lrz Leibniz-Rechenzentrum
der Bayerischen Akademie der Wissenschaften

DEEP LEARNING INSTITUTE

Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften

# DEEP LEARNING FUNDAMENTALS FOR COMPUTER VISION

PD Dr. Juan J. Durillo – Big Data and AI Team, LRZ

# DEEP LEARNING INSTITUTE

## DLI Mission

Training you to solve the world's most challenging problems.

- Developers, data scientists and engineers
- Self-driving cars, healthcare and robotics
- Training, optimizing, and deploying deep neural networks

# This Course: Deep Learning Fundamentals

## Training

- Train first network
  - Introduce Image Classification
- Necessary ingredients for successful training

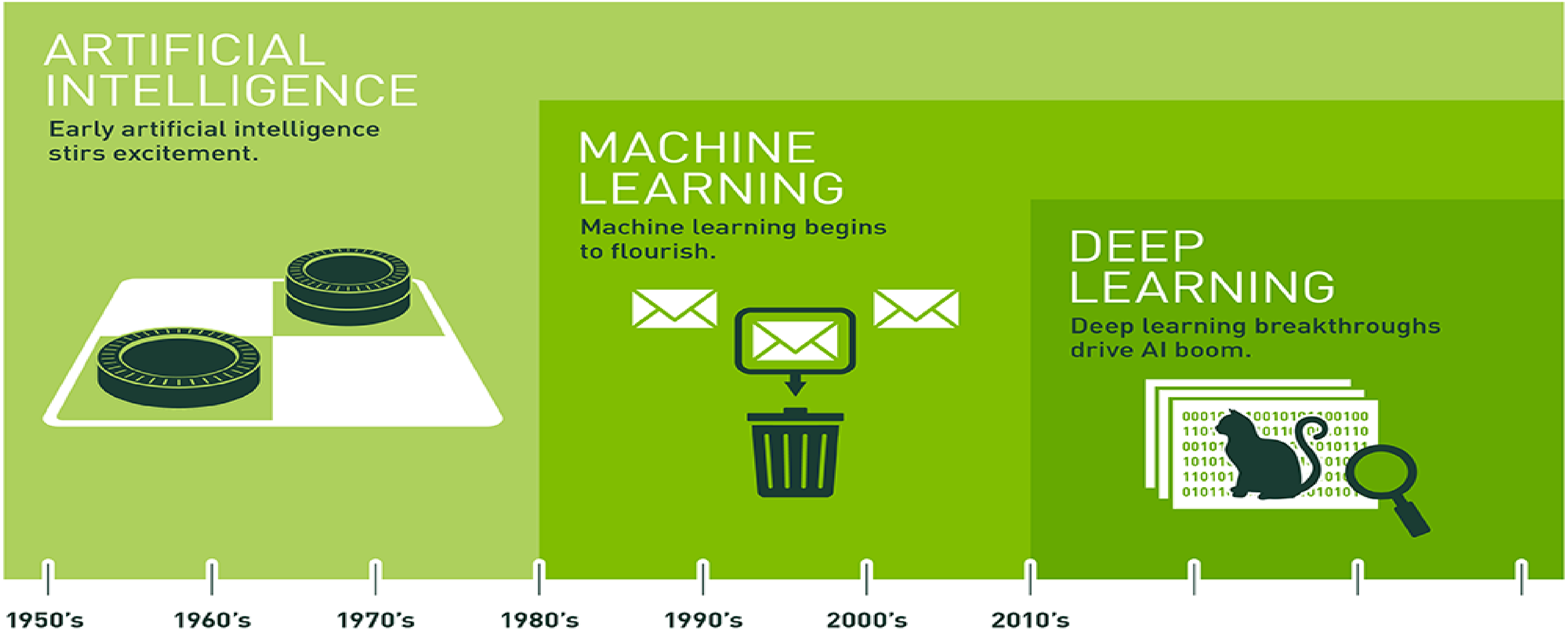
## Deployment

- Role of Neural Networks in Applications
- Building around networks

## Performance

- Improving accuracy
- Improving capability
- Solving novel problems
  - Changing layers of network
  - Beyond Image Classification

# ACCOMPLISHING COMPLEX GOALS




NVIDIA DEEP LEARNING INSTITUTE

# DEEP LEARNING IN PRODUCTION

Speech Recognition

Recommender Systems

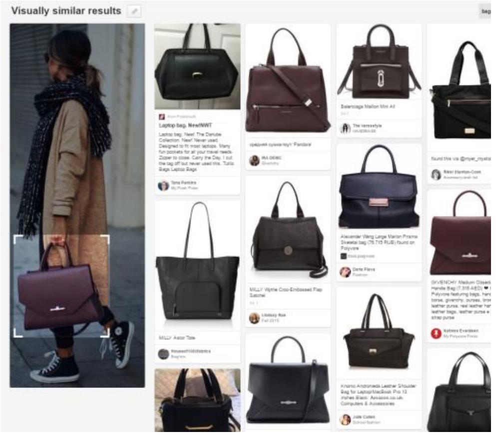
Autonomous Driving

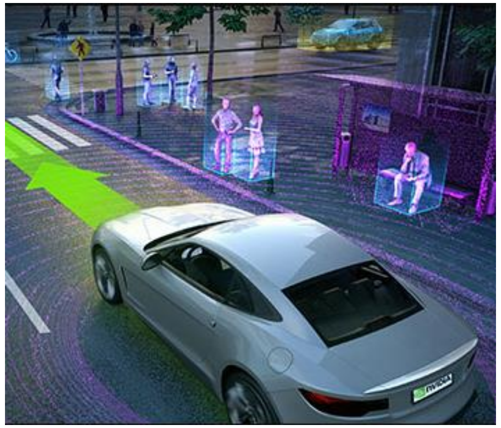
Real-time Object Recognition

Robotics

Real-time Language Translation

Many More…

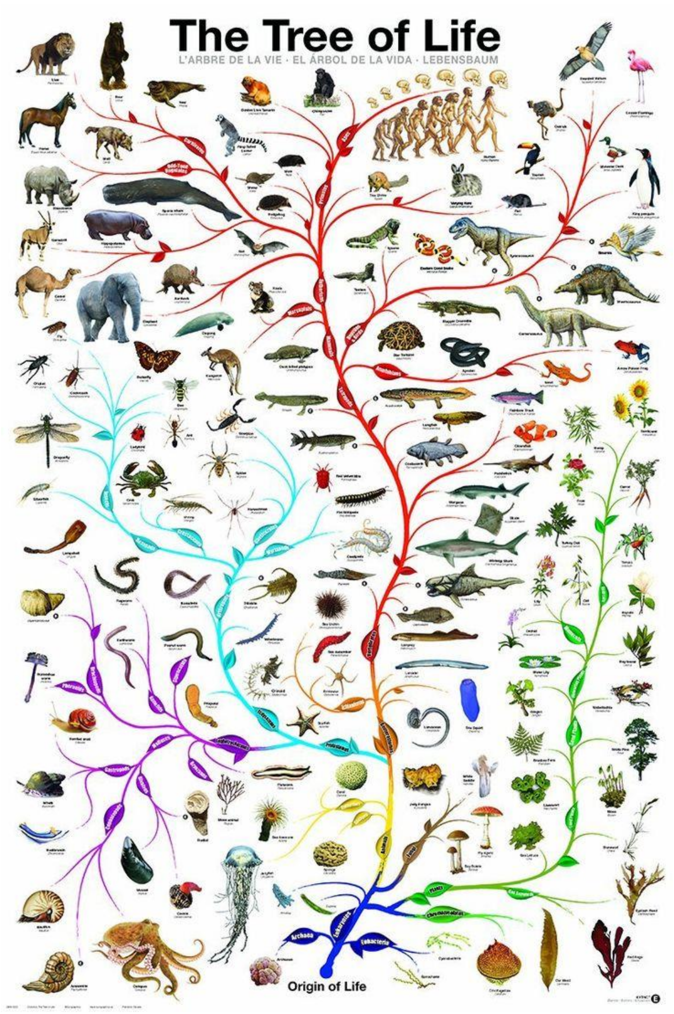




# Achieving Complex Goals

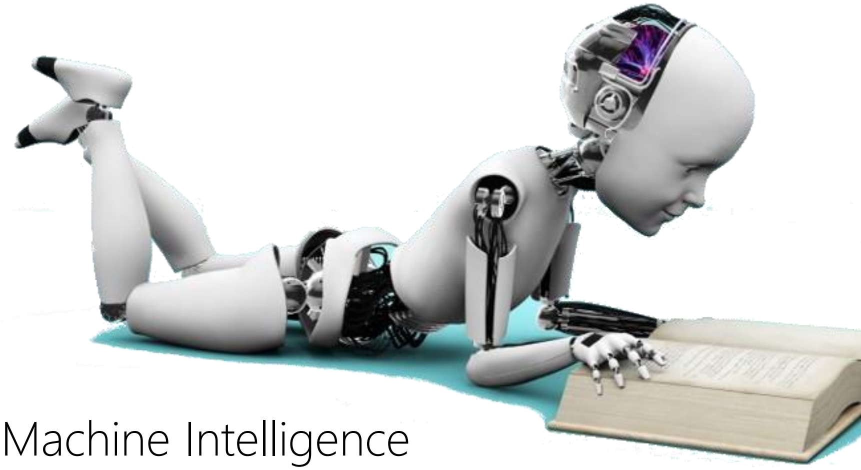
4th revolution in knowledge acquisition

1st - Evolution



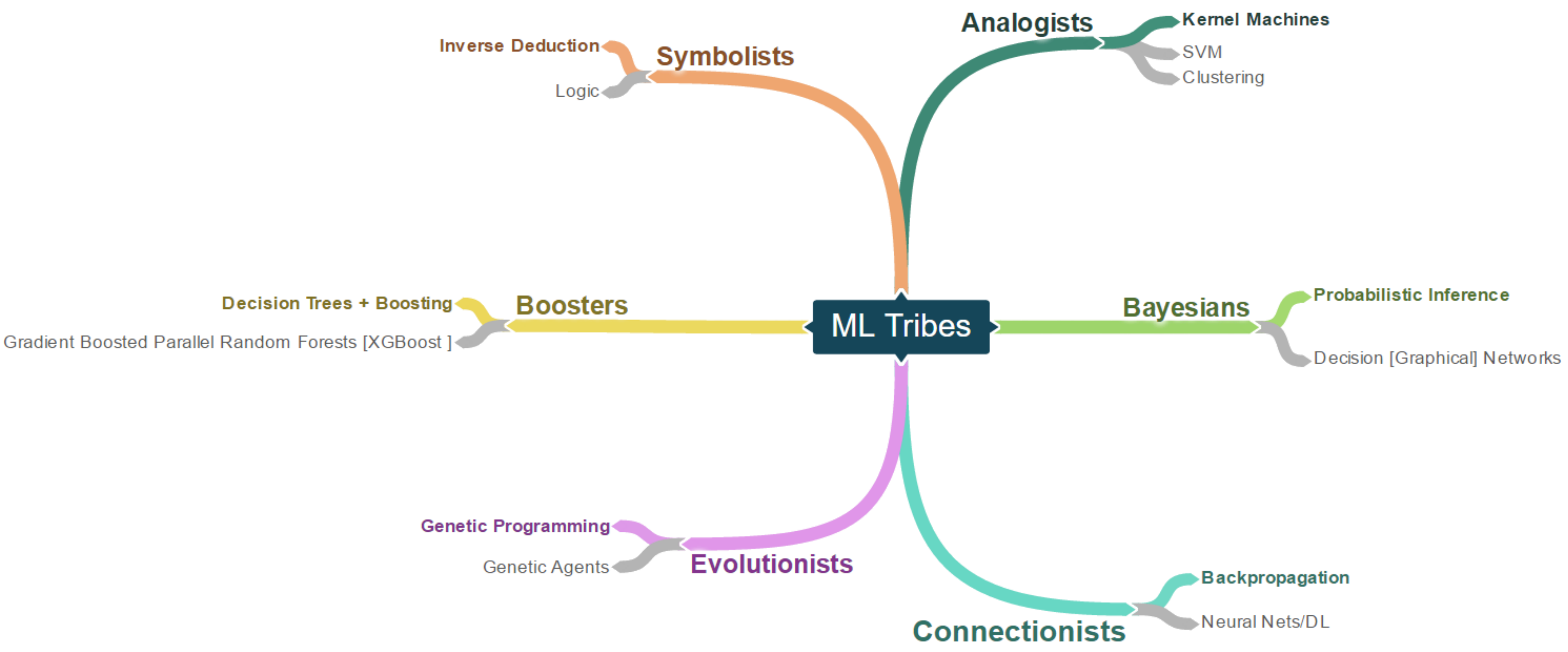

2nd - Experience/Brain

3rd - Culture

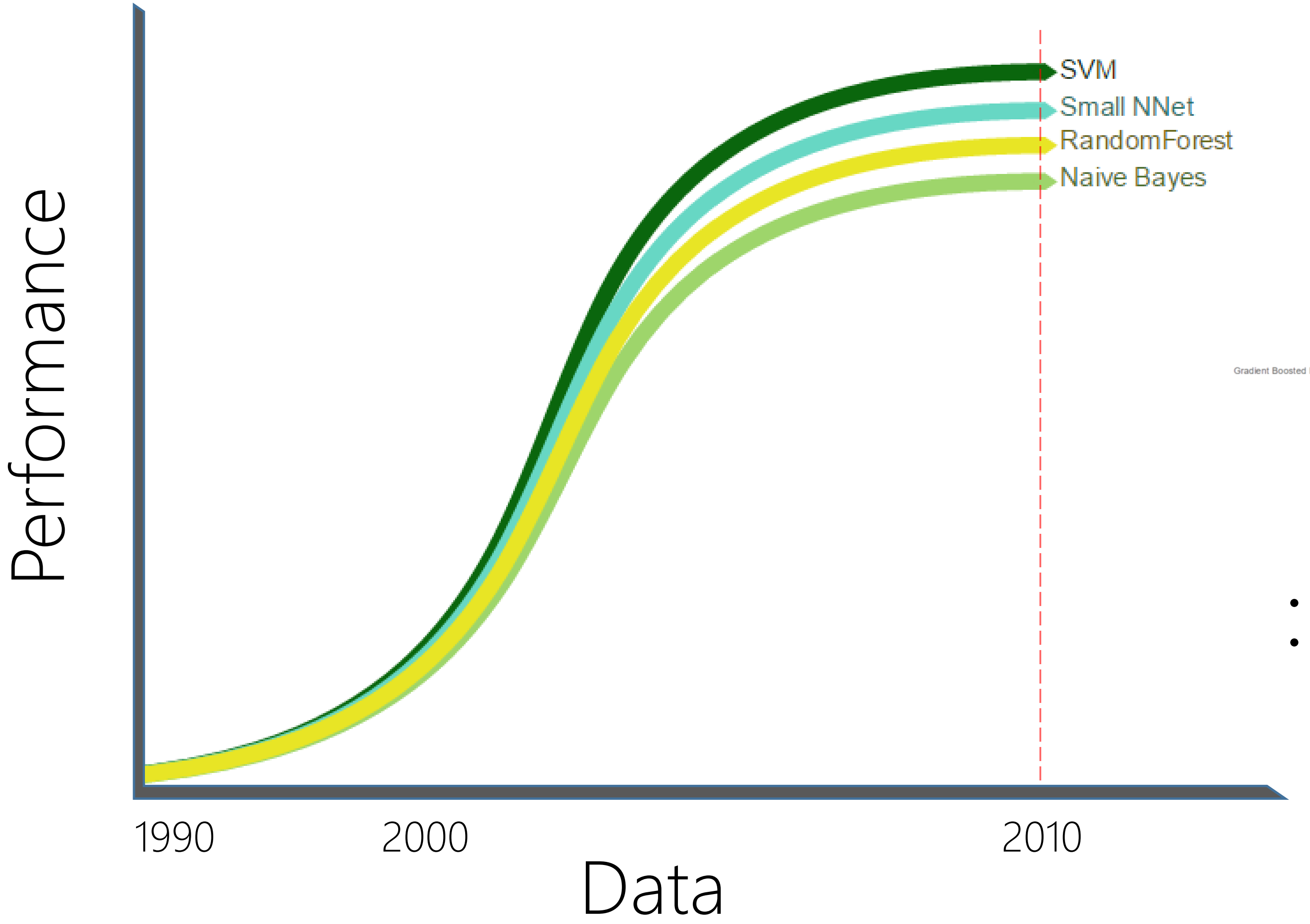
4th – Machine Intelligence

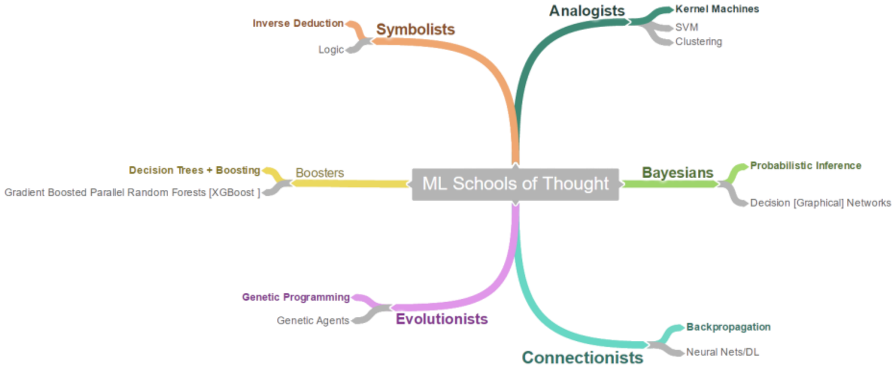
# ML Tribes




| Tribe | Origins | Master Algorithm |
|---|---|---|
| Symbolists | Logic, philosophy | Inverse deduction |
| Connectionists | Neuroscience | Backpropagation |
| Evolutionaries | Evolutionary biology | Genetic programming |
| Bayesians | Statistics | Probabilistic inference |
| Analogizers | Psychology | Kernel machines |

