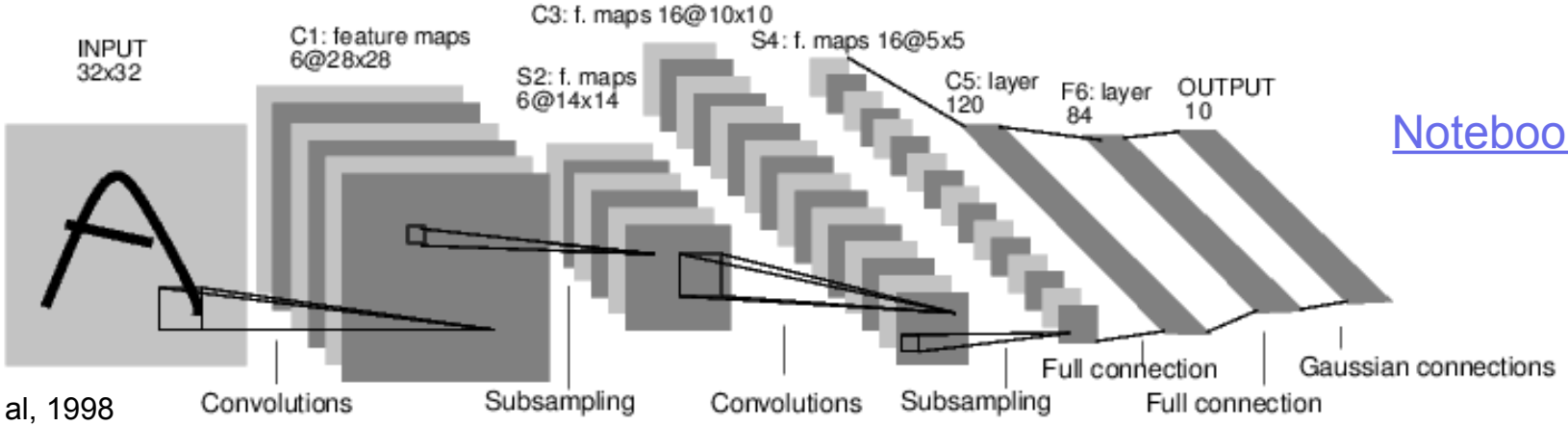
Basic concepts

- NDArray - multi-dimensional array computation
- Symbol - symbolic expression for neural networks
- Module - neural network training and inference

Applications

- MNIST: recognize handwritten digits
- Check distributed training results
- Predict with pre-trained models
- LSTMs for sequence learning
- Recommender systems

Notebook - MNIST and LeNet



Le Cun et al, 1998

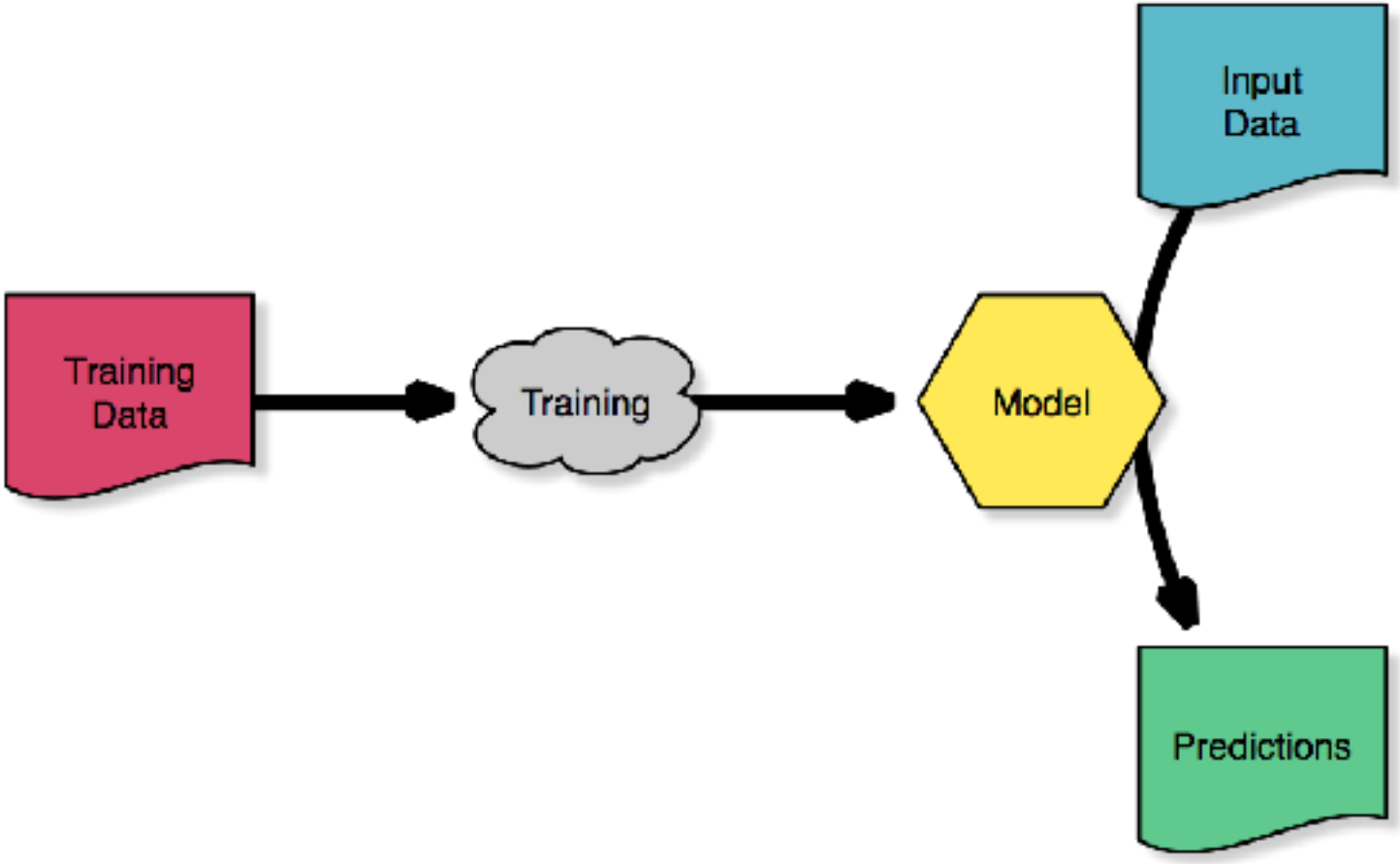
Notebook

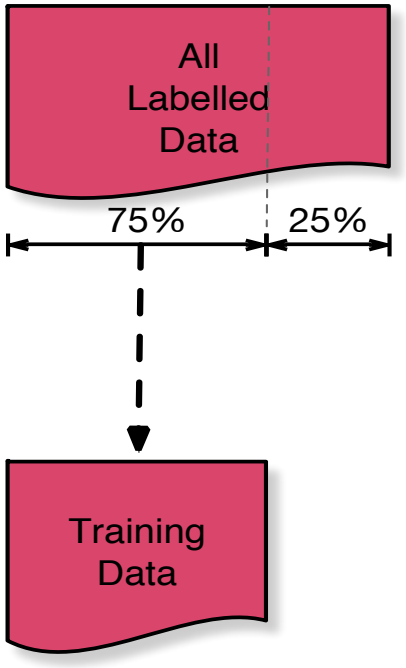
```
fc1  = mx.symbol.FullyConnected(data = data, name='fc1', num_hidden=128)
act1 = mx.symbol.Activation(data = fc1, name='relu1', act_type="relu")
fc2  = mx.symbol.FullyConnected(data = act1, name = 'fc2', num_hidden = 64)
act2 = mx.symbol.Activation(data = fc2, name='relu2', act_type="relu")
fc3  = mx.symbol.FullyConnected(data = act2, name='fc3', num_hidden=10)
mlp  = mx.symbol.SoftmaxOutput(data = fc3, name = 'softmax')
```

