AWS re:INVENT

Large Scale Deep Learning with BigDL

AWS re:Invent

Large Scale Deep Learning with BigDL

Tim Fox | Big Data and Machine Learning Consultant | Elephant Scale

AWS re:Invent

aws

ABOUT ME

Tim Fox,

Principal @ Elephant Scale

Practitioner and Trainer in Data Engineering and Data Science

Author of “Data Science in Python” on LinkedIn Learning

tim@elephantscale.com

Linkedin: tim-fox-0063541

ABOUT **Elephant Scale**

- Training in Big Data and AI technologies
- BigData : Spark, Hadoop, Cloud, NoSQL, Streaming
- AI : Machine Learning, Deep Learning, BigDL, Tensorflow
- BigDL training available!
- Public and Private trainings available
- BigDL Sandbox : elephantscale.com/sandbox

Elephantscale.com
info@elephantscale.com

Quick Roundup of AI / Machine Learning / Deep Learning

AI / MACHINE LEARNING / DEEP LEARNING

Artificial Intelligence (AI):

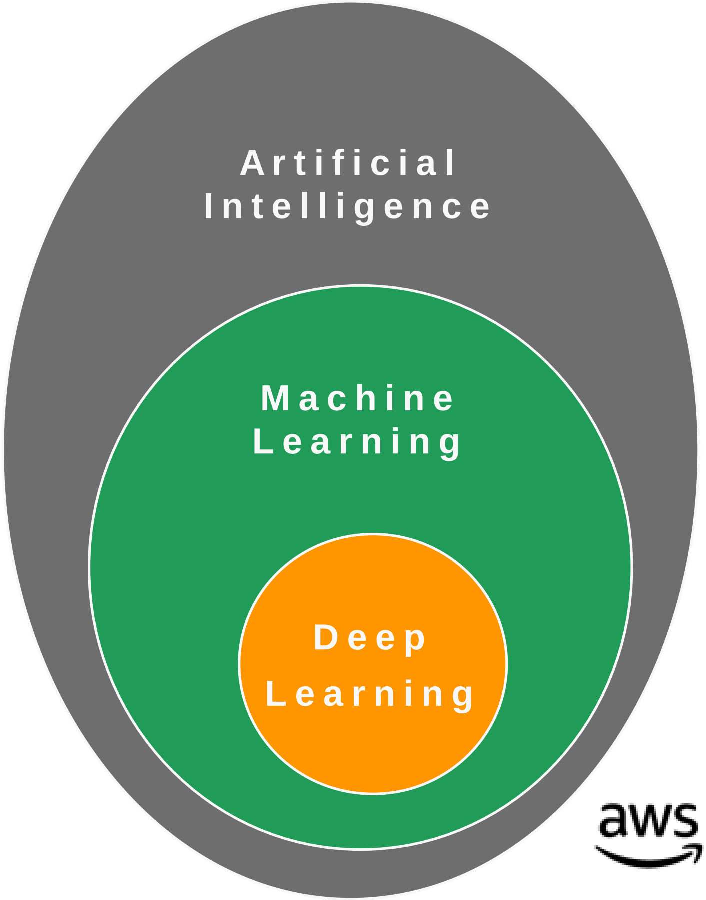
Broader concept of machines being able to carry out 'smart' tasks

Machine Learning:

A type of AI that allows software to learn from data without explicitly programmed

Deep Learning:

Using Neural Networks to solve some hard problems



DEEP LEARNING APPLICATIONS

Self Driving Cars

- ML system using image recognition
- Where the edge of the road / road sign / car in front

Face recognition

- Facebook images
- System learns from images manually tagged and then automatically detects faces in uploaded photos

DEEP LEARNING HISTORY

Early attempts at Deep Learning did not succeed.

- Compute Power was insufficient for the time.
- Training Datasets were insufficiently sized for good results.
- We lacked the ability to parallelize our work.

In the modern era, Deep Learning has been successful.

- 'Big Data' – now we have so much data to train our models
- 'Big Data ecosystem' – excellent big data platforms (Hadoop, Spark, NoSQL) are available as open source
- 'Big Compute' - cloud platforms significantly lowered the barrier to massive compute power
  - $1 buys you 16 core + 128 G + 10 Gigabit machine for 1 hr on AWS!
  - So running a 100 node cluster for 5 hrs → $500

AWS re:Invent

aws

AI SOFTWARE ECO SYSTEM

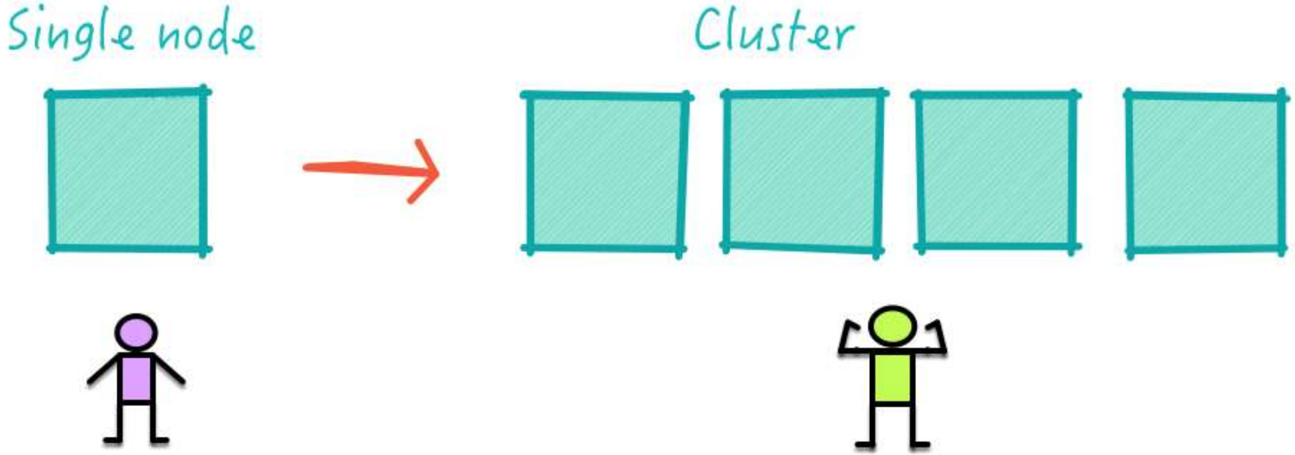
	Machine Learning	Deep Learning
Java	- Weka - Mahout	- DeepLearning4J
Python	- SciKit	- Tensorflow - Theano - Caffe
R	- Many libraries	- Deepnet - Darch
Distributed	- H20 - Spark	- H20 - Spark - BigDL
Cloud	- AWS	- AWS

MACHINE LEARNING AND BIG DATA

Until recently most of the machine learning is done on “single computer” (with lots of memory–100s of GBs)

Most R/Python/Java libraries are “single node based”

Now Big Data tools make it possible to run machine learning algorithms at massive scale– distributed across a cluster



MODERN DEEP LEARNING FRAMEWORKS

TensorFlow

torch

mxnet

BigDL

Keras

theano

TOOLS FOR SCALABLE MACHINE LEARNING

Apache Spark ML

- Runs on top of popular Spark framework
- Massively scalable
- Can use memory (caching) effectively for iterative algorithms
- Language support: Scala, Java, Python, R

BigDL

- Built for Apache Spark and Optimized for Intel Xeon
- Language Support: Scala, Java, Python

TensorFlow

- Based on “data flow graphs”
- Language support: Python, C++
- https://www.tensorflow.org/

TOOLS FOR SCALABLE CLOUD MACHINE LEARNING

Amazon Machine Learning

- Ready to go algorithms
- Visualization tools
- Wizards to guide
- Scalable on Amazon Cloud

BigDL

WHAT IS BIGDL

A distributed deep learning library for Apache Spark

Feature parity with popular deep learning frameworks

- Caffe, Torch, Tensorflow

High Performance

- Powered by Intel Math Kernel Library (MKL) and multi threaded programming

Can scale to huge datasets

- Using Apache Spark for scale

Open source! (Dec 2016)

Active Development

PRODUCTION ML/DL SYSTEMS ARE COMPLEX!

Actual ML/DL is only small portion of massive production system

BigDL running on a scalable platform like Spark helps simplify the complexity

Motivation for BigDL

Production ML/DL system is **Complex**

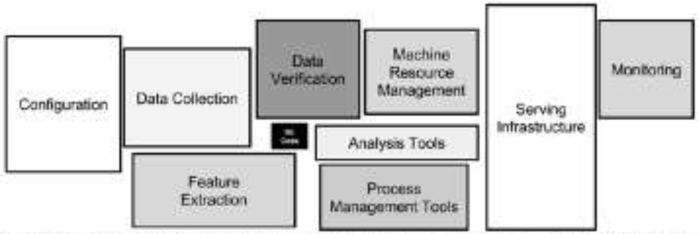


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

"Hidden Technical Debt in Machine Learning Systems", Google, NIPS 2015 Paper

BIGDL FILLS THE 'GAP' IN BIG DATA + DEEP LEARNING

Follows proven design patterns for dealing with Big Data

Sends 'compute to data' rather than reading massive data over network.

Uses 'data locality' of HDFS (Hadoop File System

Utilizes 'cluster managers' like YARN / MESOS

- Automatically handles hardware/software failures
- Elasticity and resource sharing in a cluster

BIGDL & SPARK

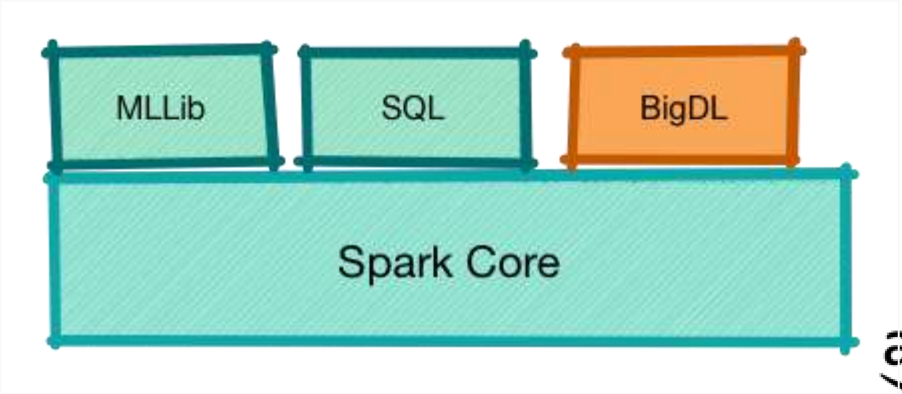
Run BigDL applications as Spark applications

Scala, Java, and Python support

Use other Spark's features

- In memory compute
- Integrate with Spark ML and Streaming

Easy development with Jupyter Notebook



BIGDL VS TENSORFLOW

	BigDL	Tensorflow
Runtime	Scala Engine with Python front-end	C++ Engine with Python front-end
Hadoop compatibility	Can run natively on Spark & Hadoop	Accesses Hadoop data as a client only
Distributed Operation	Scalable with Apache Spark for massive scale out of the box	Does not support massive distribution out of the box
Runs Tensorflow Models	Yes	Yes
Acceleration	CPU w/MKL	CPU/GPU
Summary	Excellent for distributing deep-learning models to massive scale on big-data. Great TCO value.	Excellent library for small-medium scale data, although GPU hardware costs can be significant.

AWS re:Invent

BIGDL: BIG COMPUTE PLUS BIG DATA

BigDL helps us in balancing our needs

- Big Compute: Fast Linear Algebra, Intel MKL library
- Optimized for Intel Xeon
- Big Data: I/O parallelized to run on many CPUs

BigDL Allows Massive Scalability

- Natively Designed to run on Spark
- Works with Hadoop eco system (via Spark)
  Hadoop is THE Big Data platform for on-premise deployments

Plays nicely with other BigDL frameworks

- Use existing Tensorflow or Caffe at scale in BigDL
- Train new models based on existing TF / Caffe models

BIGDL USE CASES

Fraud detection

Sentiment analysis

Image recognition

Find more at: https://github.com/intel-analytics/analytics-zoo/

GPUs and CPUs

GPUS (GRAPHICS PROCESSING UNITS)

GPUs have addressed past issues in training performance

- Example: Tensorflow - optimized to run well on GPUs.