# Trend #1 [ Scale ]

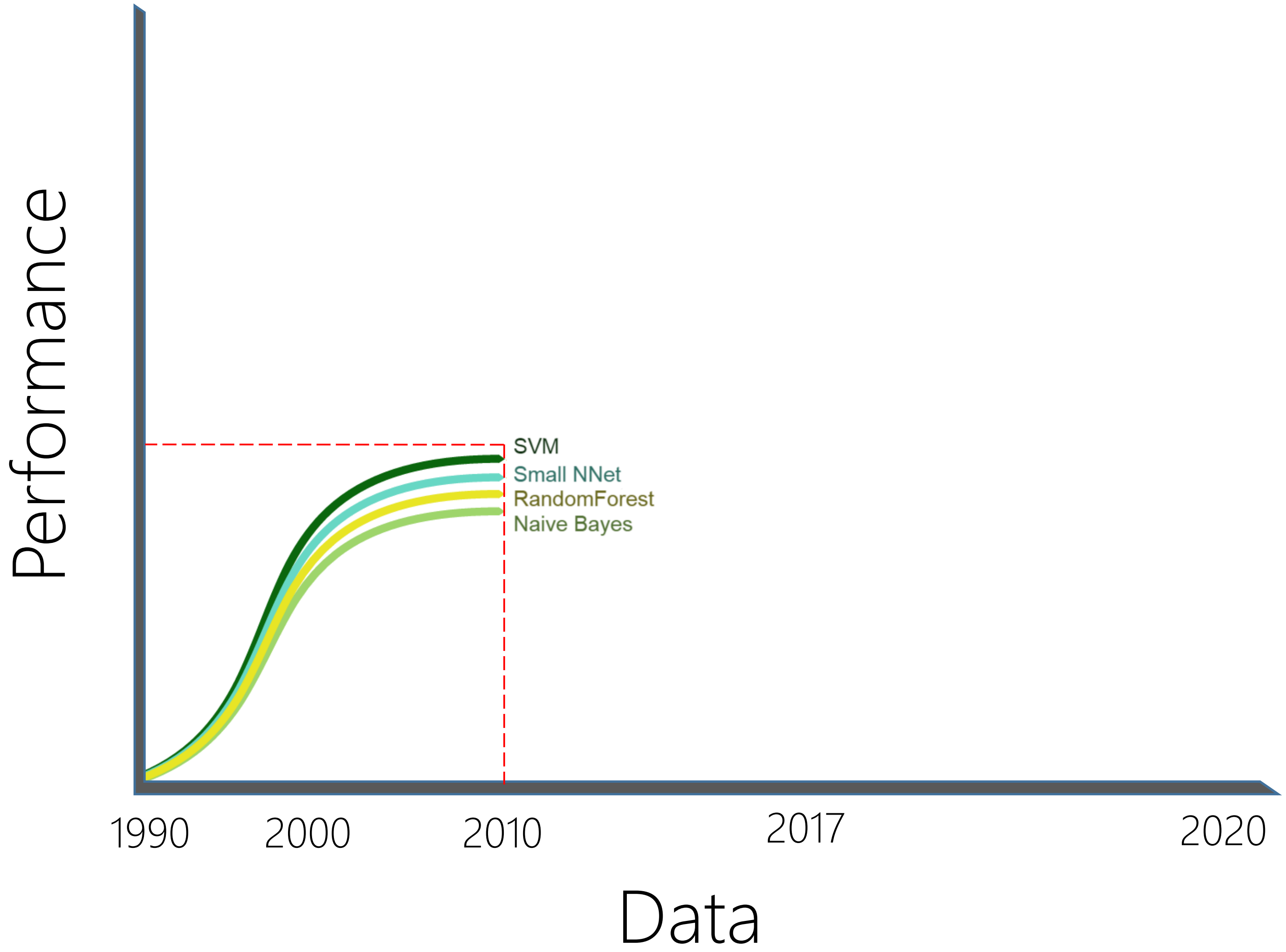


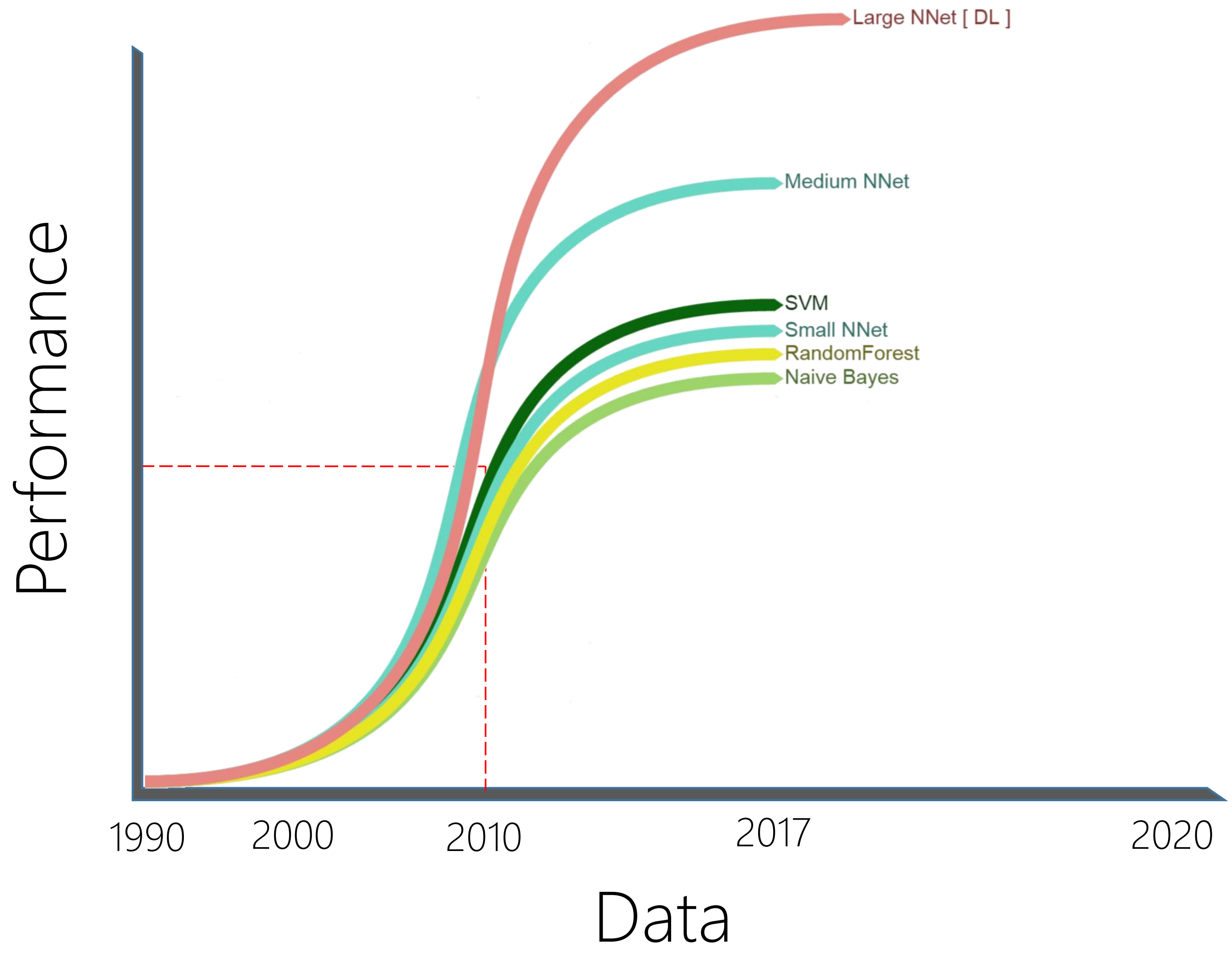

- Exponential Growth of Data
- Somewhat Arbitrary Ordering of Model Performance

  [ based on hand engineering effort ]

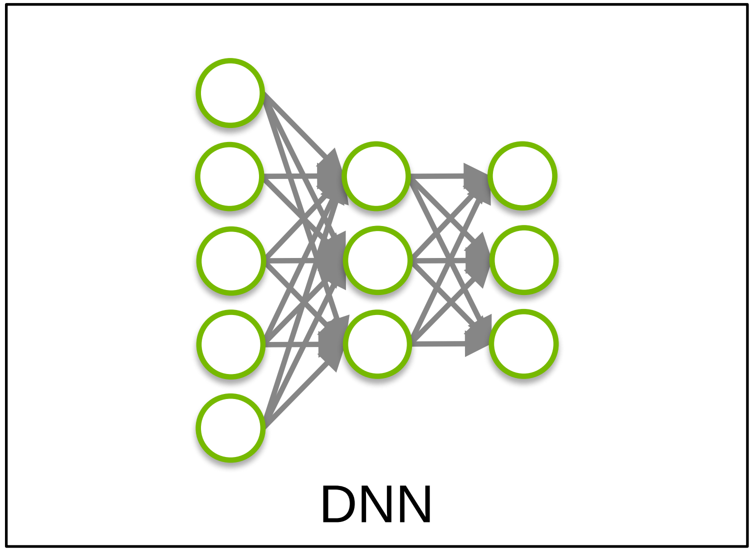
# Trend #1 [ Scale ]



# Trend #1 [ Scale ]



# THE BIG BANG IN MACHINE LEARNING






WIRED

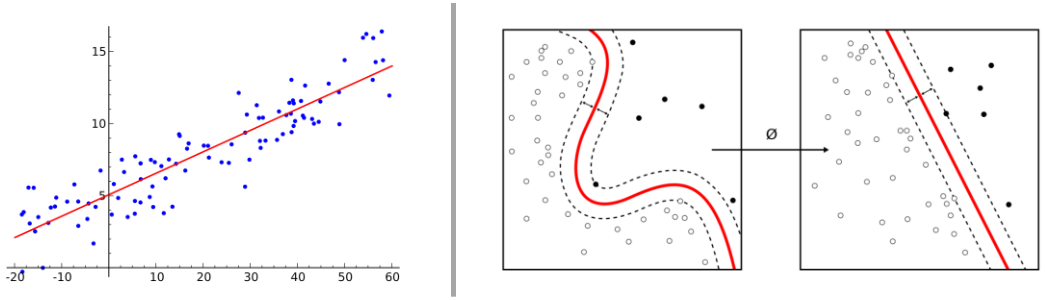
# Deep Neural Networks

# Difference in Workflow

Classic Machine Learning [ 1990 : now ]

Input → Hand Designed **Features** → Model / Mapping → Output

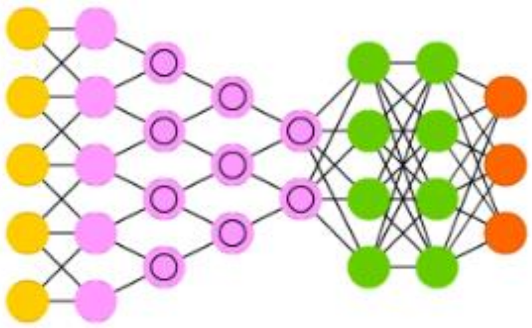
Examples [ Regression and SVMs ]




Deep/End-to-End Learning [ 2012 : now ]

Input → Simple Features → Complex Features → Model/ Mapping → Output

Example [ Conv Net ]

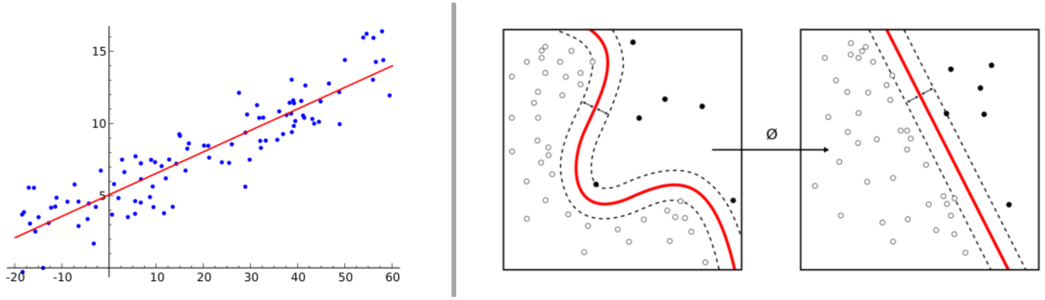


# Difference in Workflow

Classic Machine Learning [ 1990 : now ]

Input → Hand Designed Features → Model / Mapping → Output

Examples [ Regression and SVMs ]




**Challenge: How would you give instructions to differentiate the handwritten digit on the right from other digits to:**

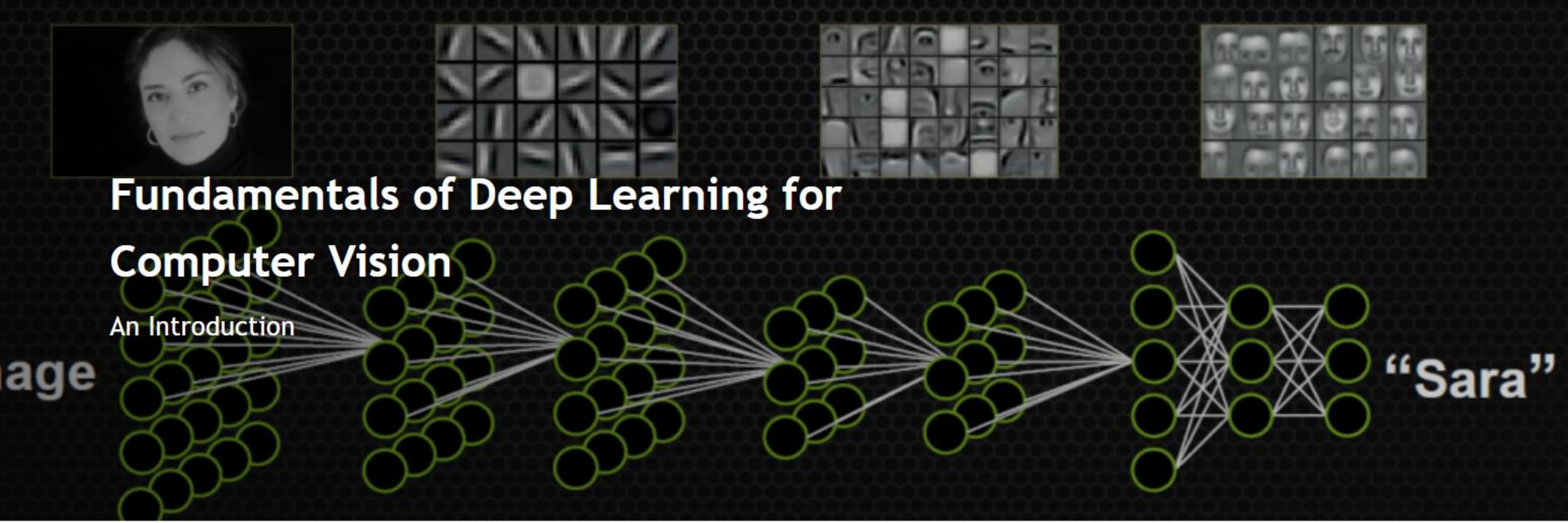
A two-year-old child learning numbers OR
A computer using any programming language (including pseudocode)

Answer in discussion section– 3 minutes

# Work through Introduction Section

- courses.nvidia.com/dli-event
- Browser Recommendation: Chrome

- Event code: [will be given during the lecture]
- Create an Account
- Work through the Introduction Section and ‘Start’ launching your first GPU task




Start Date: Mar 25, 2018

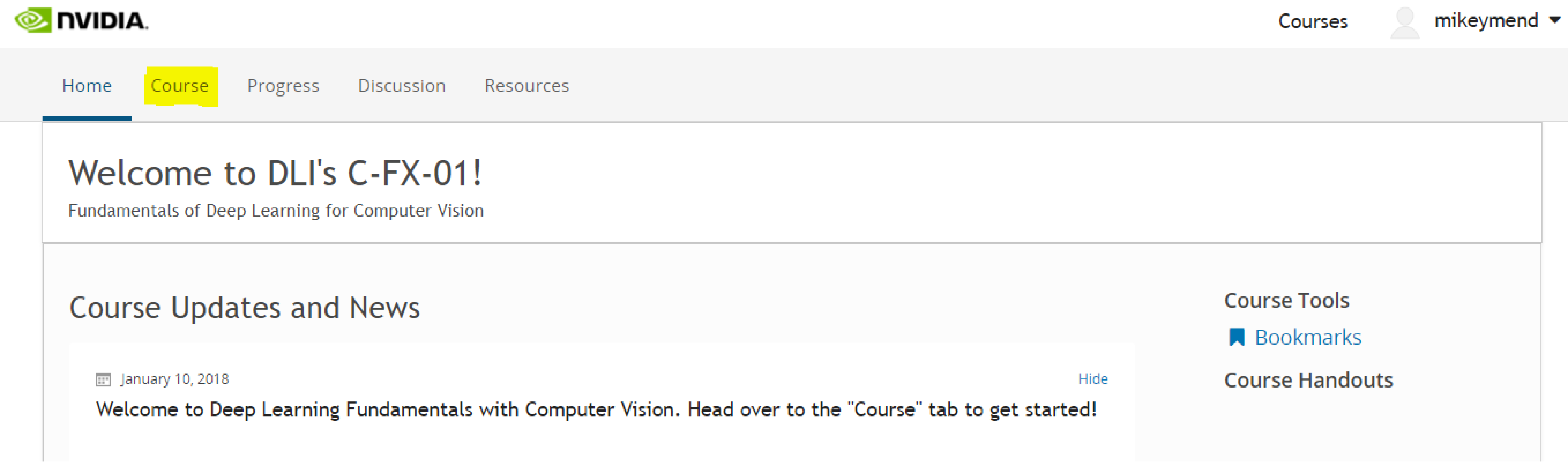
Duration: 8 Hours

Price: $90.00

You are enrolled in this course

Join course with the big green button above

# Select the “Course” Tab



# Open the first hands-on section

## Fundamentals of Deep Learning for Computer Vision

### Introduction

Welcome

### Unlocking New Capabilities

Training Deep Neural Networks: 120 minutes (Hands-On) | Resume Course

Deploying Trained Neural Networks: 40 minutes (Hands-On)