https://github.com/dmlc/mxnet/tree/master/example/image-classification

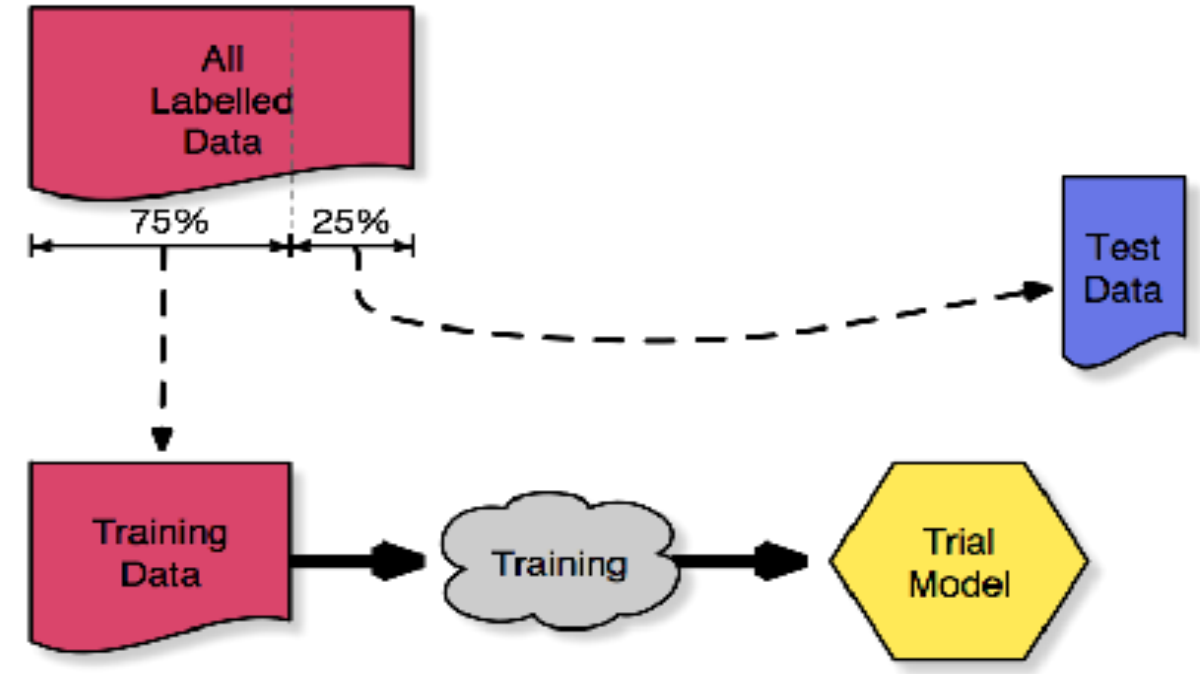
amazon web services

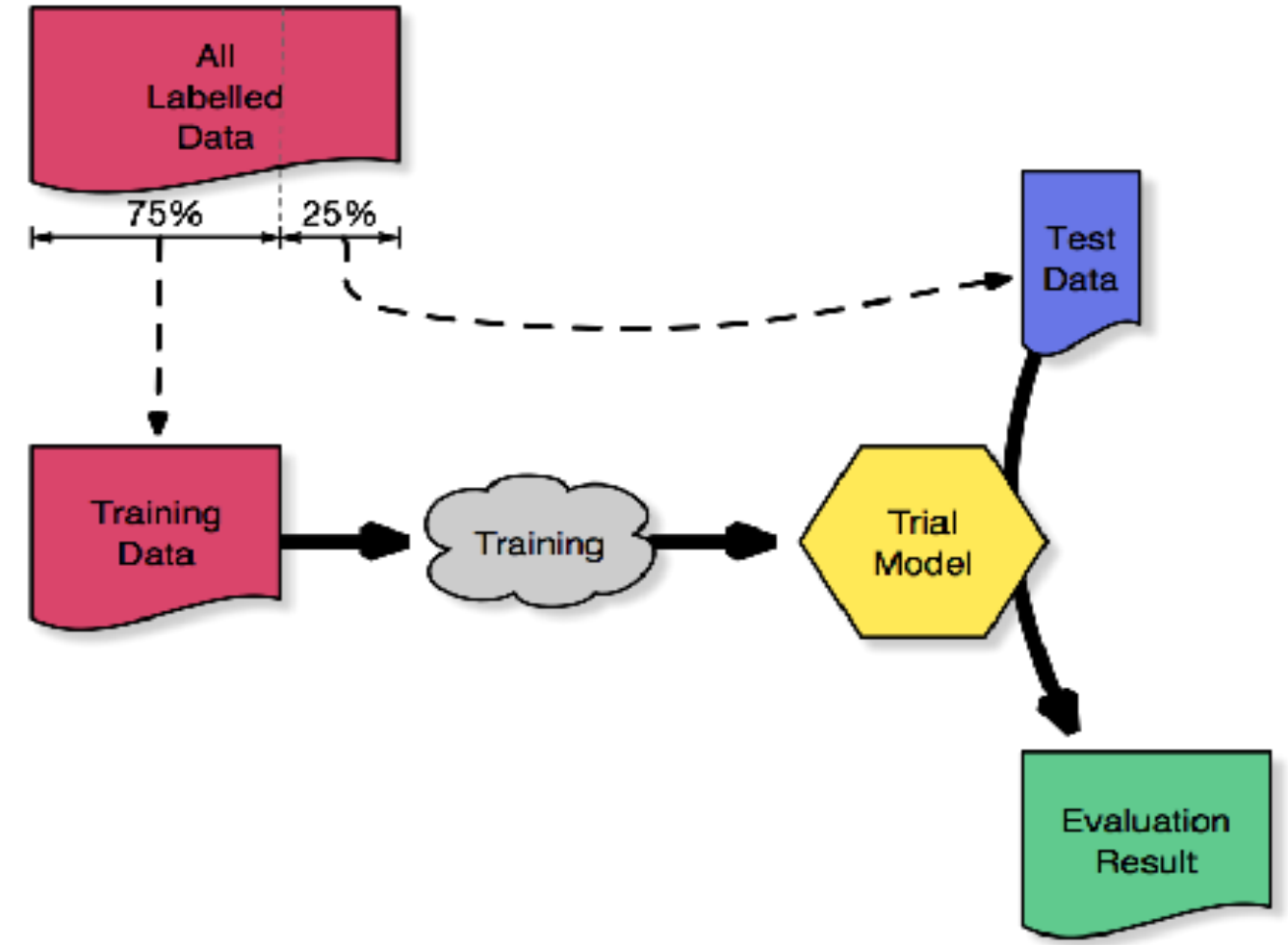
Training a machine learning model

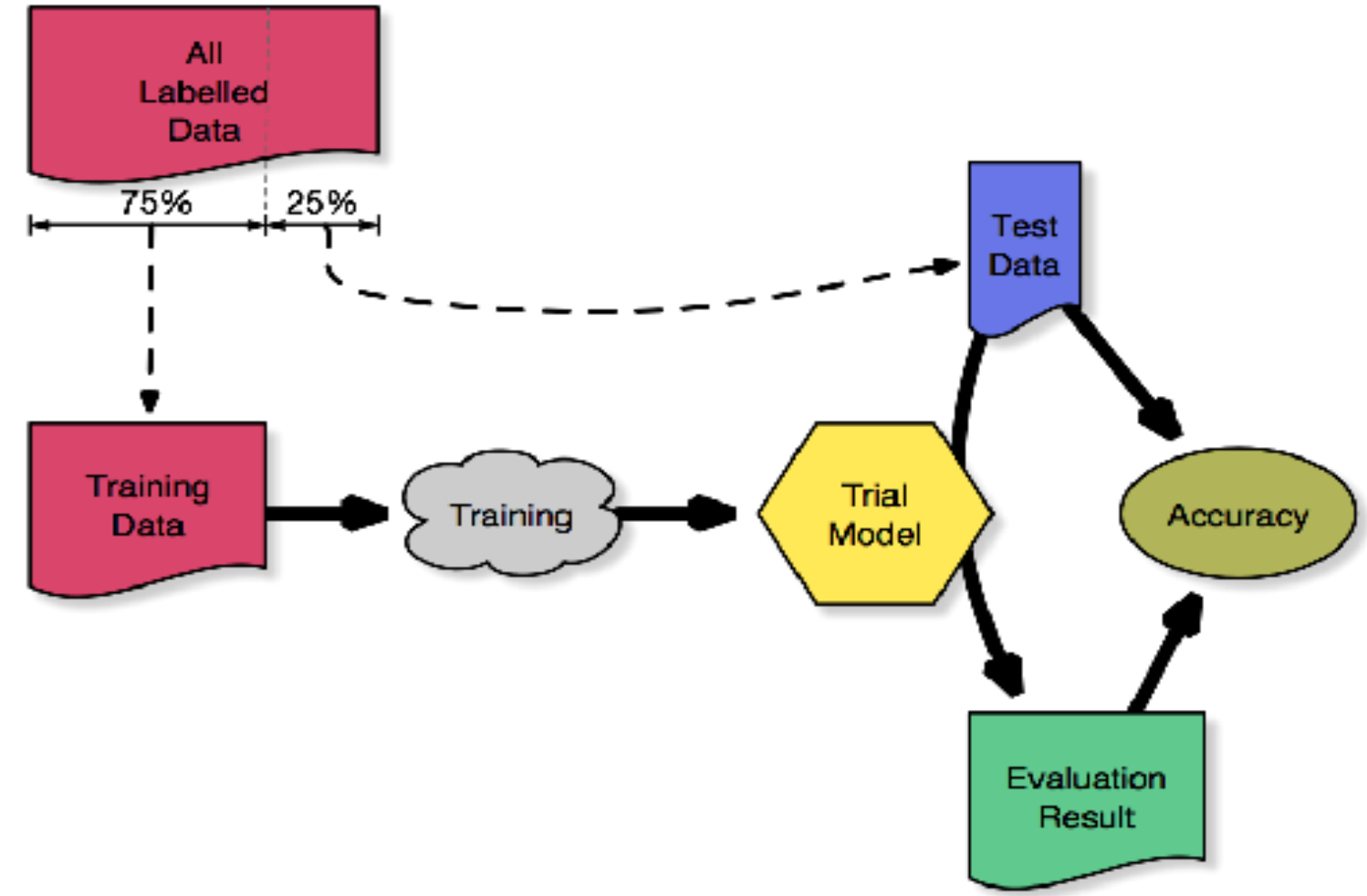



All
Labelled
Data
75%
25%
Training
Data


All
Labelled
Data
75%
25%
Training
Data
Training
Trial
Model

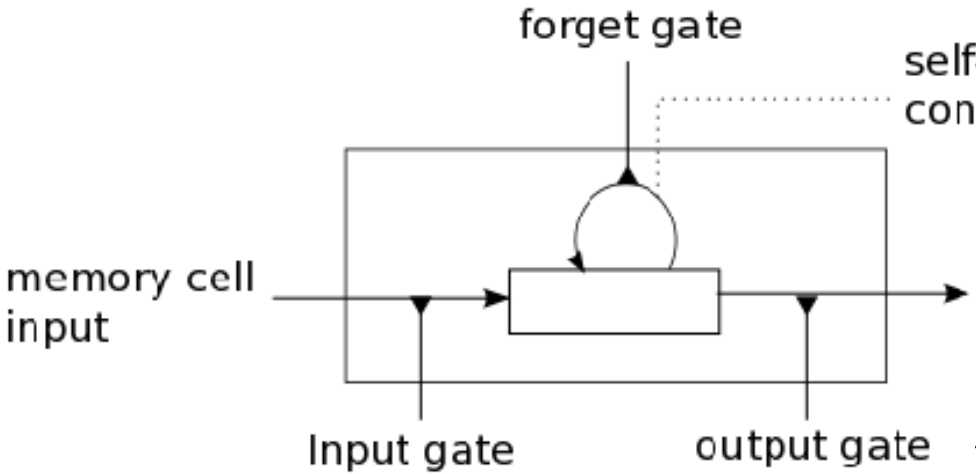

All
Labelled
Data
75%
25%
Test
Data
Training
Data
Training
Trial
Model


All
Labelled
Data
75%
25%
Test
Data
Training
Data
Training
Trial
Model
Evaluation
Result


All
Labelled
Data
75%
25%
Test
Data
Training
Data
Training
Trial
Model
Accuracy
Evaluation
Result

Notebook - Long Short Term Memory

Schmidhuber and Hochreiter, 1998



input

$$i_t = \sigma(W_i(x_t, h_t) + b_i)$$

forgetting

$$f_t = \sigma(W_f(x_t, h_t) + b_f)$$

$$z_{t+1} = f_t \cdot z_t + i_t \cdot \tanh(W_z(x_t, h_t) + b_z)$$

output

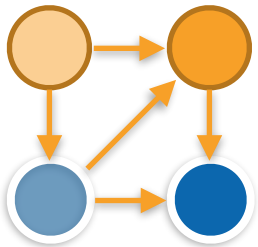
$$o_t = \sigma(W_o(x_t, h_t, z_{t+1}) + b_o)$$

$$h_{t+1} = o_t \cdot \tanh z_{t+1}$$

state

h

x



Recommender Systems

Total price: $233.42

Add all three to Cart

Add all three to List

- This item: Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series) by Kevin P. Murphy Hardcover $92.37
- Pattern Recognition and Machine Learning (Information Science and Statistics) by Christopher Bishop Hardcover $68.03
- The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition... by Trevor Hastie Hardcover $73.02

Customers Who Bought This Item Also Bought

Pattern Recognition and Machine Learning (Information Science and...
Christopher Bishop
★★★★☆ 129
Hardcover
$68.03 Prime

The Elements of Statistical Learning: Data Mining, Inference, and...
Trevor Hastie
★★★★☆ 83
Hardcover
$73.02 Prime

Deep Learning (Adaptive Computation and Machine Learning series)
› Ian Goodfellow
★★★★★ 18
#1 Best Seller in Artificial Intelligence...
Hardcover
$72.00 Prime

Probabilistic Graphical Models: Principles and Techniques (Adaptive...
› Daphne Koller
★★★★☆ 37
Hardcover
$109.46 Prime

Learning From Data
› Yaser S. Abu-Mostafa
★★★★½ 124
Hardcover

An Introduction to Statistical Learning: with Applications in R...
› Gareth James
★★★★★ 132
Hardcover
$70.36 Prime

amazon web services

Recommender Systems

Users u, movies m (or projects)