CPU in past not vectorized for parallel compute

- Meant that GPUs were much faster for deep learning

Modern Intel Xeon CPUs have vectorized linear algebra

- Properly optimized, approaches speed of GPUs
- CPUs are now a credible alternative to running on GPUs
- Cost Advantage and Scalability

INTEL MATH KERNEL LIBRARY (MKL)

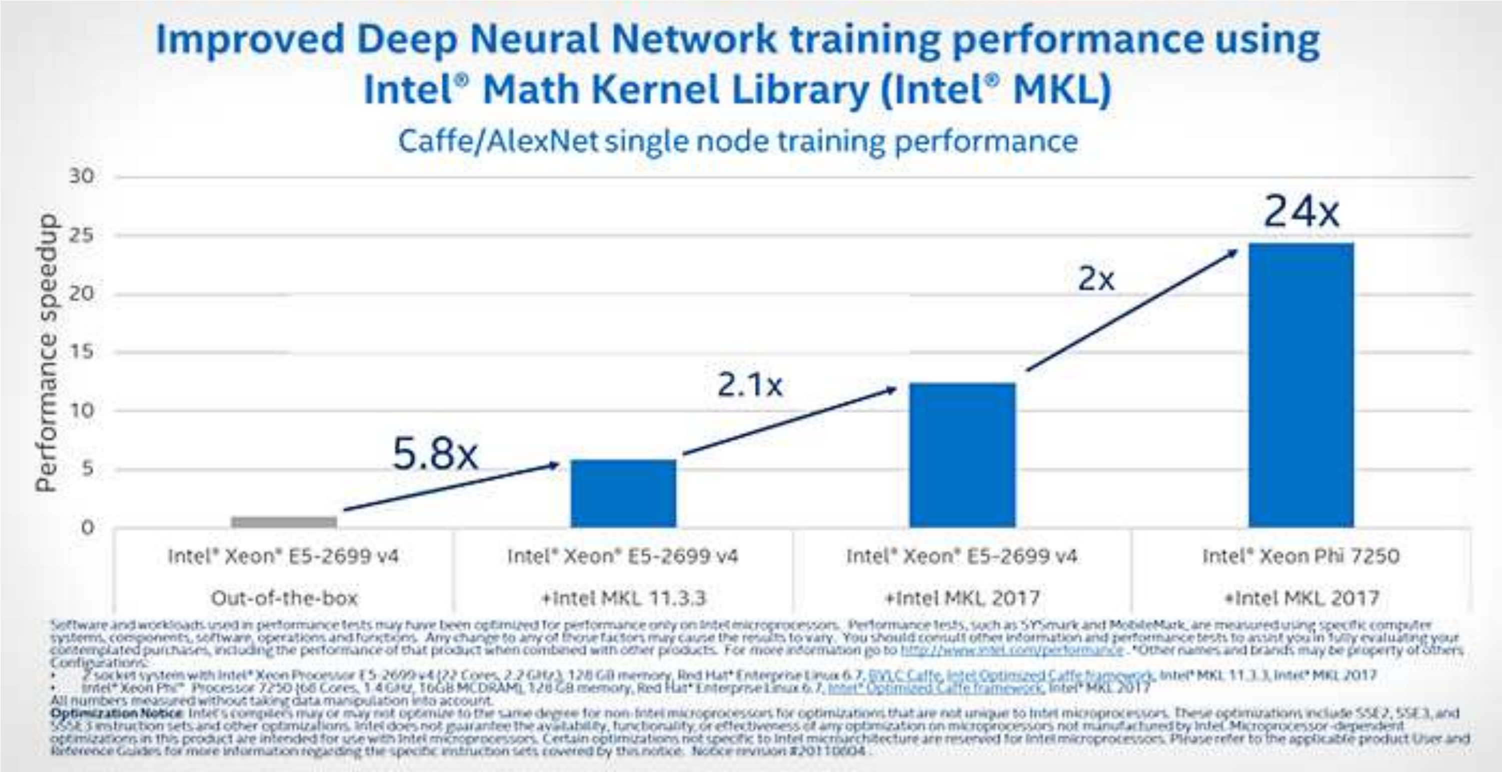
Features highly optimized, threaded, and vectorized math functions that maximize performance on each processor family.

Utilizes industry-standard C and Fortran APIs for compatibility with popular BLAS, LAPACK, and FFTW functions—no code changes required.

Dispatches optimized code for each processor automatically without the need to branch code.

Provides priority support, connecting you directly to Intel engineers for confidential answers to technical questions

INTEL MKL PERFORMANCE



CPU VERSUS GPU FOR BIG DATA

CPU offers higher scalability at lower cost versus GPU

Optimized Software and libraries on CPU allow single-node performance to approach GPU performance.

GPU plus CPU architectures can be effective for smaller number of nodes, when cost is not a concern.

”Big Compute” versus “Big Data”

Running BigDL

AWS re:Invent

RUNNING BIGDL

Developing:
Use the following to develop your BigDL apps effortlessly

- Docker
- VM Sandbox

Deploying:
Cloud ready deployment

- Amazon AMI

DEMO: GETTING STARTED WITH BIGDL

We will provide:

- Docker
- Sandbox VM
- AWS Marketplace AMI

BigDL Summary

BigDL offers outstanding scalability and performance

BigDL optimizes TCO by running being tuned and optimized for Intel Xeon Processors

BigDL brings deep learning to Spark Clusters and Hadoop Datasets

BigDL can be used to deploy Tensorflow and Caffe models to big data.

IMAGE RECOGNITION WITH APACHE SPARK AND BIGDL

Alex Kalinin | VP, AI/Machine Learning | Sizmek

AWS re:Invent

ABOUT ME

Alex Kalinin
VP, AI/Machine Learning | Sizmek
alex.kalinin@sizmek.com
Linkedin: linkedin.com/in/alexkalinin/

