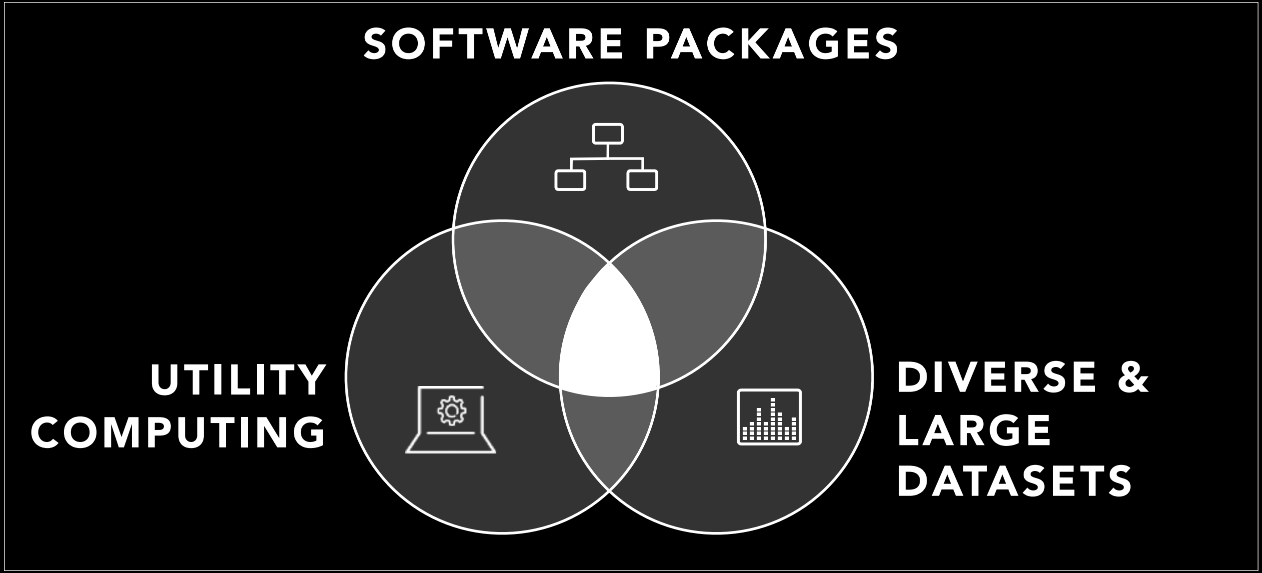
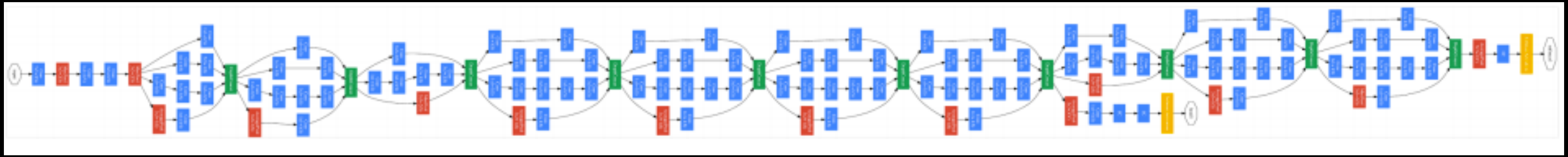
ANIMA ANANDKUMAR

DISTRIBUTED DEEP LEARNING

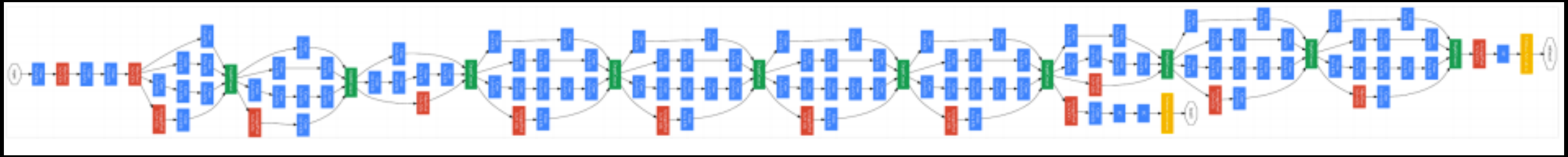
PRACTICAL CONSIDERATIONS FOR MACHINE LEARNING



CHALLENGES IN DEPLOYING LARGE-SCALE LEARNING



CHALLENGES IN DEPLOYING LARGE-SCALE LEARNING



- Complex deep network
  - Coding from scratch is impossible
- A single image requires billions floating-point operations
  - Intel i7 ~500 GFLOPS
  - Nvidia Titan X: ~5 TFLOPS
- Memory consumption is linear with number of layers

DESIRABLE ATTRIBUTES IN A ML SOFTWARE PACKAGE

PROGRAMMABILITY

Simplifying network definitions

PORTABILITY

Efficient use of memory

EFFICIENCY

In training and inference

mxnet

TensorFlow

Caffe

theano

Torch

CNTK

mxnet

MXNET IS AWS'S DEEP LEARNING FRAMEWORK OF CHOICE

mxnet

MOST OPEN

Apache

BEST ON AWS

(Integration with AWS)

PROGRAMMABILITY

frontend

backend

single implementation of backend system and common operators

performance guarantee regardless which front-end language is used

IMPERATIVE PROGRAMMING

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

Easy to tweak
with python codes

PROS

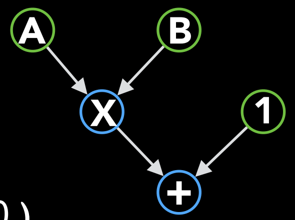
- Straightforward and flexible.
- Take advantage of language native features (loop, condition, debugger)
- E.g. Numpy, Matlab, Torch, …

CONS

- Hard to optimize

DECLARATIVE PROGRAMMING

```
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

PROS

- More chances for optimization
- Cross different languages
- E.g. TensorFlow, Theano, Caffe

CONS

- Less flexible

MXNET: MIXED PROGRAMMING PARADIGM

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_hidde
>>> net = mx.symbol.SoftmaxOutput(data=net)
>>> texec = mx.module.Module(net)
>>> texec.forward(data=c)
>>> texec.backward()
```

NDArray can be set as input to the graph

MXNET: MIXED PROGRAMMING PARADIGM

```
 texec = mx.module.Module(net)
for batch in train_data:
    texec.forward(batch)
    texec.backward()
          for param, grad in zip(texec.get_params(), texec.get_grads()):
        param -= 0.2 * grad
```