Function class

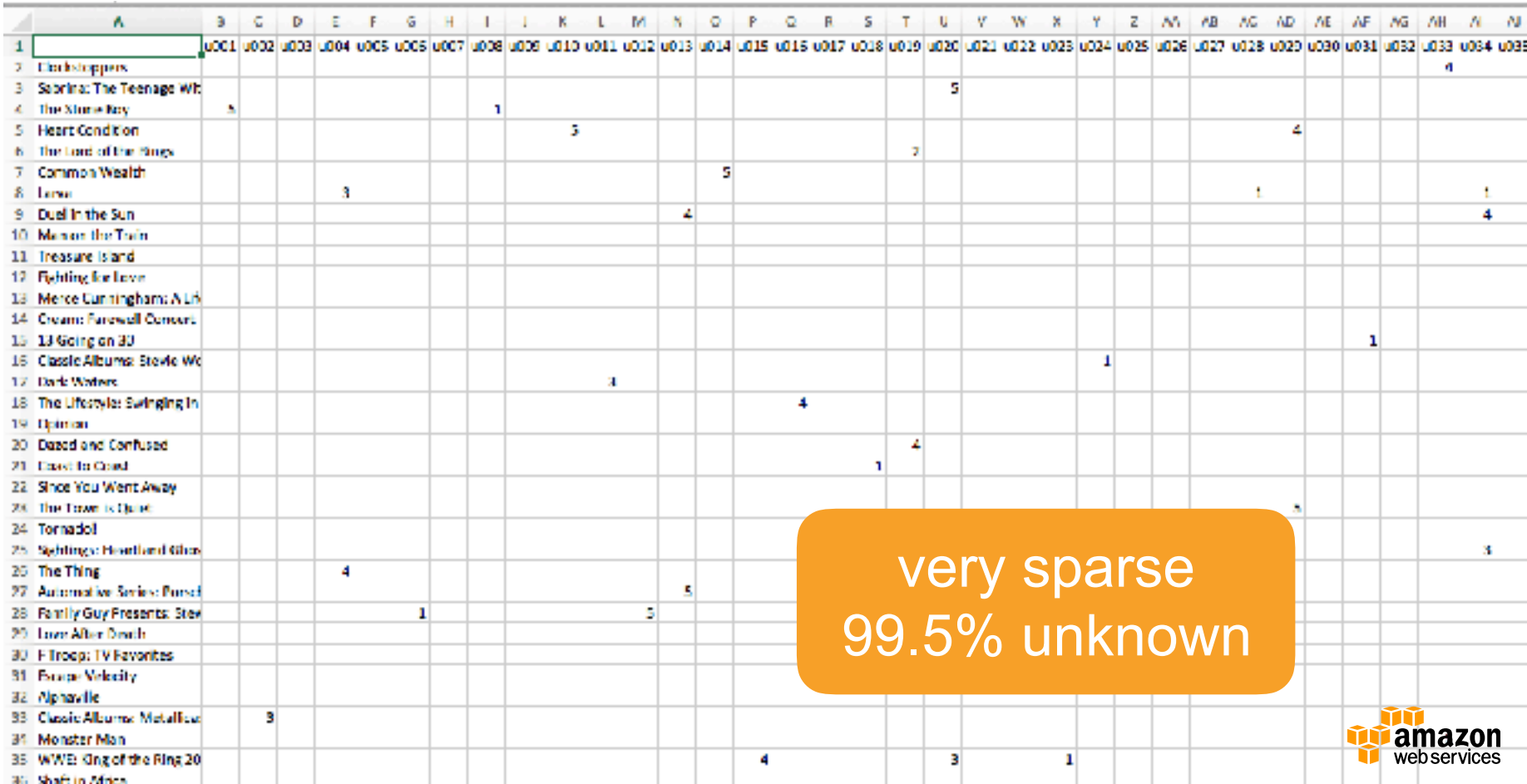
$$r_{um} = \langle v_u, w_m \rangle + b_u + b_m$$

Loss function for recommendation (Yelp, Netflix)

$$\sum_{u \sim m} \left(\langle v_u, w_m \rangle + b_u + b_m - y_{um} \right)^2$$

amazon web services

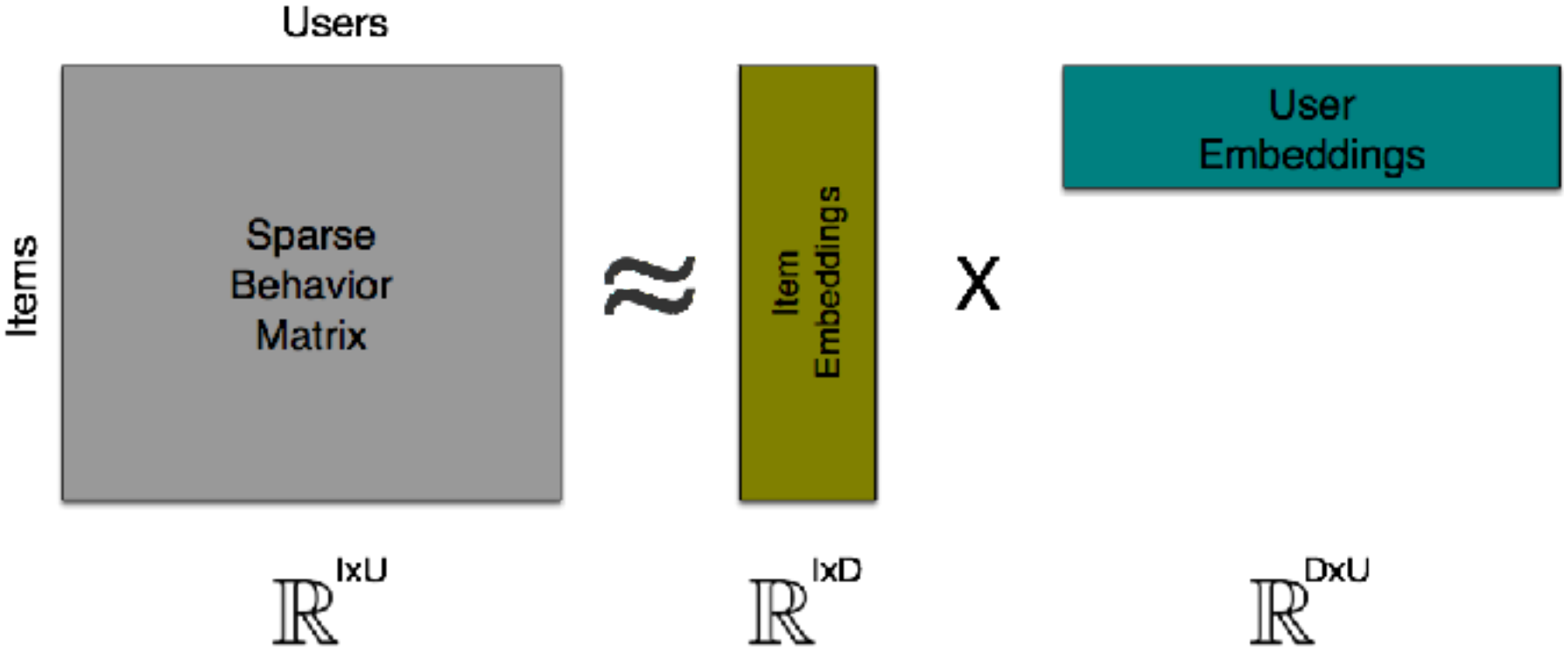
User-Item Ratings Matrix



very sparse
99.5% unknown

amazon web services

Matrix Factorization as Math



Recommender Systems

Regularized Objective

$$\sum_{u \sim m} \left(\langle v_u, w_m \rangle + b_u + b_m + b_0 - r_{um} \right)^2 + \frac{\lambda}{2} \left[\|U\|_{\text{Frob}}^2 + \|V\|_{\text{Frob}}^2 \right]$$