AI-POWERED MARKETING AND OPTIMIZATION

100,000,000,000 requests per day

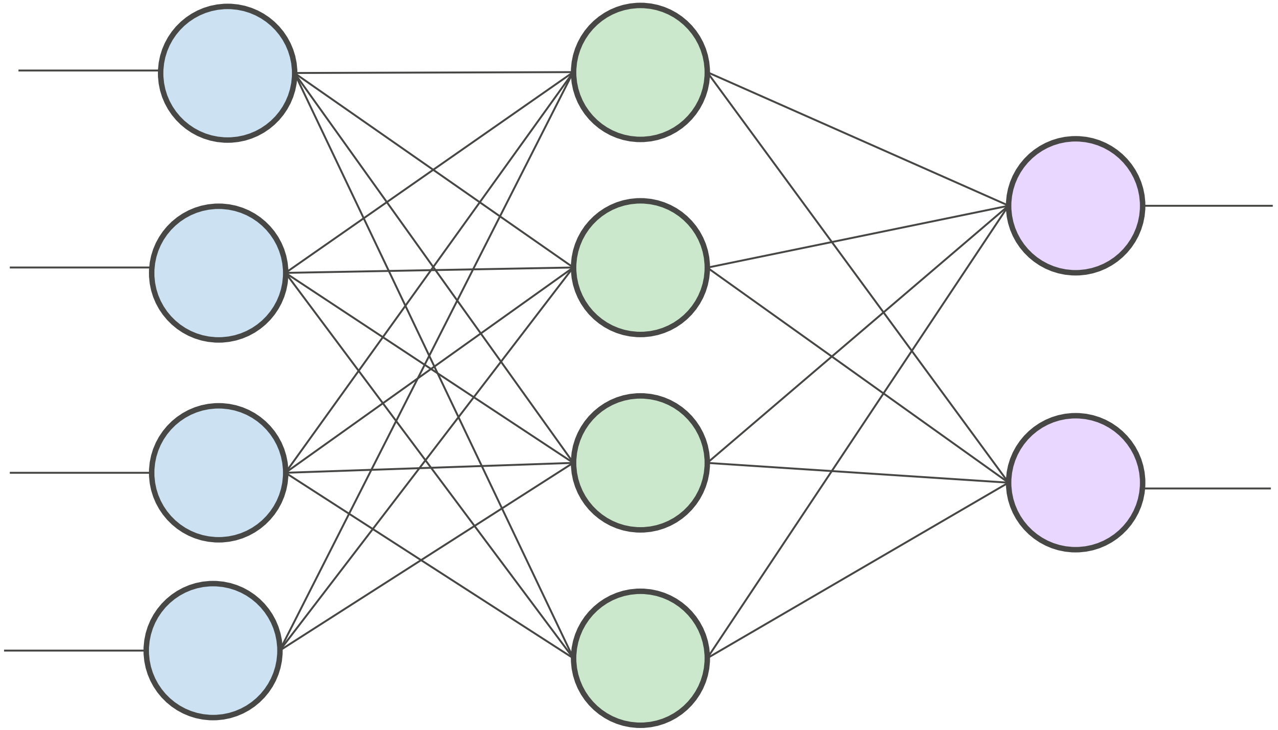
70,000,000 / minute

1,200,000 / sec

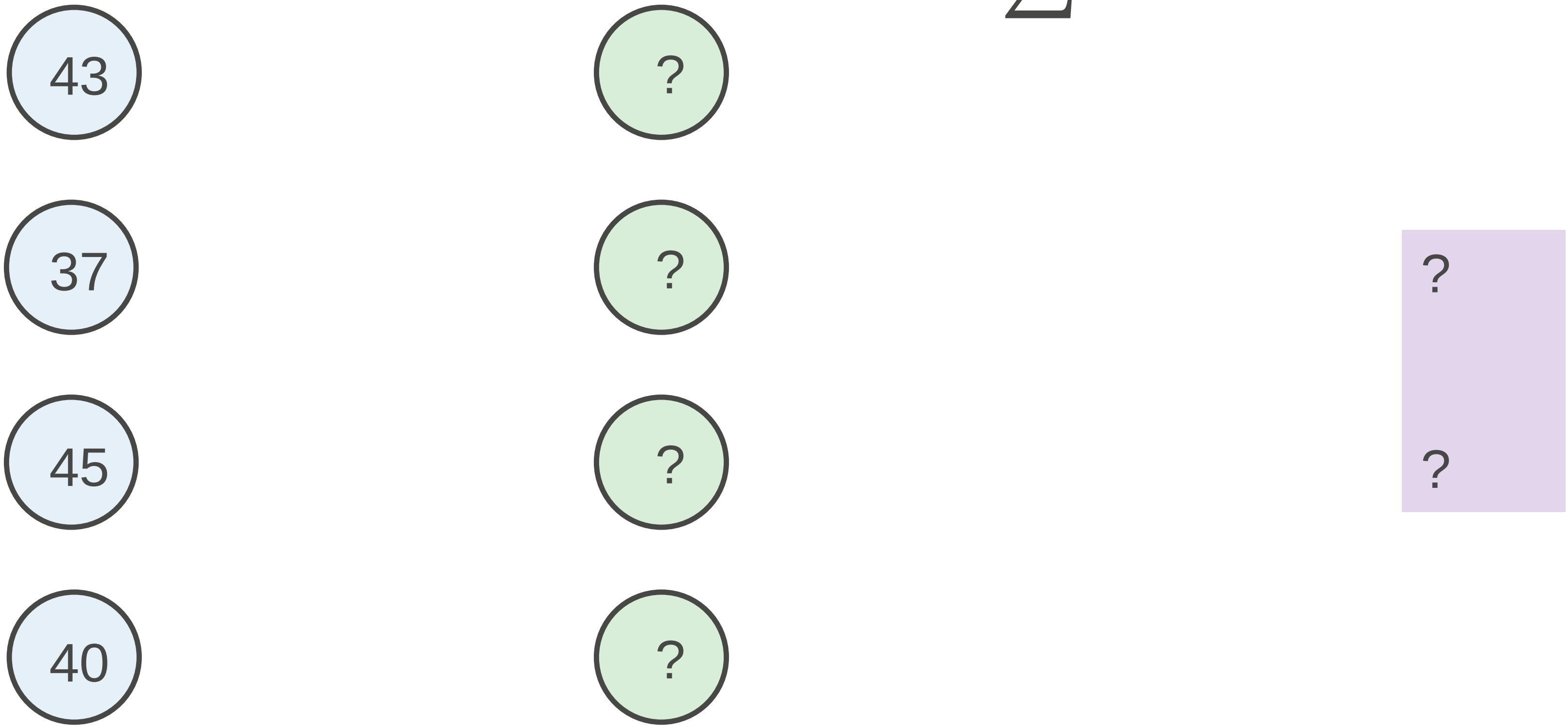
PBs of training data

https://www.linkedin.com/in/alexkalinin/
alex.kalinin@sizmek.com

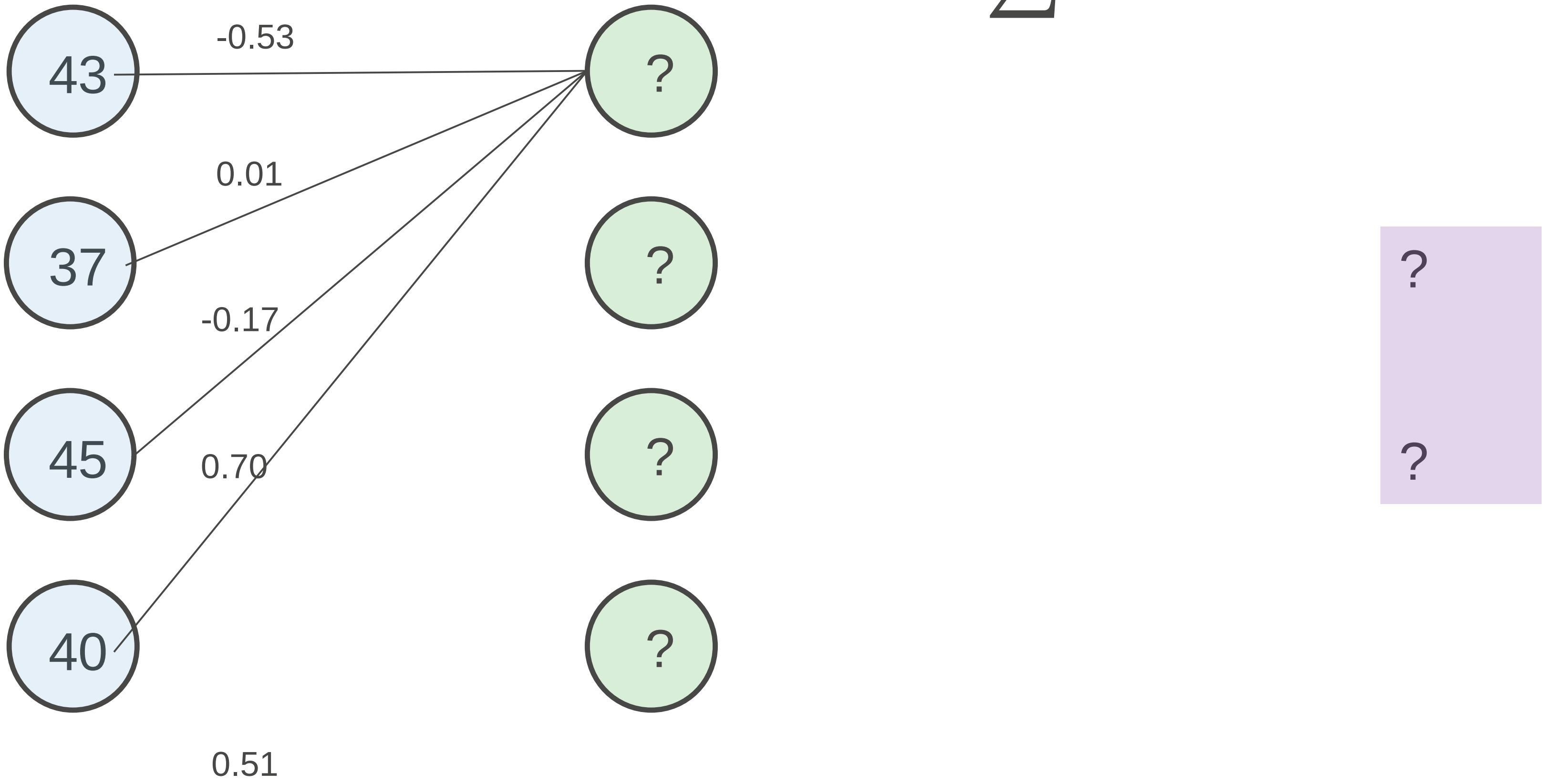
FEED-FORWARD NETWORK



$$y = \sum (w_i * x_i)$$

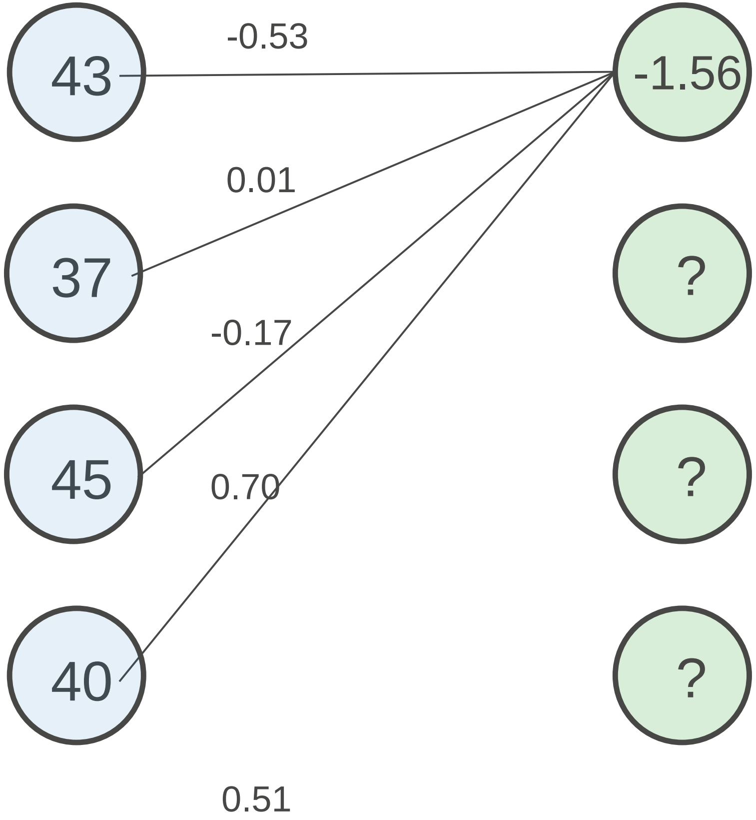


$$y = \sum (w_i * x_i)$$



re:Invent

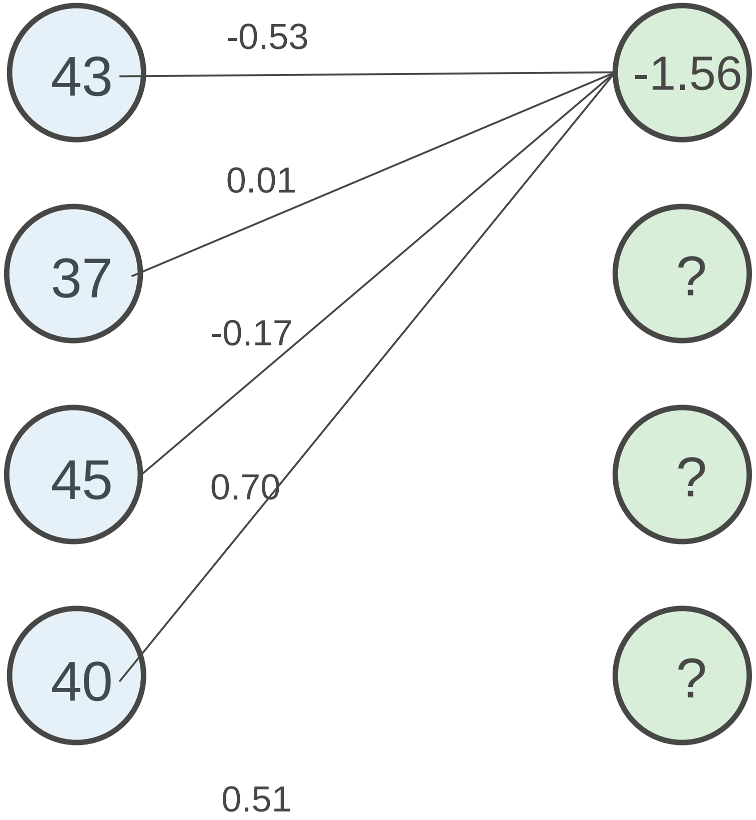
$$y = \sum (w_i * x_i)$$

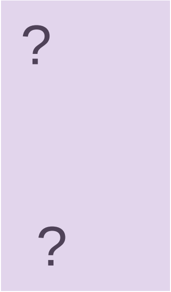


?

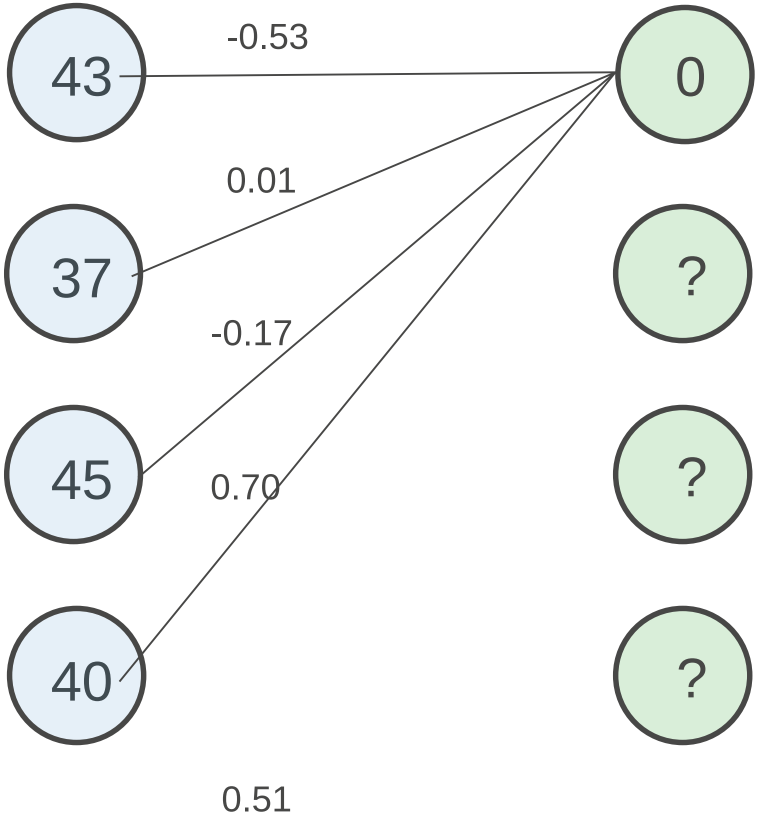
?

$$y = ReLU\left(\sum (w_i * x_i)\right)$$





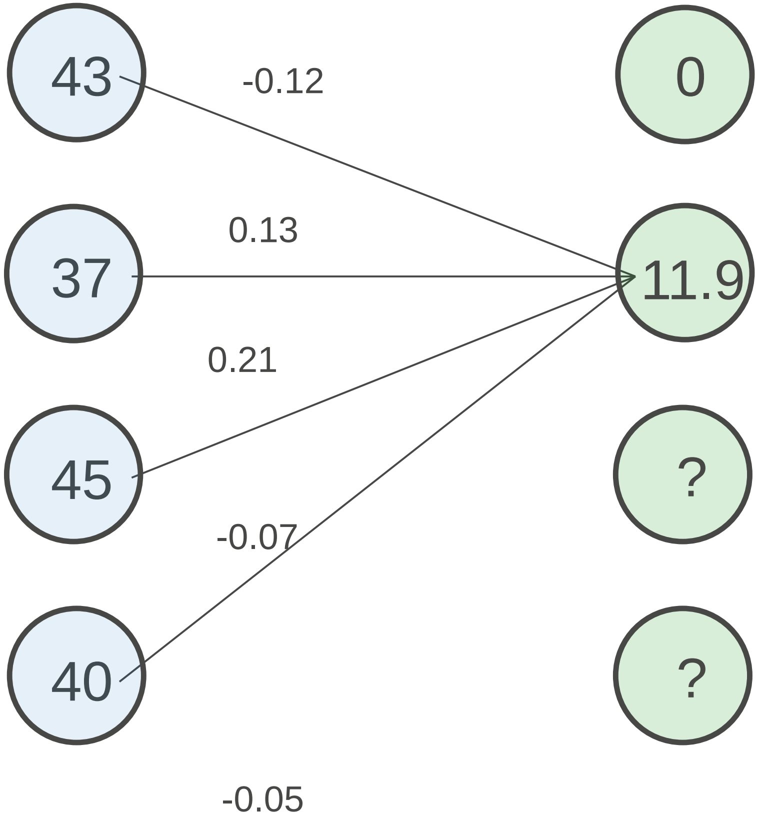
$$y = ReLU(\sum (w_i * x_i))$$

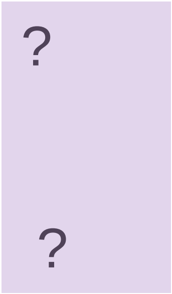


?

