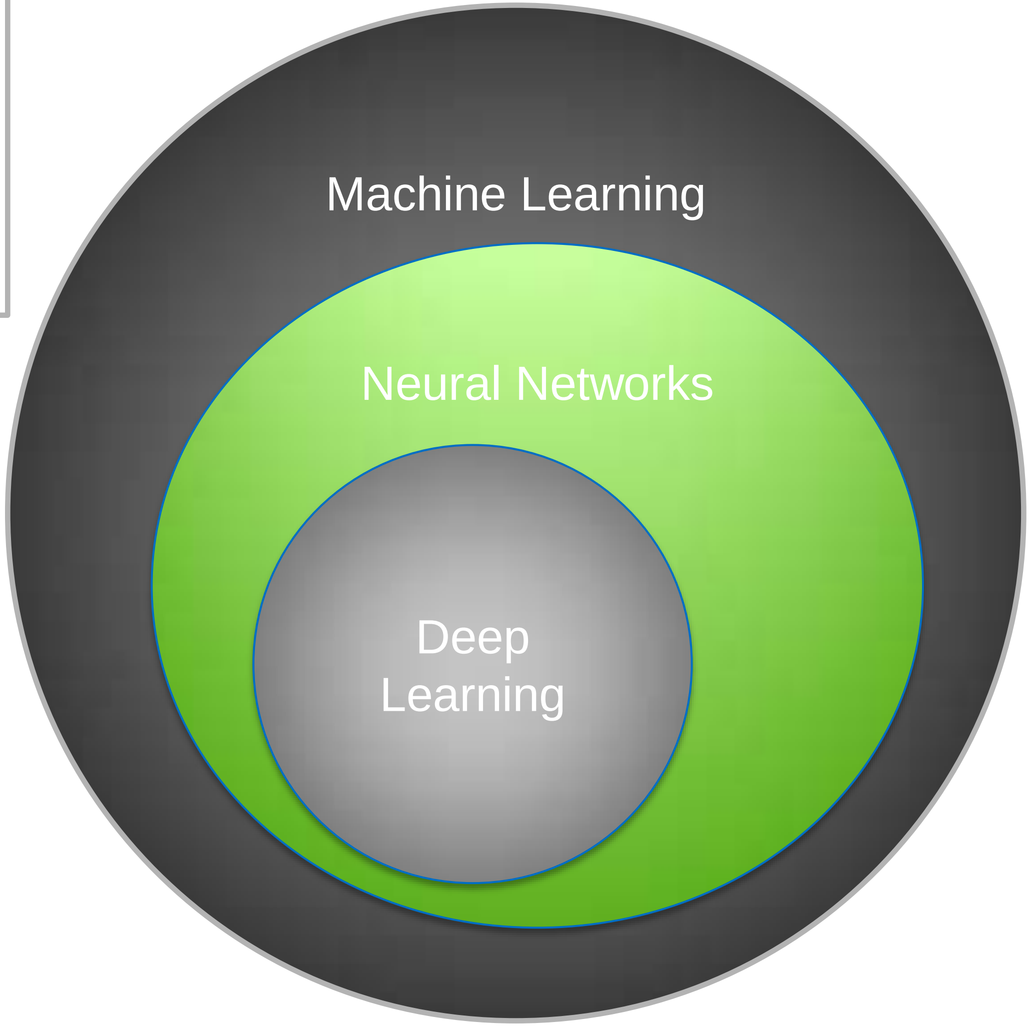
Deep Learning and GPUs
Intro and hands-on tutorial

Julie Bernauer – HPC Advisory Council Stanford Tutorial – 2017/02/07

NVIDIA

ML, Neural Nets and Deep Learning

Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed (Arthur Samuel, 1959).



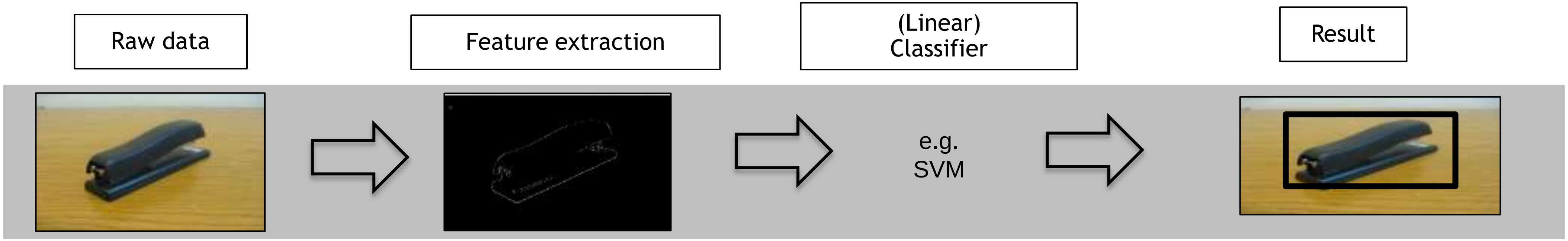
An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is inspired by the structure and functional aspects of biological neural networks.

Deep learning [...] consists of multiple hidden layers in an artificial neural network

from https://en.wikipedia.org/wiki/Machine_learning

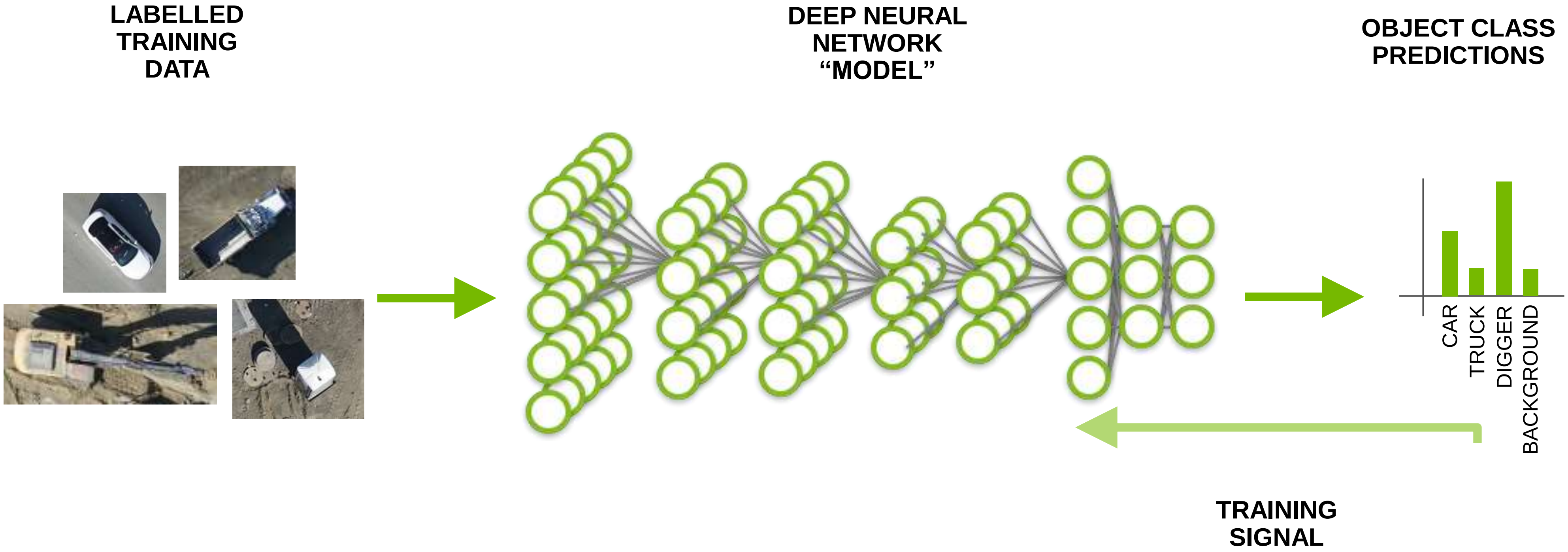
Object Recognition

Traditional Machine Learning / Computer Vision Approach



Object recognition

The Deep Learning Approach

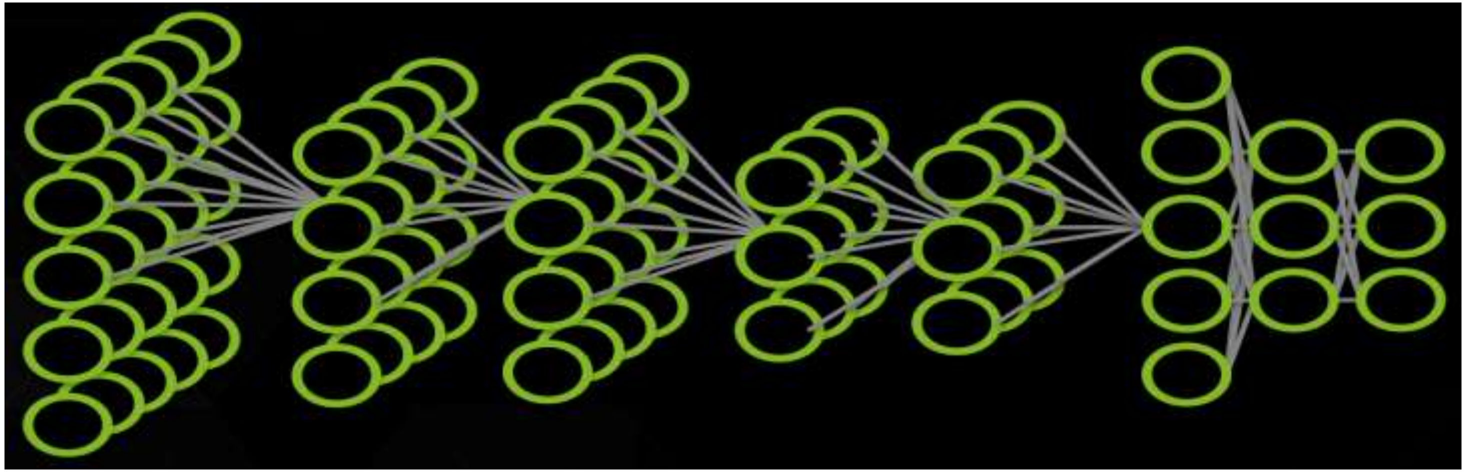


Artificial Neural Network

A collection of simple, trainable mathematical units that collectively learn complex functions