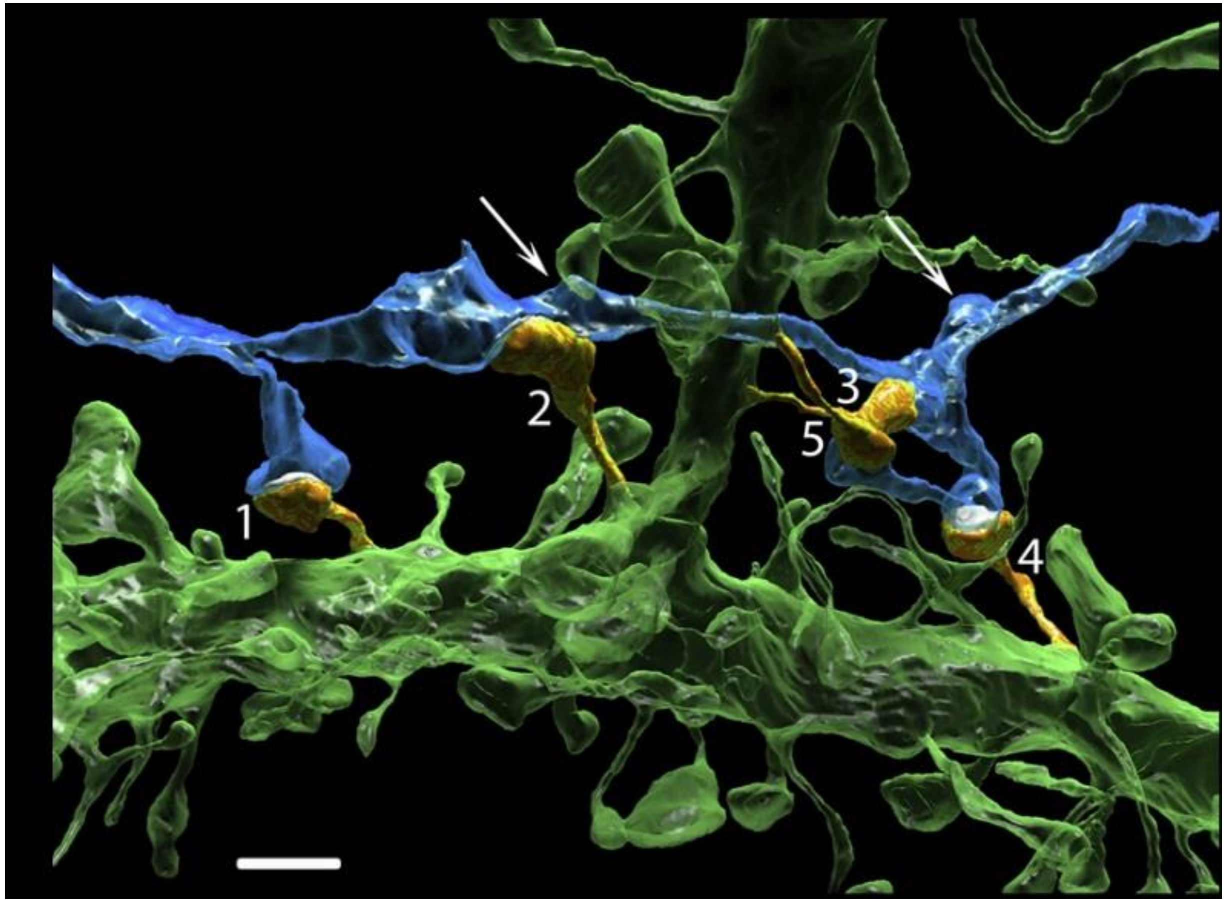
# Biological Inspiration

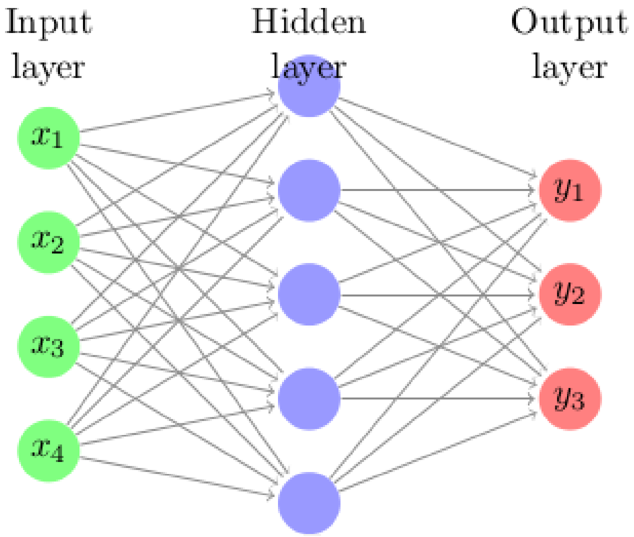
## Louie Classifier

Let's play a game where you learn something new. Identify Louie, a dog who you have never met before, by studying 4 images. While playing, we'll highlight similarities to the training process in Deep Learning. After you learn who Louie is, you'll train a neural network to do the same.

start press ENTER

# Biological Inspiration



# Artificial Neural Networks: GPU Task 1

Get ready to train a neural network. This first training session will take about 20 minutes. Select **Start** below when ready.

Hosted on:

START

# Inputs and Outputs

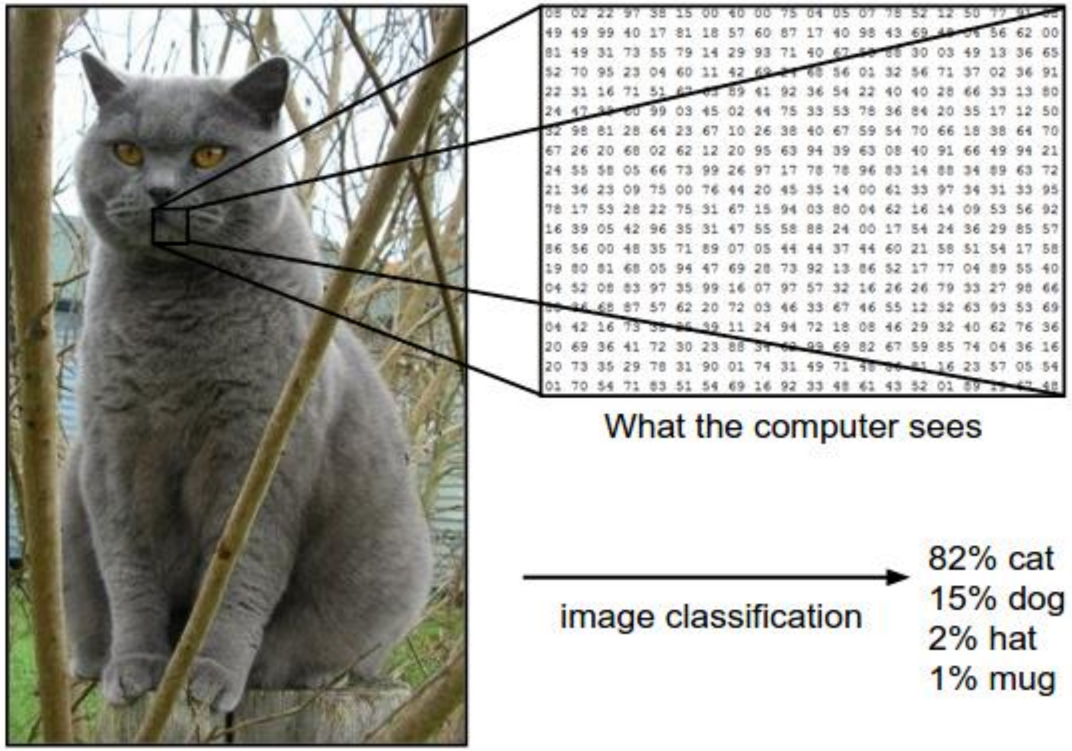
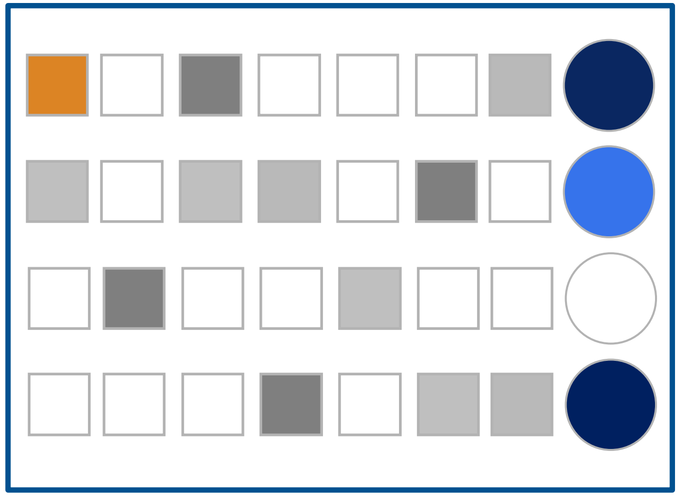
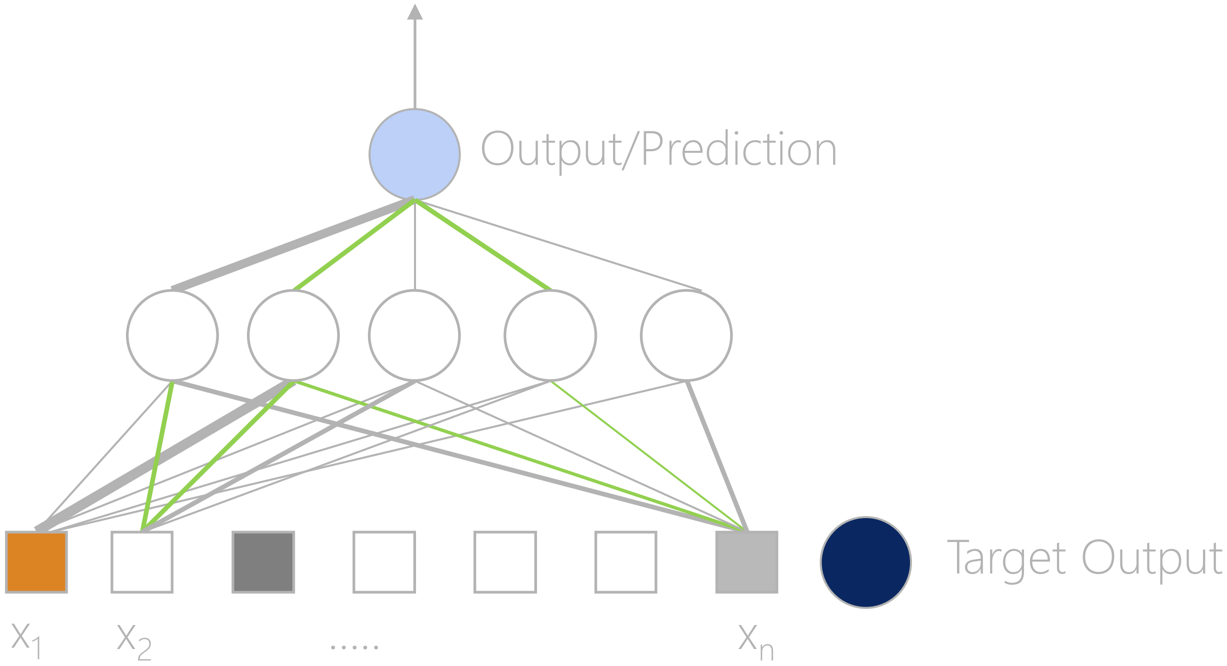