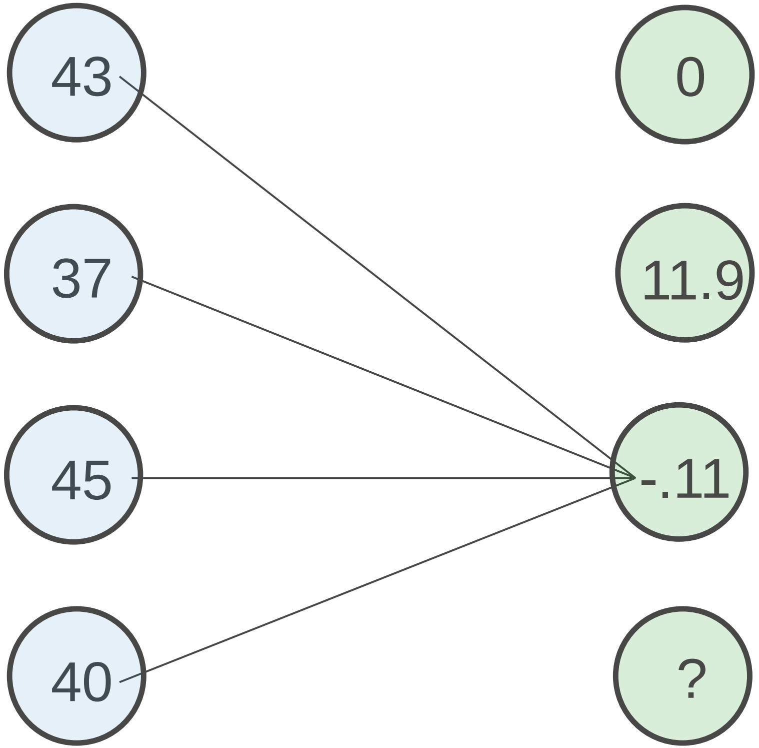
?

$$y = ReLU(\sum(w_i * x_i))$$





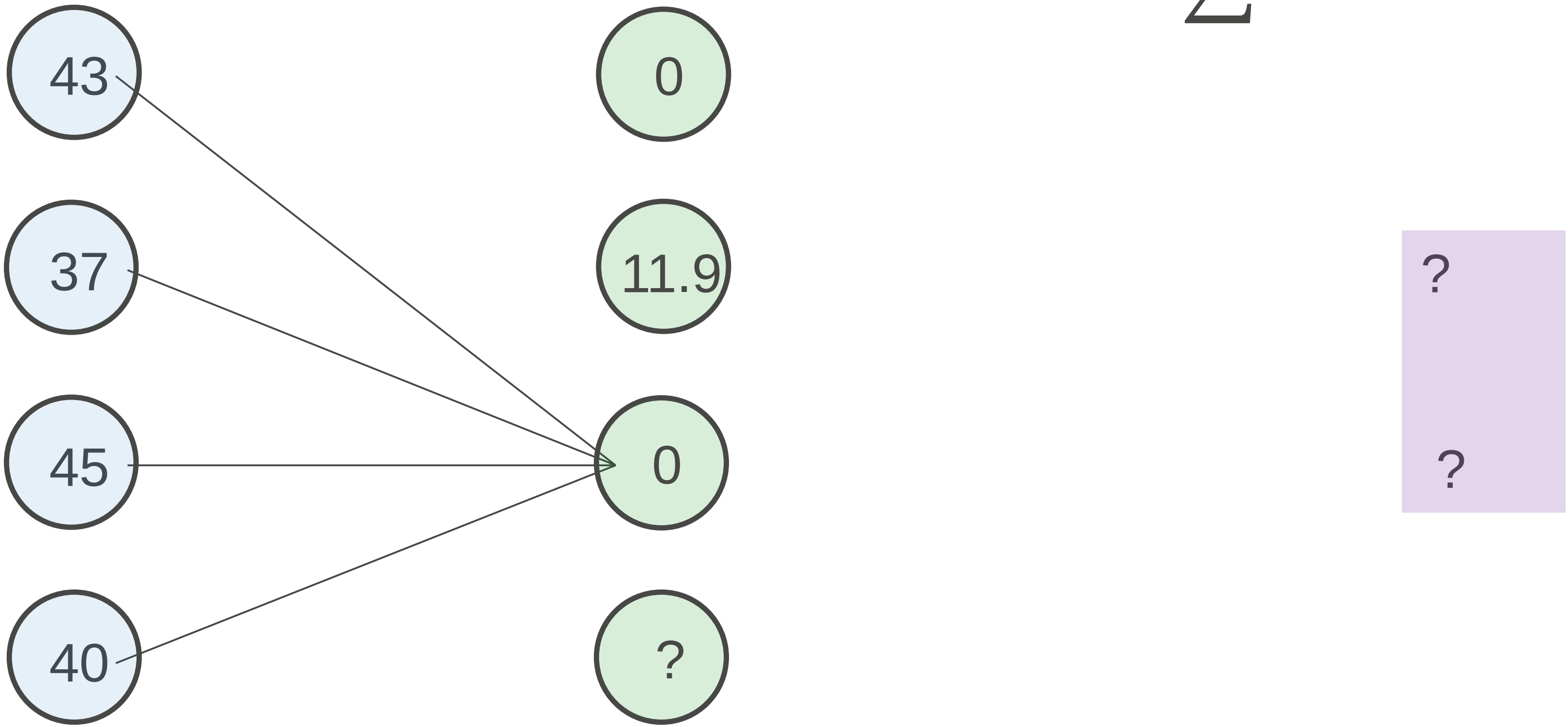
$$y = ReLU(\sum(w_i * x_i))$$



?

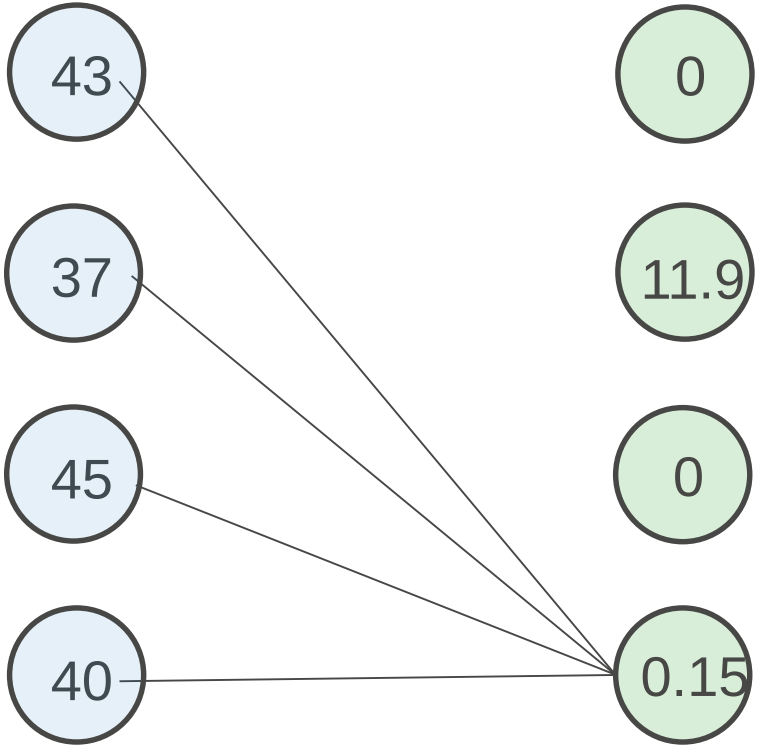
?