Hidden layers

Input layer



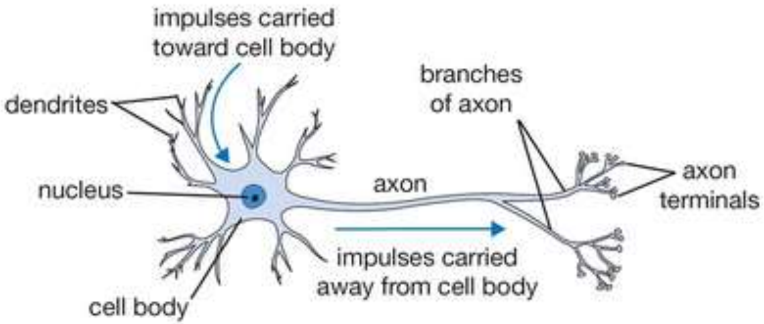
Output layer

Given sufficient training data an artificial neural network can approximate very complex functions mapping raw data to output decisions

Artificial Neuron

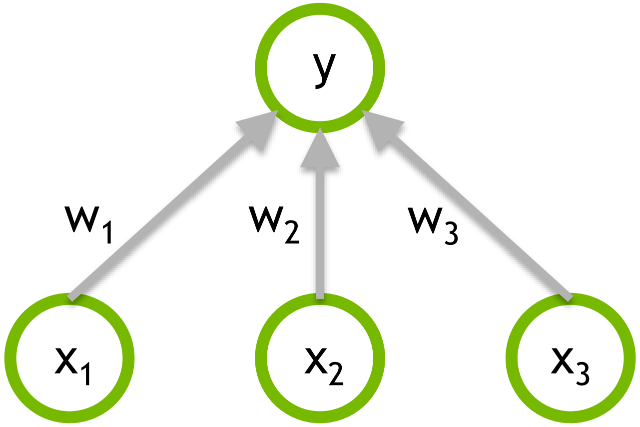
Mathematical Unit

Biological neuron



From Stanford cs231n lecture notes

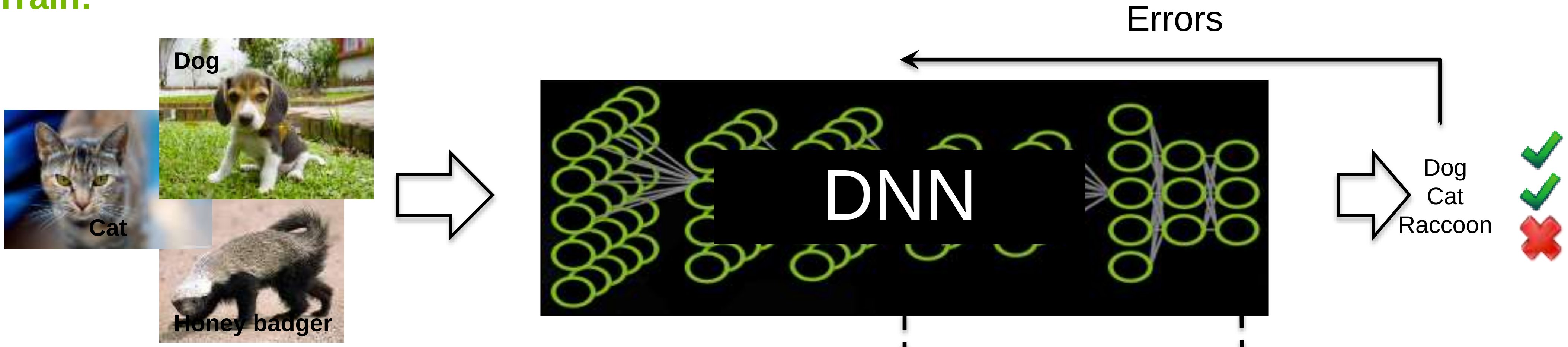
Artificial neuron



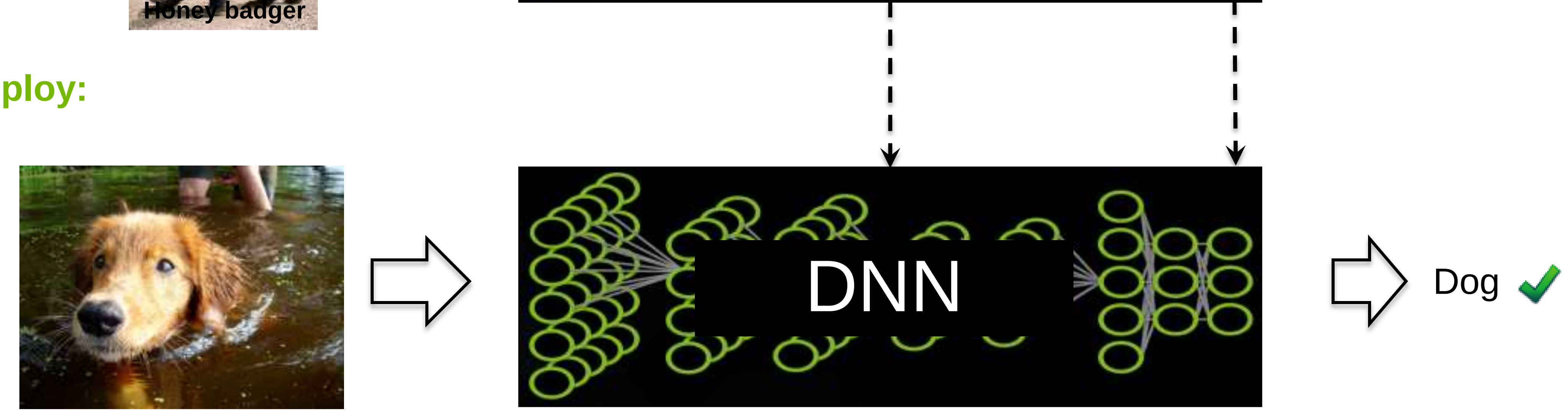
$$y=F(w_1x_1+w_2x_2+w_3x_3)$$

Deep Learning Approach

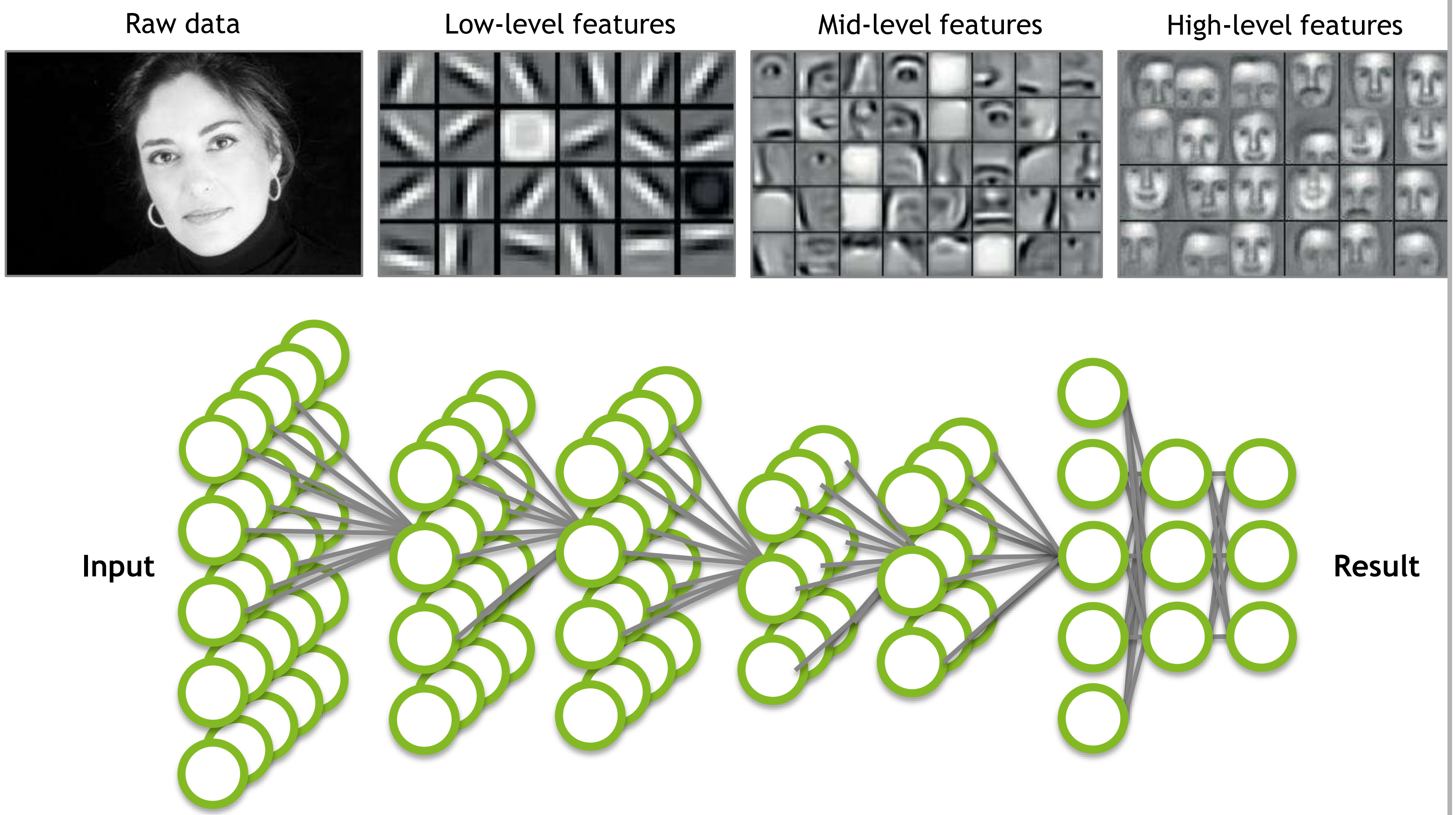
Train:



Deploy:



Deep Neural Network (DNN)



Application components:

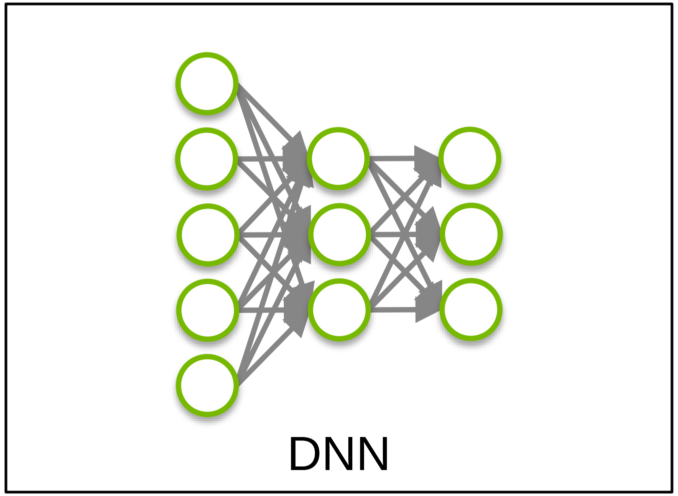
Task objective
e.g. Identify face

Training data
10-100M images

Network architecture
~10s-100s of layers
1B parameters

Learning algorithm
~30 Exaflops
1-30 GPU days

THE BIG BANG IN MACHINE LEARNING


DNN


BIG DATA

GPU

“Google’s AI engine also reflects how the world of computer hardware is changing. (It) depends on machines equipped with GPUs… And it depends on these chips more than the larger tech universe realizes.”