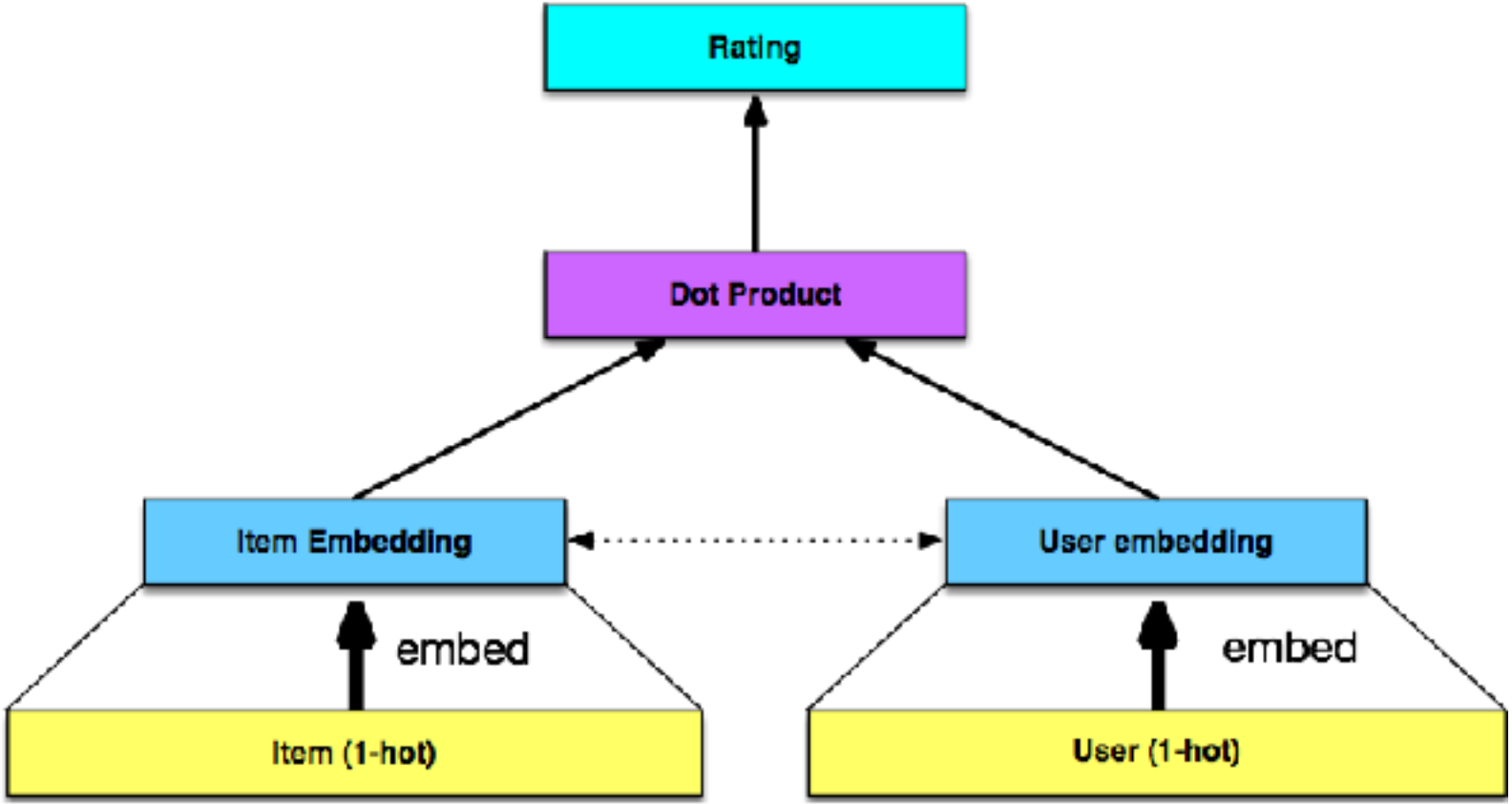
Update operations

$$v_u \leftarrow (1 - \eta_t \lambda) v_u - \eta_t w_m \left(\langle v_u, w_m \rangle + b_u + b_m + b_0 - r_{um} \right)$$
$$w_m \leftarrow (1 - \eta_t \lambda) w_m - \eta_t v_u \left(\langle v_u, w_m \rangle + b_u + b_m + b_0 - r_{um} \right)$$

Very simple SGD algorithm (random pairs)