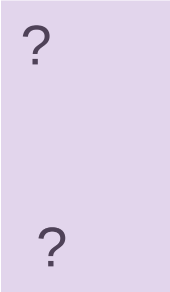
$$y = ReLU(\sum(w_i * x_i))$$



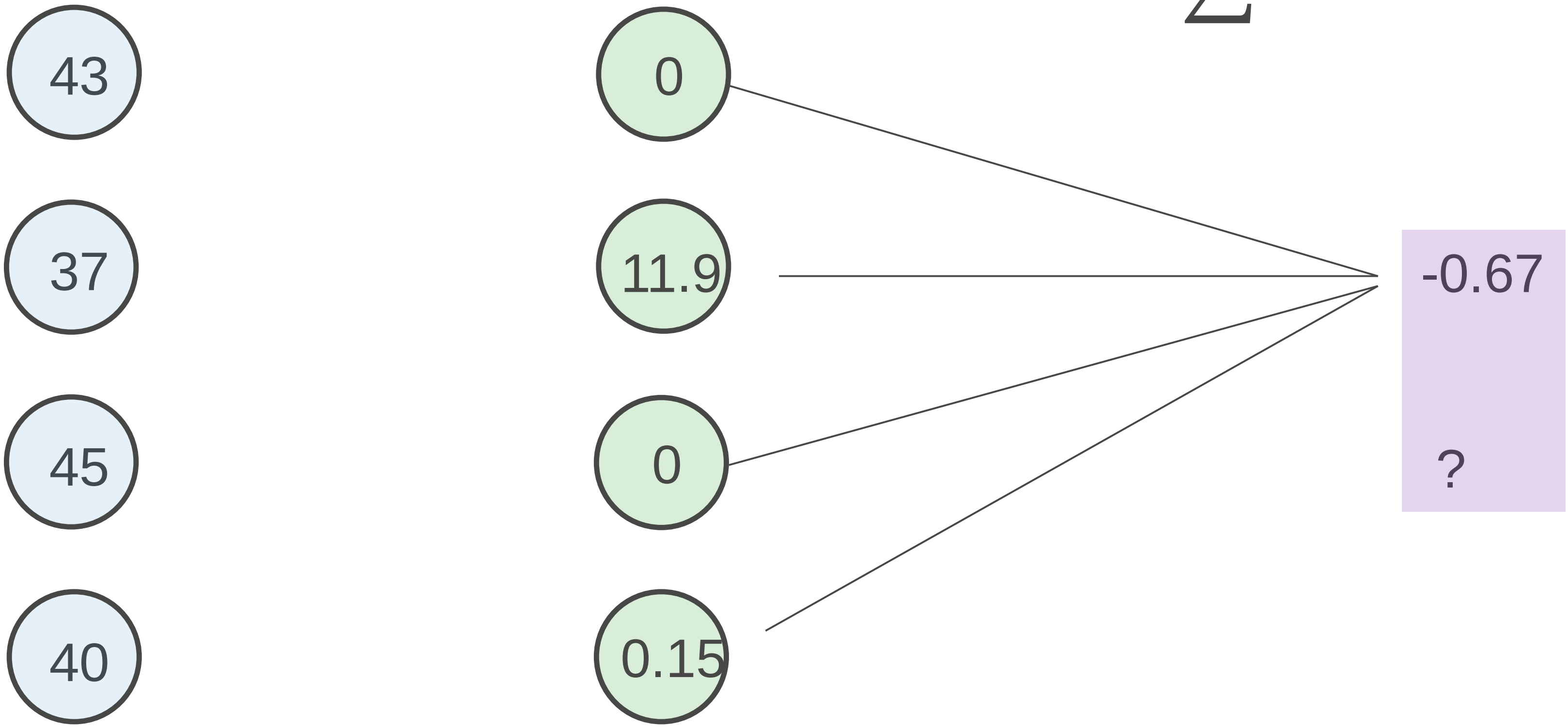
AWS re:Invent

$$y = ReLU\left(\sum (w_i * x_i)\right)$$

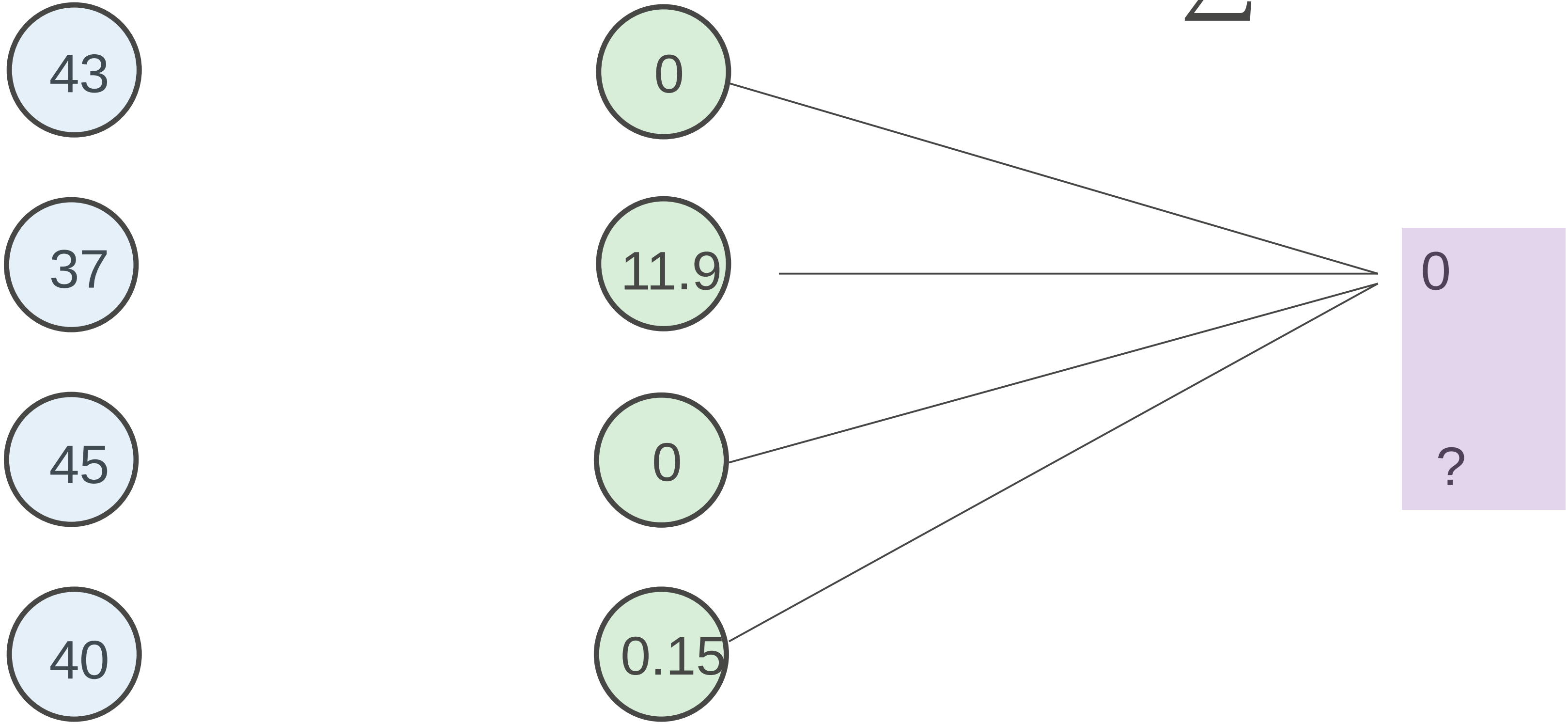




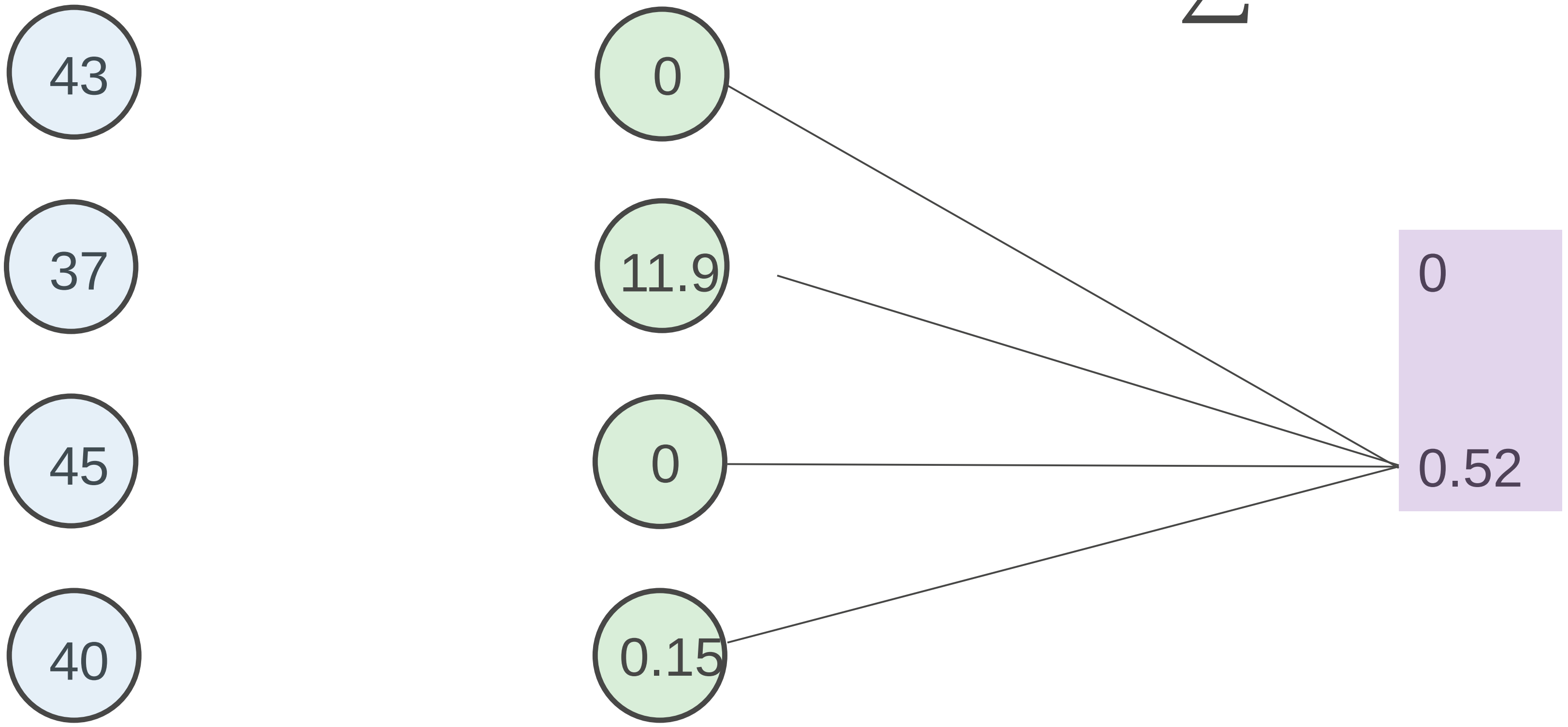
$$y = ReLU\left(\sum (w_i * x_i)\right)$$