Embed symbolic expressions into imperative programming

PORTABILITY

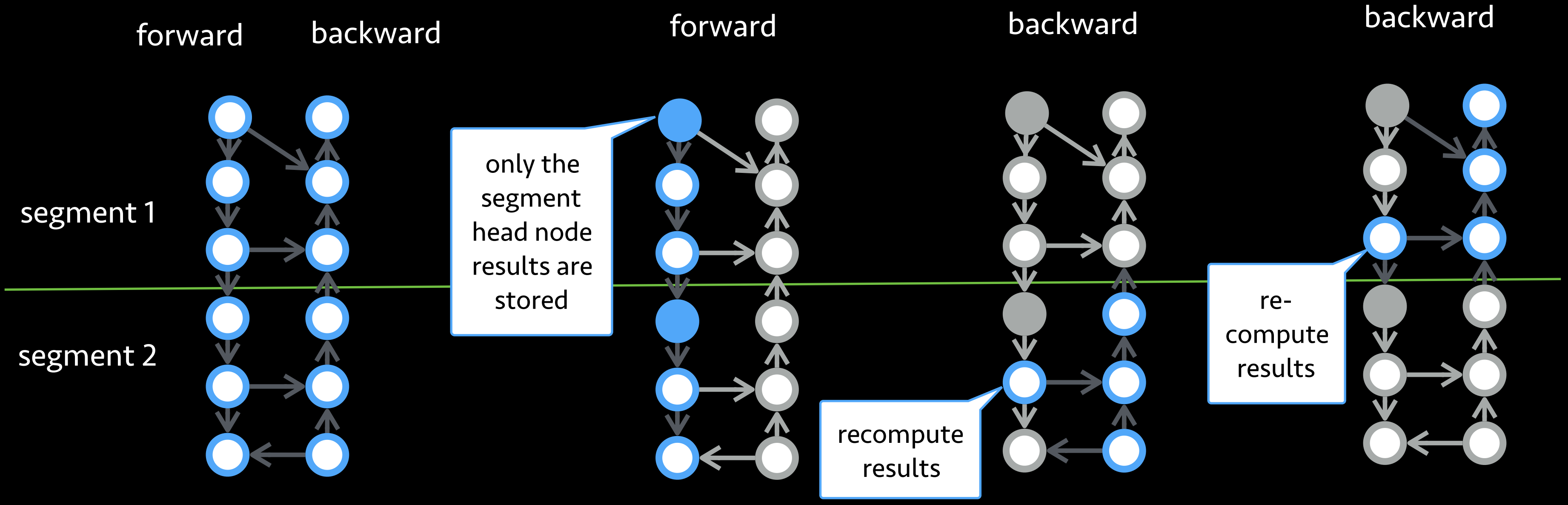
AMALGAMATION

RUNS IN BROWSER WITH JAVASCRIPT

- Fit the core library with all dependencies into a single C++ source file
- easy to compile on any platform

BlindTool by Joseph Paul Cohen, demo on Nexus 4

MEMORY OPTIMIZATION



TRADEOFF MEMORY FOR COMPUTATION

- Needs an extra forward pass
- Reduces the memory complexity from O(n) to O(sqrt(n)), where n is the number of layers
- *Training Deep Nets with Sublinear Memory Cost. T. Chen et al 2016*