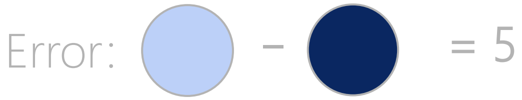

Image from the Stanford CS231 Course

# Learning Principle

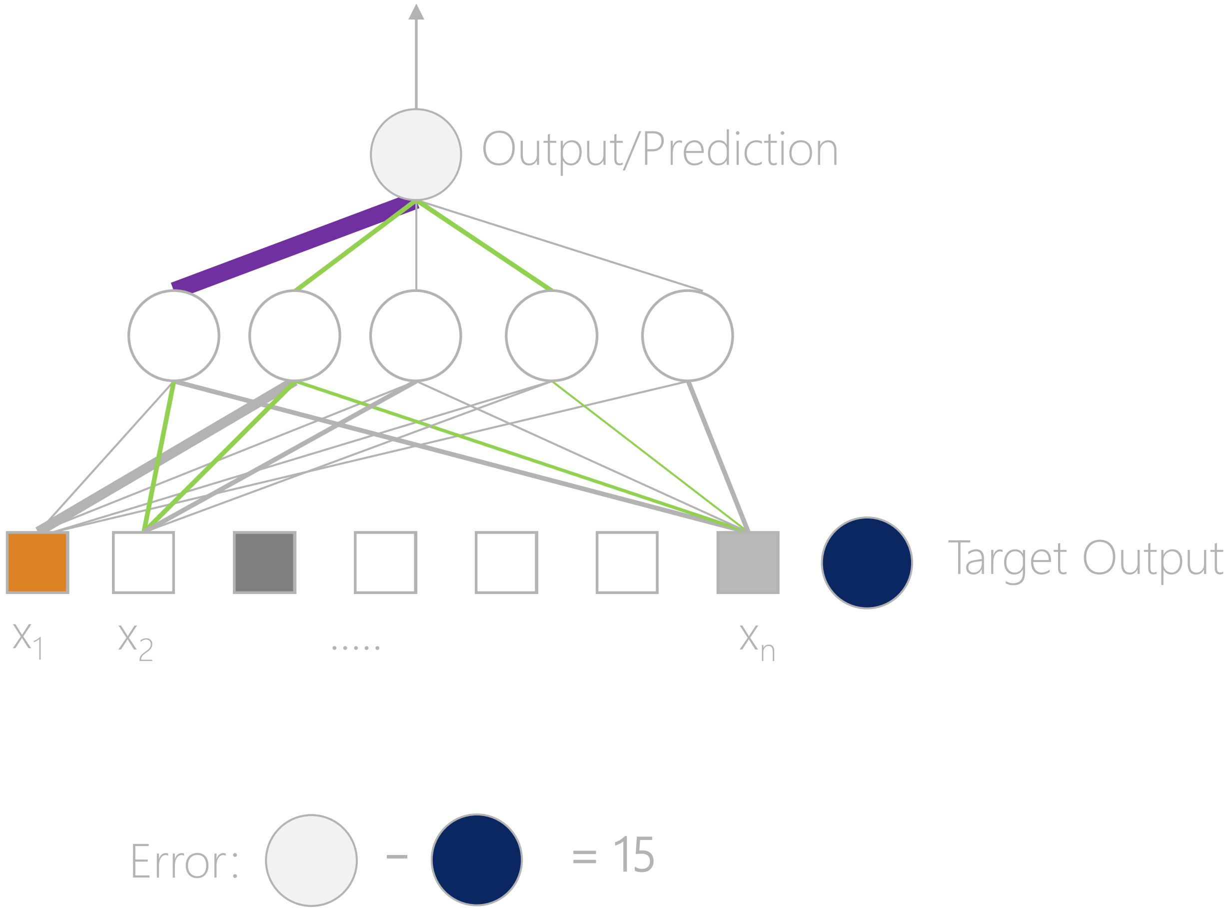
Dataset



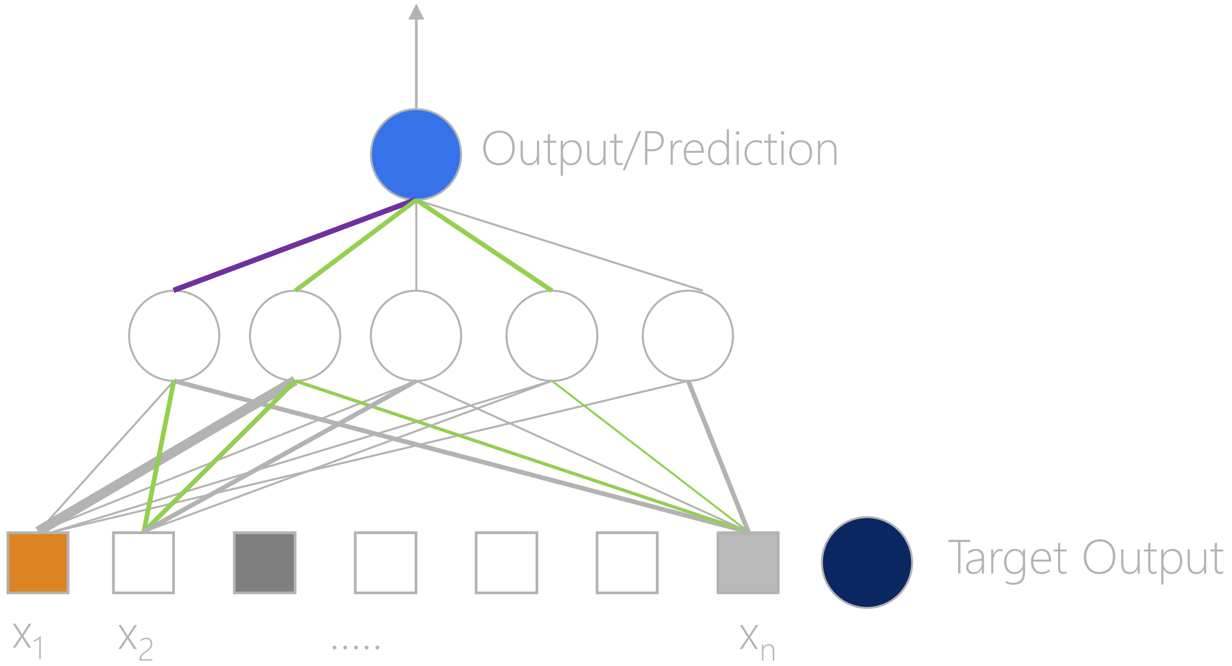


# Learning Principle

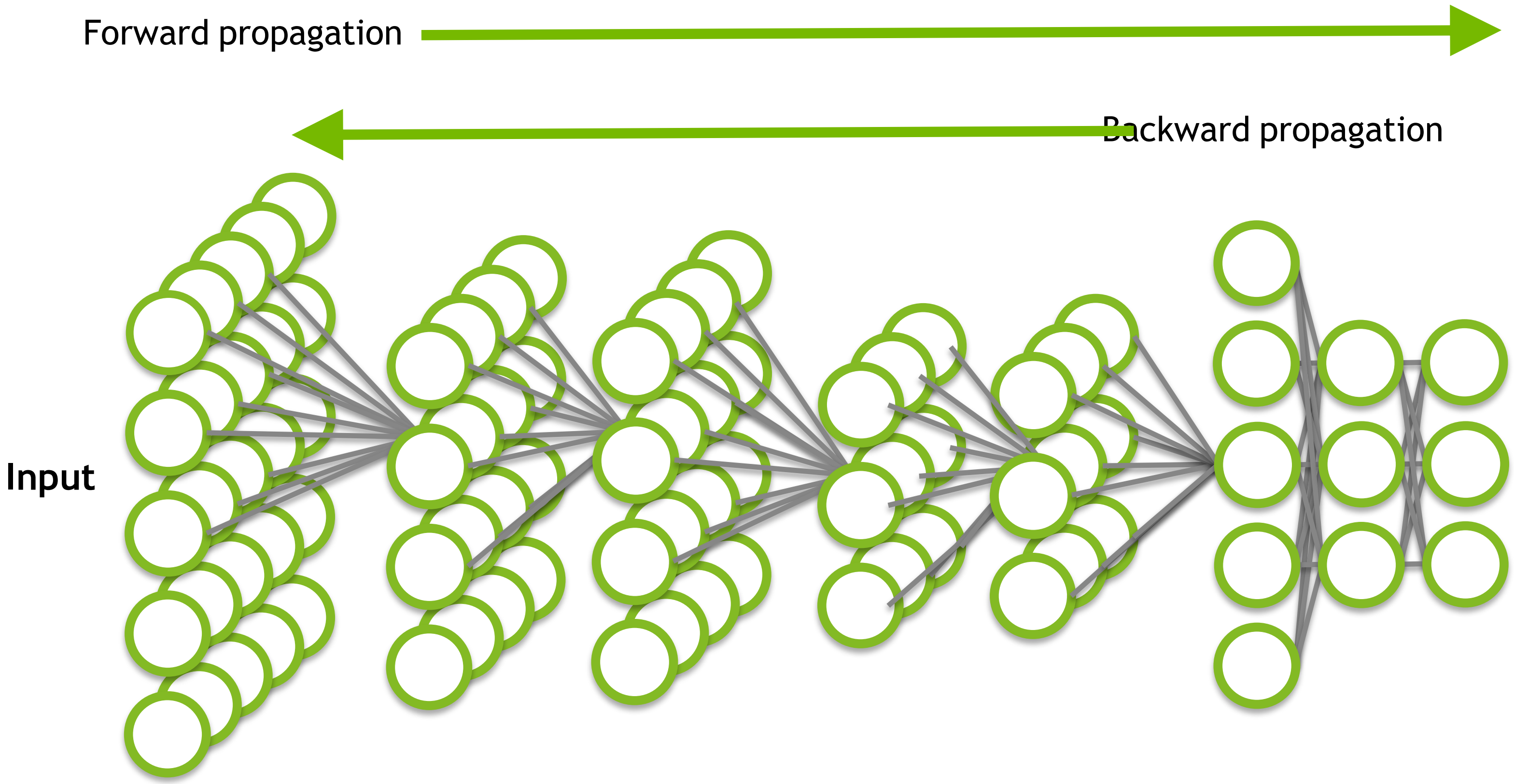


# Learning Principle




Error: ● – ● = 2.5

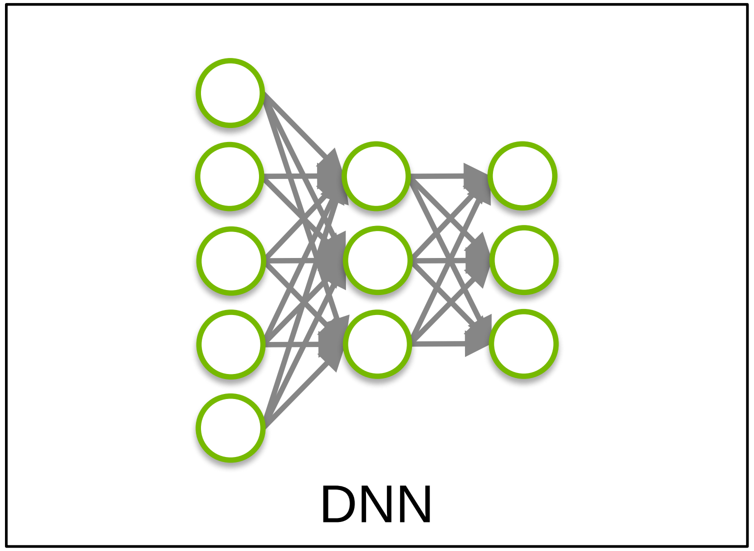
# DEEP LEARNING APPROACH - TRAINING




## Process

- Forward propagation yields an inferred label for each training image
- Loss function used to calculate difference between known label and predicted label for each image
- Weights are adjusted during backward propagation
- Repeat the process

# THE BIG BANG IN MACHINE LEARNING

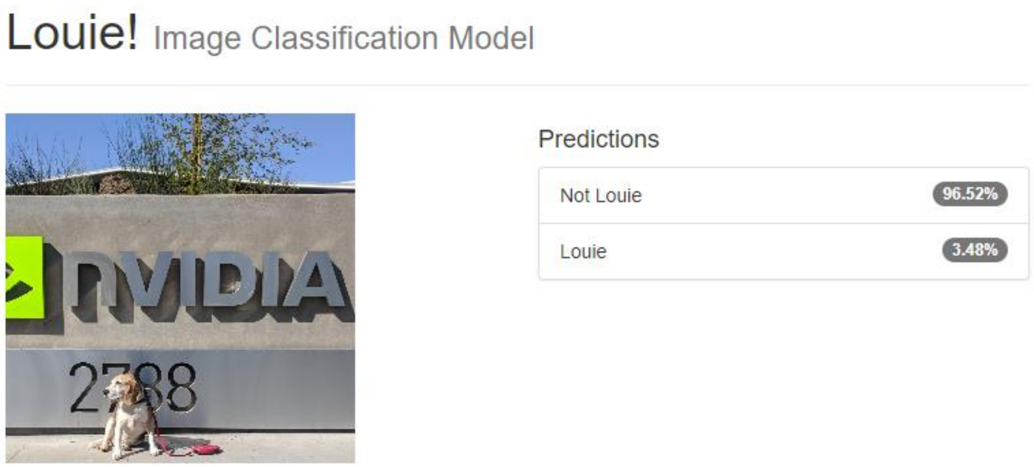





WIRED

# OVERFITTING

## What to do about it?





**Two Violets – Who is right?**

One person with two outfits. Just like Reed sometimes wears a blue shirt

# Features vs Data

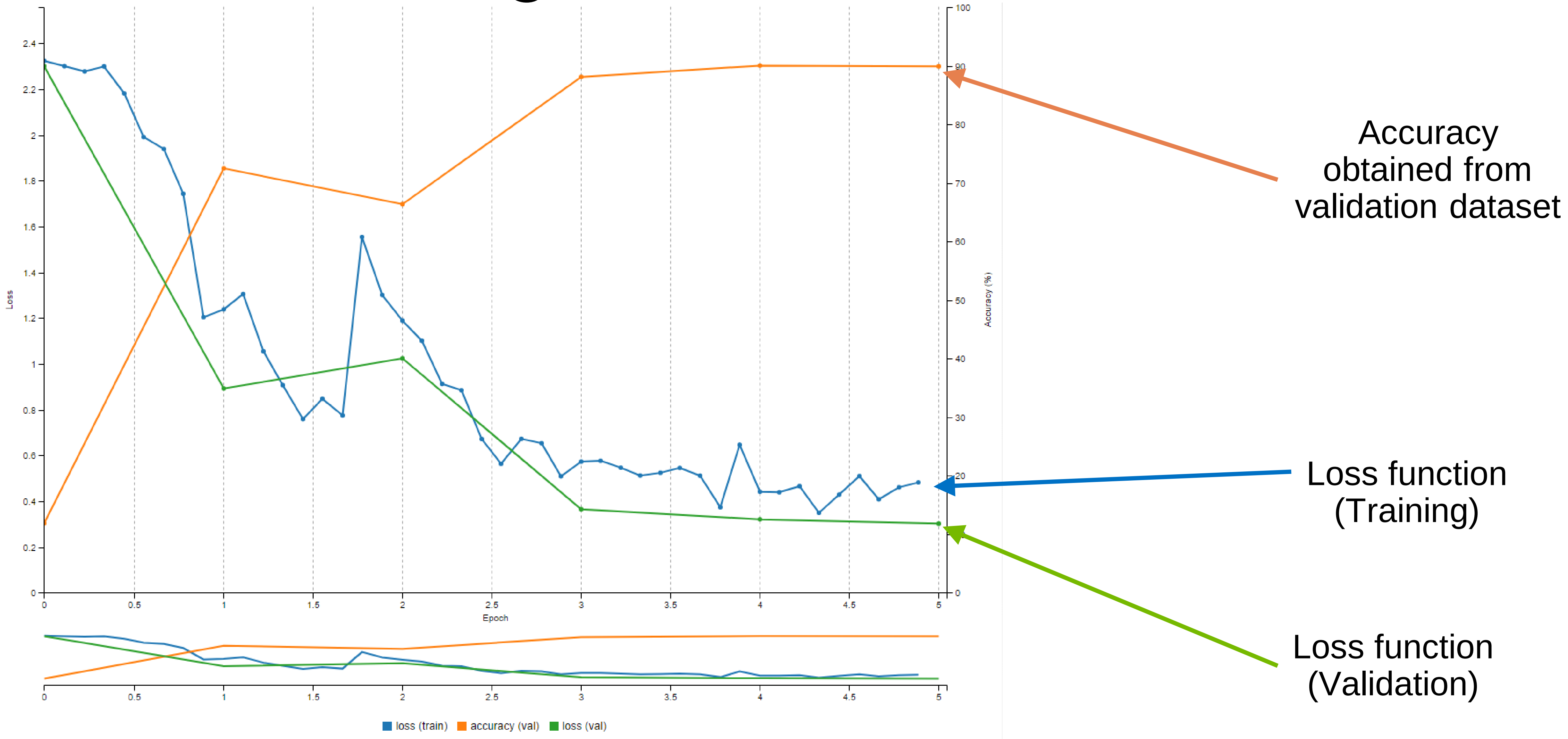
# Big Data: GPU Task 2

Get ready to train a neural network. This first training session will take about 20 minutes. Select **Start** below when ready.

Hosted on:

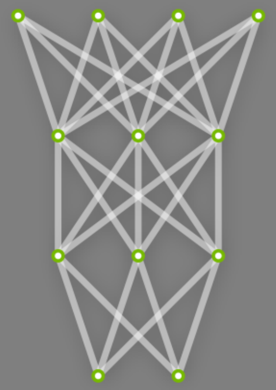
START

# Next Page - EVALUATE THE MODEL



# DEEP LEARNING

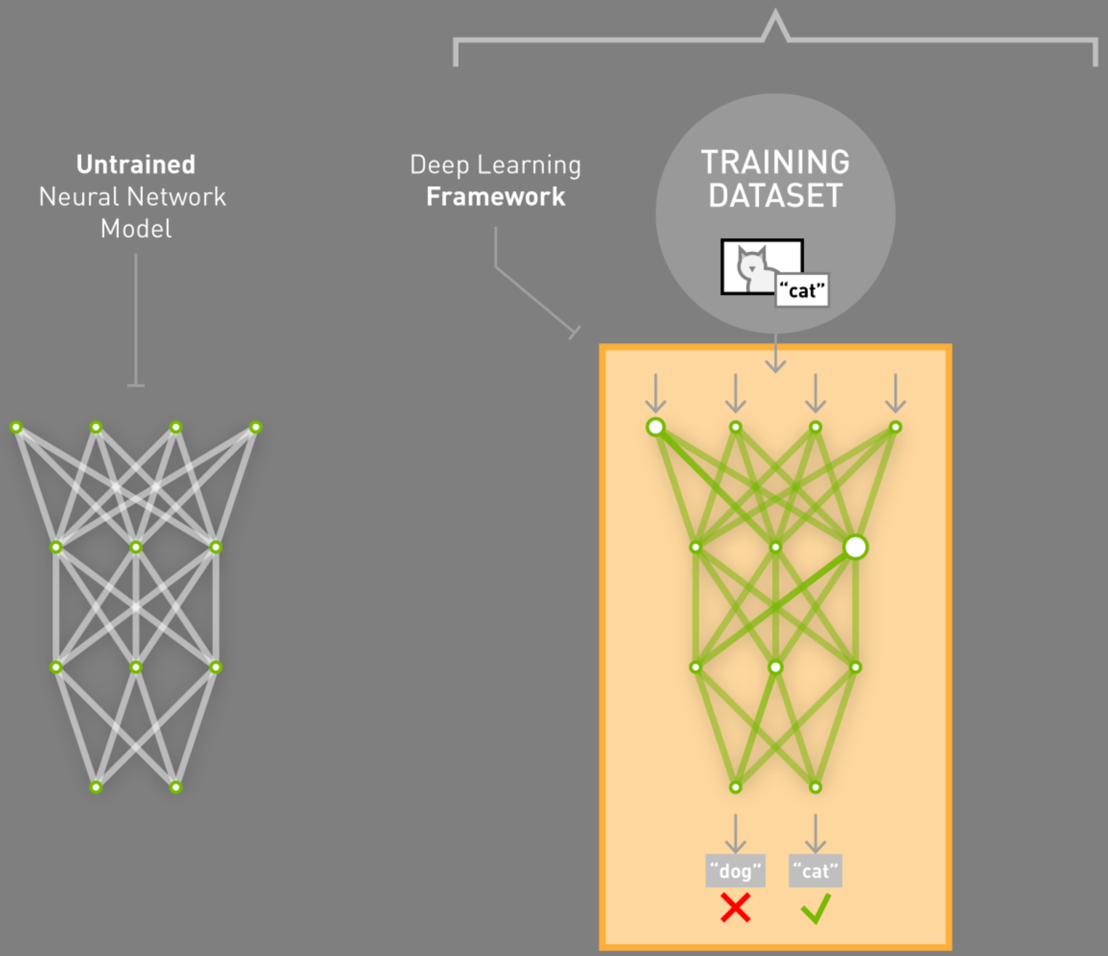
**Untrained**
Neural Network
Model



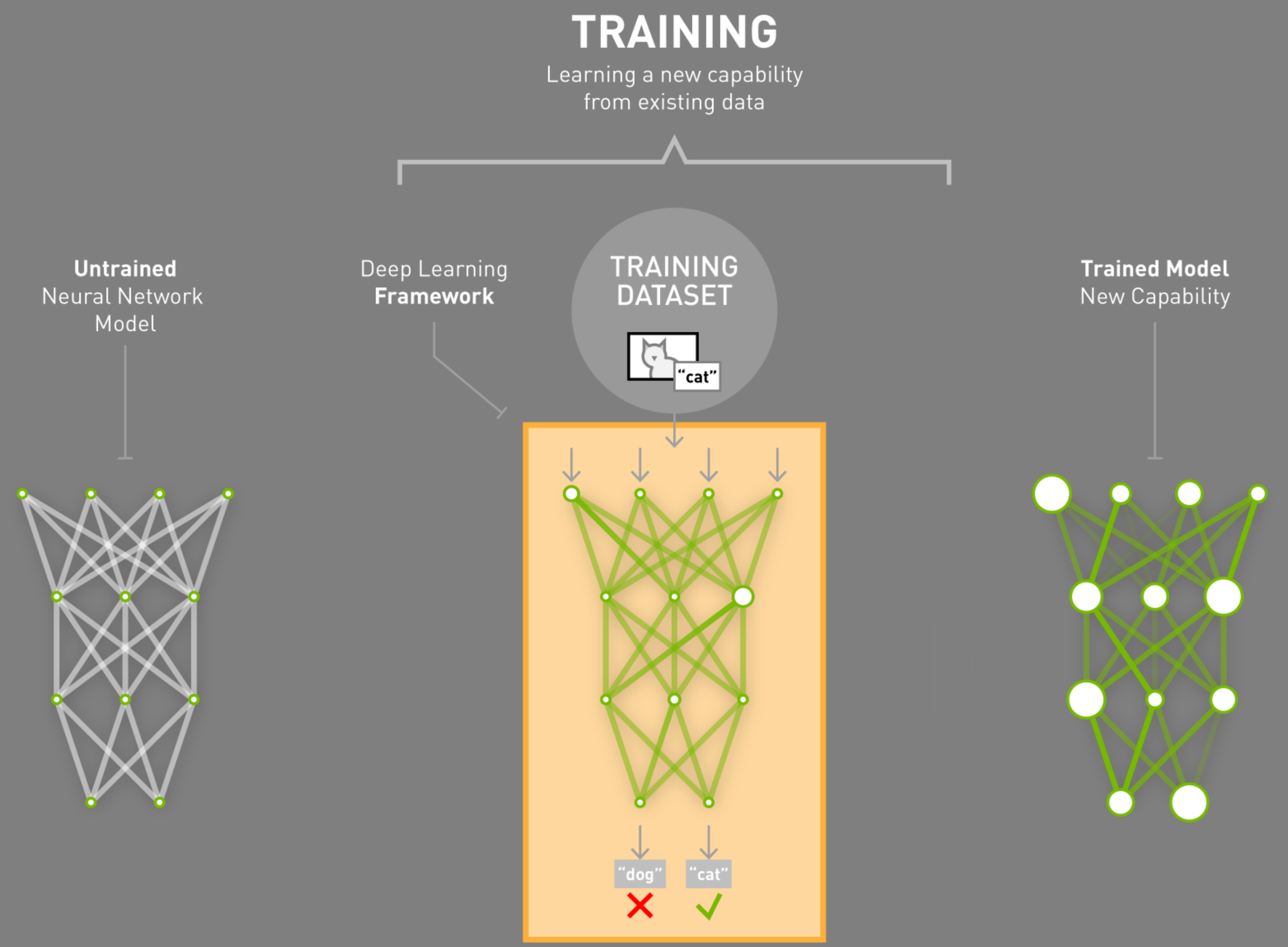
# DEEP LEARNING

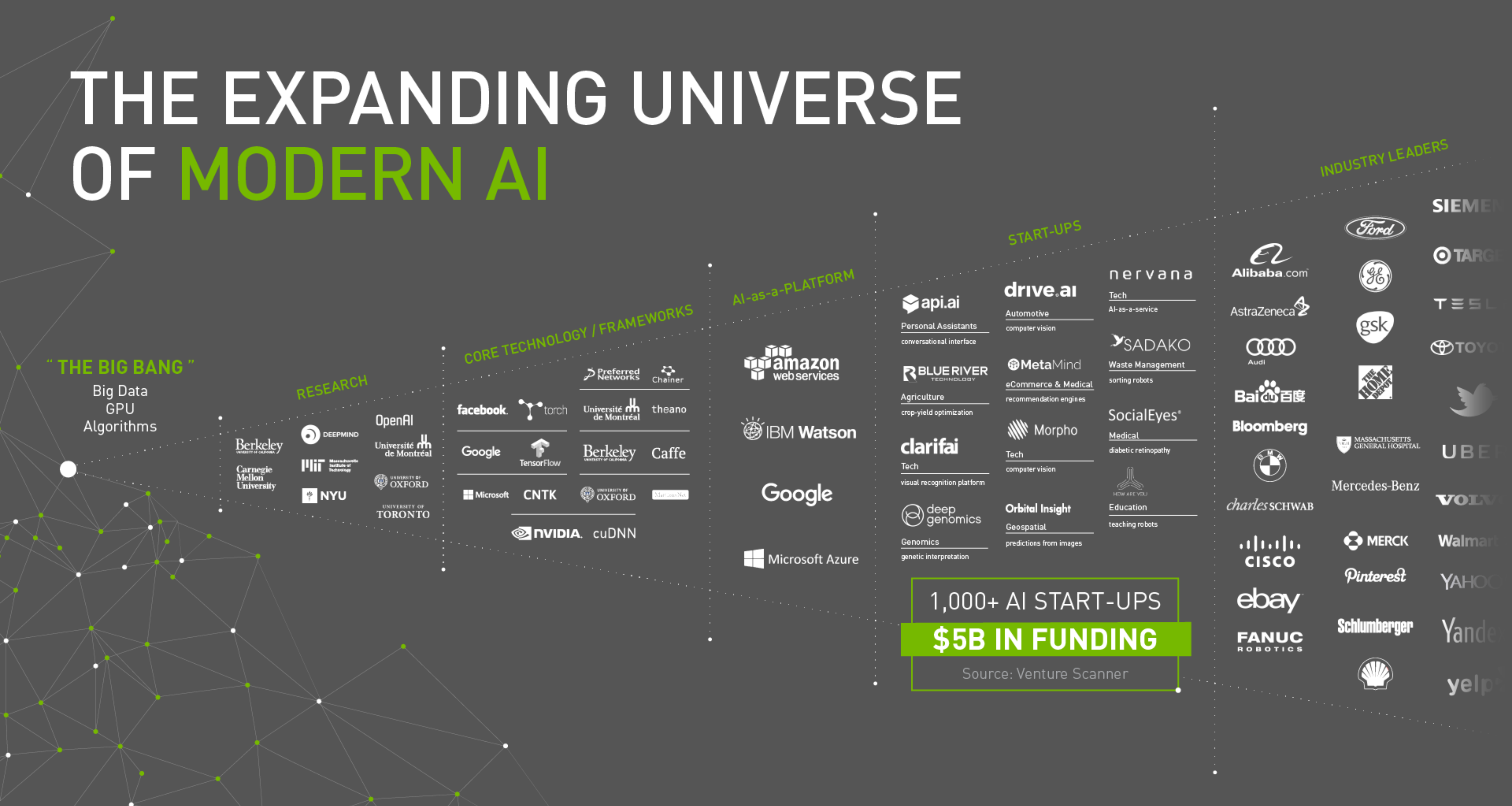
## TRAINING

Learning a new capability
from existing data



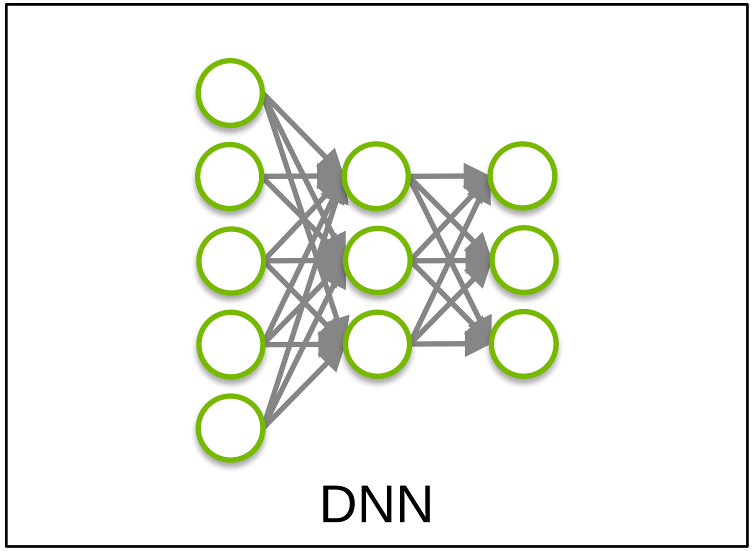
# DEEP LEARNING





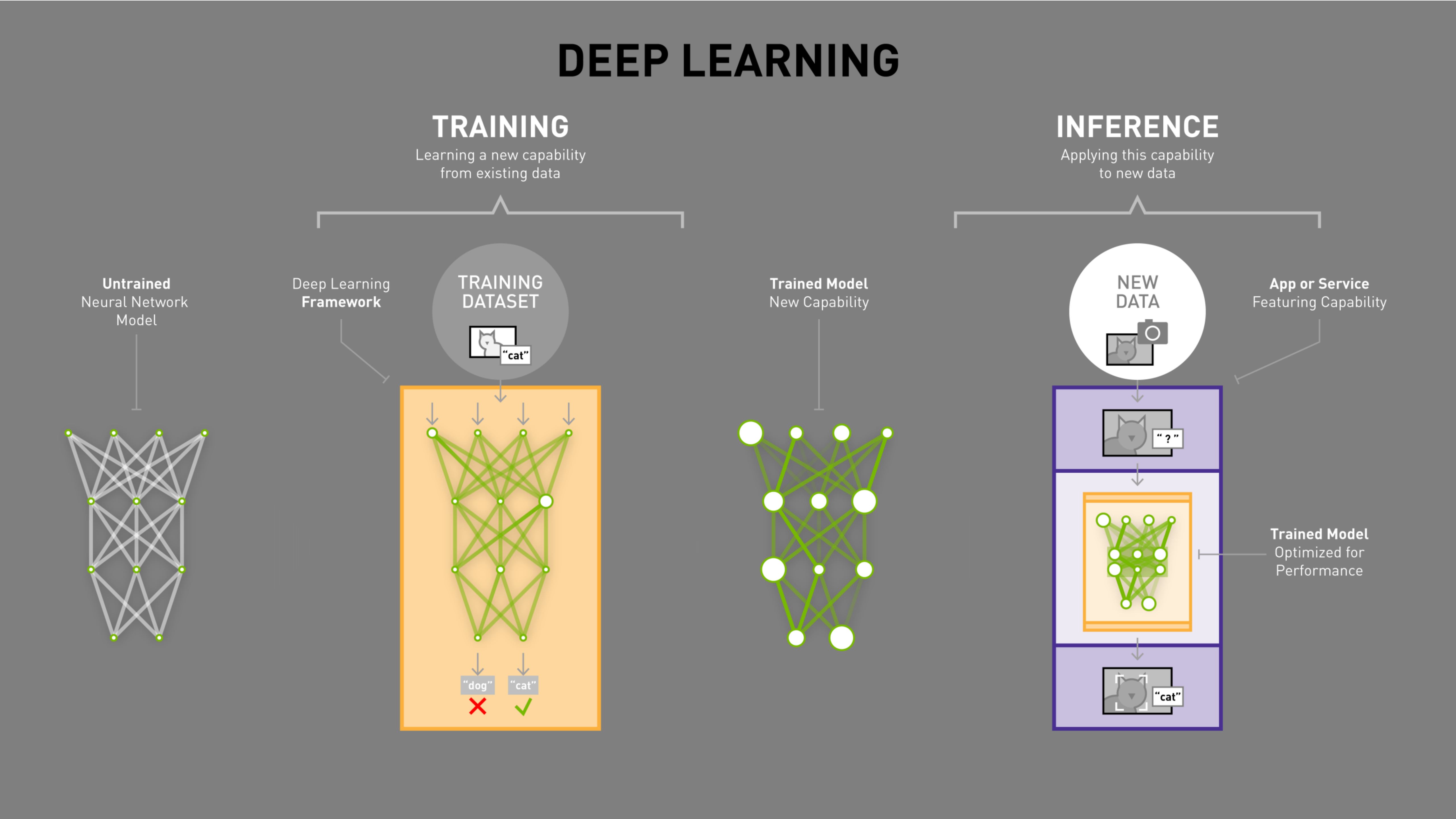
THE EXPANDING UNIVERSE OF MODERN AI
" THE BIG BANG "
Big Data
GPU
Algorithms
RESEARCH
Berkeley
Carnegie Mellon University
DEEPMIND
MIT
NYU
OpenAI
Université de Montréal
OXFORD
TORONTO
CORE TECHNOLOGY / FRAMEWORKS
Preferred Networks
Chainer
facebook
torch
Université de Montréal
theano
Google
TensorFlow
Berkeley
Caffe
Microsoft
CNTK
OXFORD
NVIDIA
cuDNN
AI-as-a-PLATFORM
amazon web services
IBM Watson
Google
Microsoft Azure
START-UPS
api.ai
Personal Assistants
conversational interface
BLUE RIVER TECHNOLOGY
Agriculture
crop-yield optimization
clarifai
Tech
visual recognition platform
deep genomics
Genomics
genetic interpretation
drive.ai
Automotive
computer vision
MetaMind
eCommerce & Medical
recommendation engines
Morpho
Tech
computer vision
Orbital Insight
Geospatial
predictions from images
nervana
Tech
AI-as-a-service
SADAKO
Waste Management
sorting robots
SocialEyes
Medical
diabetic retinopathy
Education
teaching robots
1,000+ AI START-UPS
$5B IN FUNDING
Source: Venture Scanner
INDUSTRY LEADERS
Alibaba.com
AstraZeneca
Audi
Baidu 百度
Bloomberg
charles SCHWAB
CISCO
ebay
FANUC ROBOTICS
Ford
gsk
MASSACHUSETTS GENERAL HOSPITAL
Mercedes-Benz
MERCK
Pinterest
Schlumberger

# THE BIG BANG IN MACHINE LEARNING

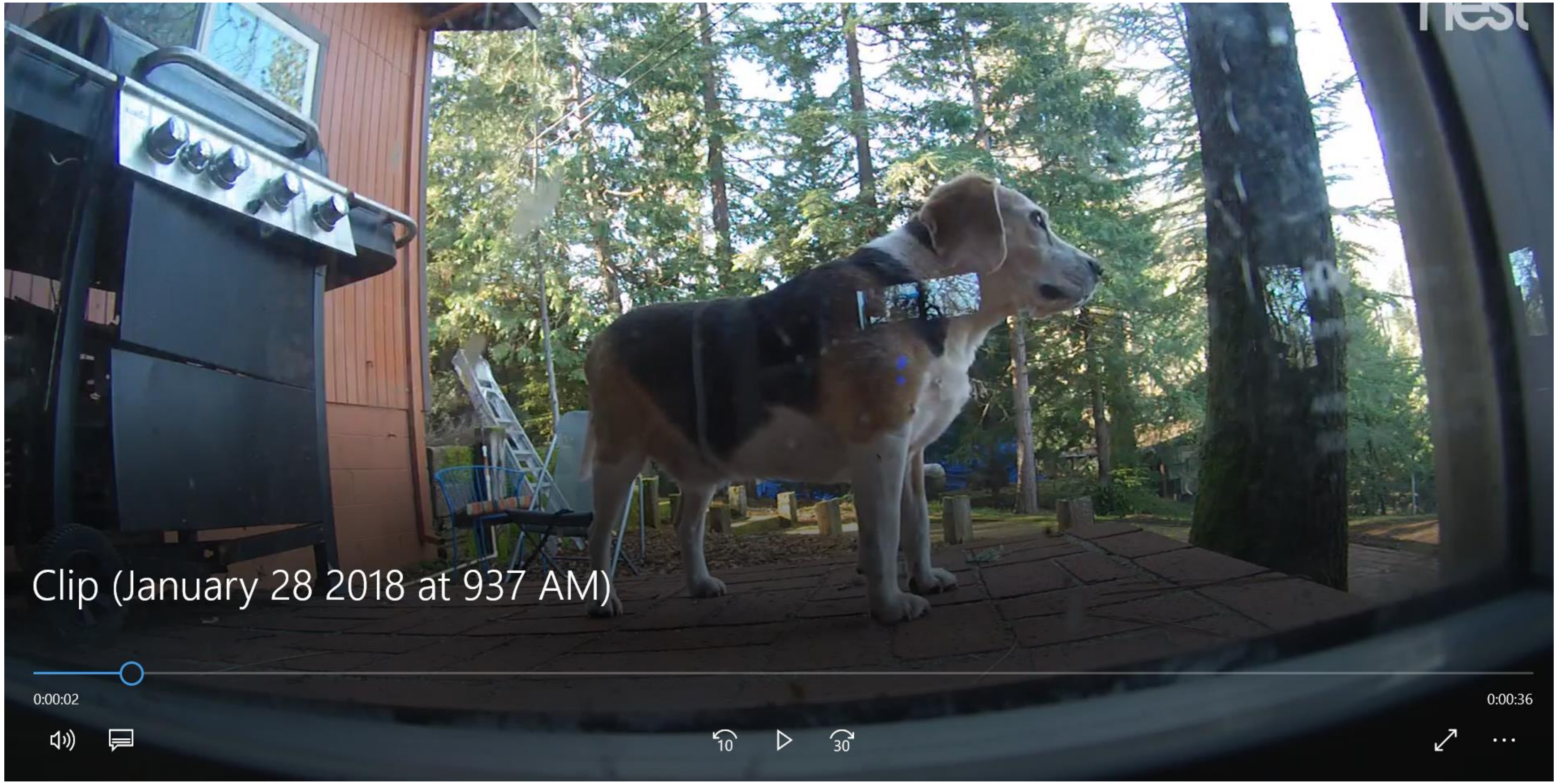





WIRED


DEEP LEARNING
TRAINING
Learning a new capability from existing data
INFERENCE
Applying this capability to new data
Untrained Neural Network Model
Deep Learning Framework
TRAINING DATASET
"cat"
"dog"
"cat"
Trained Model New Capability
NEW DATA
App or Service Featuring Capability
" ? "
Trained Model Optimized for Performance
"cat"