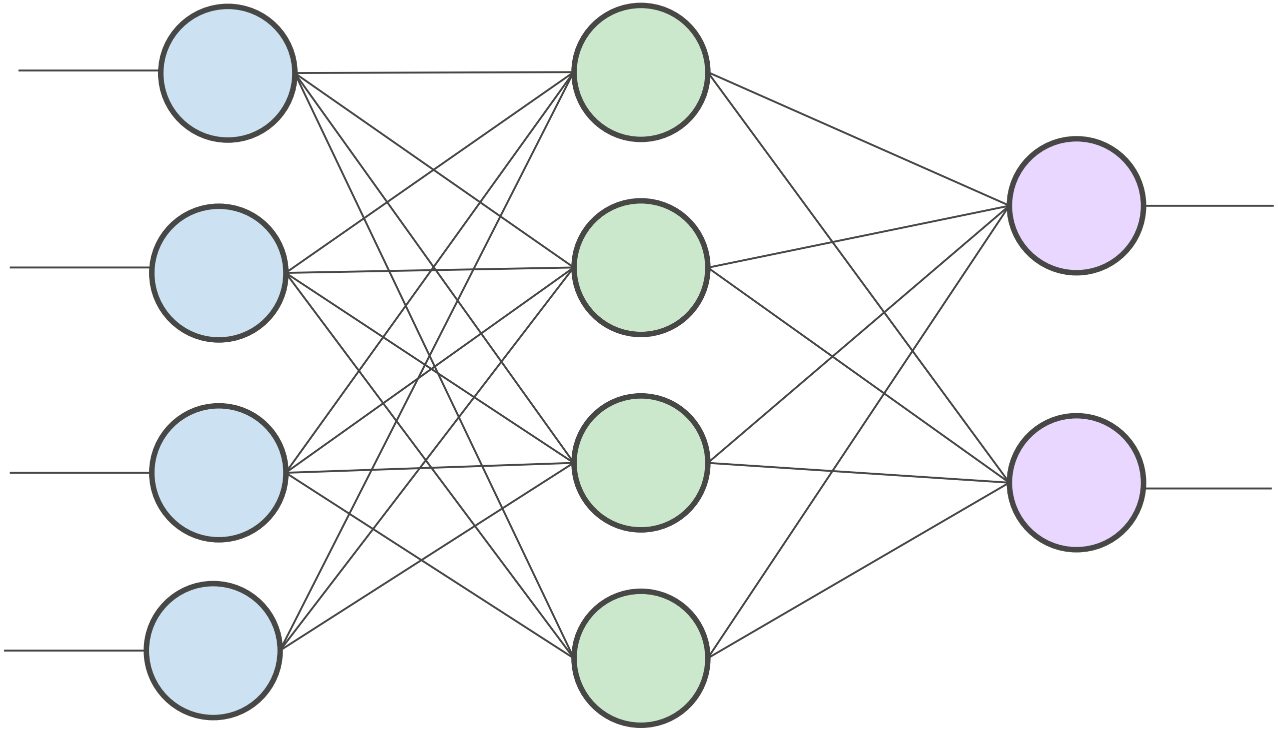
$$y = ReLU(\sum(w_i * x_i))$$

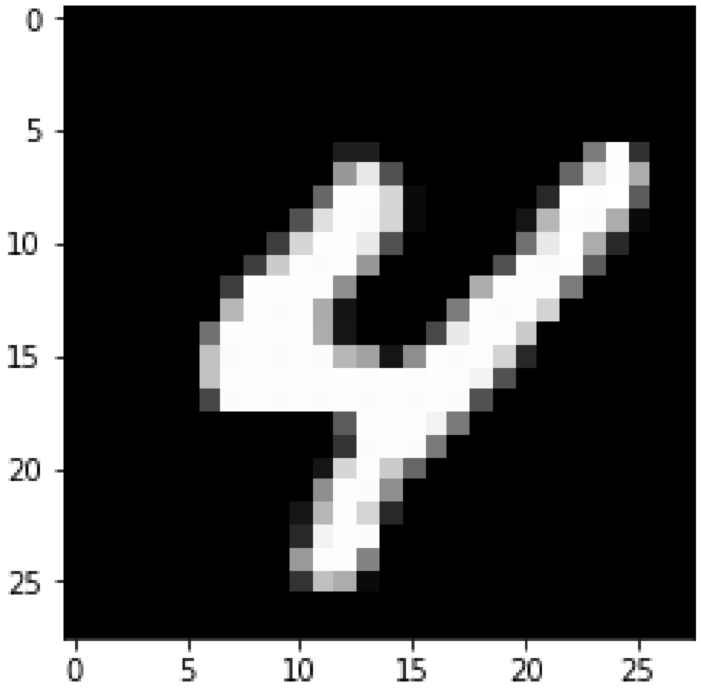


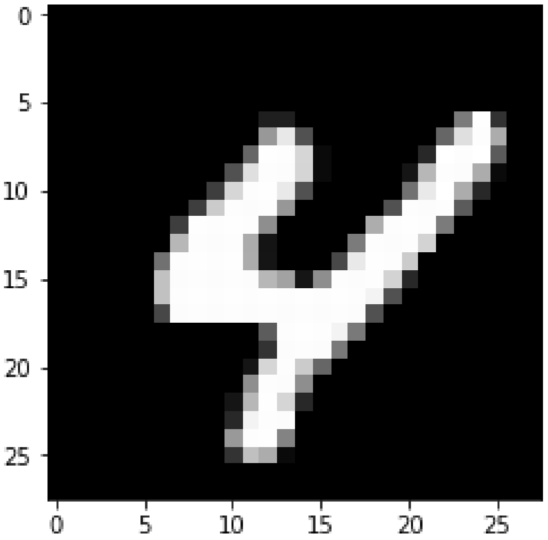
$$y = ReLU(\sum(w_i * x_i))$$



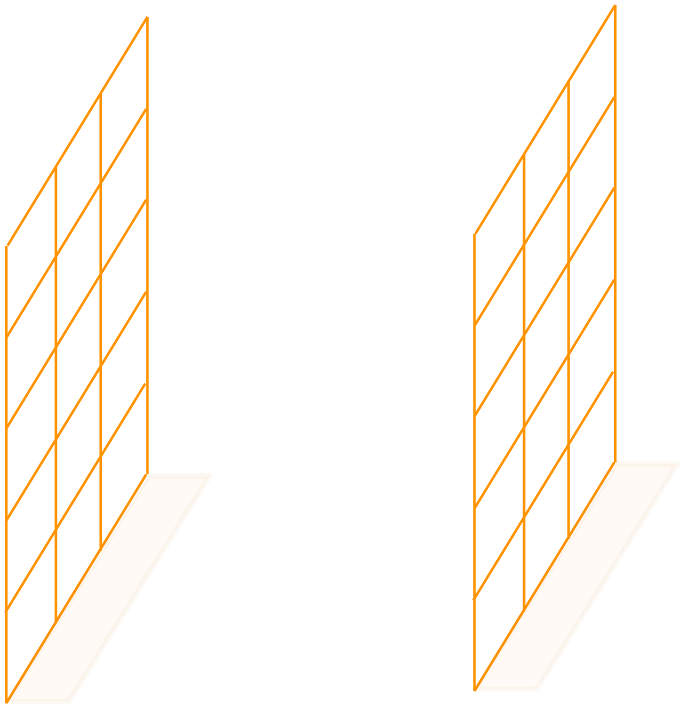
FEED-FORWARD NETWORK



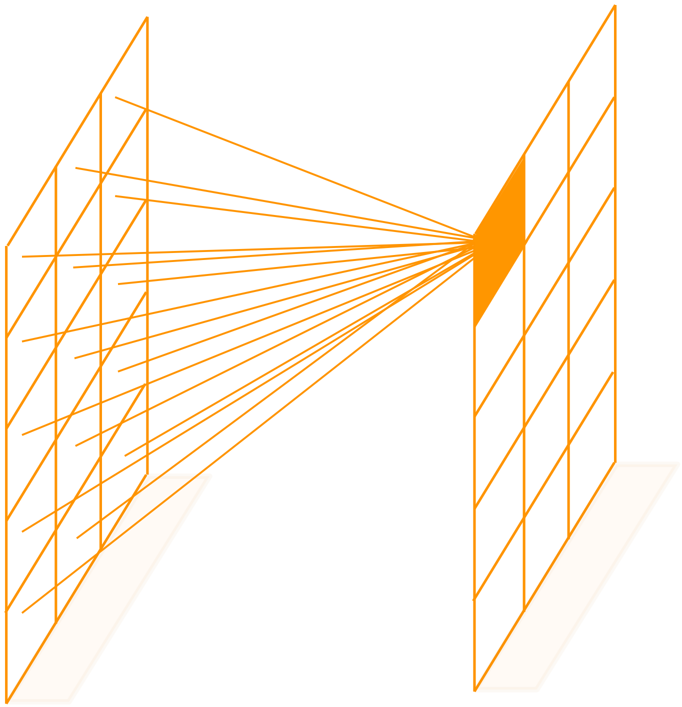

0
5
10
15
20
25
0
5
10
15
20
25




```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  31  31   0   0   0   0   0   0   0   0   0 123 254  50   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0 152 232  82   0   0   0   0   0   0   0 102 223 253 172   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 102 254 253 214  10   0   0   0   0   0  41 255 253 255  91   0   0]
 [  0   0   0   0   0   0   0   0   0   0  82 223 253 252 213  10   0   0   0   0  21 183 253 252 172  10   0   0]
 [  0   0   0   0   0   0   0   0   0  62 214 253 254 233  82   0   0   0   0   0 113 233 255 172  41   0   0   0]
 [  0   0   0   0   0   0   0   0  62 203 253 252 253 151   0   0   0   0   0  82 253 252 253  91   0   0   0   0]
 [  0   0   0   0   0   0   0  62 254 253 254 253 142   0   0   0   0   0 173 253 254 253 123   0   0   0   0   0]
 [  0   0   0   0   0   0   0 183 253 252 253 171  20   0   0   0   0 123 253 252 253 212   0   0   0   0   0   0]
 [  0   0   0   0   0   0 113 253 254 253 254 172  21   0   0   0  72 233 254 253 203   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0 193 252 253 252 253 252 183 162  21 142 253 252 253 212  40   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0 193 253 254 253 254 253 254 253 254 253 254 253 244  81   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  71 252 253 252 253 252 253 252 253 252 253 252  81   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  92 253 254 253 244 122   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  51 252 253 252 122   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  21 214 253 203 102   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 142 253 252 142   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  21 183 254 213  41   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  41 243 253 252   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 153 253 254 131   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  51 192 172  10   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

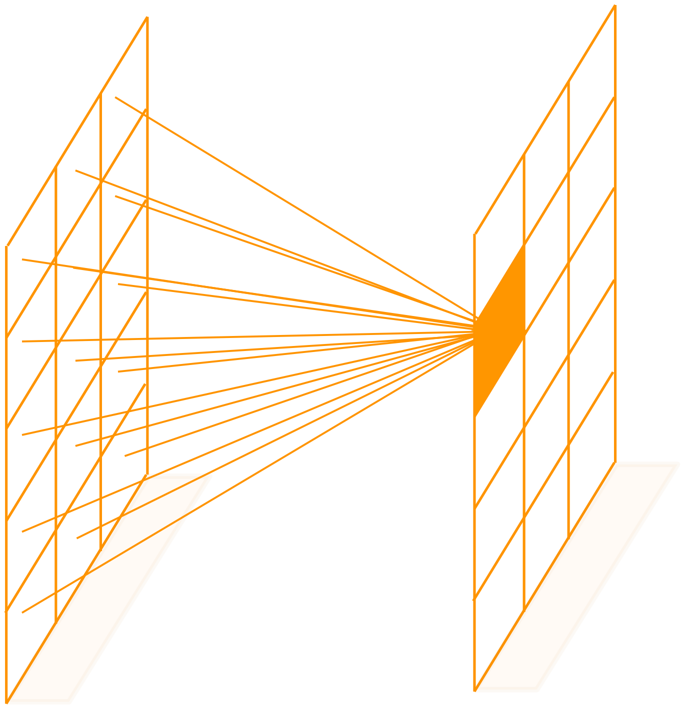


FULLY CONNECTED

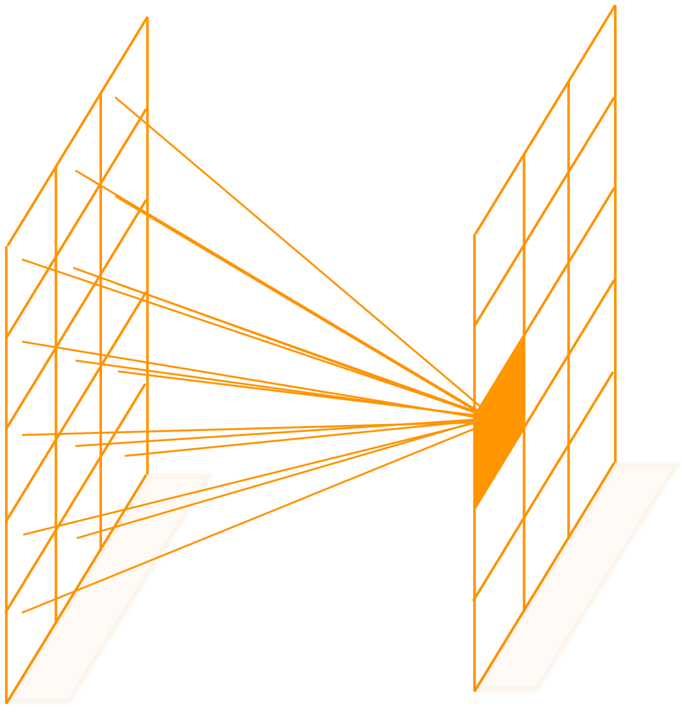


AWS re:Invent

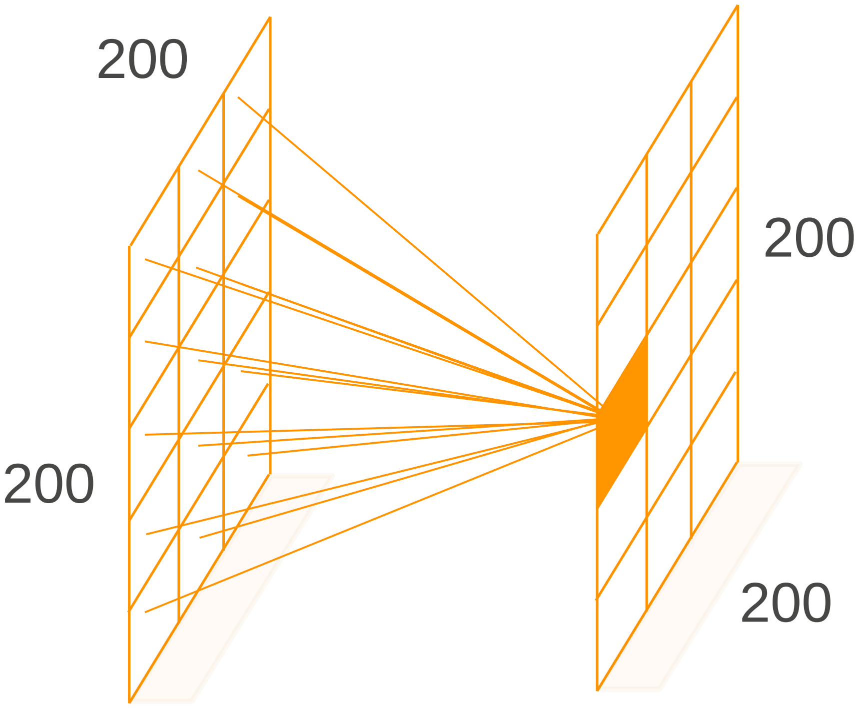
FULLY CONNECTED



FULLY CONNECTED



FULLY CONNECTED



Input Size: 40,000

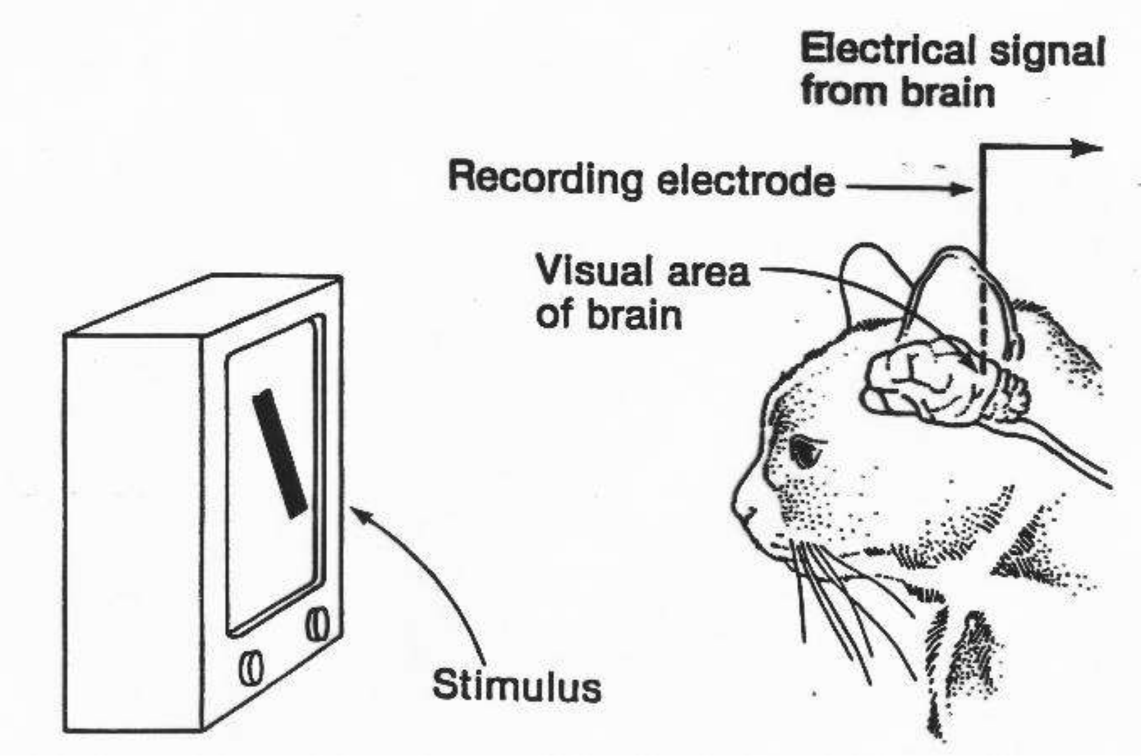
Connections: 1,600,000,000

10 layers: 16 billion

INVENTION OF CONVOLUTIONAL NEURAL NETWORKS

- LeNet-5 network developed in 1998 by Yann LeCun