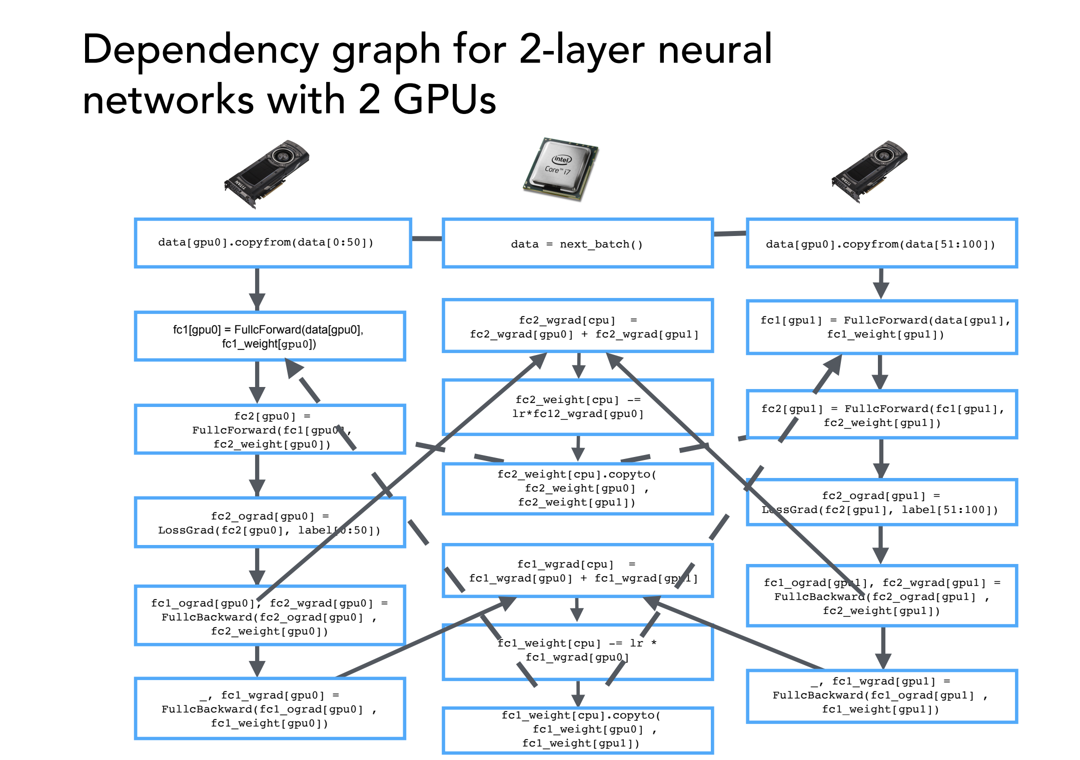
EXAMPLES

- 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

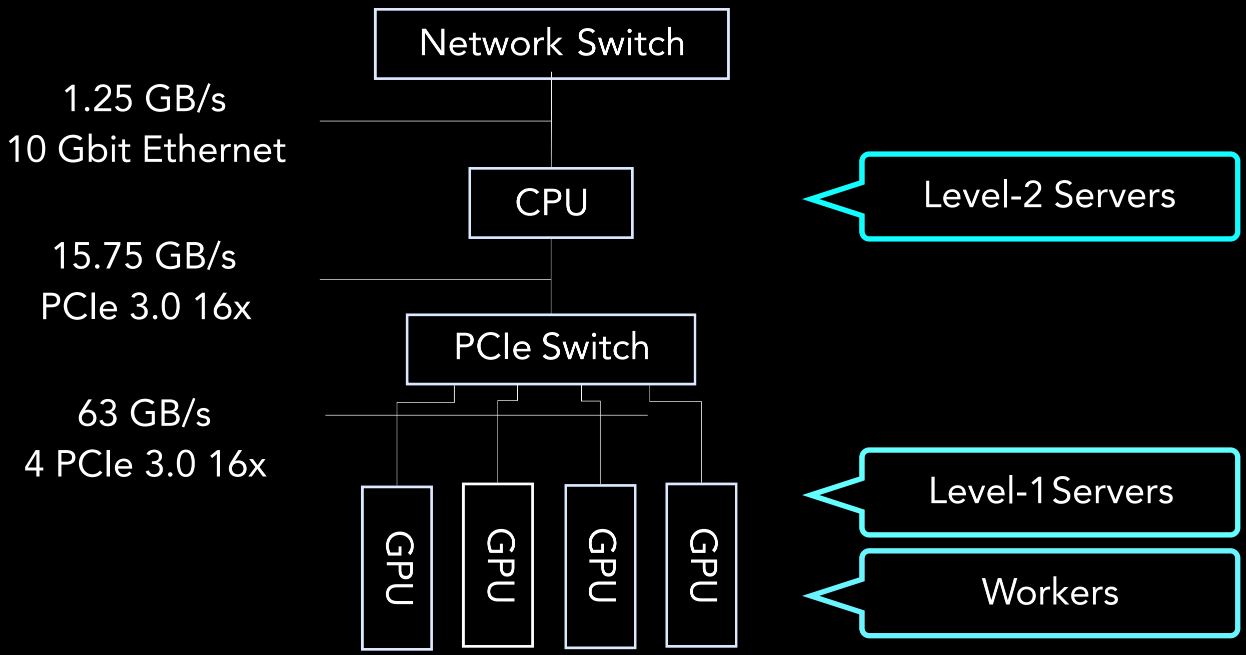
PERFORMANCE

WRITING PARALLEL PROGRAMS IS HARD



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

HIERARCHICAL PARAMETER SERVER IN MXNET



SCALABILITY OF MXNET

SCALE TO MULTIPLE CORES

Deep learning well suited to GPUs

SCALE ACROSS GPUS

Up to 16 available on P2.16xl

SCALE ACROSS NODES

Lots and lots of p2.16xl ;)