# LUNCH

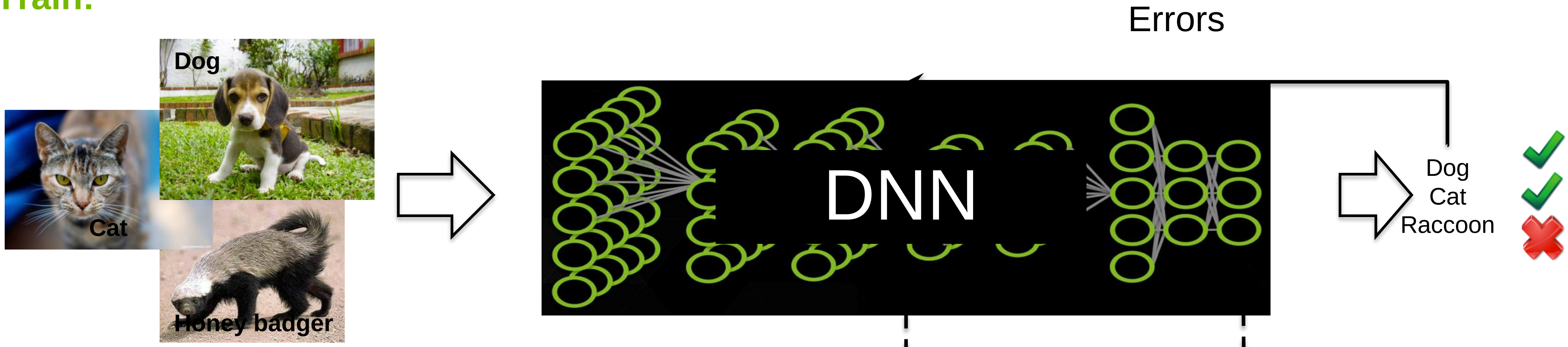



**Exclusive Doggy Door**

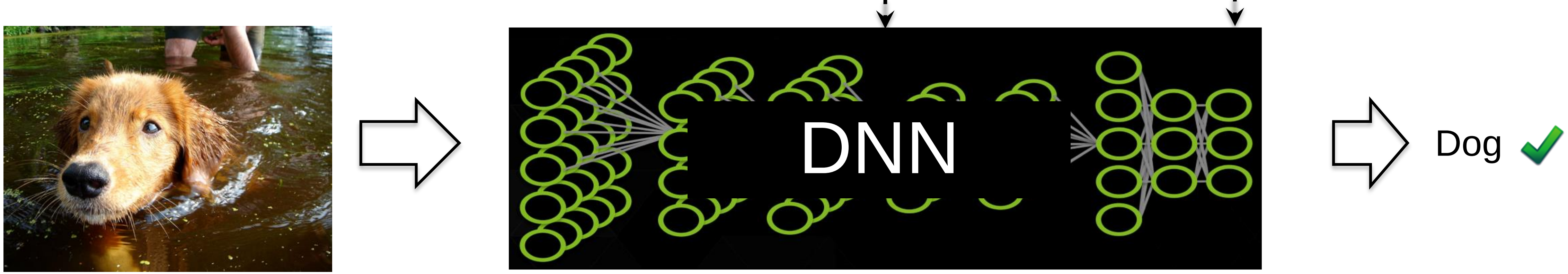
# Deployment

# Deep Learning Approach

**Train:**

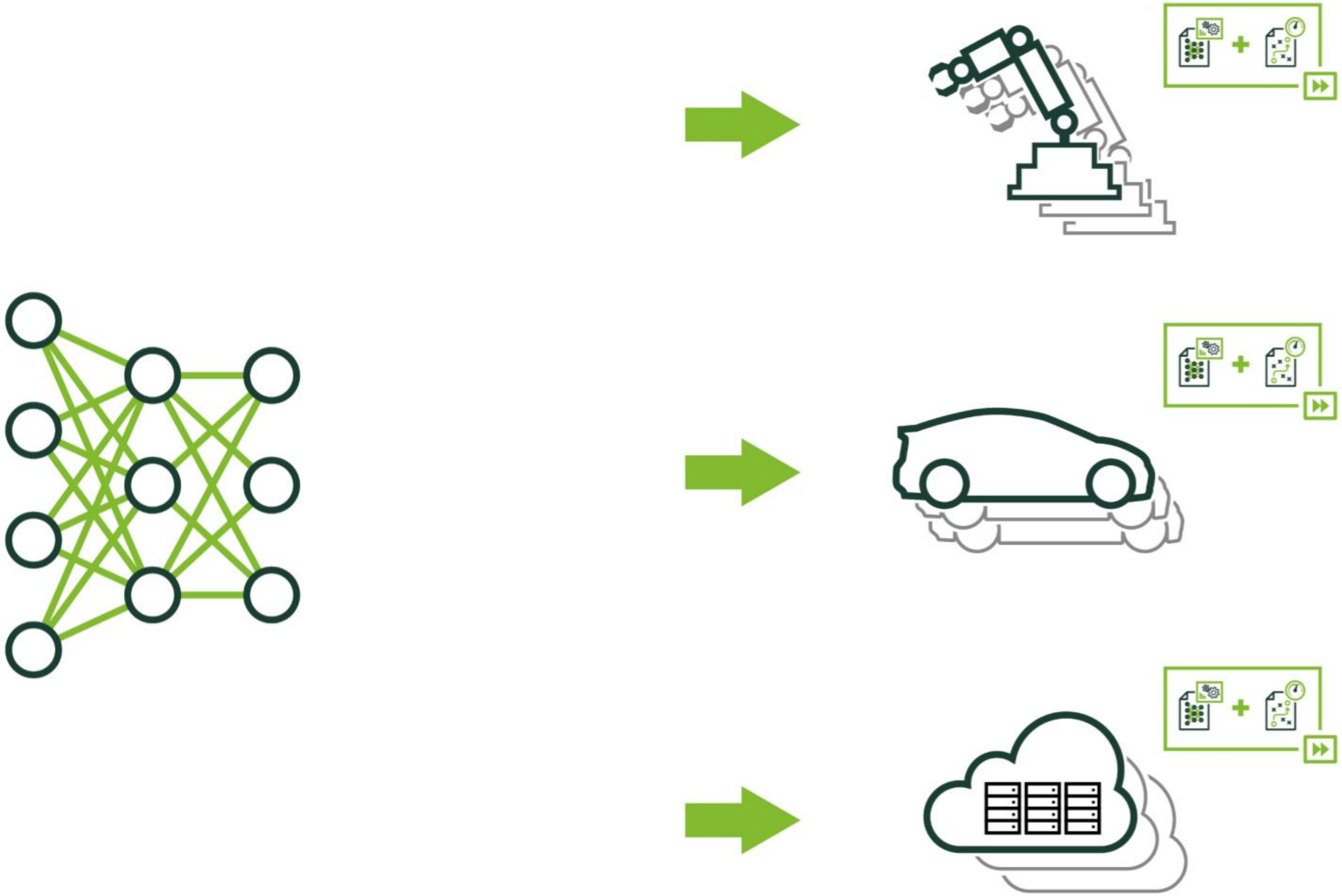



**Deploy:**



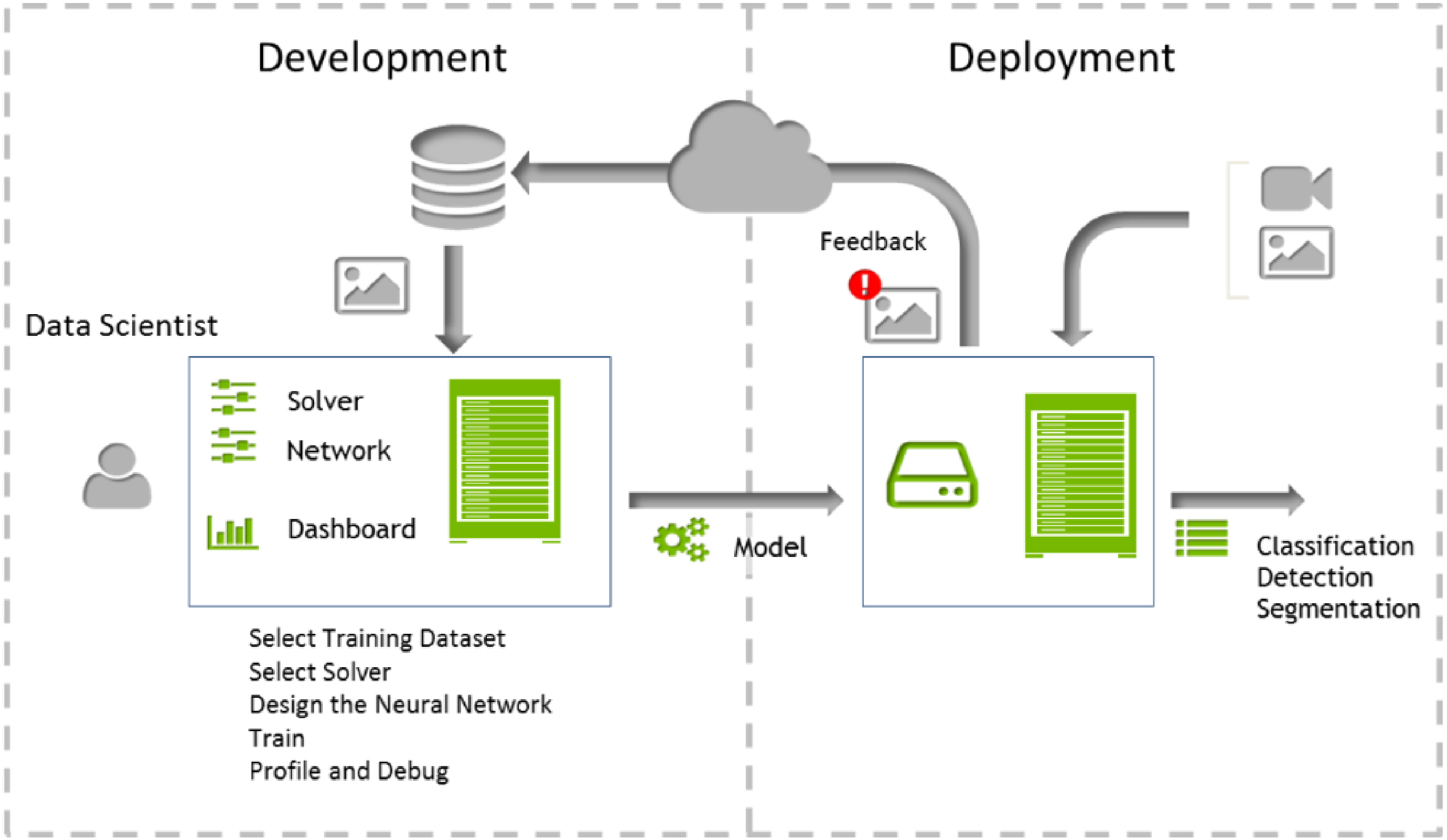
# Deployment

## How do I use a trained neural network as part of a solution?

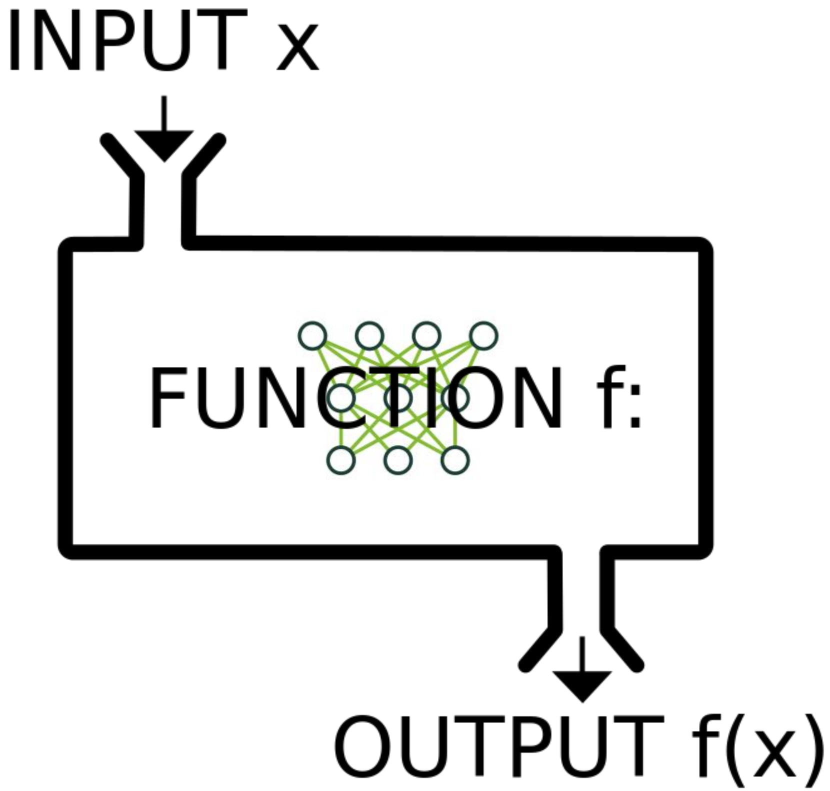


# Deep Learning Approach

## Neural network training and inference



# Expected Inputs and Useful Outputs



# Our current architecture

## FRAMEWORK

We've been working in a framework called Caffe.

Each framework requires a different way (syntax) of describing architectures and hyperparameters.

Other frameworks include TensorFlow, MXNet, etc.

## NETWORK

We've been working with a network called AlexNet.

Each network can be described and trained using ANY framework.

Different networks learn differently: different training rates, methods, etc. Think different learners.

## TOOL - UI

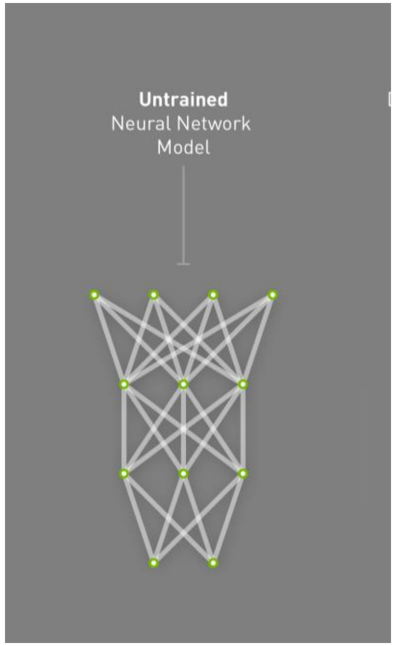
We've been working with a UI called DIGITS

The community works to make model building and deployment easier.

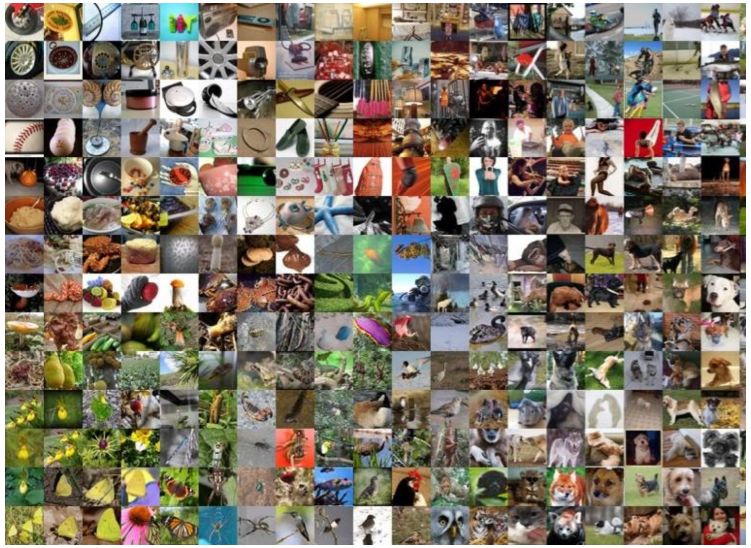
Other tools include Keras, Tensorboard, or APIs with common programming languages.

# Components of a Model

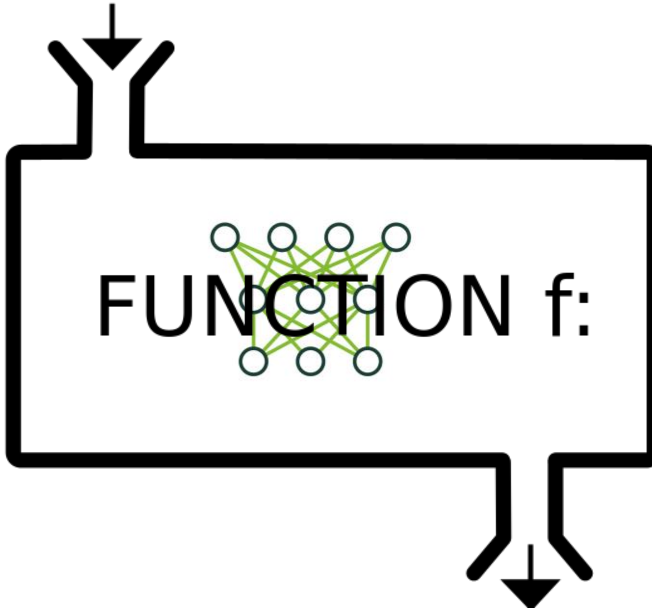
Model Architecture = deploy.prototxt




Learned Weights = ***.caffemodel



Model



# Caffe files

***.caffemodel**: a binary file containing the weights for the model at the iteration it was saved

***.prototxt**: a text file describing the network model and its layers

**image_mean.binaryproto**: the image mean of the dataset, the model requires this to be subtracted from each image before classifying

# Deploying Our Model: GPU Task 3

## Deploying our Model: GPU Task 3

VIEW UNIT IN STUDIO

Bookmark this page

NVIDIA

Select **Start** to launch our GPU task.