- Torsten Hubel and David Wiesel

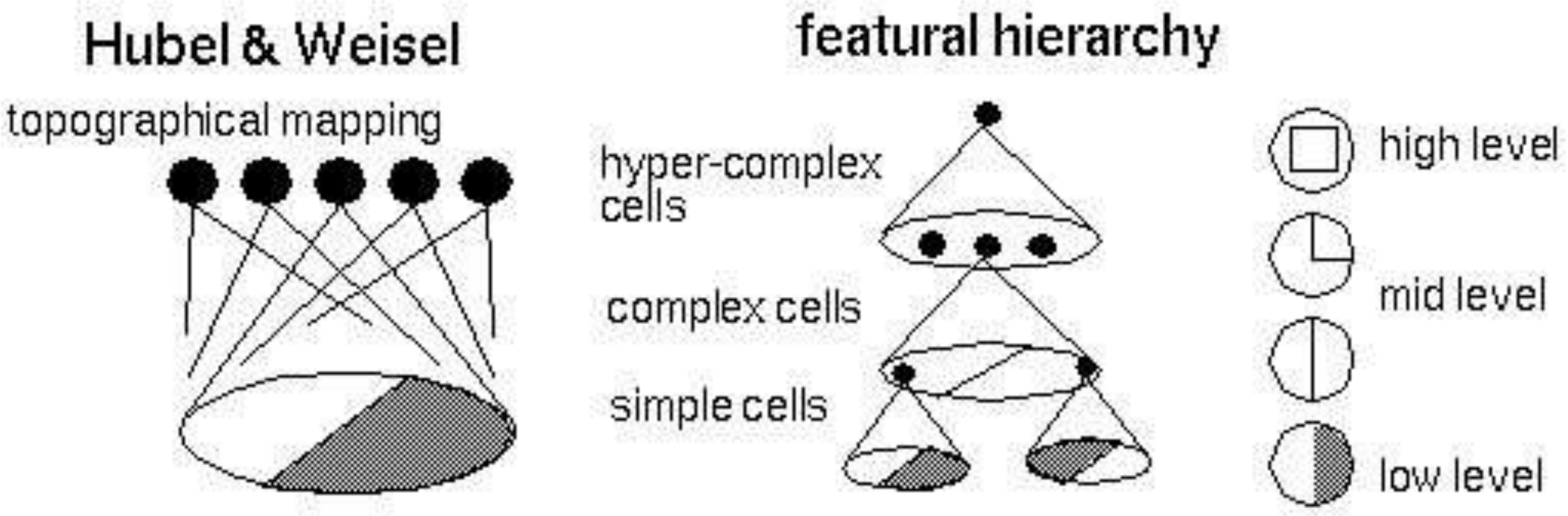
HUBEL & WIESEL



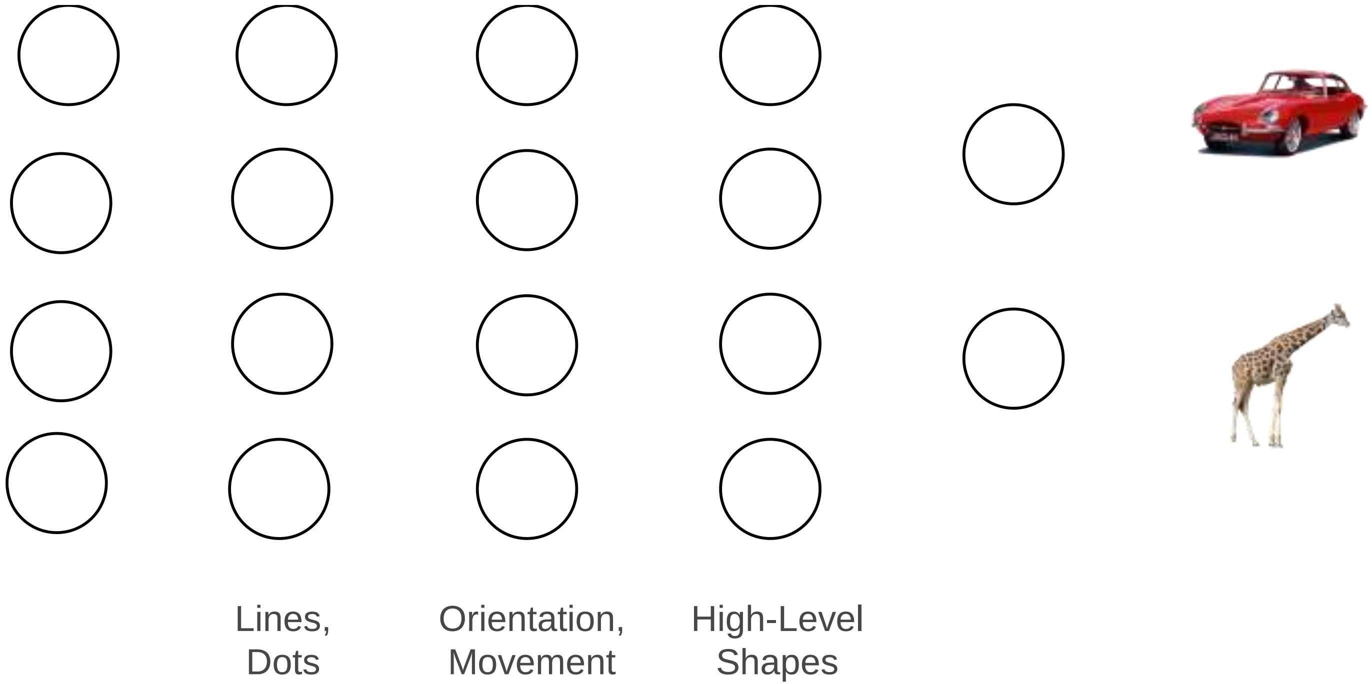
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1363130/

AWS re:Invent

HIERARCHICAL & LOCAL VISUAL CORTEX



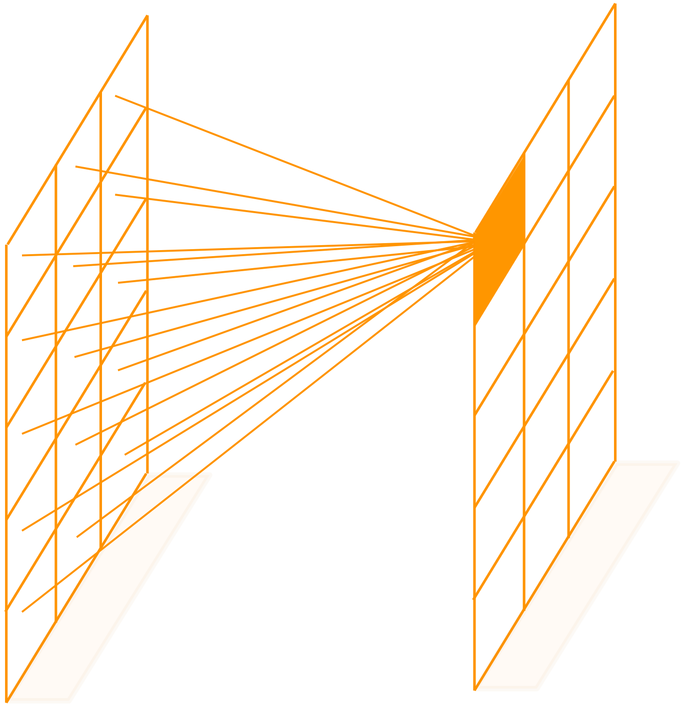
HIERARCHICAL & LOCAL VISUAL CORTEX



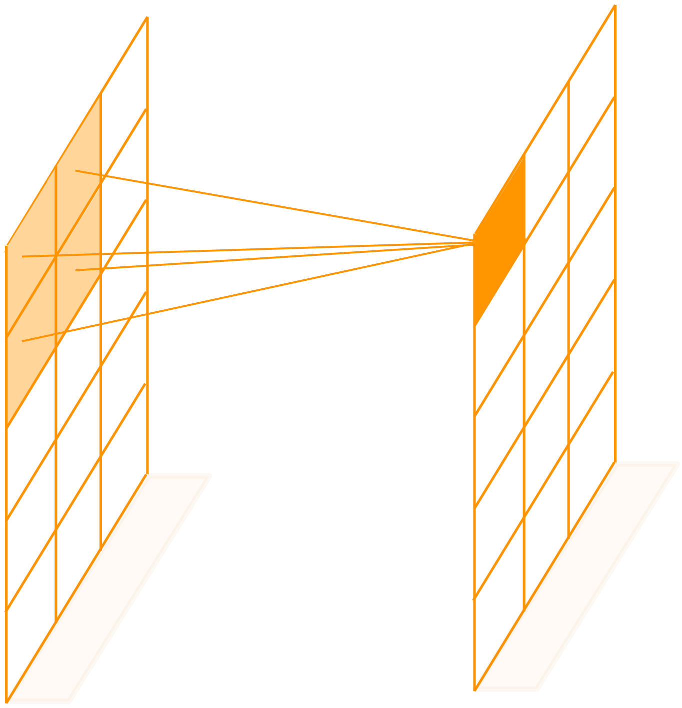
KEY FEATURES OF CONVOLUTIONAL NETWORK

- Convolution
- Pooling

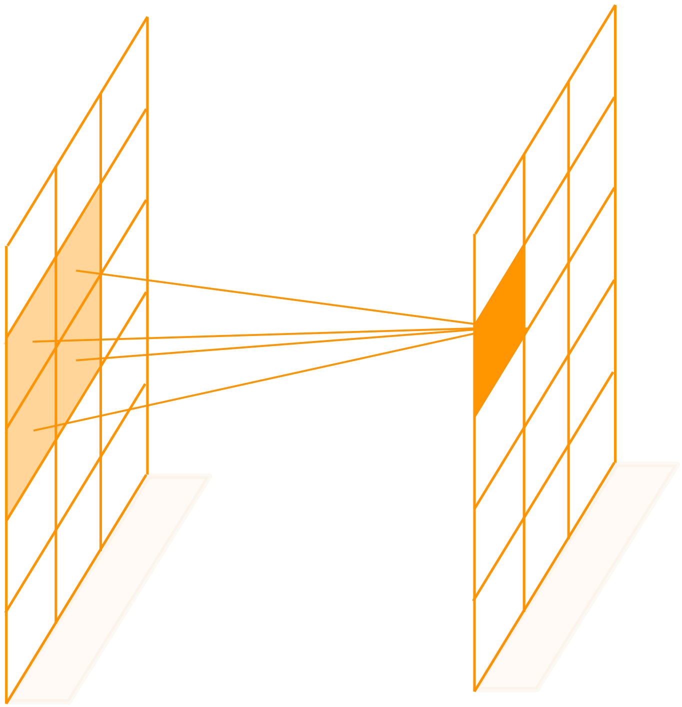
FULLY CONNECTED



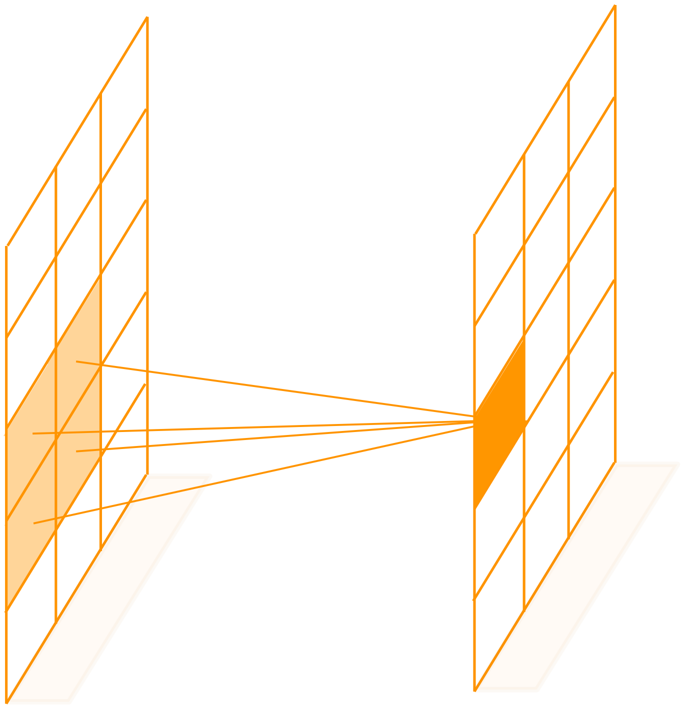
CONVOLUTION



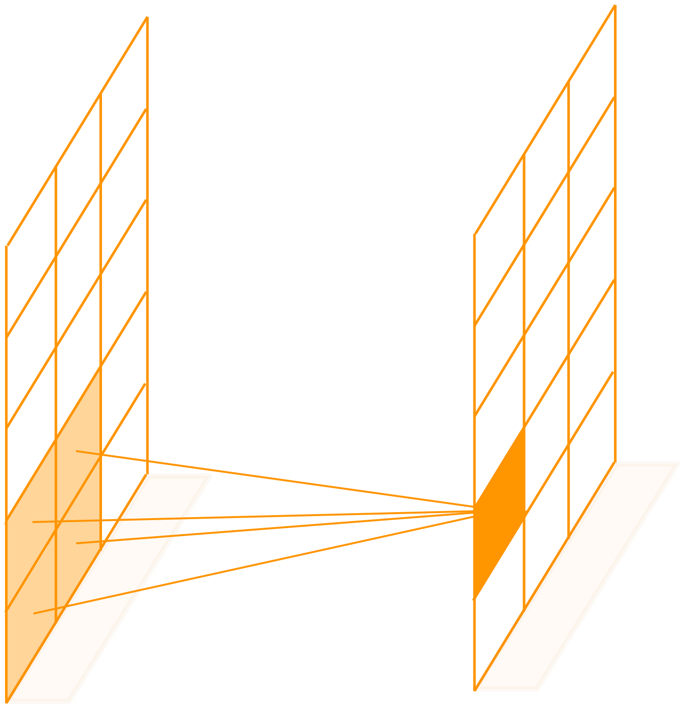
CONVOLUTION



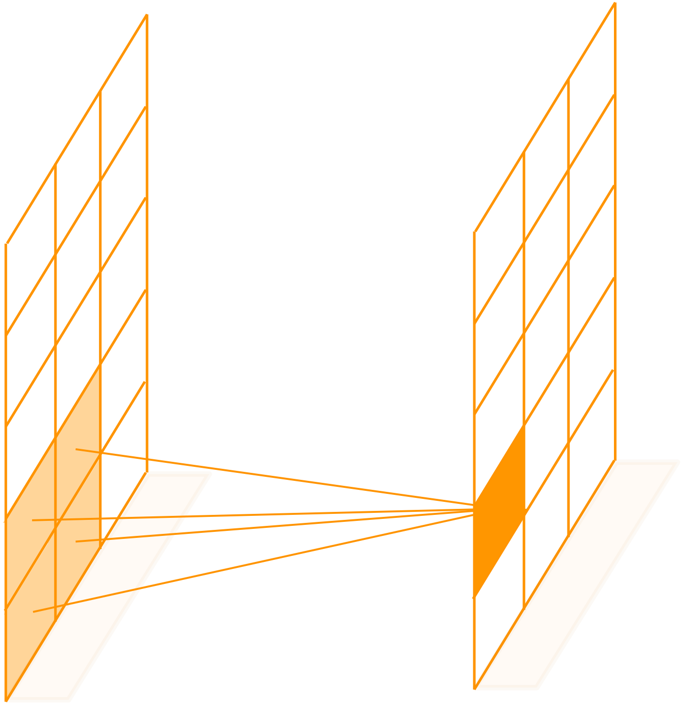
CONVOLUTION



CONVOLUTION

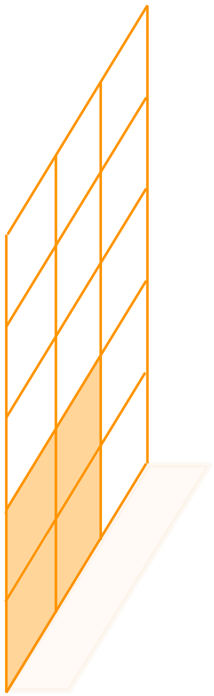


CONVOLUTION



Only **four** weights

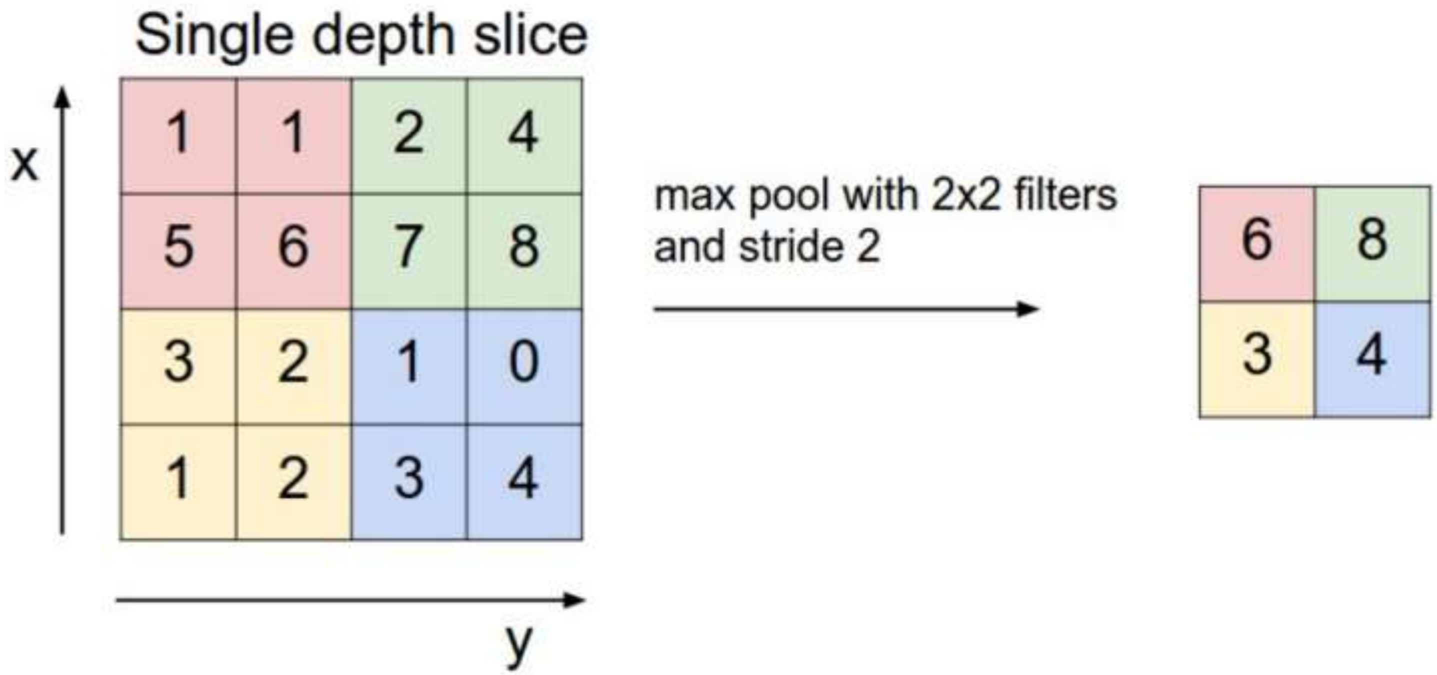
CONVOLUTION



Filter

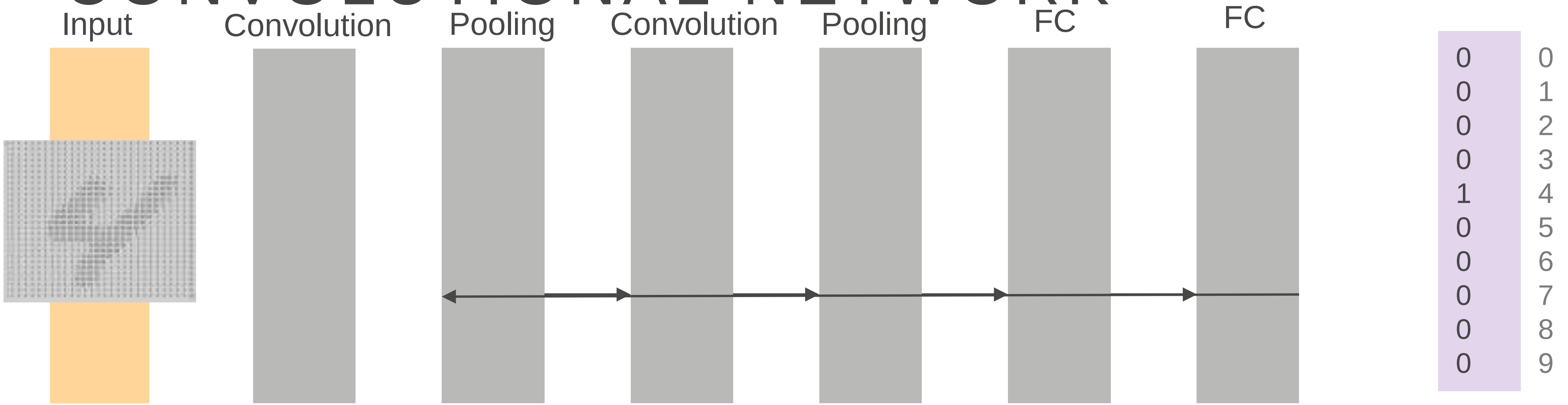
0.10	-0.06
0.24	0.17

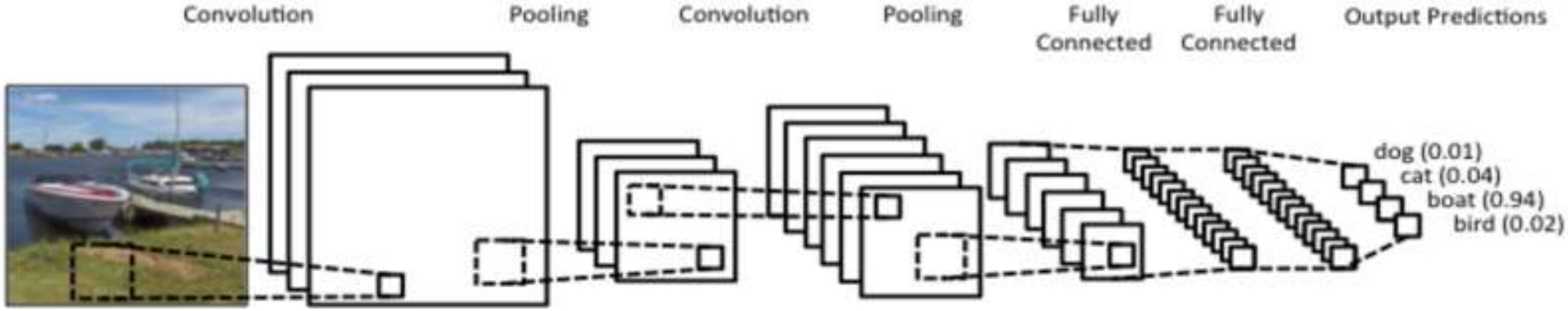
POOLING



Source: https://cs231n.github.io/convolutional-networks/

CONVOLUTIONAL NETWORK





Source: https://www.clarifai.com/technology

http://scs.ryerson.ca/~aharley/vis/conv/flat.html

AWS re:Invent

DEMO

GitHub: https://github.com/alex-kalinin/lenet-bigdl

Getting Started With BigDL

AWS re:Invent

ABOUT ME

Sujee Maniyam
Founder / Principal @ Elephant Scale
Practitioner and Trainer in Data Engineering and Data Science

Author

- "**Hadoop and Spark**" video training on O'Reilly Media
- "**HBase Design Patterns**"
- "**Hadoop illuminated**"

elephant SCALE

sujee@elephantscale.com
Linkedin: linkedin.com/in/sujeemaniyam

AWS re:Invent

aws

RUNNING BIGDL

Developing:
Use the following to develop your BigDL apps effortlessly

- Docker
- VM Sandbox
- Amazon AMI

Deploying:
Cloud ready deployment

- Amazon AMI

GETTING STARTED WITH BIGDL

We will demonstrate

- Docker
- Sandbox VM
- AWS Marketplace AMI

Docker

Step 1 : Install Docker on your laptop

Step 2 : get docker image

docker pull elephantscale/bigdl-sandbox

Step 3 : download tutorials

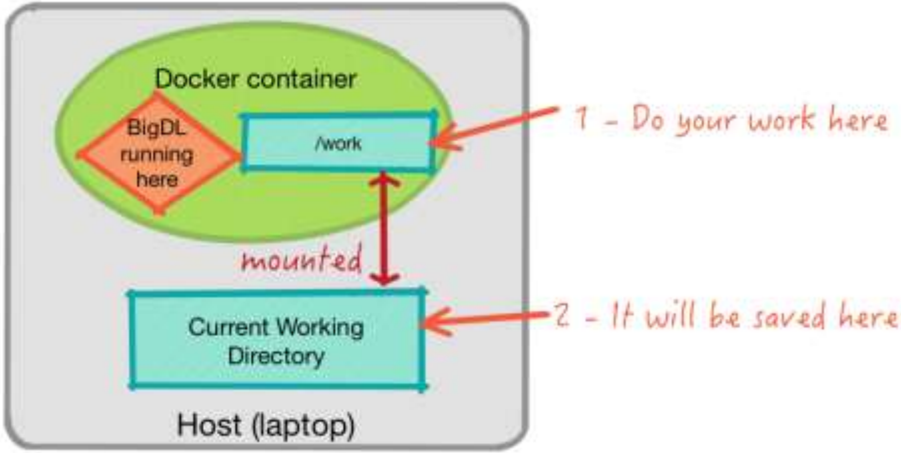
git clone https://github.com/elephantscale/bigdl-tutorials

Step 4 : Launch docker

cd bigdl-tutorials

./run-bigdl-docker.sh   elephantscale/bigdl-sandb

Step 5 : Go to Jupyter notebook



VM-Sandbox

Step 1 : Install VMware Player or VirtualBox on your laptop

Step 2 : Download BigDL-Sandbox image

http://elephantscale.com/sandbox/

Step 3 : (In Sandbox) download tutorials

git clone https://github.com/elephantscale/bigdl-tutorials

Step 4 : (In Sandbox) Run BigDL natively

cd bigdl-tutorials

./run-bigdl-native.sh

Step 5 : (In Sandbox) Go to Jupyter notebook

Docker on AWS

Step 1 : Spin up an AMI (Ubuntu recommended)

Step 2 : Install Docker on the instance

Step 3 : get docker image