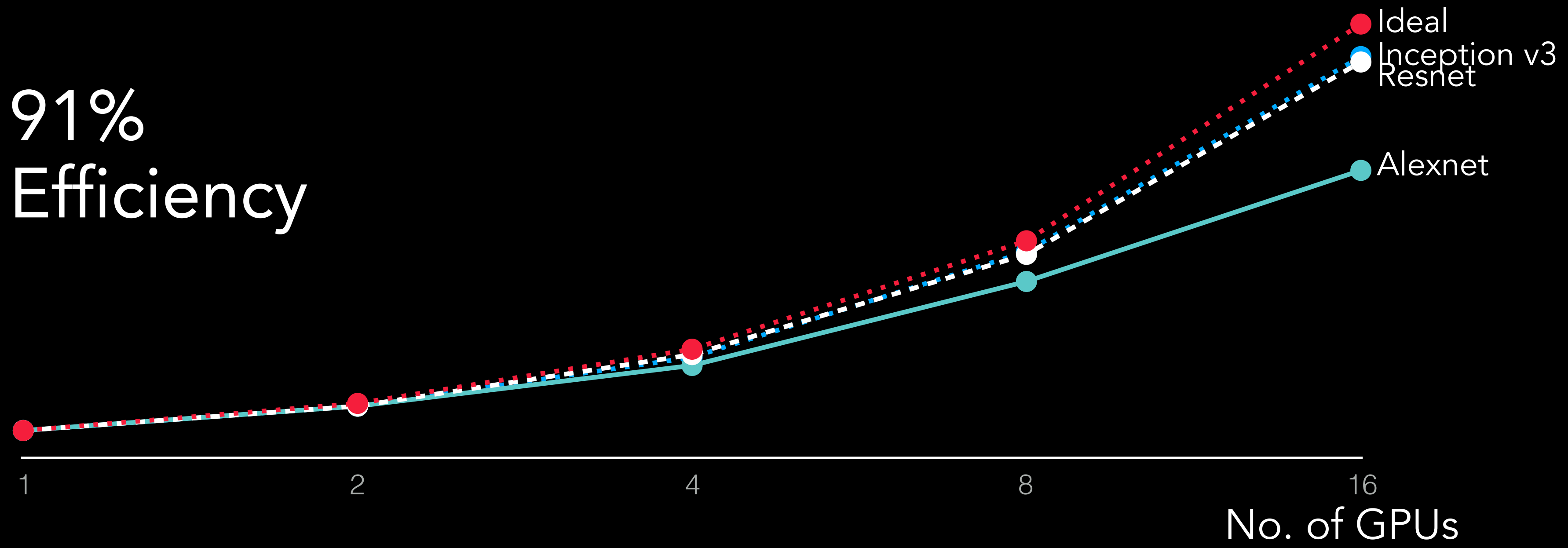
SPIN UP

LOG IN

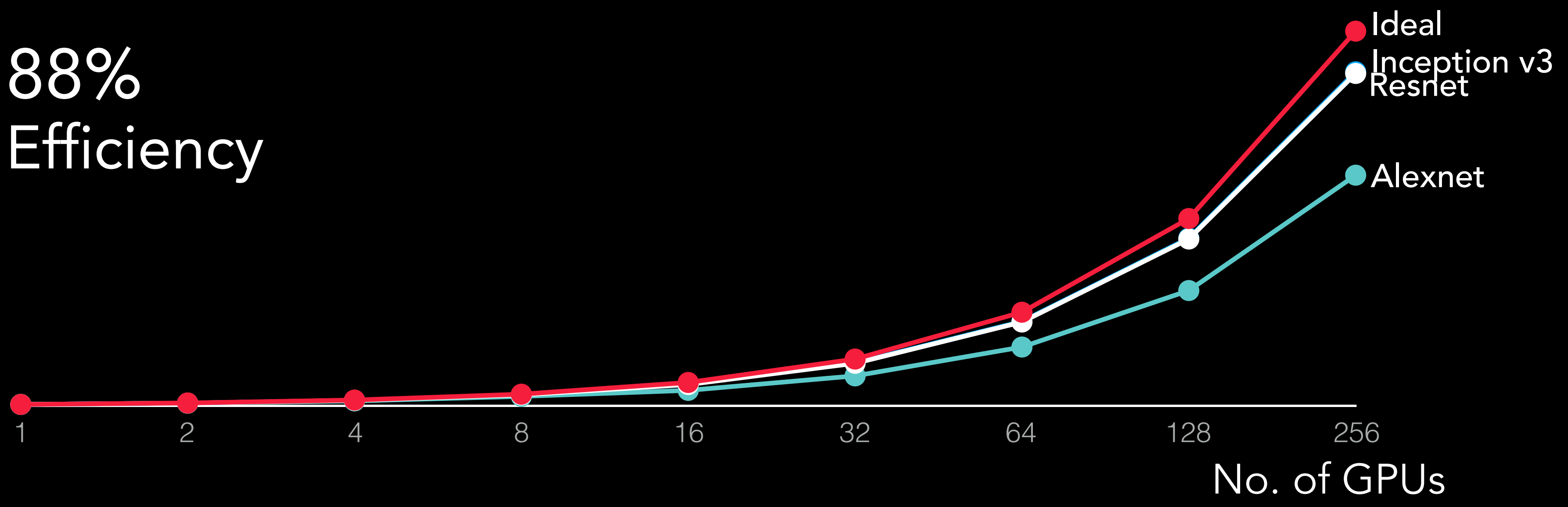
RUN

github.com/awslabs/deeplearning-benchmark

91% Efficiency

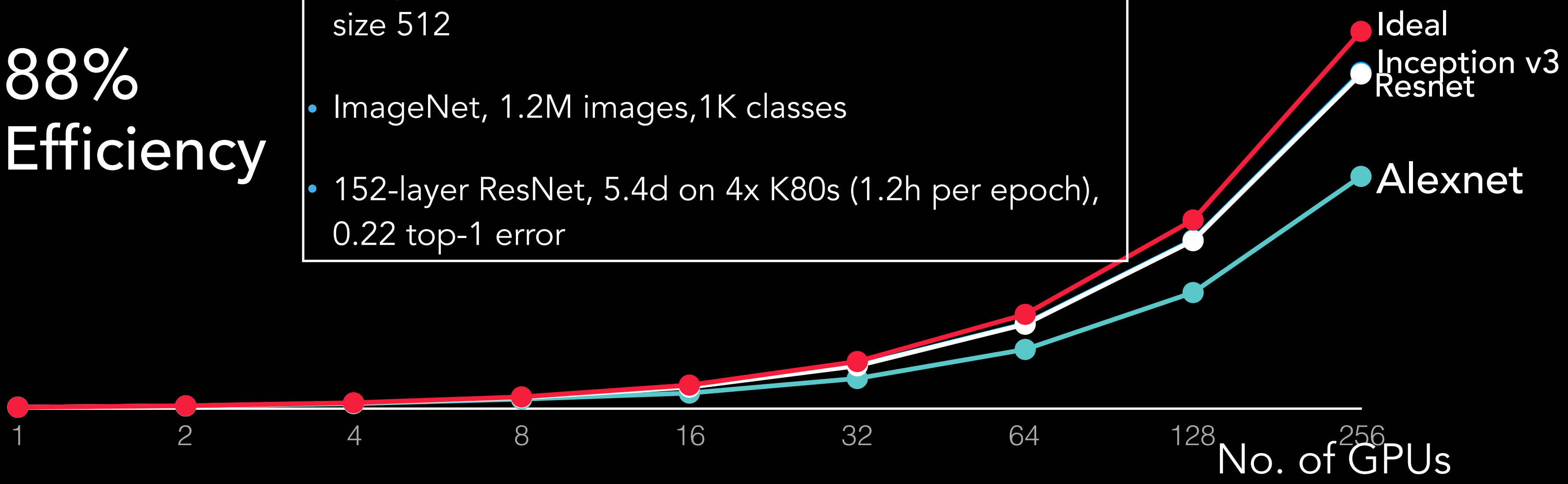


88%
Efficiency



- Cloud formation with Deep Learning AMI
- 16x P2.16xlarge. Mounted on EFS
- Inception and Resnet: batch size 32, Alex net: batch size 512
- ImageNet, 1.2M images,1K classes
- 152-layer ResNet, 5.4d on 4x K80s (1.2h per epoch), 0.22 top-1 error

88% Efficiency



ROADMAP FOR MXNET

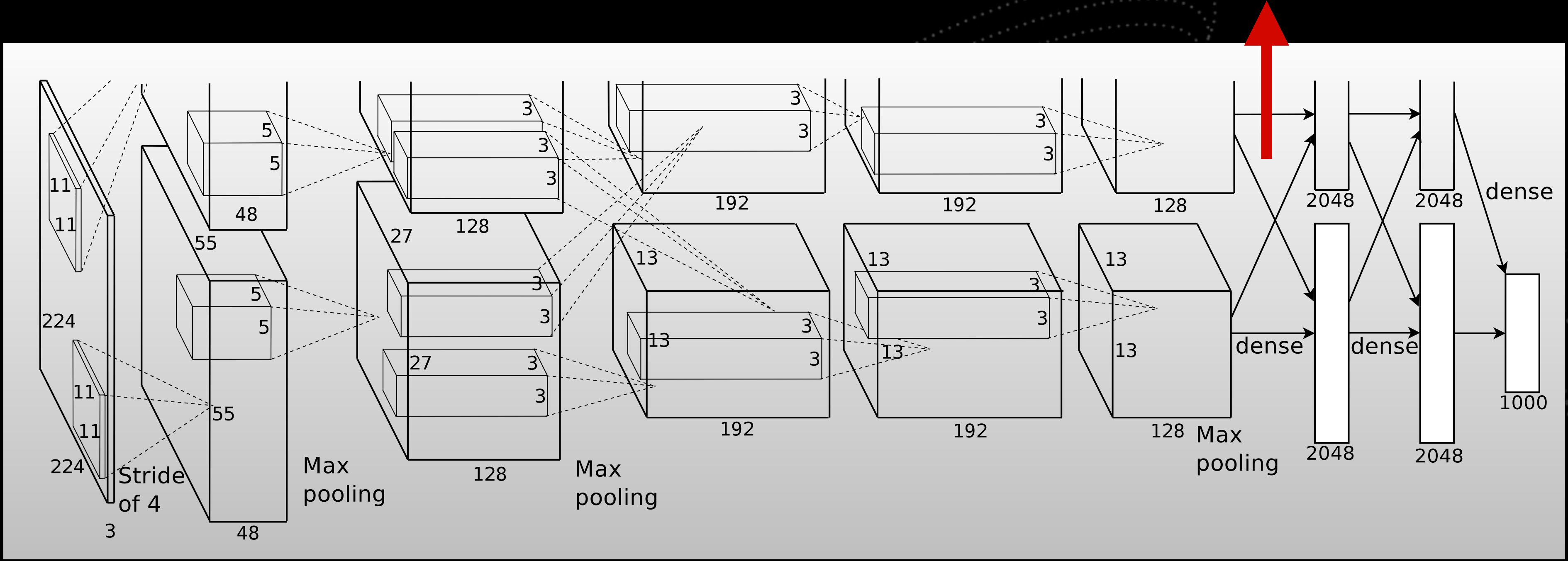
- Documentation (installation, native documents, etc.)
- Platform support (Linux, Windows, OS X, mobile …)
- Sparse datatypes and tensor operations
- Platform for general distributed machine learning algorithms

TENSORS, DEEP LEARNING & MXNET

- Tensors = natural representations for many data in Machine Learning (e.g. images are third order tensors (height, width, channels)
- Great tool to better understand Deep Learning
- Tensor decomposition has ability to discover multi-dimensional dependencies and produce compact low-rank approximation of data
- Tensors are first class citizens in MxNet