WIRED

Deep Learning Benefits

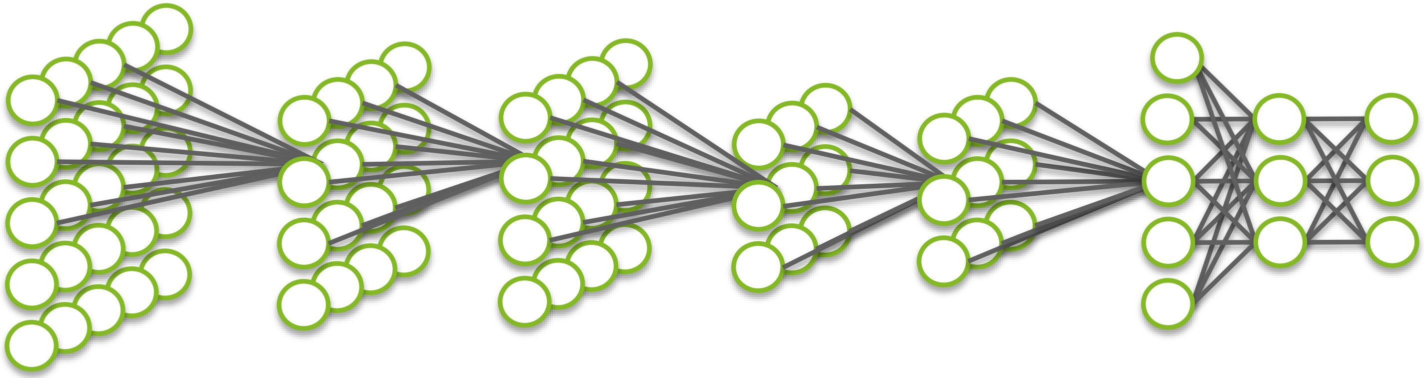
- Robust
  - No need to design the features ahead of time – features are automatically learned to be optimal for the task at hand
  - Robustness to natural variations in the data is automatically learned
- Generalizable
  - The same neural net approach can be used for many different applications and data types
- Scalable
  - Performance improves with more data, method is massively parallelizable

Three main kinds of networks

DNN – all fully connected layers

CNN – some convolutional layers

RNN – recurrent neural network, LSTM



Neural Network

In practice

Interpret AI task as the evaluation of complex function

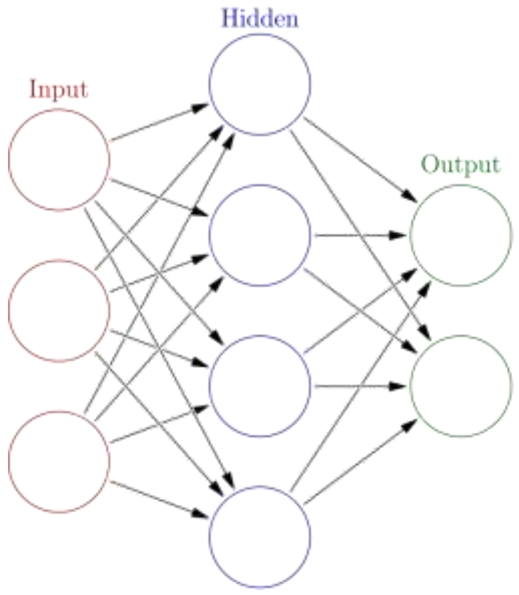
- Facial Recognition: Map a bunch of pixels to a name
- Handwriting Recognition: Image to a character

Neural Network: Network of interconnected simple “neurons”

Neuron typically made up of 2 stages:

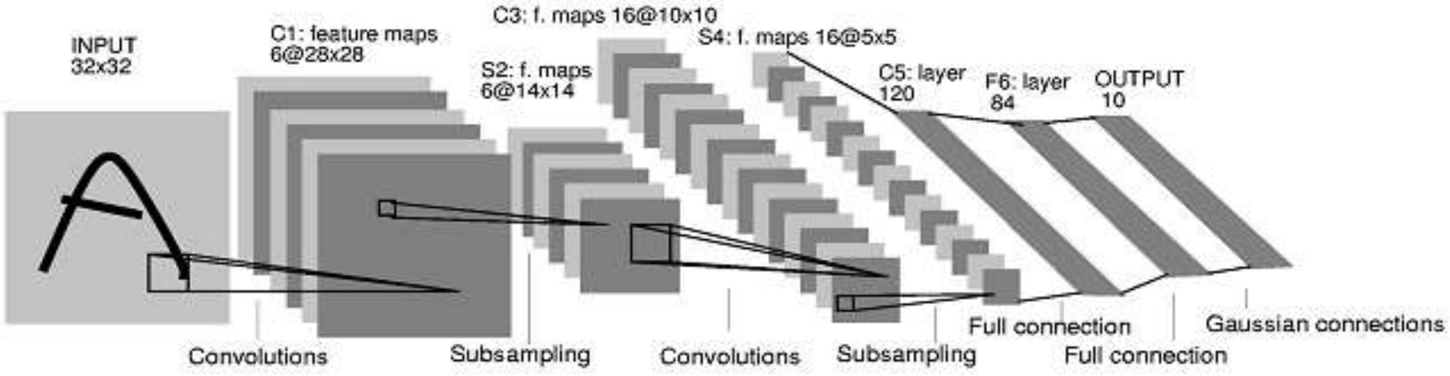
- Linear Transformation of data
- Point-wise application of non-linear function

In a CNN, Linear Transformation is a convolution

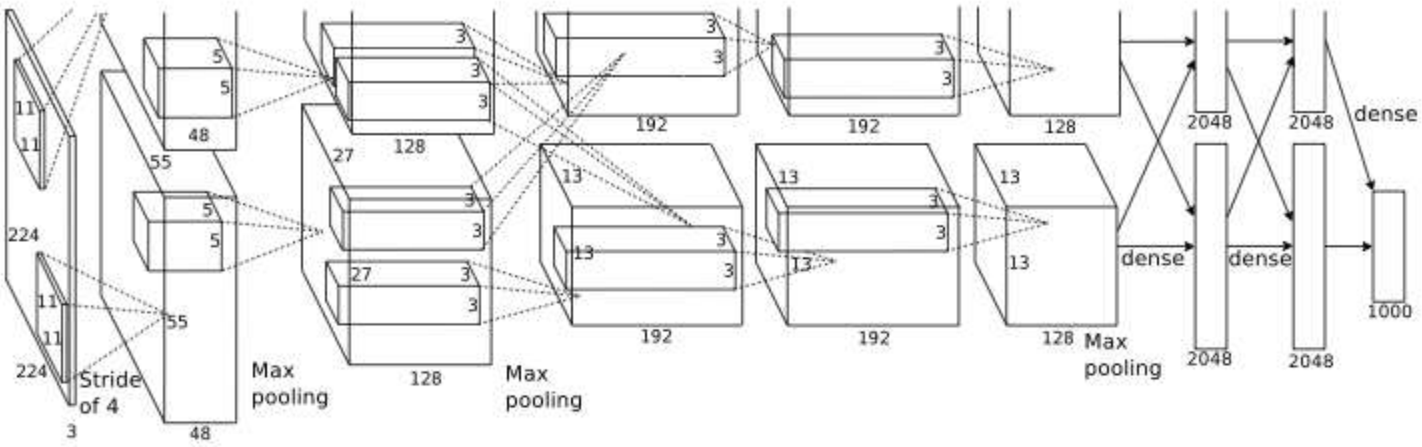


https://en.wikipedia.org/wiki/Artificial_neural_network

Convolutional Networks breakthrough



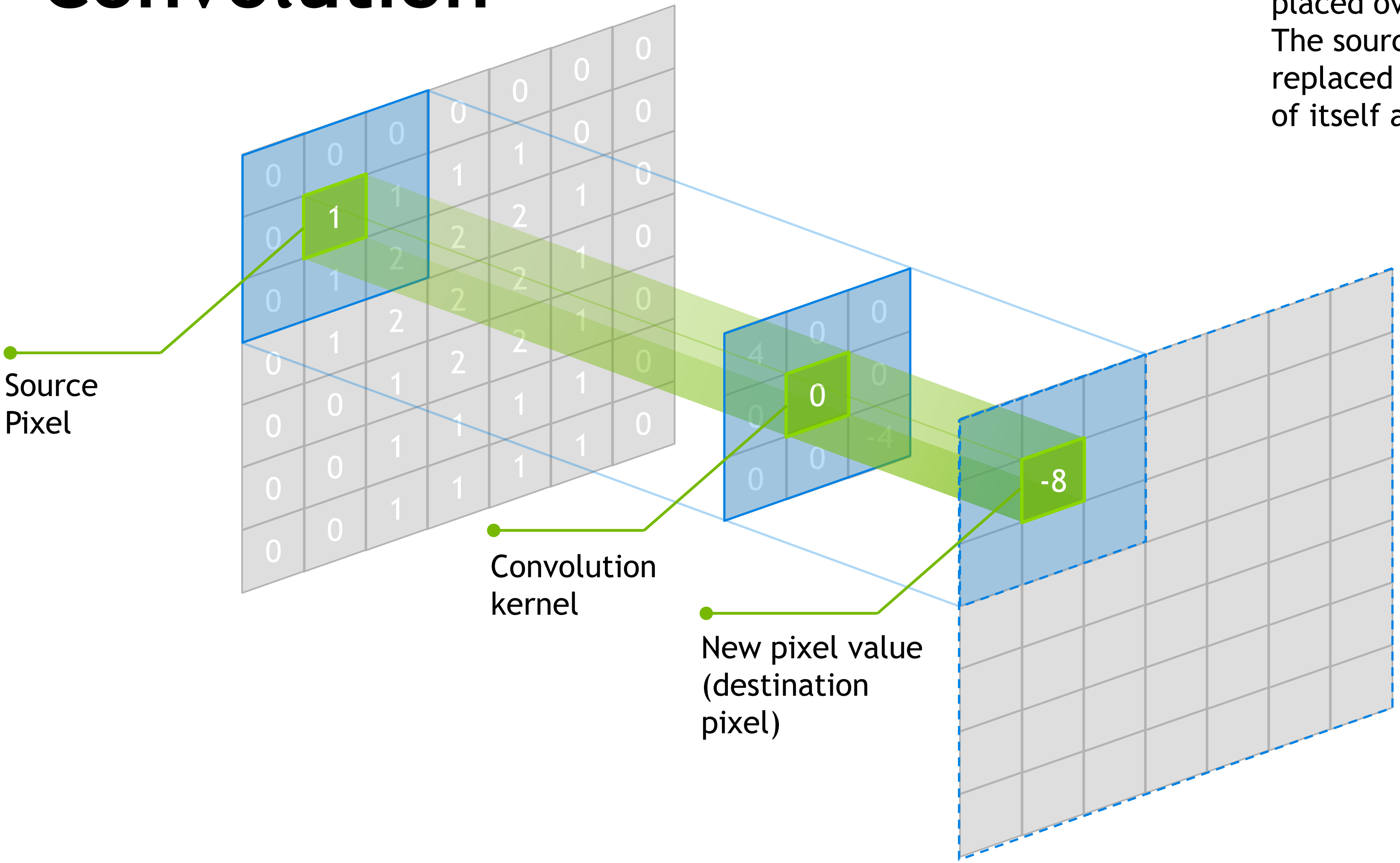
Y. LeCun et al. 1989-1998 : Handwritten digit reading