```
docker pull  elephantscale/bigdl-sandbox
```

Step 4 : download tutorials

```
git clone https://github.com/elephantscale/bigdl-tutorials
```

Step 5 : Launch docker

```
cd  bigdl-tutorials
./run-bigdl-docker.sh    elephantscale/bigdl-sandbox
```

Step 6 : Go to Jupyter notebook

AMI on AWS

Step 1 : Spin up BigDL AMI

Step 2 : download tutorials

git clone https://github.com/elephantscale/bigdl-tutorials

Step 3 : Run BigDL

cd bigdl-tutorials

./run-bigdl-native.sh

Step 4 : Go to Jupyter notebook

QUESTIONS

GitHub: https://github.com/alex-kalinin/lenet-bigdl

LinkedIn: https://www.linkedin.com/in/alexkalinin/

Notebooks and Resources

BigDL: software.intel.com/bigdl

Tutorials: github.com/dnielsen/bigdl-resources

Sandbox: elephantscale.com/sandbox

BigDL AMI: aws.amazon.com/marketplace/

Training: elephantscale.com

Slides: slideshare.net/dcnielsen/

Tim Fox
Elephant Scale

Sujee Maniyam
Elephant Scale

Alex Kalinin
Sizmek

Dave Nielsen
Intel Software

THANK YOU!