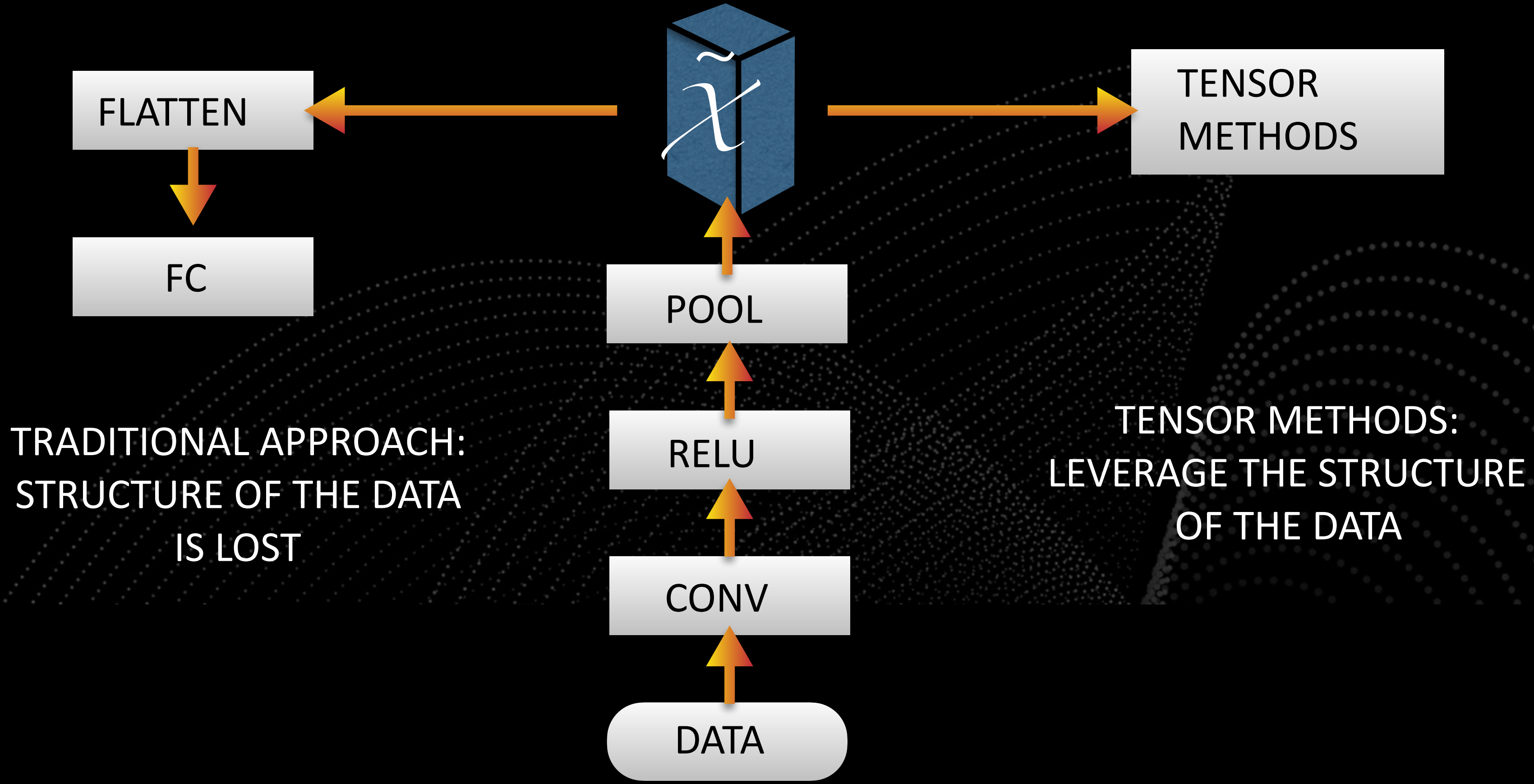
TENSORS, DEEP LEARNING & MXNET

Structure is lost when flattening

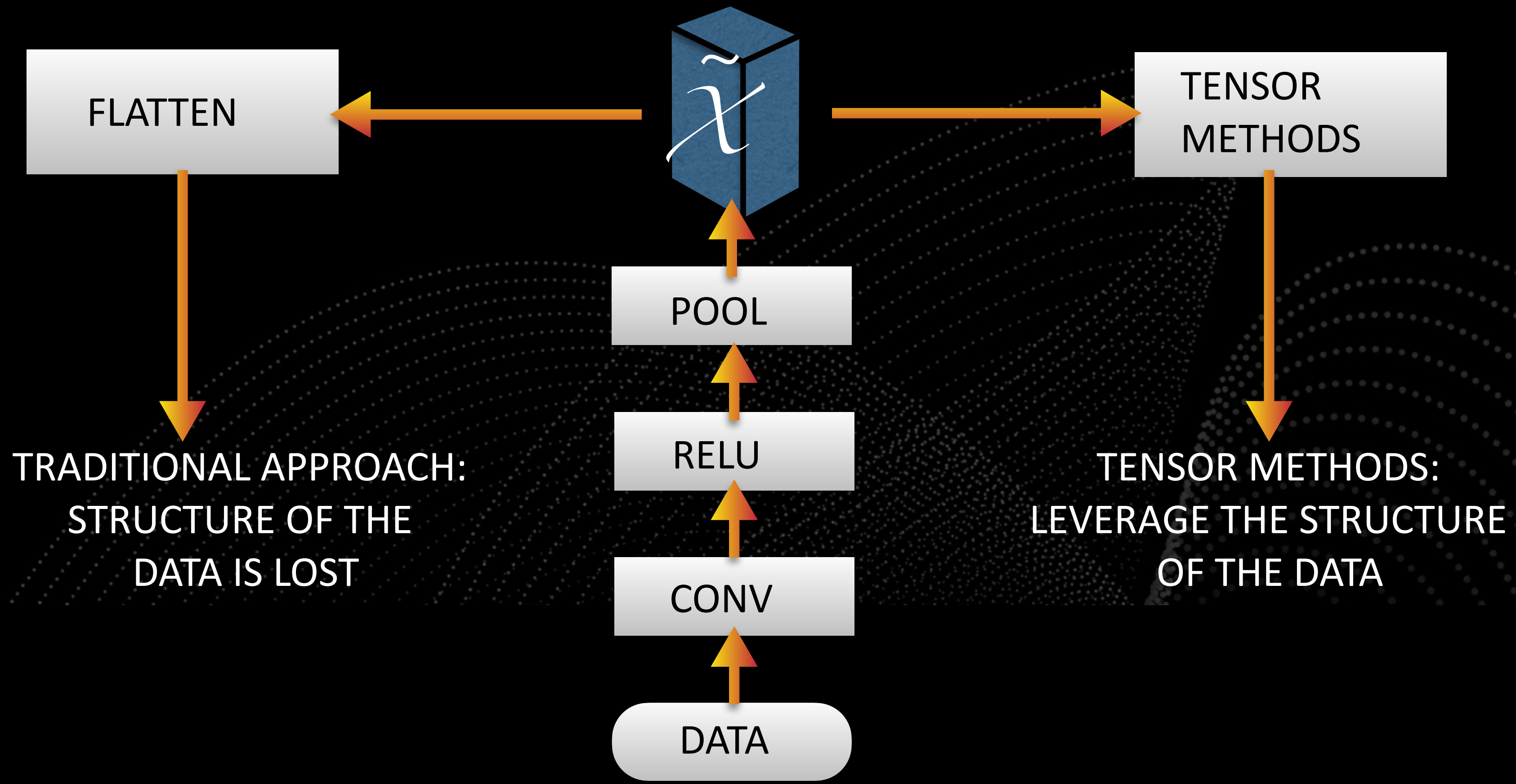


AlexNet, ImageNet classification with deep convolutional neural networks, NIPS'12, Alex Krizhevsky et. al.