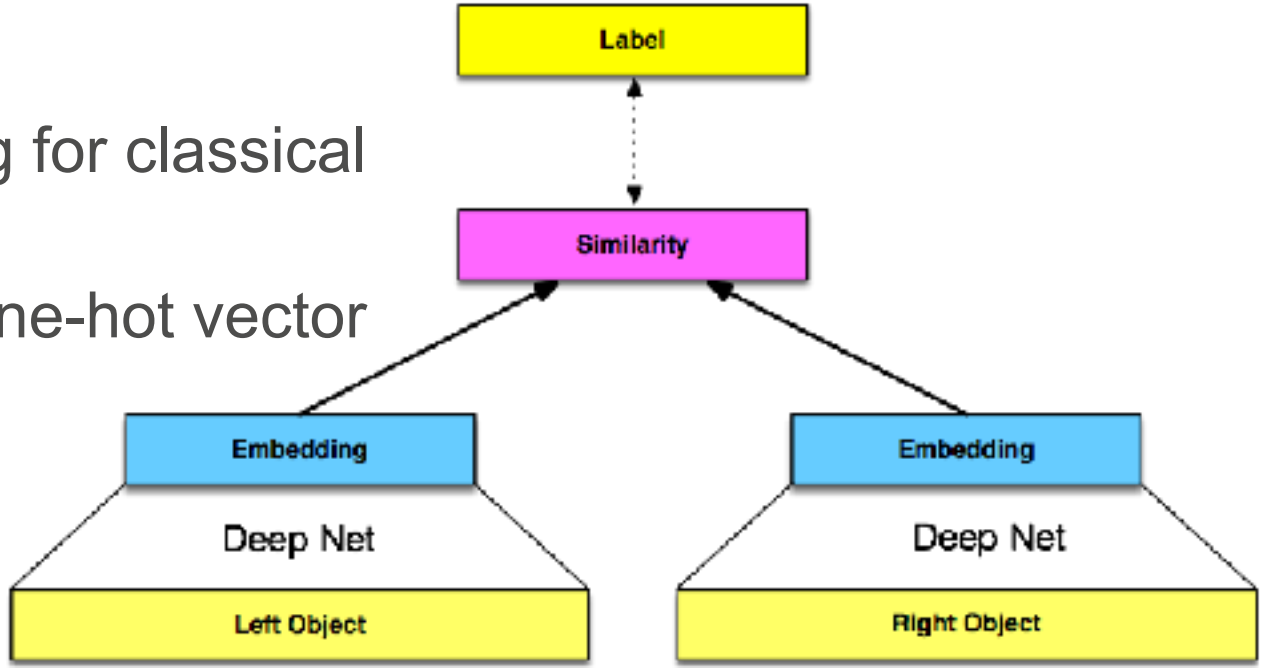
This should be cheap …

Matrix Factorization as Deep Network



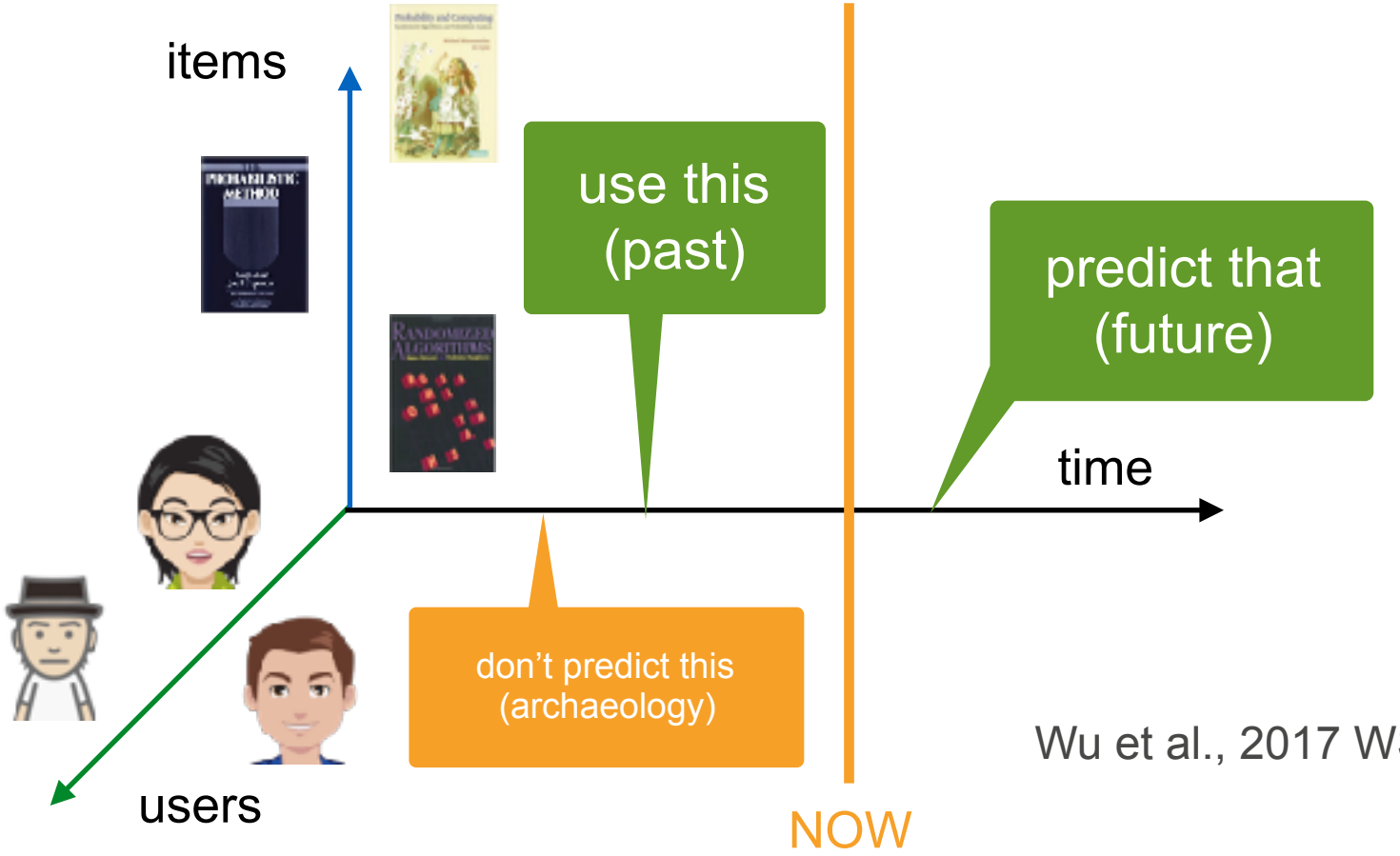
Adding Features

- One-hot encoding for classical recommenders
- Add features to one-hot vector

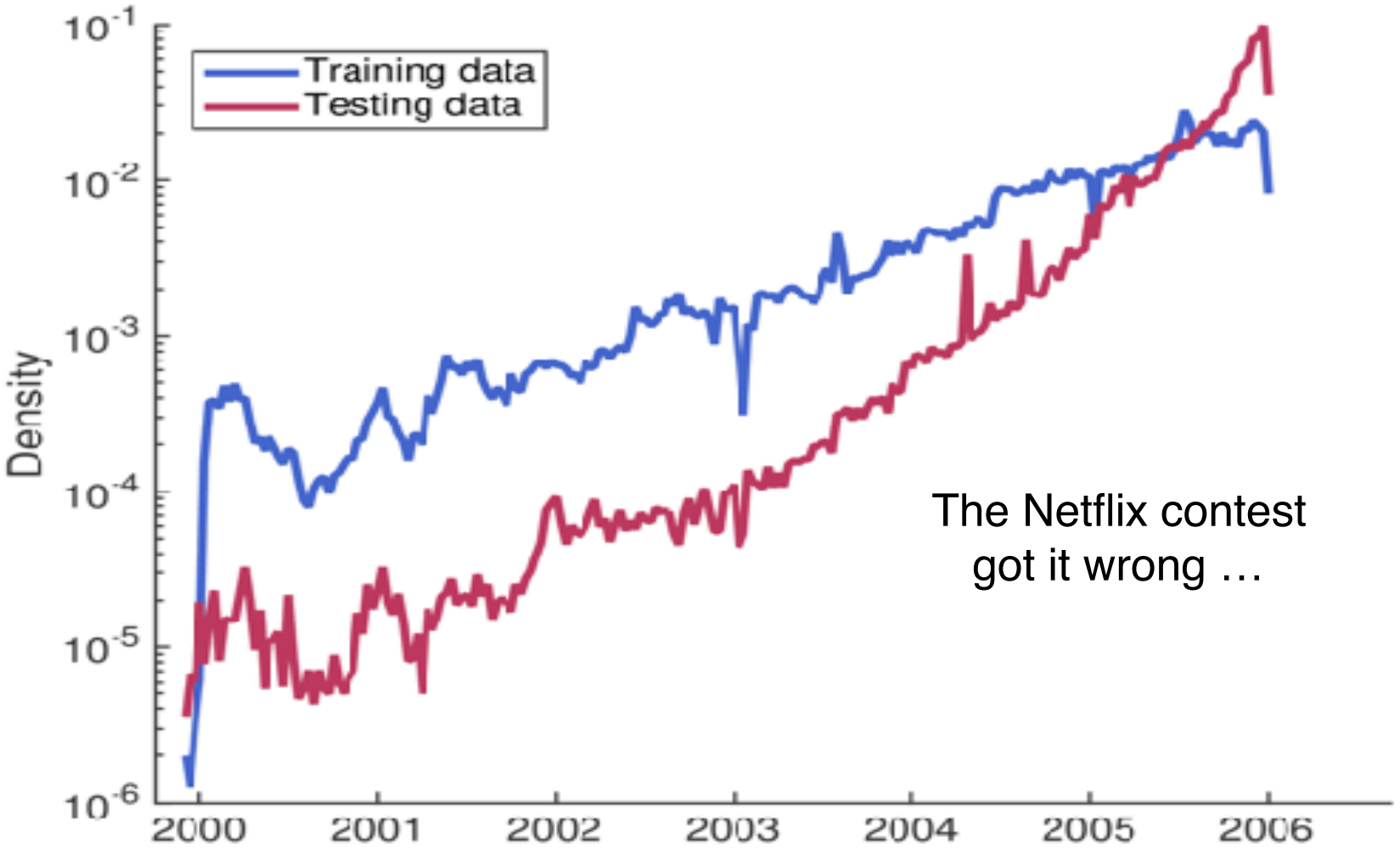


- Images - via a Convolutional Network (e.g. ResNet)
- Text - LSTM or (much simpler) via Bag of Words