# Performance

# Performance - Deployment

# CATEGORIES OF PERFORMANCE

| Requirement | Challenges |
| --- | --- |
| **High Throughput** | **Unable to processing high-volume, high-velocity data**<br>➢ Impact: Increased cost ($, time) per inference |
| **Low Response Time** | **Applications don't deliver real-time results**<br>➢ Impact: Negatively affects user experience (voice recognition, personalized recommendations, real-time object detection) |
| **Power and Memory Efficiency** | **Inefficient applications**<br>➢ Impact: Increased cost (running and cooling), makes deployment infeasible |
| **Deployment-Grade Solution** | **Research frameworks not designed for production**<br>➢ Impact: Framework overhead and dependencies increases time to solution and affects productivity |

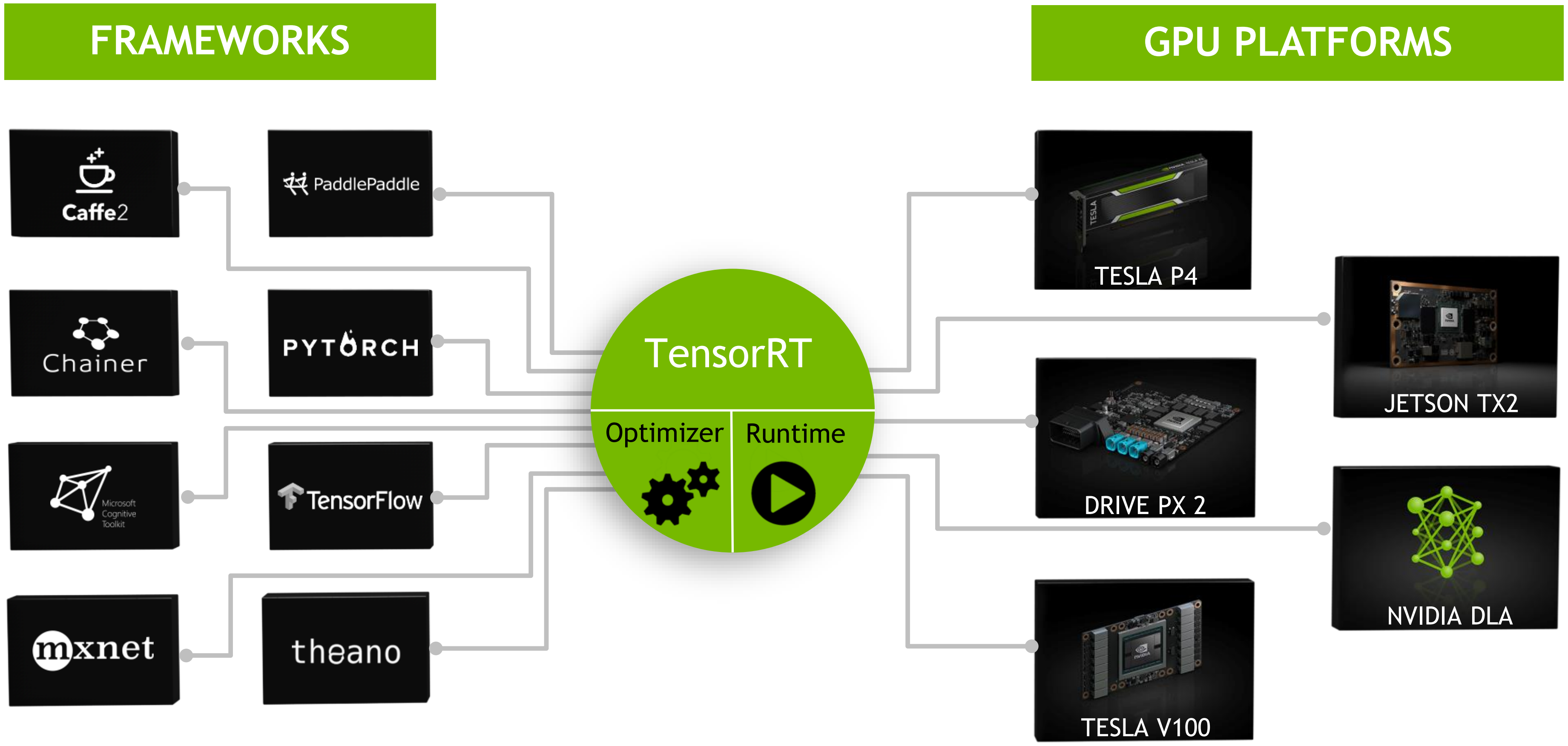
# Levers

- Batch size
  - Reduce for less latency
  - Increase for more throughput
- Tools
  - The right deployment platform
  - TensorRT

# NVIDIA TENSORRT

# NVIDIA TENSORRT

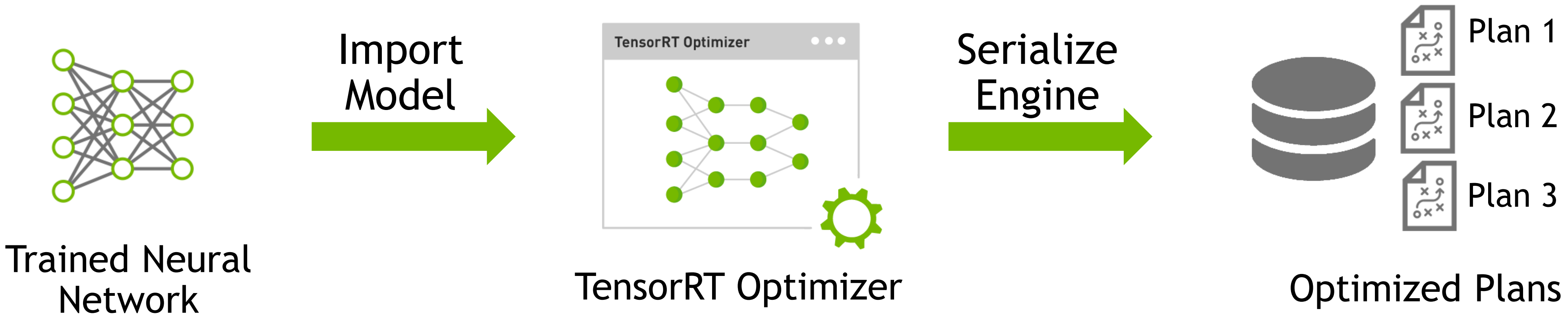
## Programmable Inference Accelerator



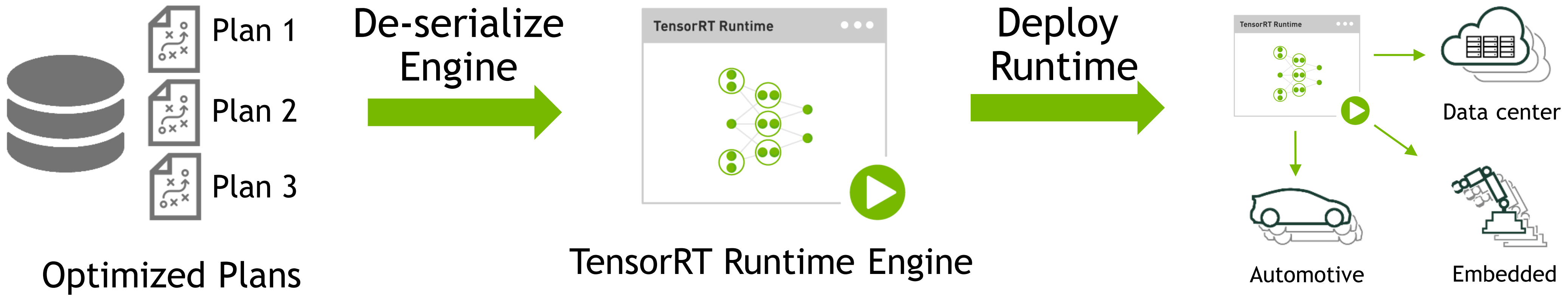

developer.nvidia.com/tensorrt

# TENSORRT DEPLOYMENT WORKFLOW

## **Step 1:** Optimize trained model

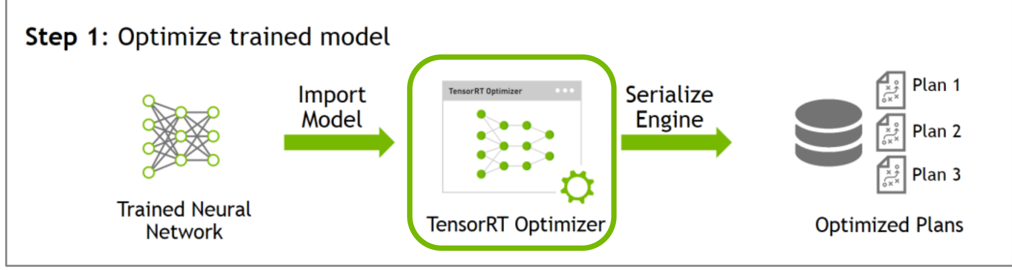



## **Step 2:** Deploy optimized plans with runtime

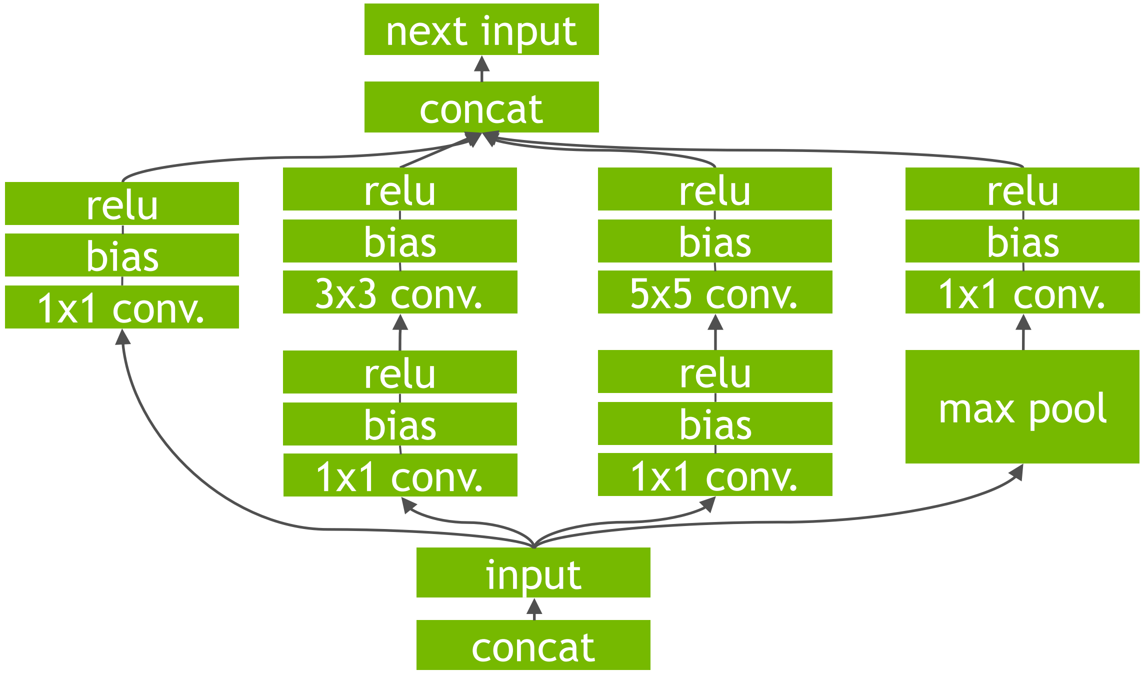


# TENSORRT OPTIMIZATIONS

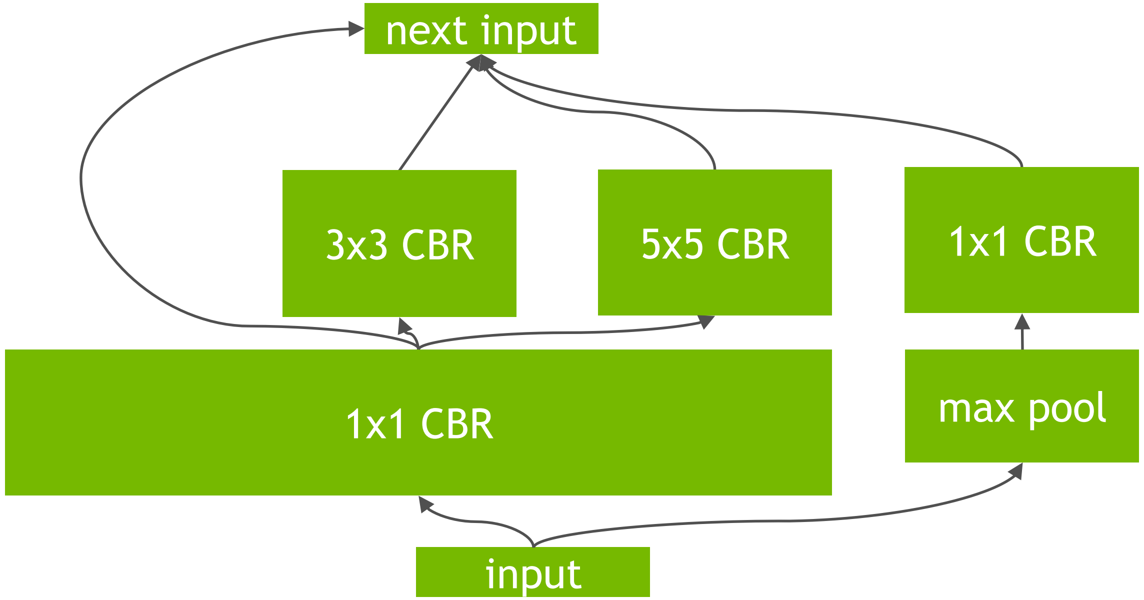
# LAYER & TENSOR FUSION



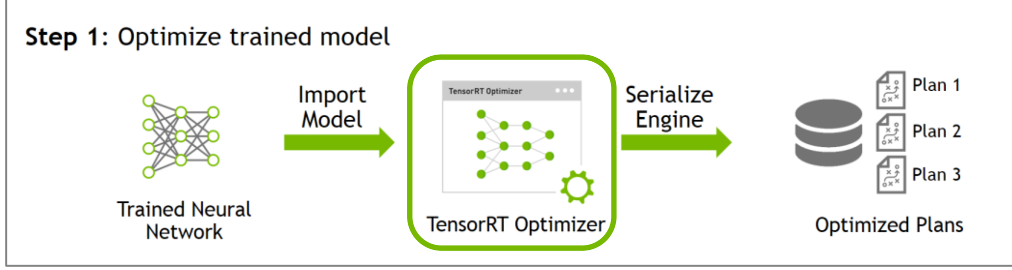

## Un-Optimized Network




## TensorRT Optimized Network



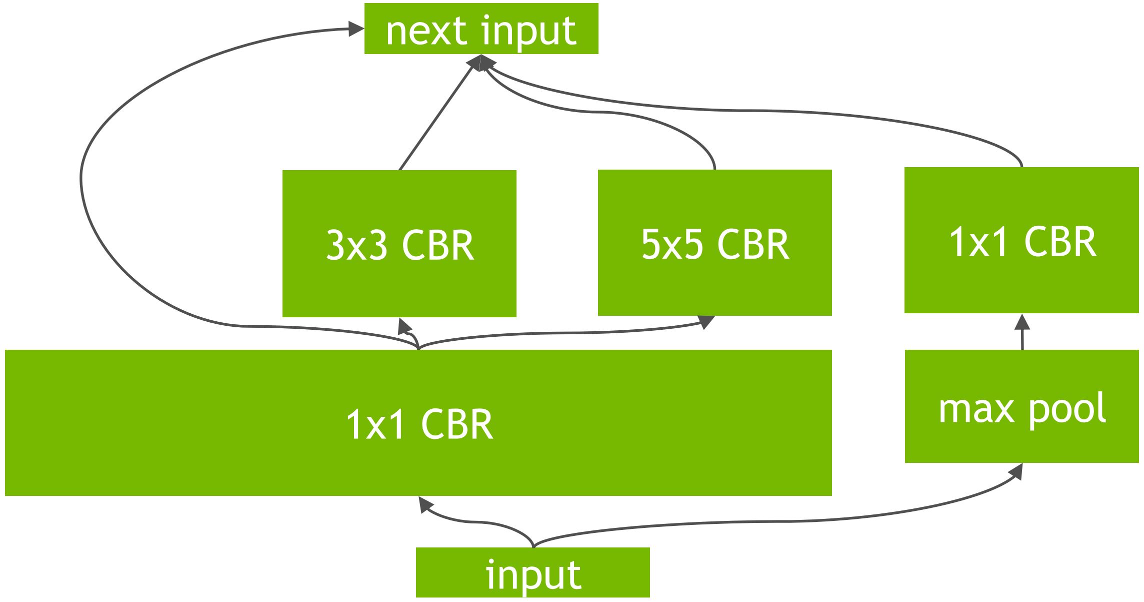
# LAYER & TENSOR FUSION




- Vertical Fusion
- Horizonal Fusion
- Layer Elimination

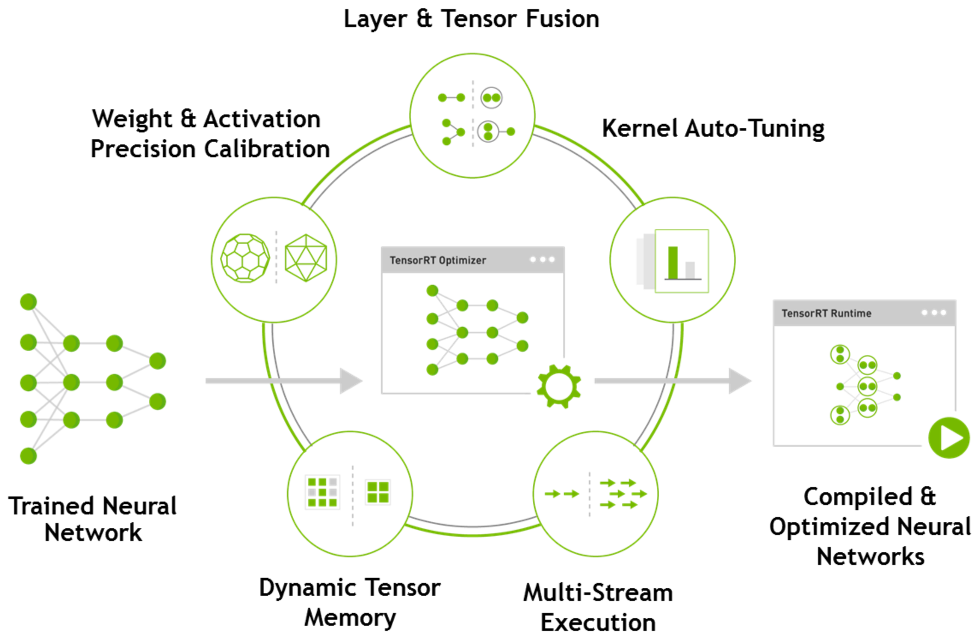
| Network | Layers before | Layers after |
|---|---|---|
| VGG19 | 43 | 27 |
| Inception V3 | 309 | 113 |
| ResNet-152 | 670 | 159 |