TENSOR METHODS, DEEP LEARNING & MXNET

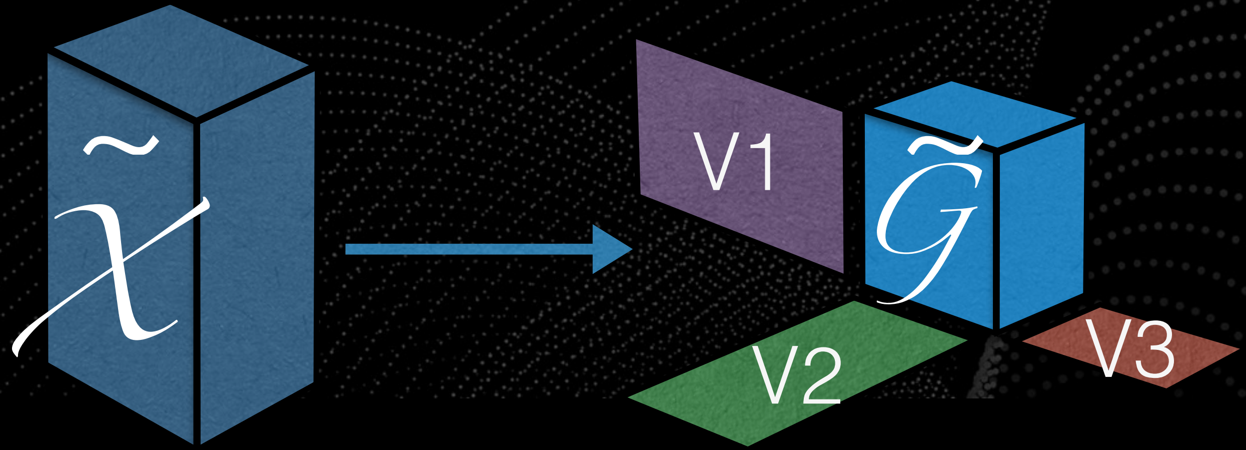


TENSORS, DEEP LEARNING & MXNET



TENSOR CONTRACTION

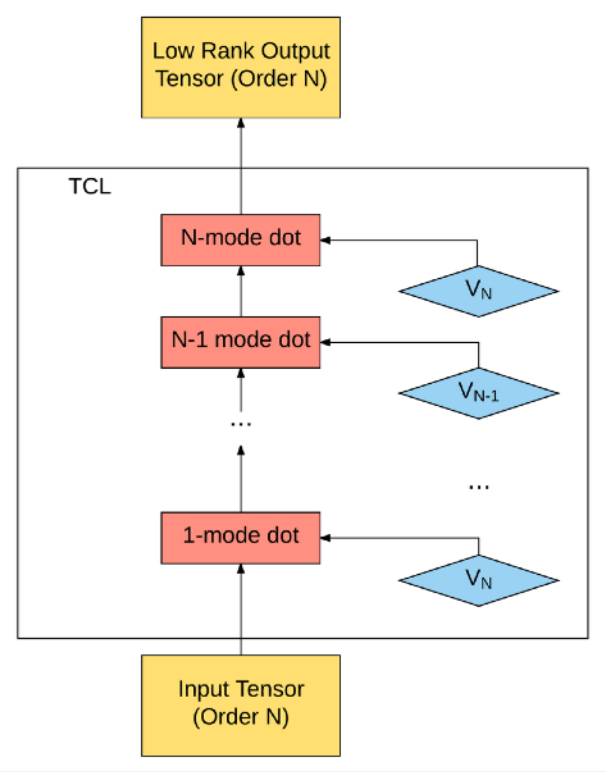
Tucker tensor decomposition: express a tensor as a function of a low rank tensor and projection matrices



$$\tilde{\mathcal{X}} = \tilde{\mathcal{G}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times \cdots \times_N \mathbf{U}^{(N)}$$

TENSOR CONTRACTION AS A LAYER

- Take activation tensor as input
- Feed it through a tensor contraction layer (TCL)
- Output a low rank activation tensor

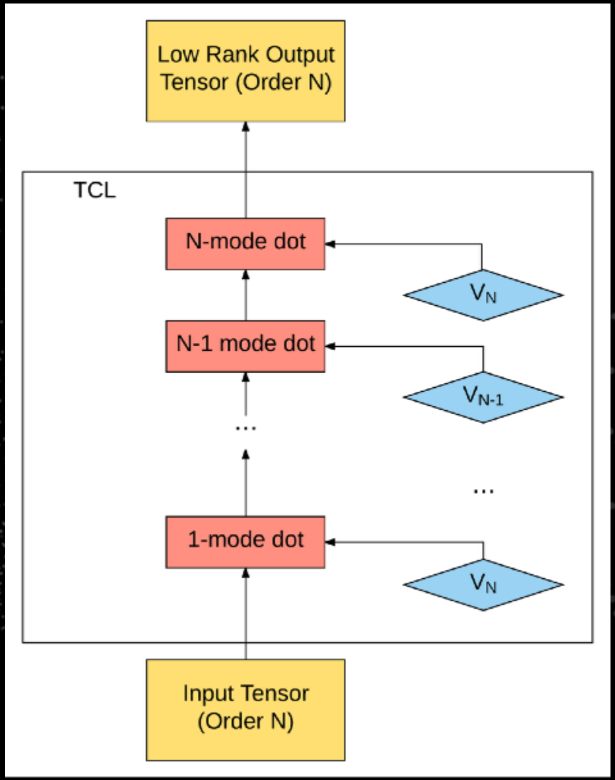


TENSOR CONTRACTION AS A LAYER

- Compact representation
  -> less parameters
  (measured as Space Savings)

$$\text{space saving} = 1 - \frac{n_{\text{original parameters}}}{n_{\text{parameters in compact model}}}$$

- Similar and sometimes better performance



PRELIMINARY RESULTS

Method - Hidden Units in Fully Connected Layers	Accuracy (%)	Space savings (%)
Baseline Traditional AlexNet, 4096 hidden units	56.29	0
Adding a TCL (256, 5, 5), 4096 hidden units	57.54	-0.11
Adding a TCL (200, 5, 5), 3276 hidden units	56.11	35.73
Replace a FCL with (256, 5, 5) TCL, 4096 Hidden Units	56.63	44.45