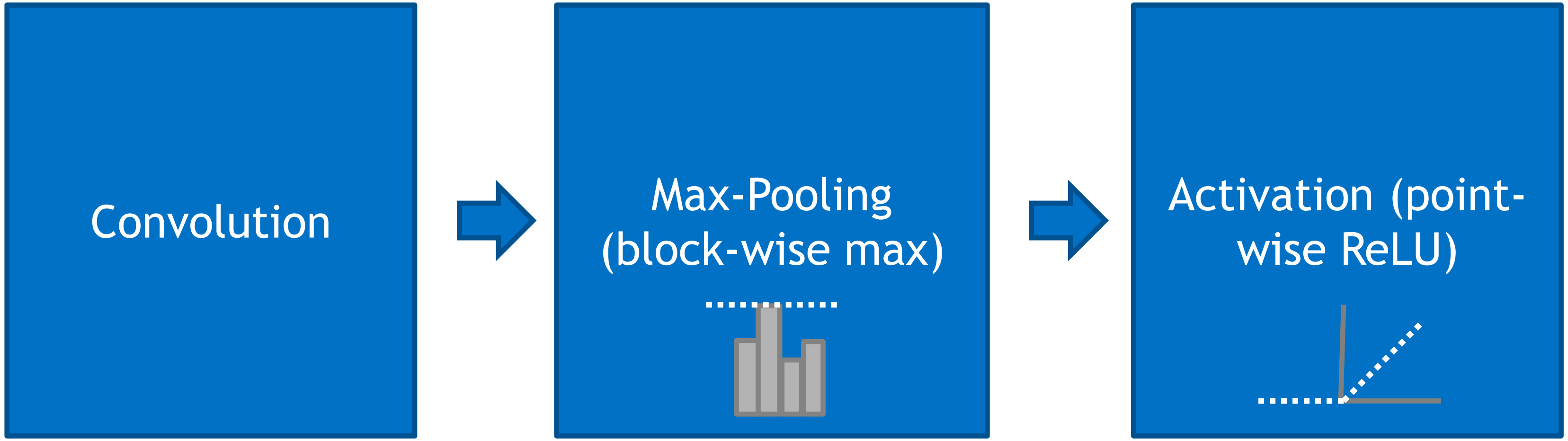
A. Krizhevsky, G. Hinton et al. 2012 : Imagenet classification winner

Convolution

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.



CNNs: Stacked Repeating Triplets



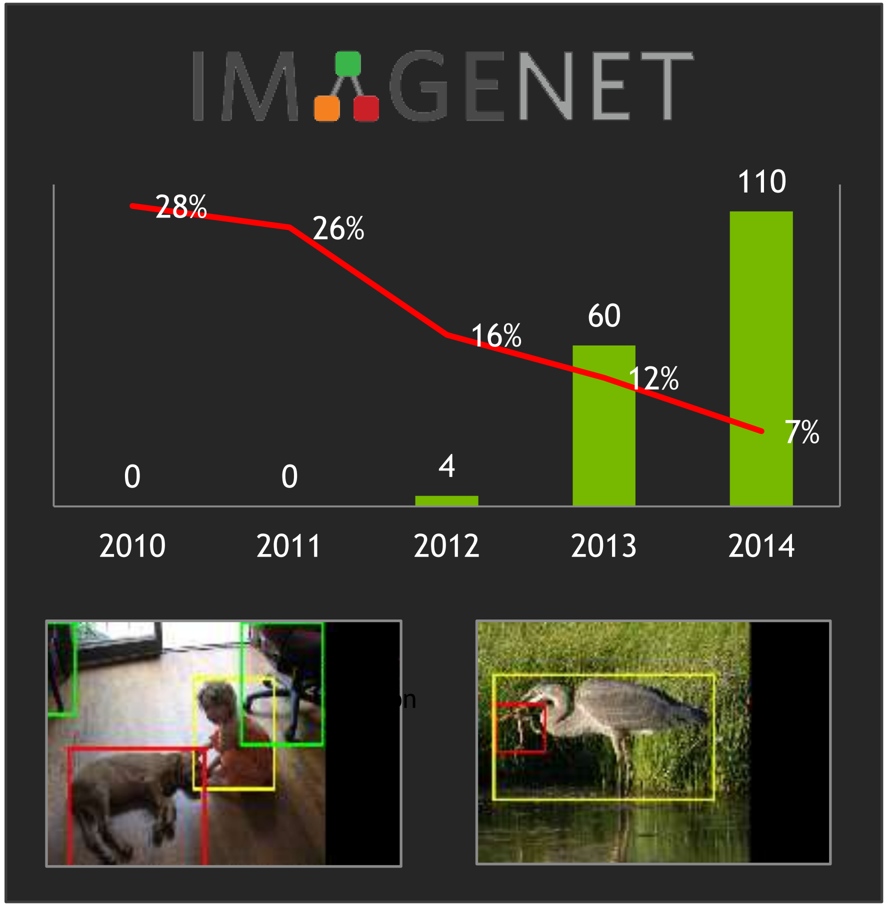
OverFeat Network, 2014

Layer	1	2	3	4	5	6	7	8	Output 9
Stage	conv + max	conv + max	conv	conv	conv	conv + max	full	full	full
# channels	96	256	512	512	1024	1024	4096	4096	1000
Filter size	7x7	7x7	3x3	3x3	3x3	3x3	-	-	-
Conv. stride	2x2	1x1	1x1	1x1	1x1	1x1	-	-	-
Pooling size	3x3	2x2	-	-	-	3x3	-	-	-
Pooling stride	3x3	2x2	-	-	-	3x3	-	-	-
Zero-Padding size	-	-	1x1x1x1	1x1x1x1	1x1x1x1	1x1x1x1	-	-	-
Spatial input size	221x221	36x36	15x15	15x15	15x15	15x15	5x5	1x1	1x1

Why are GPUs good for deep learning?

	Neural Networks	GPUs
Inherently Parallel	✓	✓
Matrix Operations	✓	✓
FLOPS	✓	✓

- GPUs deliver --
  - prediction accuracy
  - faster results
  - smaller footprint
  - lower power



Compute Platform

ACCELERATING INSIGHTS

"Now You Can Build Google's $1M Artificial Brain on the Cheap"

WIRED

GOOGLE DATACENTER

1,000 CPU Servers
2,000 CPUs • 16,000 cores

600 kWatts
$5,000,000

STANFORD AI LAB

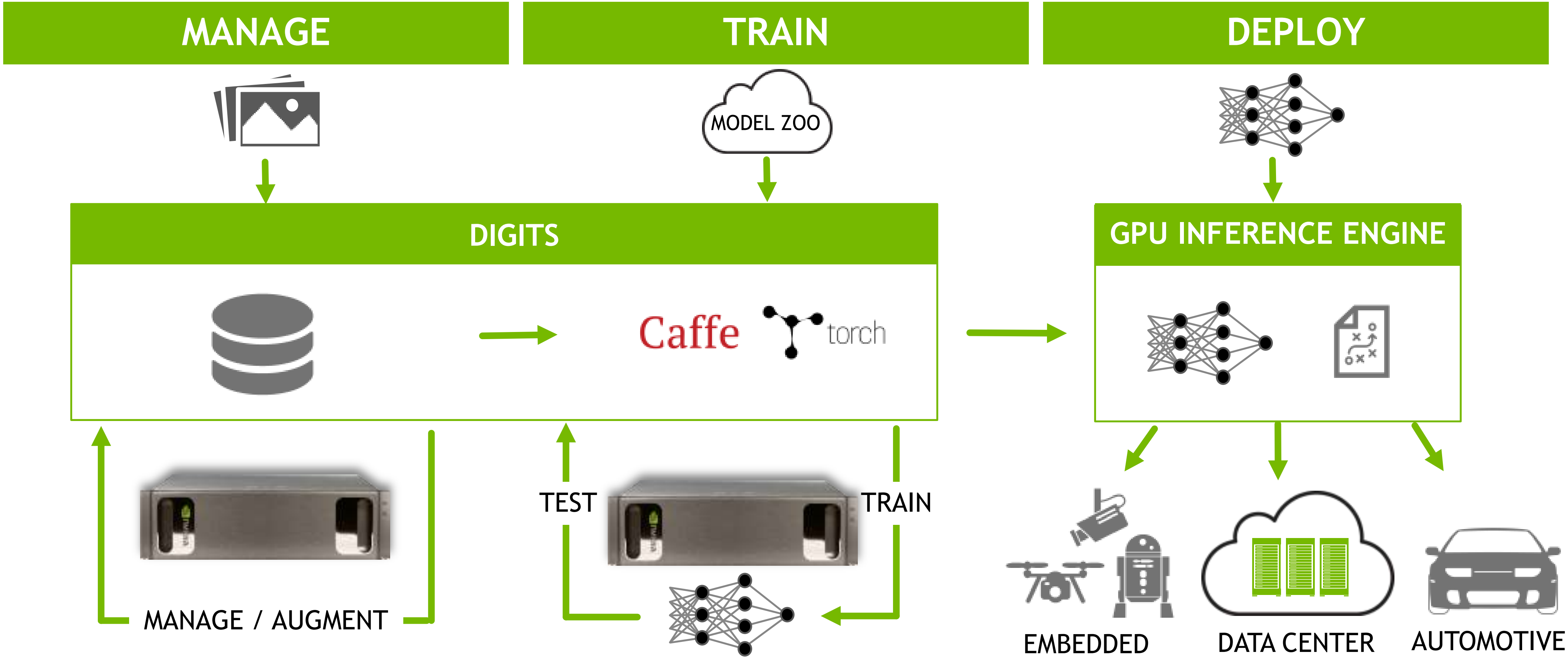
3 GPU-Accelerated Servers
12 GPUs • 18,432 cores