Recommender systems, not recommender archaeology

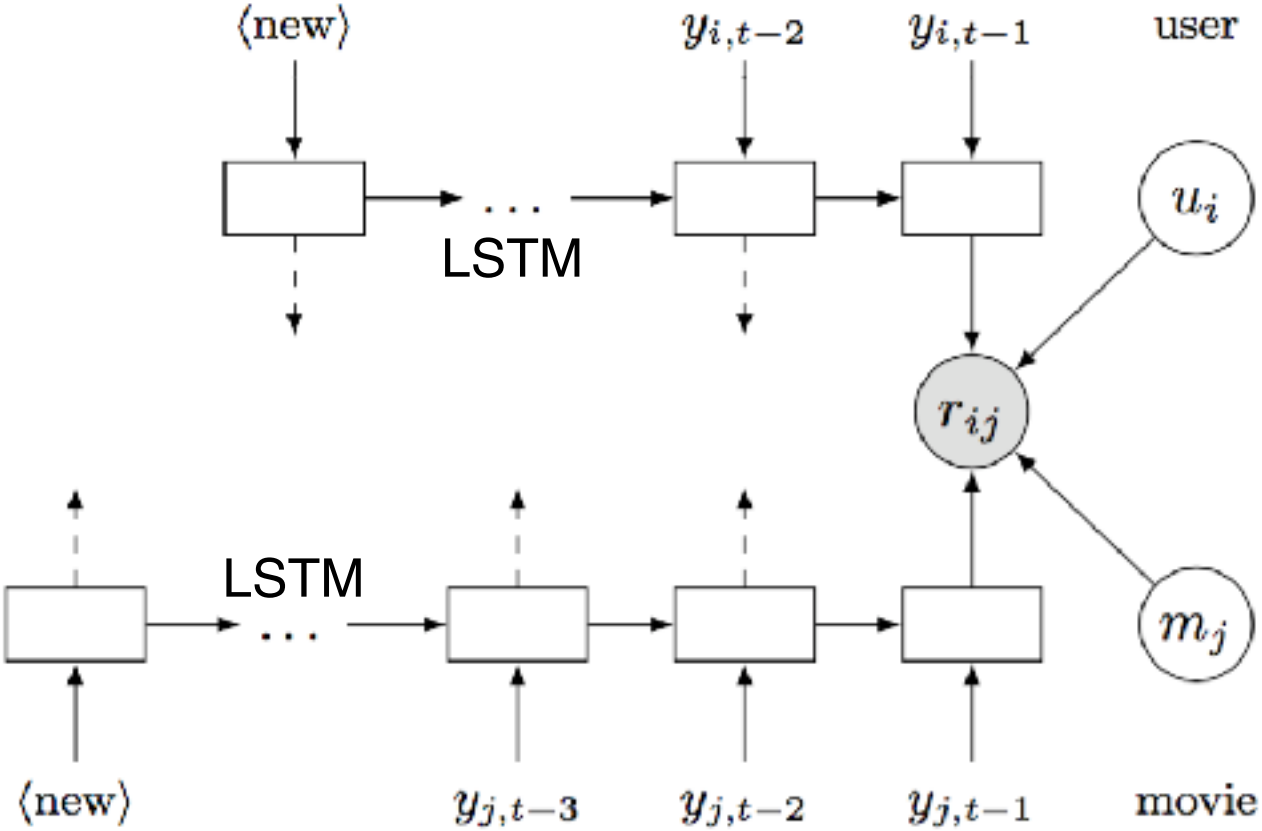


Wu et al., 2017 WSDM



The Netflix contest got it wrong …

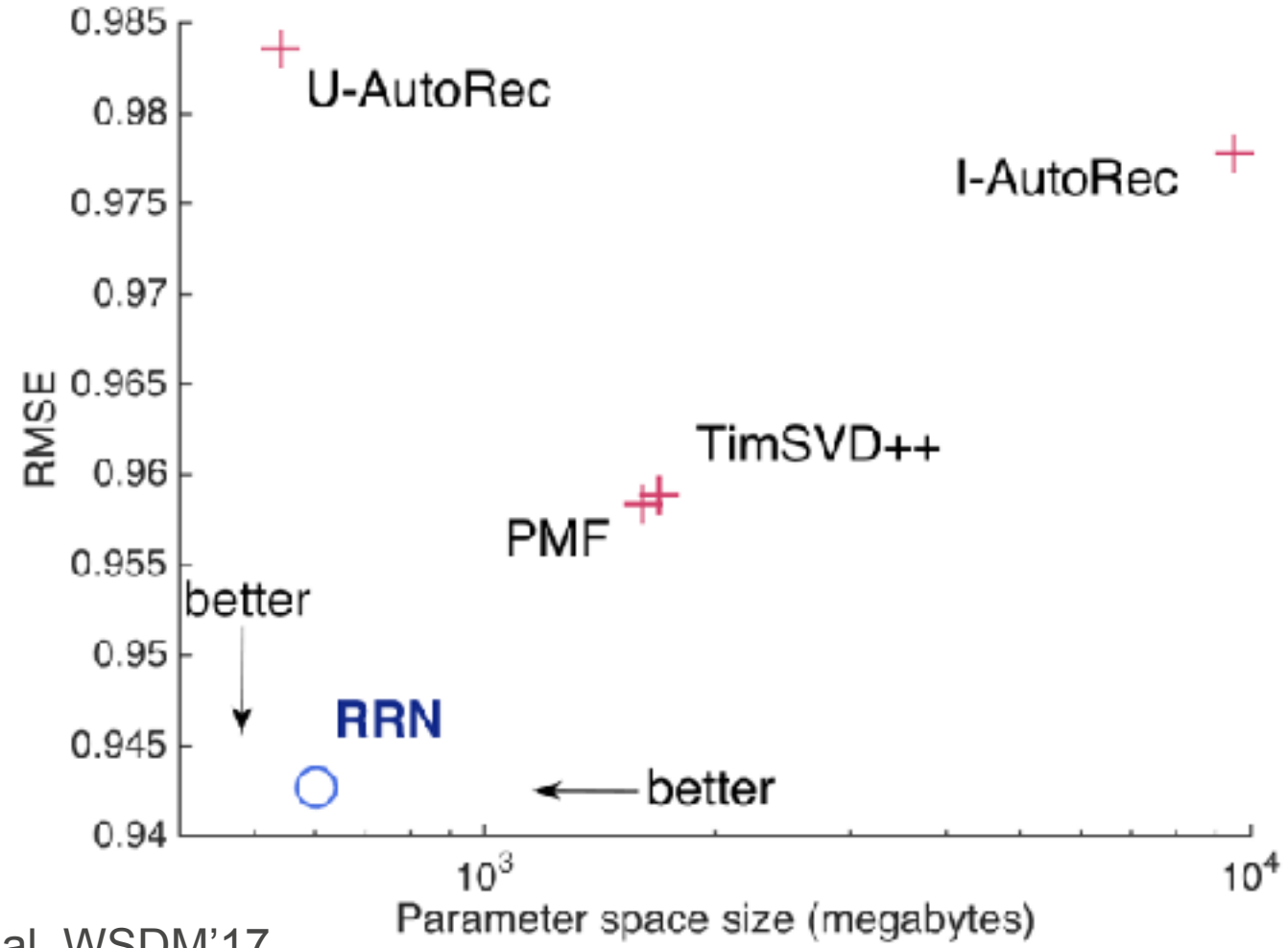
Getting it right



change in taste and expertise

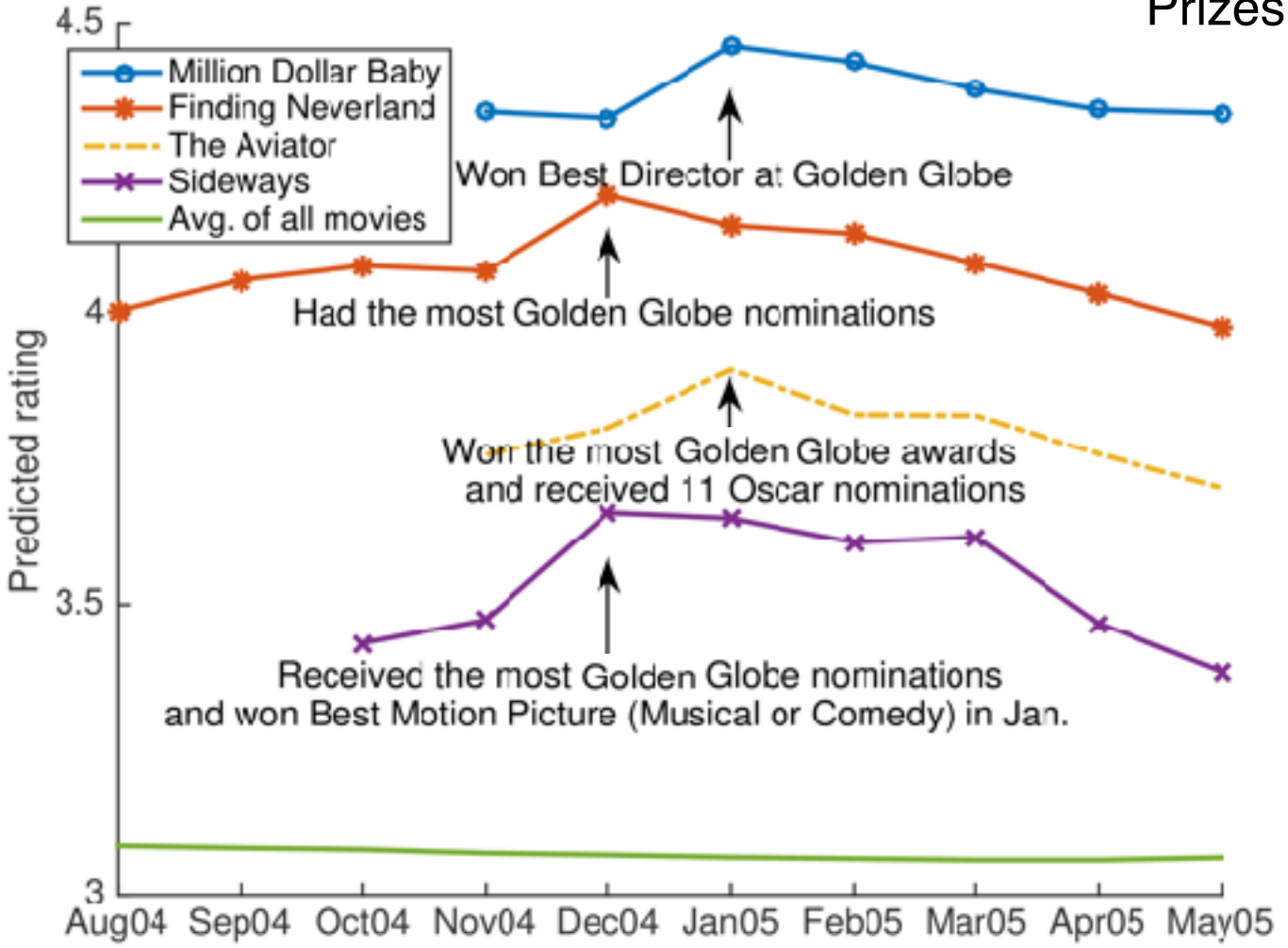