Results on ImageNet with an AlexNet. *J. Kossafi et. al 2017*

AMIs, Cloud Formation and DL

amazon web services

One-Click Deep Learning

image credit - publicdomainpibtures

P2 INSTANCES

Up to 40k
CUDA cores

DEEP AMI

Pre-configured for
deep learning

DEEP TEMPLATE

Deep learning
clusters

P2 INSTANCES

Up to 40k
CUDA cores

p2.16xl instance= 16 K80 GPUs ~ 70 tera flops

16 p2.16xl instances ~ 1.1 peta flops

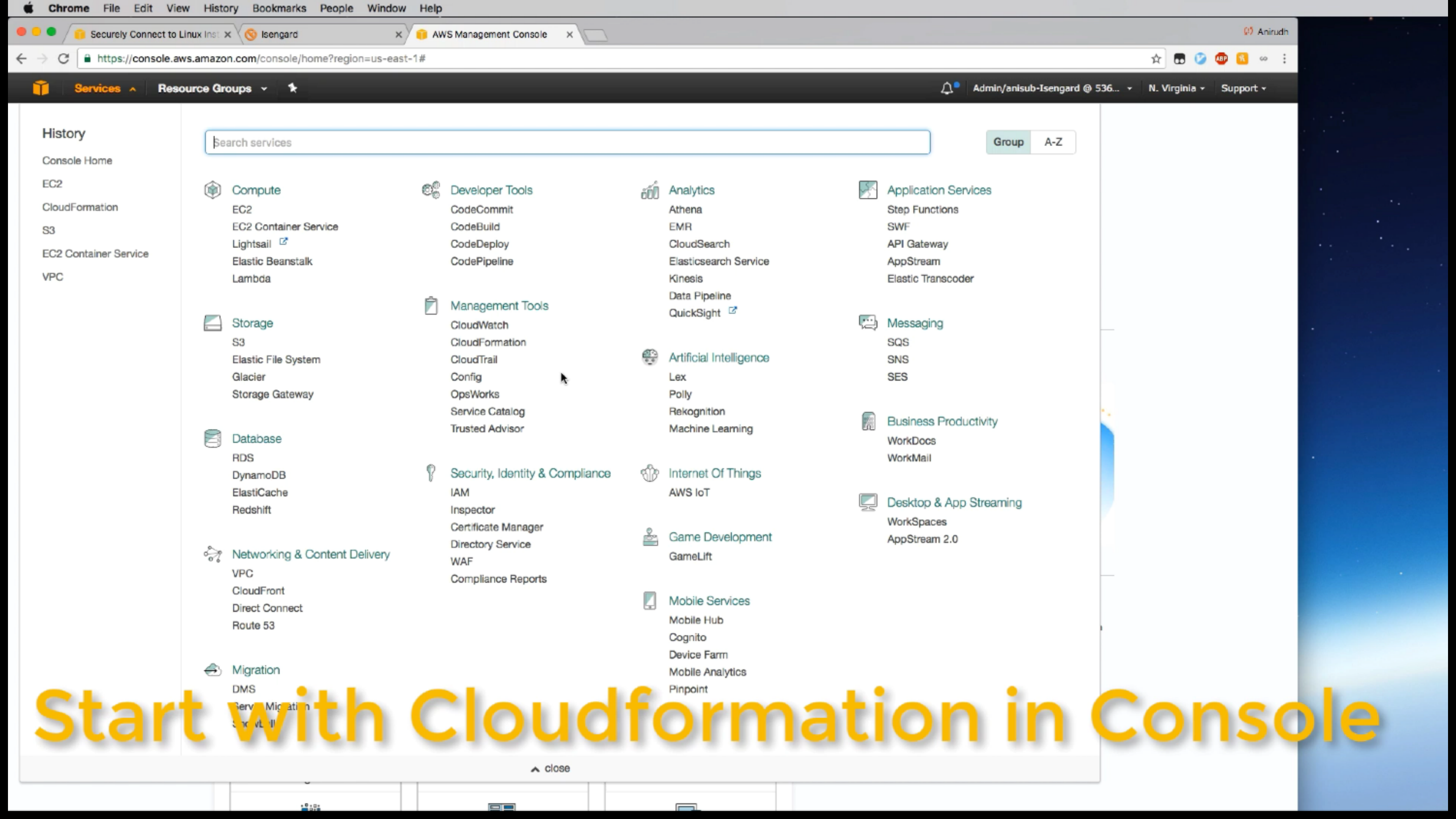
World's fastest supercomputer ~ 93 peta flops

GPUDirect™ (peer-to-peer GPU communication)

AMAZON MACHINE IMAGES

http://bit.ly/deepami

- Tool for data scientists and developers
- Setting up a DL system takes (install) time & skill
- Keep packages up to date and compiled (MXNet, TensorFlow, Caffe, Torch, Theano, Keras)
- Anaconda, Jupyter, Python 2 and 3
- **NVIDIA** Drivers for G2 and P2 instances
- **Intel MKL** Drivers for all other instances (C4, M4, …)


Chrome File Edit View History Bookmarks People Window Help
Securely Connect to Linux Inst
Isengard
AWS Management Console
Anirudh
https://console.aws.amazon.com/console/home?region=us-east-1#
Services
Resource Groups
Admin/anisub-Isengard @ 536...
N. Virginia
Support
History
Console Home
EC2
CloudFormation
S3
EC2 Container Service
VPC
Search services
Group
A-Z
Compute
EC2
EC2 Container Service
Lightsail
Elastic Beanstalk
Lambda
Storage
S3
Elastic File System
Glacier
Storage Gateway
Database
RDS
DynamoDB
ElastiCache
Redshift
Networking & Content Delivery
VPC
CloudFront
Direct Connect
Route 53
Migration
DMS
Developer Tools
CodeCommit
CodeBuild
CodeDeploy
CodePipeline
Management Tools
CloudWatch
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Trusted Advisor
Security, Identity & Compliance
IAM
Inspector
Certificate Manager
Directory Service
WAF
Compliance Reports
Analytics
Athena
EMR
CloudSearch
Elasticsearch Service
Kinesis
Data Pipeline
QuickSight
Artificial Intelligence
Lex
Polly
Rekognition
Machine Learning
Internet Of Things
AWS IoT
Game Development
GameLift
Mobile Services
Mobile Hub
Cognito
Device Farm
Mobile Analytics
Pinpoint
Application Services
Step Functions
SWF
API Gateway
AppStream
Elastic Transcoder
Messaging
SQS
SNS
SES
Business Productivity
WorkDocs
WorkMail
Desktop & App Streaming
WorkSpaces
AppStream 2.0
Start with Cloudformation in Console
close