4 kWatts
$33,000

Deep learning with COTS HPC systems, A. Coates, B. Huval, T. Wang, D. Wu, A. Ng, B. Catanzaro ICML 2013

Deep Learning Workflow

A complete GPU-accelerated workflow



NVIDIA Deep Learning SDK

High Performance GPU-Acceleration for Deep Learning

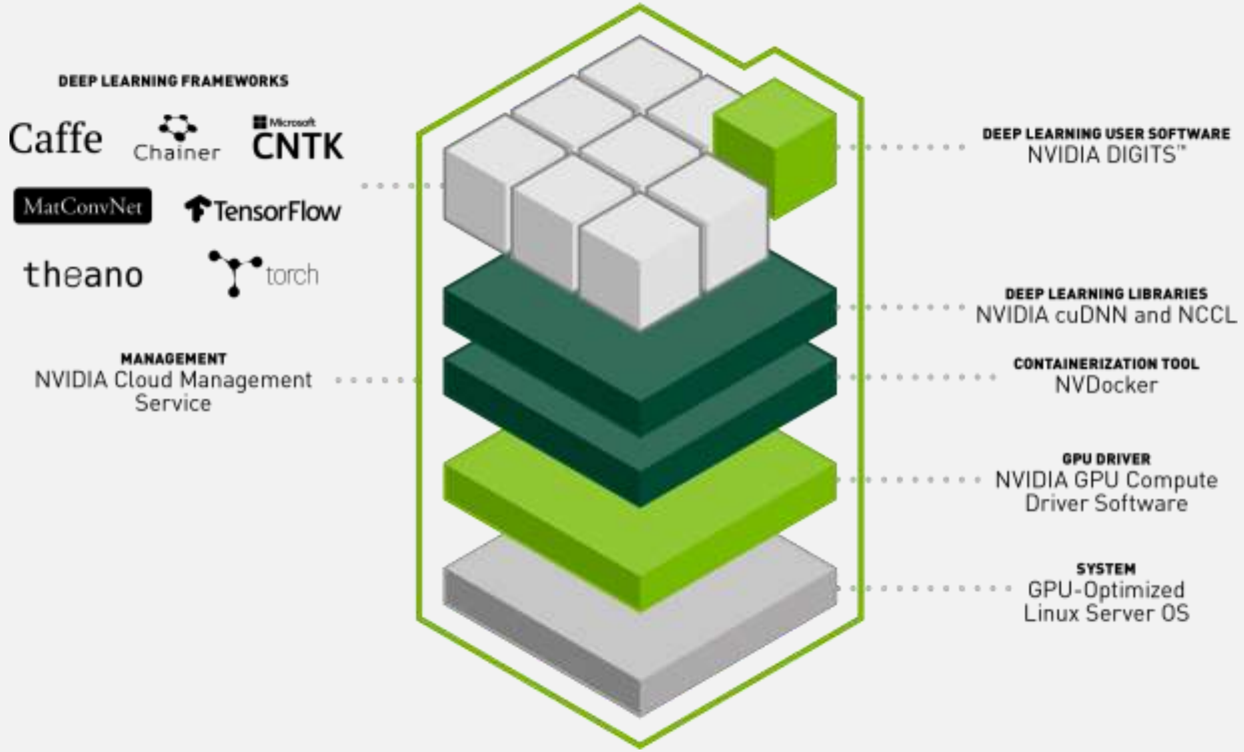
DGX-1

AI Supercomputer-in-a-Box

170 TFLOPS | 8x Tesla P100 16GB | NVLink Hybrid Cube Mesh

2x Xeon | 8 TB RAID 0 | Quad IB 100Gbps, Dual 10GbE | 3U — 3200W

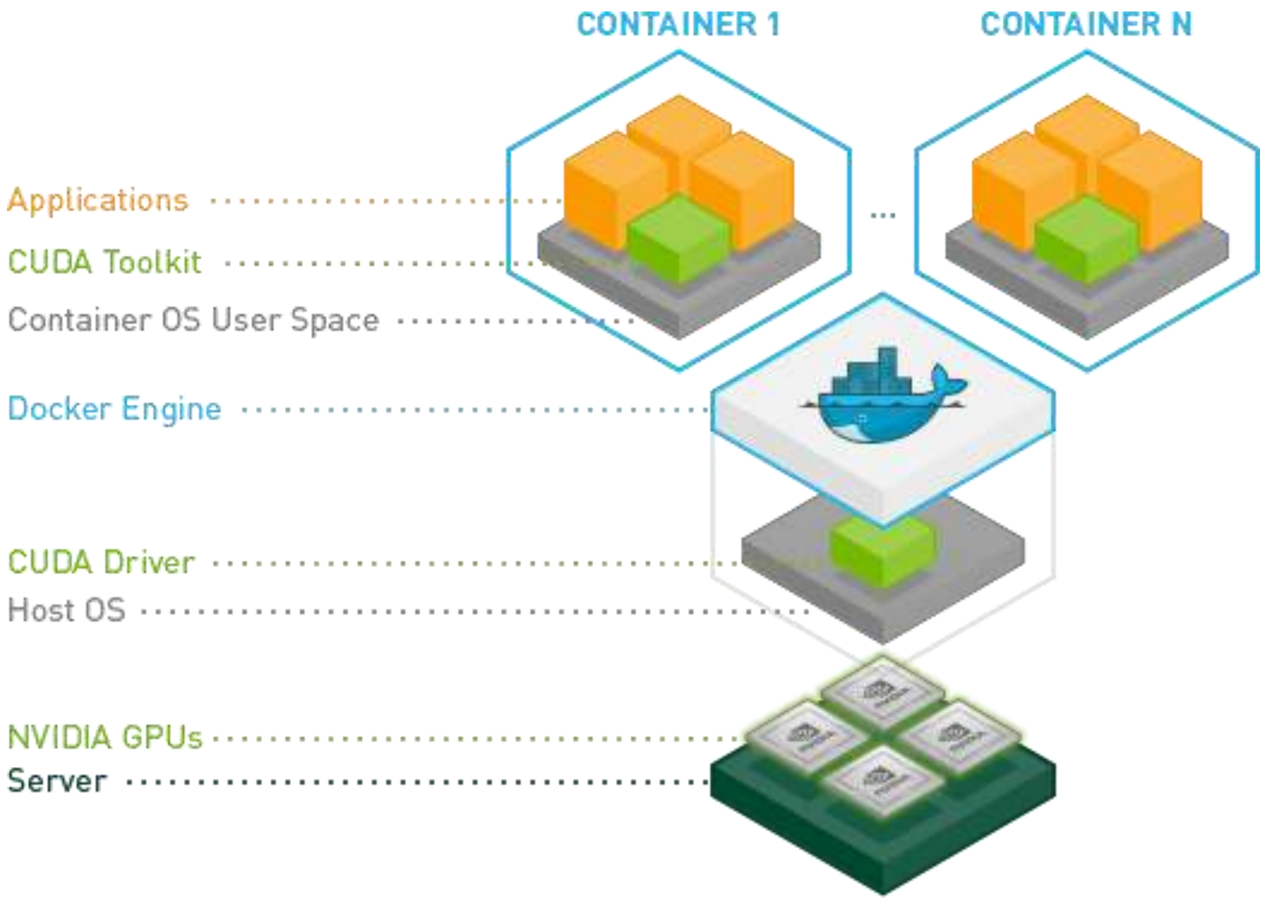
DGX-1 DL STACK



Performance optimized across the entire stack

Mixed framework environments — containerized

NVIDIA DOCKER Containers



NVIDIA cuDNN

Accelerating Deep Learning

High performance building blocks for deep learning frameworks

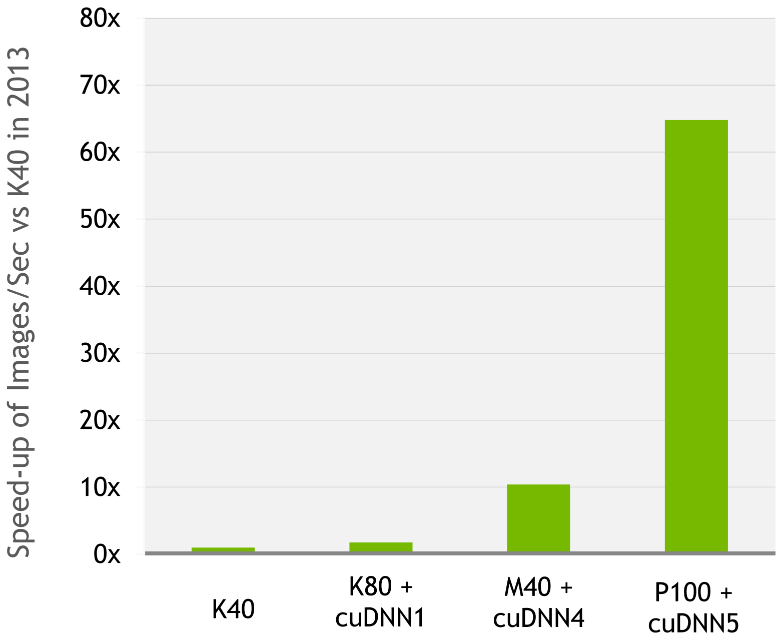
- Drop-in acceleration for widely used deep learning frameworks such as Caffe, CNTK, Tensorflow, Theano, Torch and others

- Accelerates industry vetted deep learning algorithms, such as convolutions, LSTM, fully connected, and pooling layers

- Fast deep learning training performance tuned for NVIDIA GPUs

developer.nvidia.com/cudnn

Deep Learning Training Performance
Caffe AlexNet



AlexNet training throughput on CPU: 1x E5-2680v3 12 Core 2.5GHz. 128GB System Memory, Ubuntu 14.04
M40 bar: 8x M40 GPUs in a node, P100: 8x P100 NVLink-enabled

> “NVIDIA has improved the speed of cuDNN with each release while extending the interface to more operations and devices at the same time.”