change in perception and novelty

Wu et al, WSDM'17



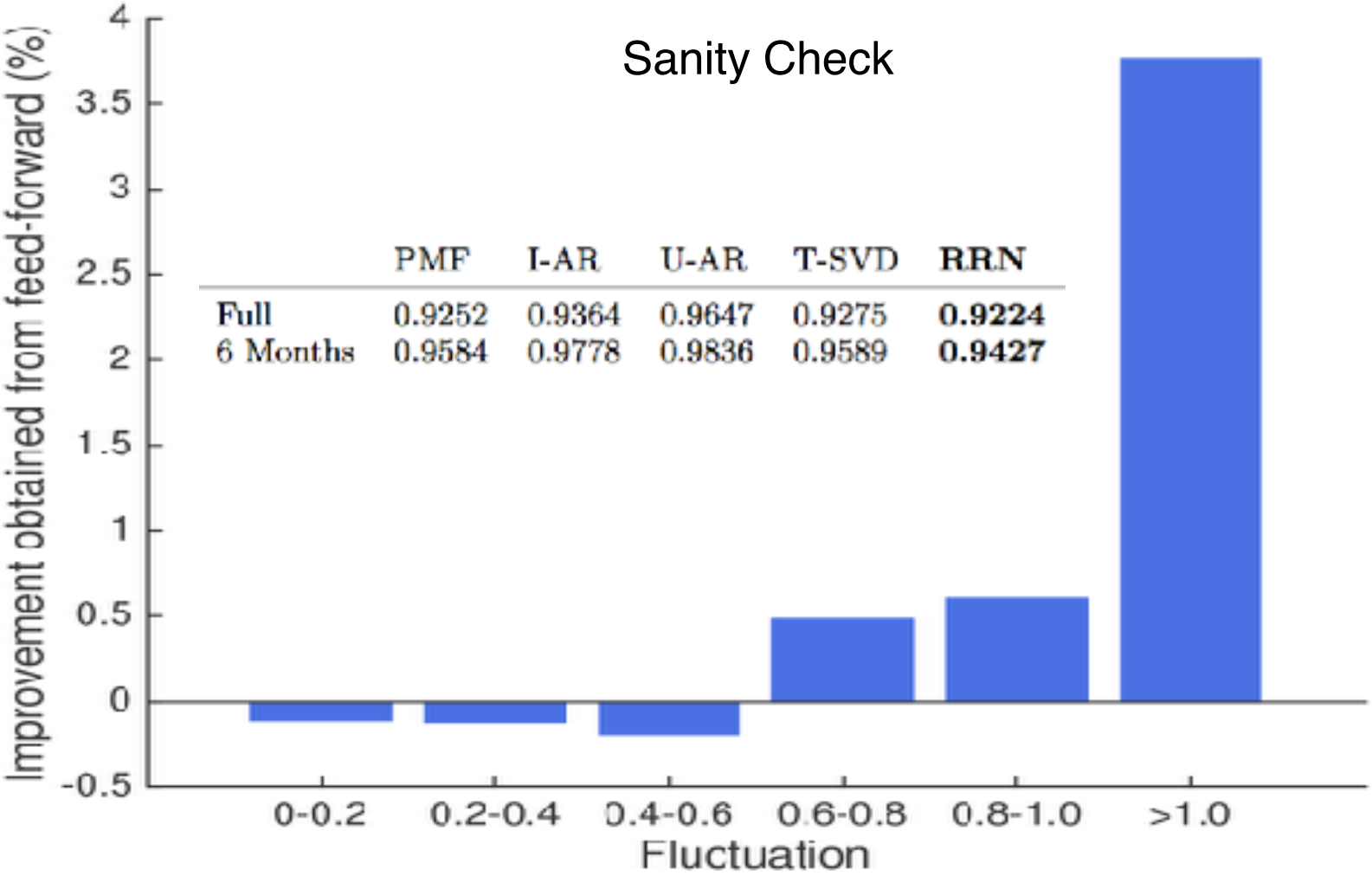
Wu et al, WSDM'17

Prizes



Sanity Check

	PMF	I-AR	U-AR	T-SVD	RRN
Full	0.9252	0.9364	0.9647	0.9275	**0.9224**
6 Months	0.9584	0.9778	0.9836	0.9589	**0.9427**



Thank You!