Introducing Amazon AI

Apache MXNet
Deep learning engine

Polly
Text-to-Speech

Rekognition
Image Analysis

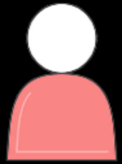
Lex
ASR & NLU

Rekognition: Search & Understand Visual Content

Real-time & batch image analysis

Object & Scene Detection

Facial Detection



Facial Analysis

Face Search

Rekognition: Object & Scene Detection

Category	Confidence
Bay	99.18%
Beach	99.18%
Coast	99.18%
Outdoors	99.18%
Sea	99.18%
Water	99.18%
Palm_tree	99.21%
Plant	99.21%
Tree	99.21%
Summer	58.3%
Landscape	51.84%
Nature	51.84%
Hotel	51.24%

Rekognition: Facial Analysis

Emotion: calm: 73%
Sunglasses: false (value: 0)
Mouth open wide: 0% (value: 0)
Eye closed: open (value: 0)
Glasses: no glass (value: 0)
Mustache: false (value: 0)
Beard: no (value: 0)

Lex: Build Natural, Conversational Interactions In Voice & Text

Voice & Text
“Chatbots”

Powers
Alexa

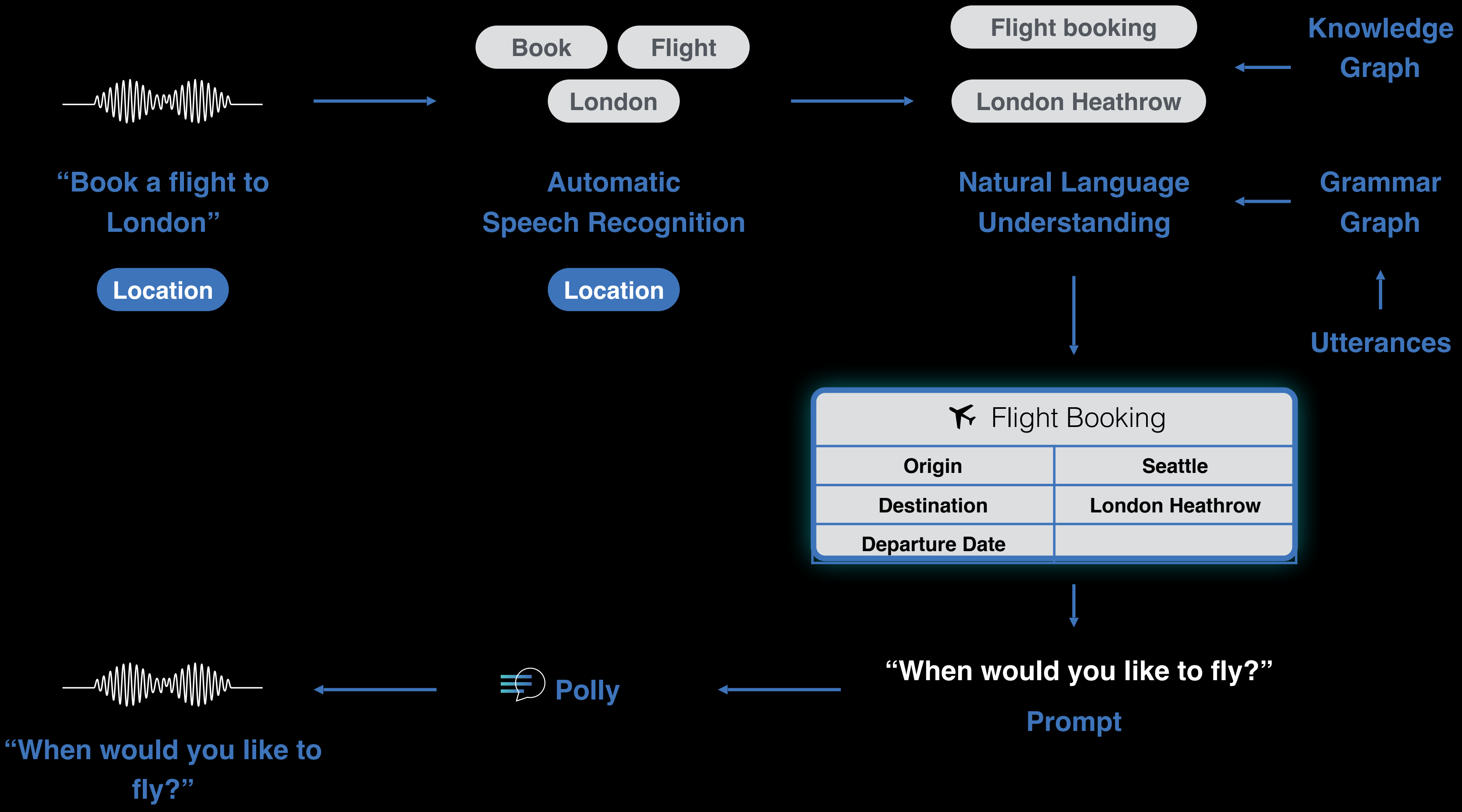
Voice interactions
on mobile, web
& devices

Text interaction
with Slack & Messenger

(with more coming)

Enterprise
Connectors

Salesforce
Microsoft Dynamics
Marketo
Zendesk
Quickbooks
Hubspot


Book
Flight
London
Flight booking
London Heathrow
Knowledge Graph
“Book a flight to London”
Automatic Speech Recognition
Natural Language Understanding
Grammar Graph
Location
Location
Utterances
Flight Booking
Origin
Seattle
Destination
London Heathrow
Departure Date
“When would you like to fly?”
Prompt
Polly
“When would you like to fly?”

"Next Friday"

"When would you like to fly?"

Flight Booking

Origin	Seattle
Destination	London Heathrow
Departure Date	

Amazon Polly: Life-like Speech Service

Converts text
to life-like speech

Fully managed

47 voices

24 languages

Low latency,
real time

Let’s listen…

Polly: A Focus On Voice Quality & Pronunciation

1. Automatic, Accurate Text Processing

“Today in Seattle, WA, it’s 11°F”

‘"We live for the music" live from the Madison Square Garden.’

Polly: A Focus On Voice Quality & Pronunciation

1. Automatic, Accurate Text Processing

2. Intelligible and Easy to Understand

Polly: A Focus On Voice Quality & Pronunciation

1. Automatic, Accurate Text Processing

2. Intelligible and Easy to Understand

3. **Add Semantic Meaning to Text**

“Richard’s number is 2122341237“

“Richard’s number is 2122341237“

Telephone Number

Polly: A Focus On Voice Quality & Pronunciation

1. Automatic, Accurate Text Processing

2. Intelligible and Easy to Understand

3. Add Semantic Meaning to Text

4. Customized Pronunciation

“My daughter’s name is Kaja.”

“My daughter’s name is Kaja.”

ACADEMIC ENGAGEMENTS

- Apply for AWS credits for your research
  https://aws.amazon.com/grants/
- Apply for AWS credits for education
  https://aws.amazon.com/education/awseducate/
- Conduct research and build products at AWS: internships and full time positions!
- Send me an email: anima@amazon.com