— Evan Shelhamer, Lead Caffe Developer, UC Berkeley

Caffe

An open framework for deep learning developed by the Berkeley Vision and Learning Center (BVLC)

caffe.berkeleyvision.org
http://github.com/BVLC/caffe

- Pure C++/CUDA architecture
- Command line, Python, MATLAB interfaces
- Fast, well-tested code
- Pre-processing and deployment tools, reference models and examples
- Image data management
- Seamless GPU acceleration
- Large community of contributors to the open-source project

Caffe features

Data pre-processing and management

Data ingest formats

LevelDB or LMDB database

In-memory (C++ and Python only)

HDF5

Image files

Pre-processing tools

LevelDB/LMDB creation from raw images

Training and validation set creation with shuffling

Mean-image generation

Data transformations

Image cropping, resizing, scaling and mirroring

Mean subtraction

Caffe features

Deep Learning model definition

Protobuf model format

- Strongly typed format
- Human readable
- Auto-generates and checks Caffe code
- Developed by Google
- Used to define network architecture and training parameters
- No coding required!

```
name: "conv1"
type: "Convolution"
bottom: "data"
top: "conv1"
convolution_param {
        num_output: 20
        kernel_size: 5
        stride: 1
        weight_filler {
                type: "xavier"
        }
}
```

Caffe features

Deep Learning model definition

Loss functions:

Classification

Softmax

Hinge loss

Linear regression

Euclidean loss

Attributes/multi-classification

Sigmoid cross entropy loss

and more…

Available layer types:

Convolution

Pooling

Normalization

Activation functions:

ReLU

Sigmoid

Tanh

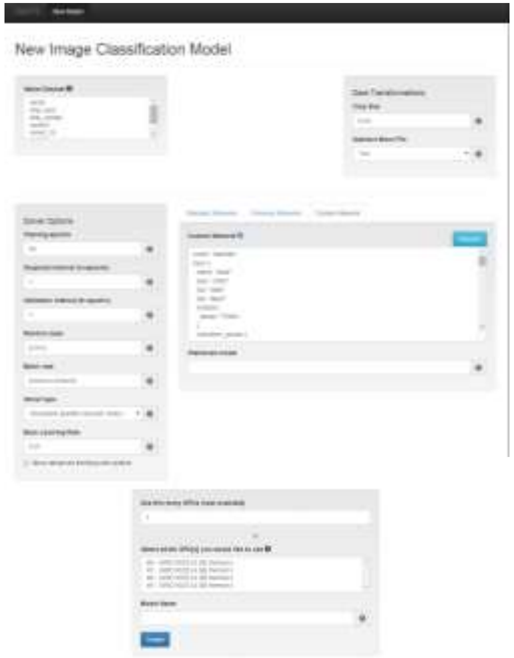
and more…

DIGITS

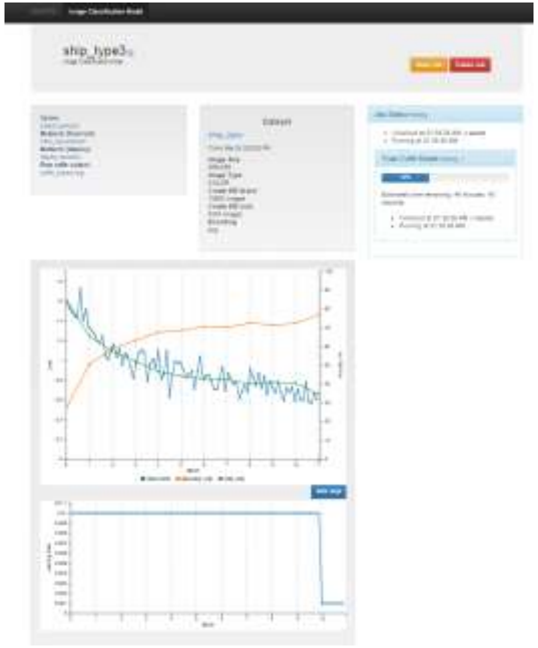
Interactive Deep Learning GPU Training System

Process Data

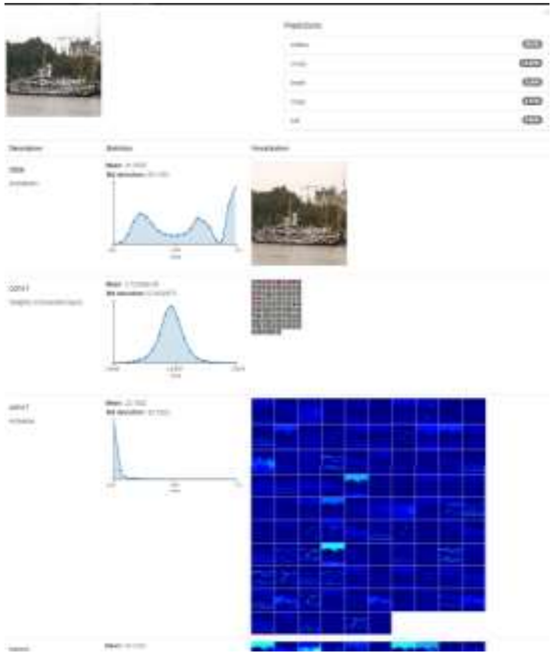
Configure DNN



Monitor Progress

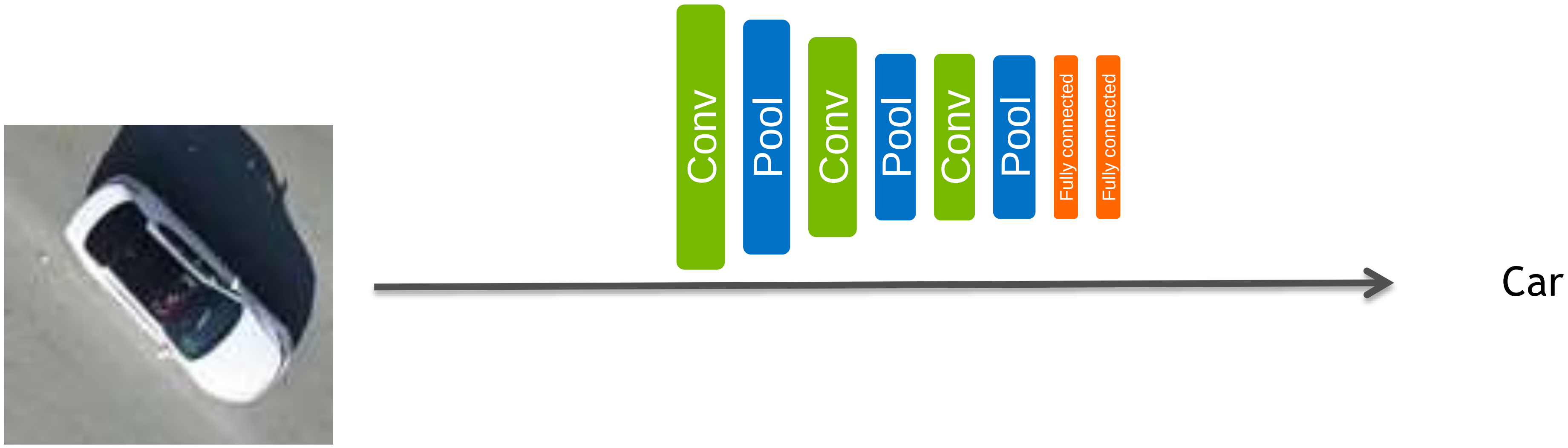


Visualization



DL examples

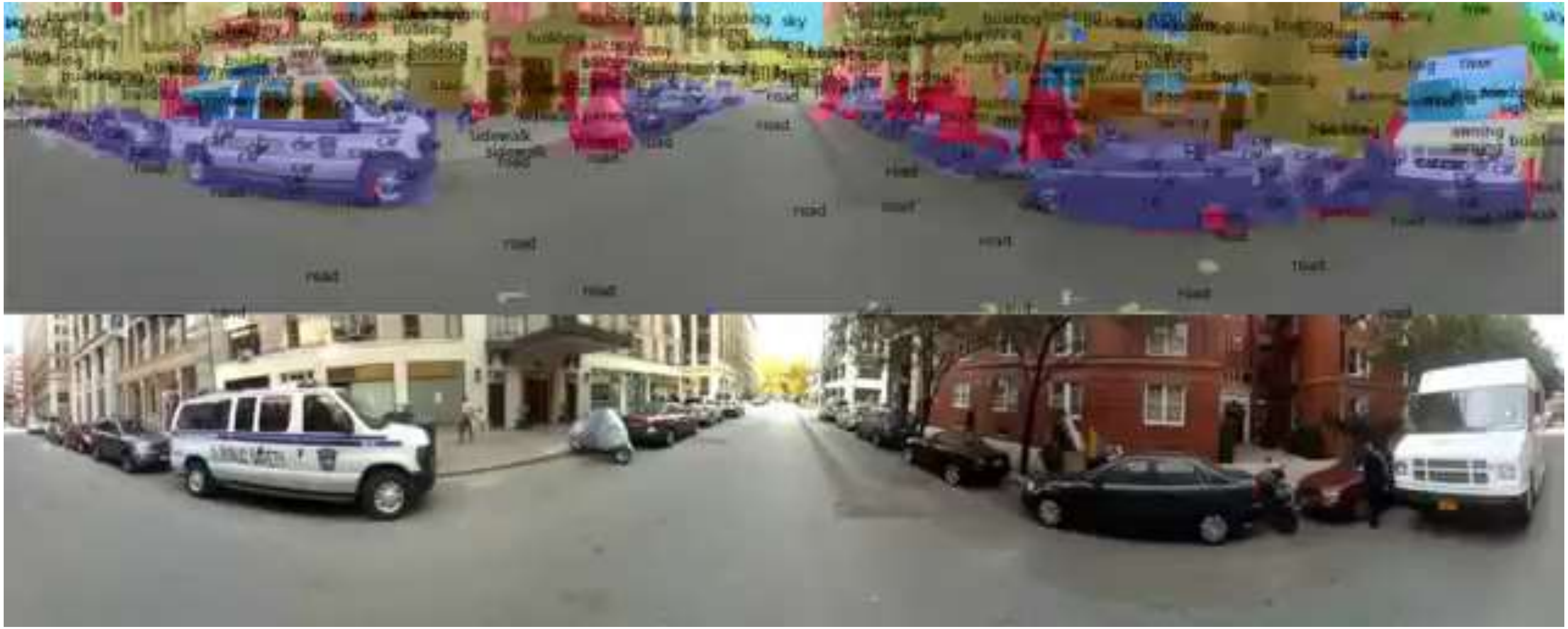
Object classification



From object classification to object detection

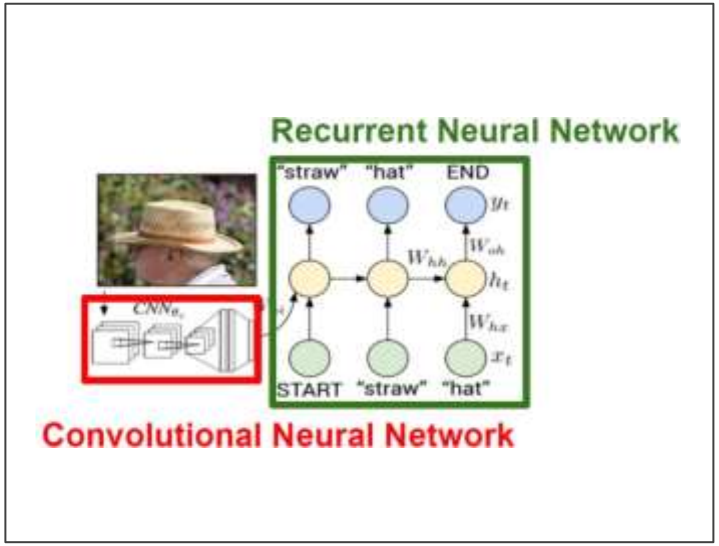
More complex examples

Segmentation



Clement Farabet, Camille Couprie, Laurent Najman and Yann LeCun: Learning Hierarchical Features for Scene Labeling, IEEE Transactions on Pattern Analysis and Machine Intelligence, August, 2013
https://www.youtube.com/watch?v=KkNhdlNs13U

Captioning



"Automated Image Captioning with ConvNets and Recurrent Nets"- Andrej Karpathy, Fei-Fei Li

Text generation

RNNs

VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Generation

GANs

NVIDIA @nvidia · Apr 5

Left: 20k images fed into a neural network. Right: the computer paints it's own pictures w/simple guidelines #GTC16

https://code.facebook.com/posts/1587249151575490/a-path-to-unsupervised-learning-through-adversarial-networks/

Selecting your lab(s)

Four labs available today

On nvlabs.qwiklab.com

1. Getting Started with Deep Learning (DIGITS, simple classification)
2. Introduction to Deep Learning (python, caffe, DIGITS, classification)
3. Approaches to Object Detection using DIGITS (python, DIGITS, detection)
4. NVIDIA-Docker (cuda, MNIST, Tensorflow, in a container)

Connection Instructions

Navigate to http://nvlabs.qwiklab.com

Login or create a new account

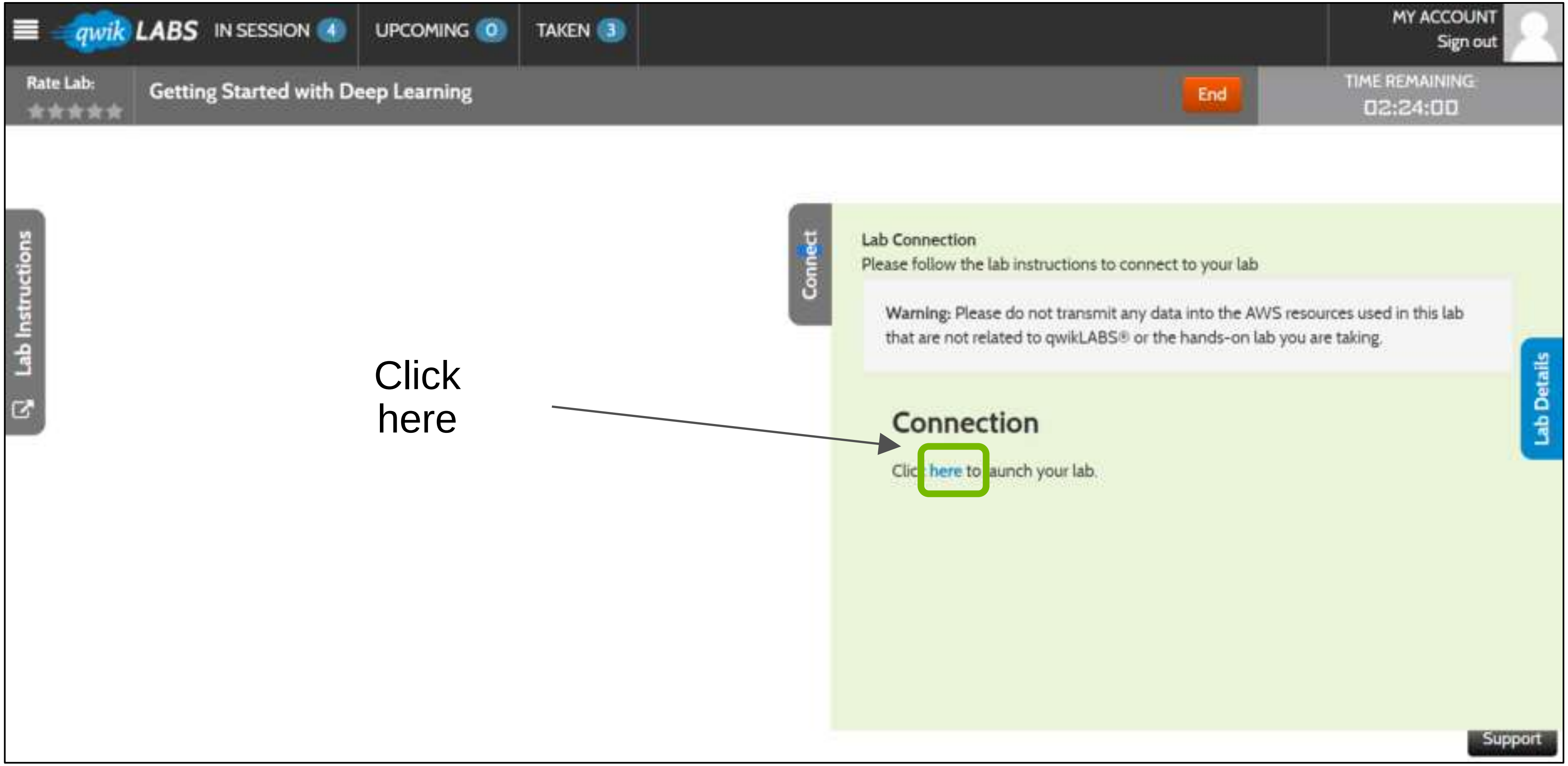
Select the "Instructor-Led Hands-on Labs" Class

Find the lab called "Getting Started with Deep Learning", select it, click Select, and finally click Start

Please ask Lab Assistants for help!

Connection Instructions

Click here



Lab1: Getting Started with DL

Goal

- Learn about the workflow of Deep Learning
- Train your own Convolutional Neural Network using Caffe and DIGITS to identify handwritten characters
- Try several different methods to improve initial results to understand the iterative nature of training and optimizing Deep Neural Networks

Handwritten Digits Recognition

HELLO WORLD of machine learning?

MNIST data set of handwritten digits from Yann Lecun’s website

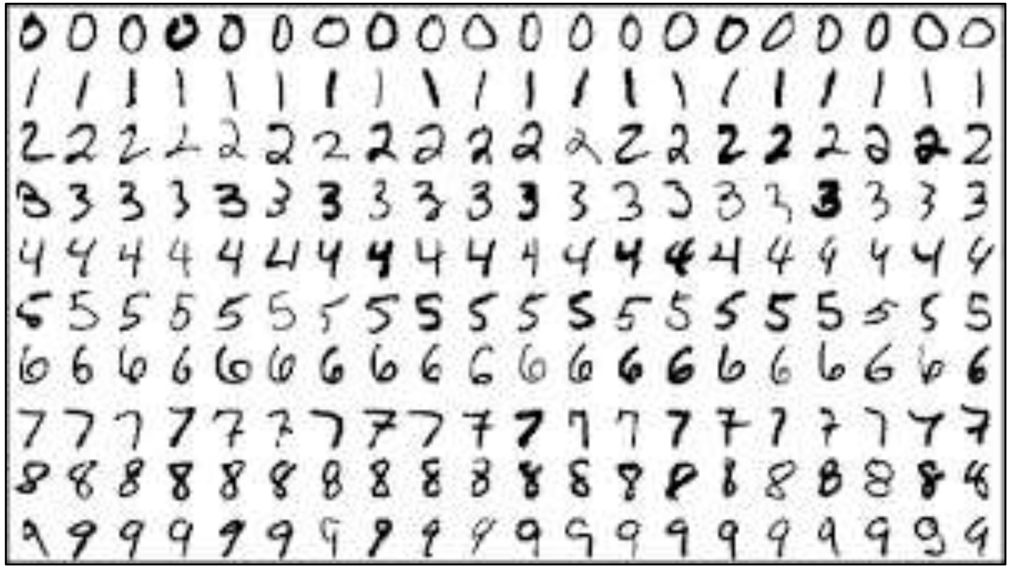
All images are 28x28 grayscale

Pixel values from 0 to 255

60k training examples, 10k test examples

Input vector of size 784

Output value is integer from 0-9



Outline

Steps

How to use DIGITS	10 min
Training with larger dataset	10 min
Data augmentation	10 min
Modifying network	10min

Dataset Setup

Login :

Use lower case letters.

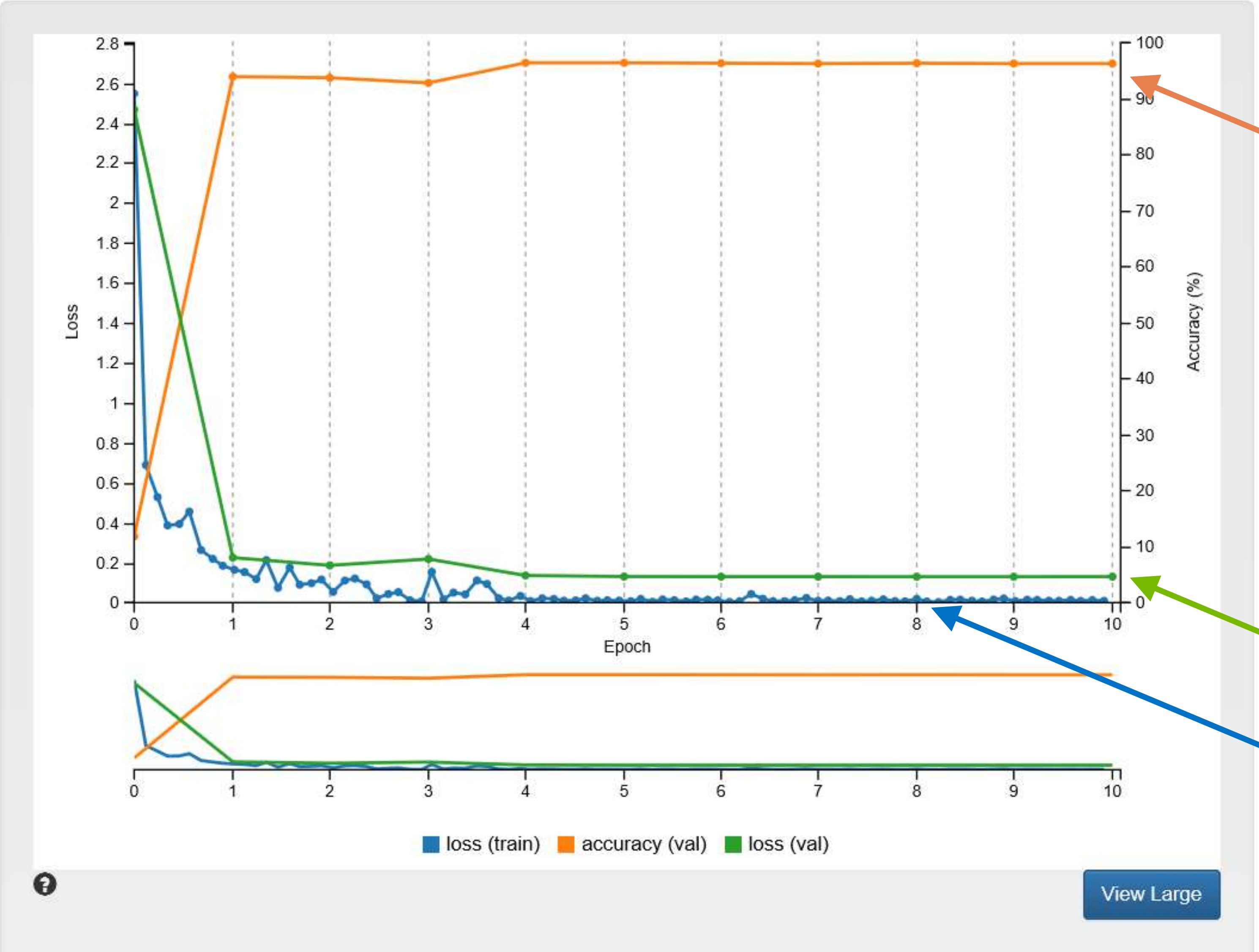
Dataset settings

- Image Type : Grayscale
- Image Size : 28 x 28
- Training Images: **/home/ubuntu/data/train_small**
- Select “**Separate test images folder**” checkbox
- Test Images : **/home/ubuntu/data/test_small**
- Dataset Name : MNIST Small

Model Setup

- Select the “**MNIST small**” dataset
- Set the number of “**Training Epochs**” to 10
- Set the framework to “**Caffe**”
- Set the model to “**LeNet**”
- Set the name of the model to “**MNIST small**”
- When training done, Classify One :

  /home/ubuntu/data/test_small/2/img_4415.png



Accuracy obtained from validation dataset

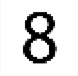
Loss function (Validation)

Loss function (Training)

First results

Small dataset (30 epochs)

- 96 % of accuracy achieved.
- Training is done within one minute.

	SMALL DATASET
	1 : 99.90 %
	2 : 69.03 %
	8 : 71.37 %
	8 : 85.07 %
	0 : 99.00 %
	8 : 99.69 %
	8 : 54.75 %

Full dataset

6x larger dataset

Dataset

Training Images : /home/ubuntu/data/train_full

Test Image : /home/ubuntu/data/test_full

Dataset Name : MNIST full

Model

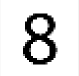
Clone “MNIST small”.

Give a new name “MNIST full” to push the create button.

Second results

Full dataset (30 epochs)

- 99 % of accuracy achieved.
- No improvements in recognizing real-world images.

	SMALL DATASET	FULL DATASET
	1 : 99.90 %	0 : 93.11 %
	2 : 69.03 %	2 : 87.23 %
	8 : 71.37 %	8 : 71.60 %
	8 : 85.07 %	8 : 79.72 %
	0 : 99.00 %	0 : 95.82 %
	8 : 99.69 %	8 : 100.0 %
	8 : 54.75 %	2 : 70.57 %

Data augmentation

Adding inverted images

Pixel(Inverted) = 255 – Pixel(original)

White letter with black background
-> Black letter with white background.

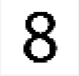
Training Images :
/home/ubuntu/data/train_invert

Test Image :
/home/ubuntu/data/test_invert

Dataset Name : MNIST invert

Data augmentation

Adding inverted images (30 epochs)

	SMALL DATASET	FULL DATASET	+INVERTED
	1 : 99.90 %	0 : 93.11 %	1 : 90.84 %
	2 : 69.03 %	2 : 87.23 %	2 : 89.44 %
	8 : 71.37 %	8 : 71.60 %	3 : 100.0 %
	8 : 85.07 %	8 : 79.72 %	4 : 100.0 %
	0 : 99.00 %	0 : 95.82 %	7 : 82.84 %
	8 : 99.69 %	8 : 100.0 %	8 : 100.0 %
	8 : 54.75 %	2 : 70.57 %	2 : 96.27 %

Modifying the network

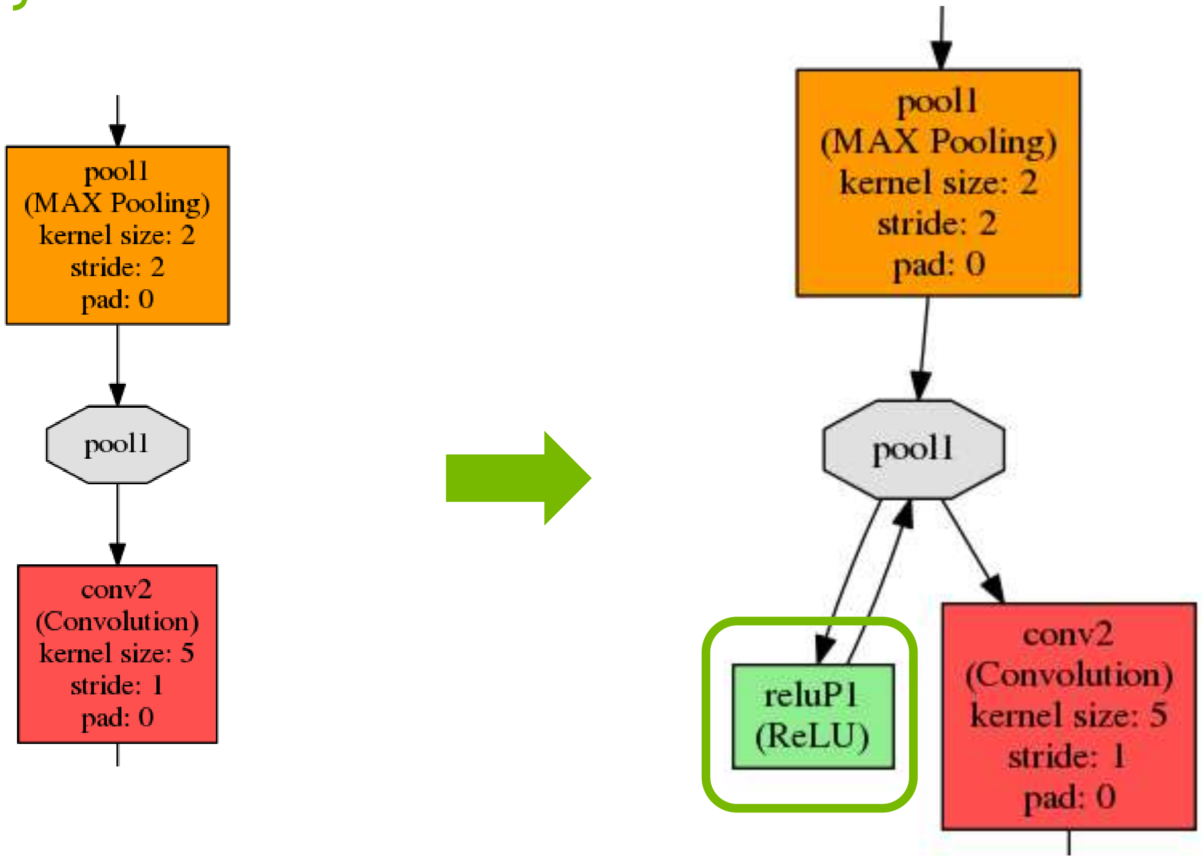
Adding filters and ReLU layer

```
layer {
        name: "pool1“
        type: "Pooling“
        …
}

layer {
        name: "reluP1"
        type: "ReLU"
        bottom: "pool1"
        top: "pool1"
}

layer {
        name: "reluP1“
```

```
layer {
  name: "conv1"
  type: "Convolution"
    ...
    convolution_param {
    num_output: 75
    ...
layer {
    name: "conv2"
    type: "Convolution"
    ...
    convolution_param {
    num_output: 100
    ...
```

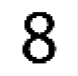
Modifying the network

Adding ReLU Layer



Modified network

Adding filters and ReLU layer (30 epochs)

	SMALL DATASET	FULL DATASET	+INVERTED	ADDING LAYER
	1 : 99.90 %	0 : 93.11 %	1 : 90.84 %	1 : 59.18 %
	2 : 69.03 %	2 : 87.23 %	2 : 89.44 %	2 : 93.39 %
	8 : 71.37 %	8 : 71.60 %	3 : 100.0 %	3 : 100.0 %
	8 : 85.07 %	8 : 79.72 %	4 : 100.0 %	4 : 100.0 %
	0 : 99.00 %	0 : 95.82 %	7 : 82.84 %	2 : 62.52 %
	8 : 99.69 %	8 : 100.0 %	8 : 100.0 %	8 : 100.0 %
	8 : 54.75 %	2 : 70.57 %	2 : 96.27 %	8 : 70.83 %

Links

GETTING STARTED WITH DEEP LEARNING

http://developer.nvidia.com/deep-learning

NVIDIA