## TensorRT Optimized Network



# TENSORRT KEY INFO

- ✓ Generate optimized, deployment-ready runtime engines for low latency inference
- ✓ Import models trained from Caffe or TensorFlow, or use Network Definition API
- ✓ Deploy in FP32 or reduced precision INT8, FP16 for higher throughput
- ✓ Optimize frequently used layers and integrate user defined custom layers



# Performance - Training

# Next Page

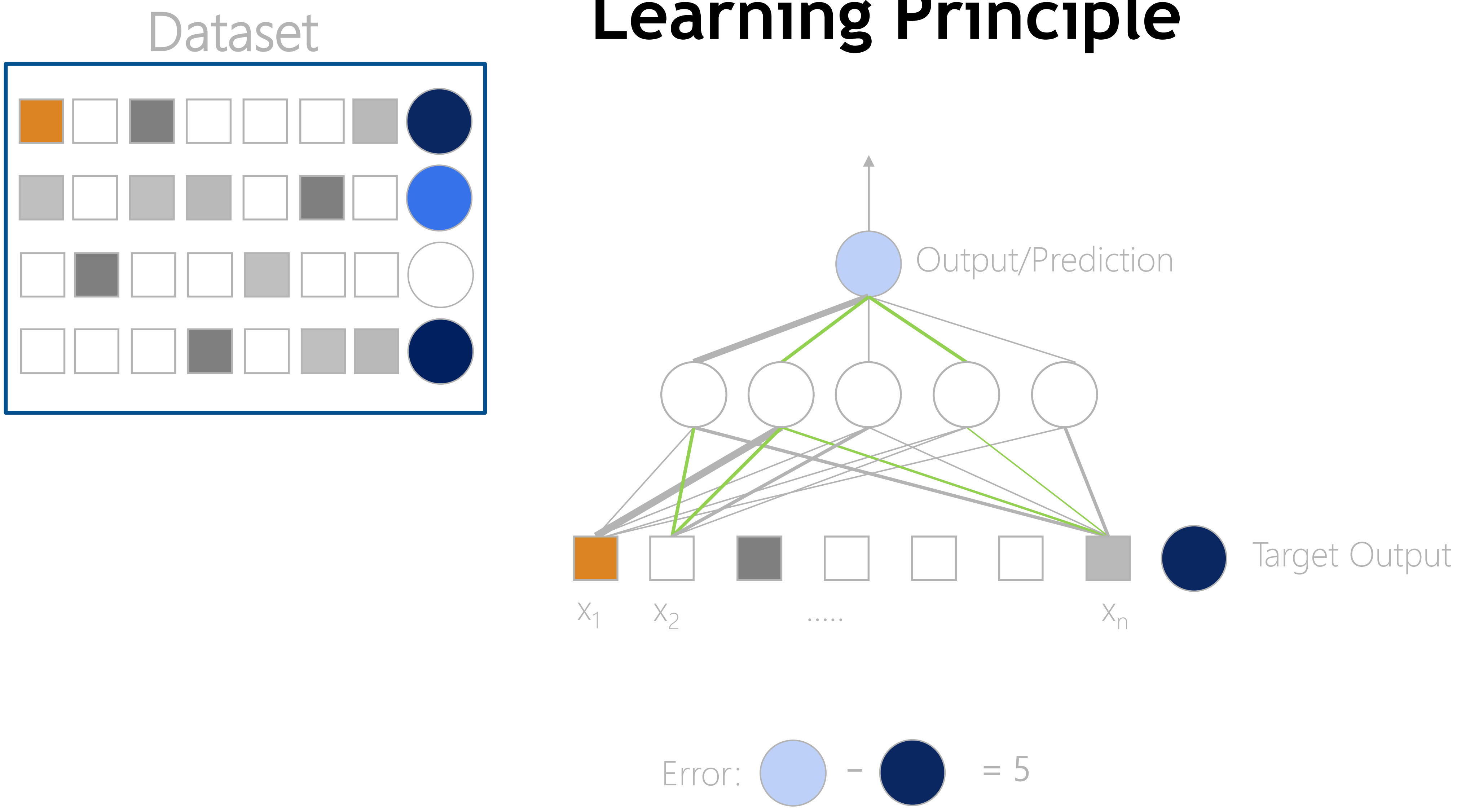
## Performance during Training: GPU Task 4

VIEW UNIT IN STUDIO

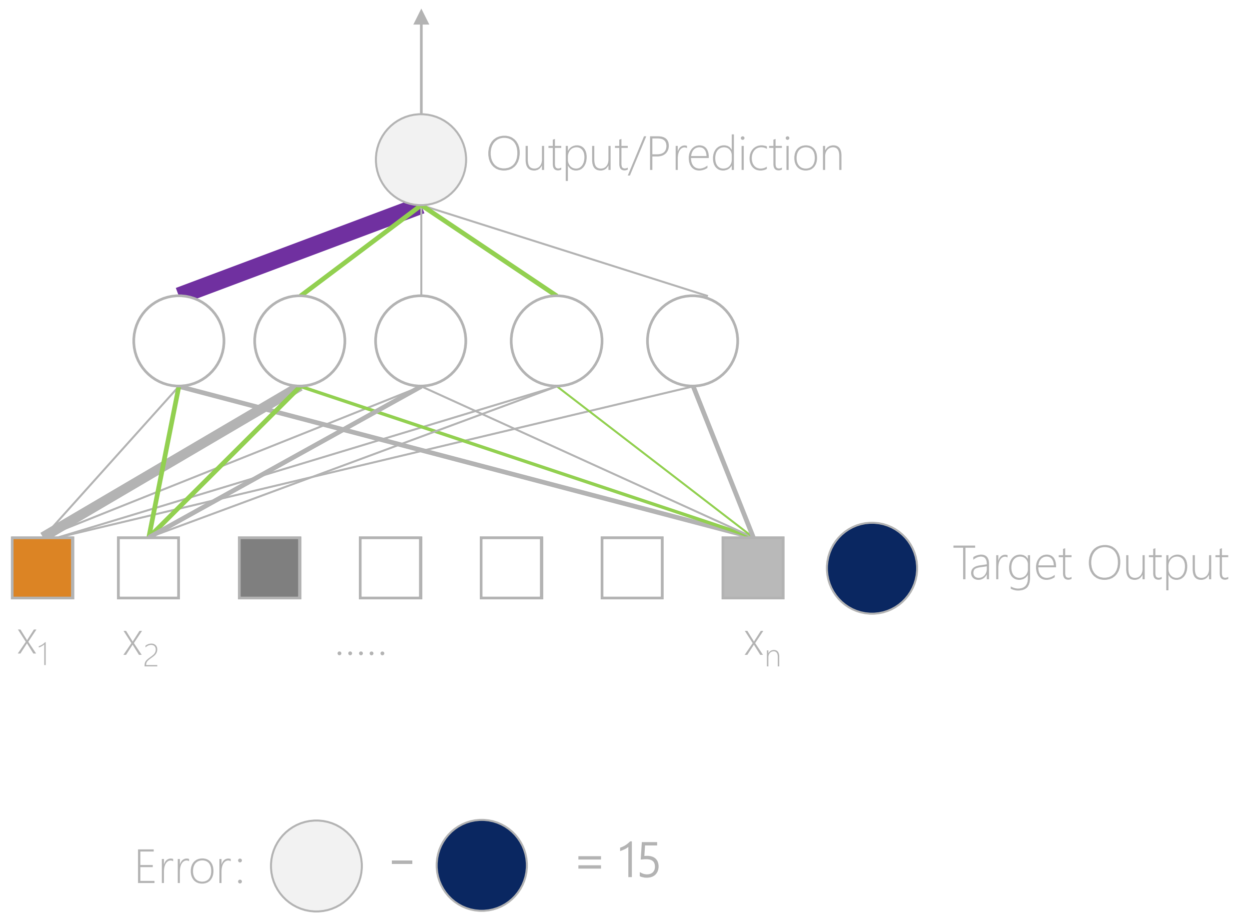
Bookmark this page

Select **START** to load your next GPU task.

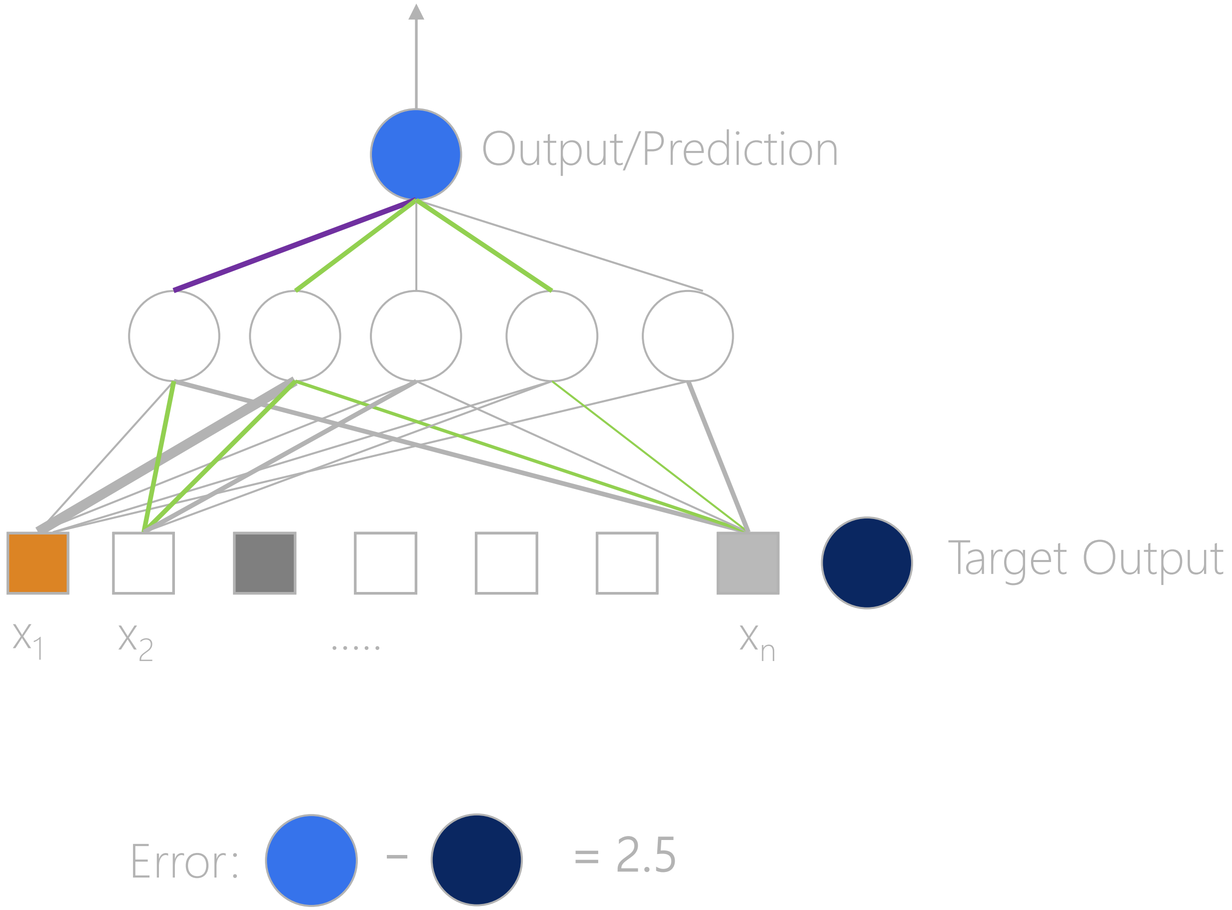
# Learning Principle



# Learning Principle

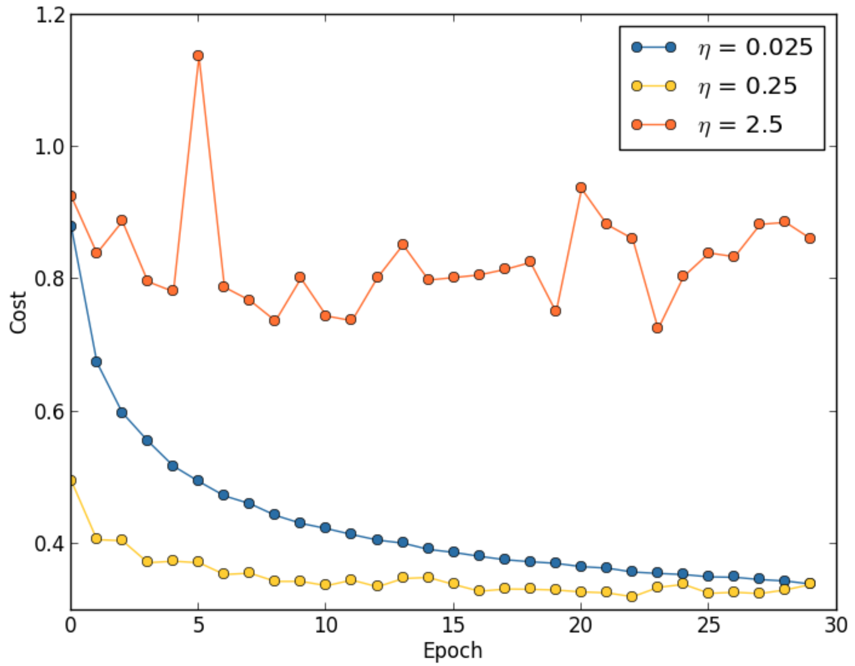


# Learning Principle



# One HyperParameter: Learning Rate

$$w_t = w_{t-1} + \eta \frac{dE}{dw_t}$$



# TECHNIQUES TO IMPROVE MODEL

- More training – GPU Time
- More/better data – Data Science
- Searching Hyperparameters – Learning Design
- Modify the network – Network Architecture – Next Section

# Performance Improvement

Ideas?

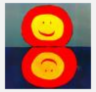
- Increase accuracy and confidence with similar data

- Generalize performance to more diverse data

# More data

## Full dataset ( 10 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 % |

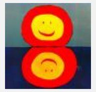
# DATA AUGMENTATION

## 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 ( 10 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 % |

# Break

# Beyond Image Classification

# COMPUTER VISION TASKS

**Image Classification**

**Image Classification + Localization**

**Object Detection**

**Image Segmentation**

(inspired by a slide found in cs231n lecture from Stanford University)

# Inputs and Outputs

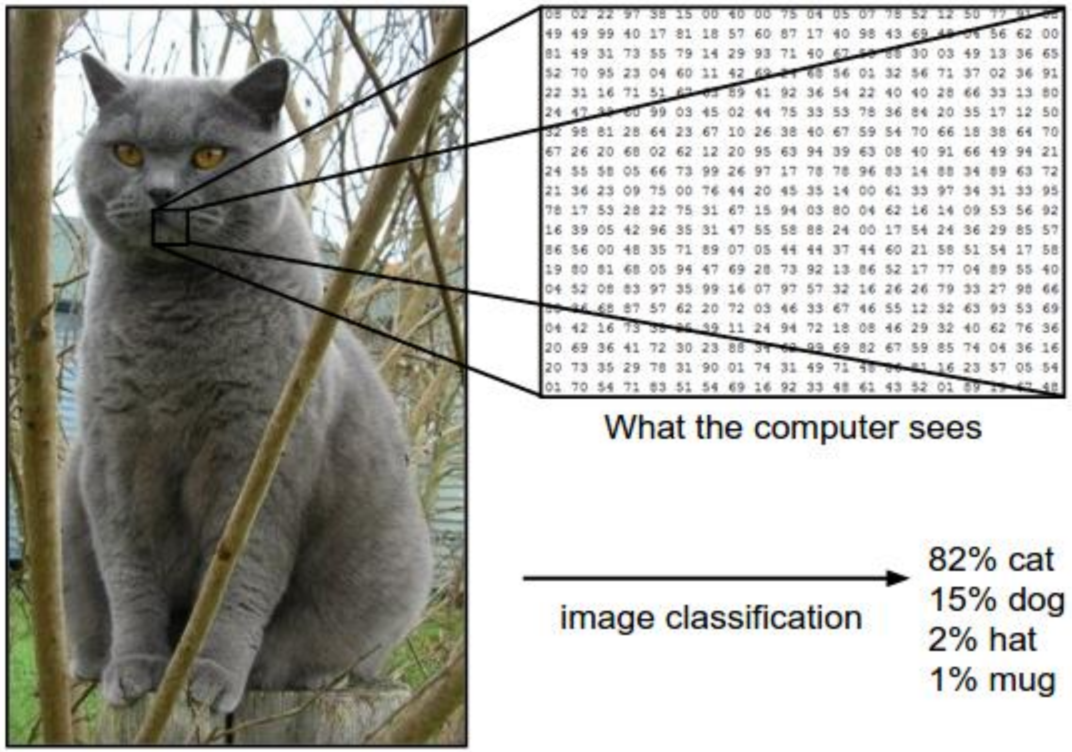



Image from the Stanford CS231 Course

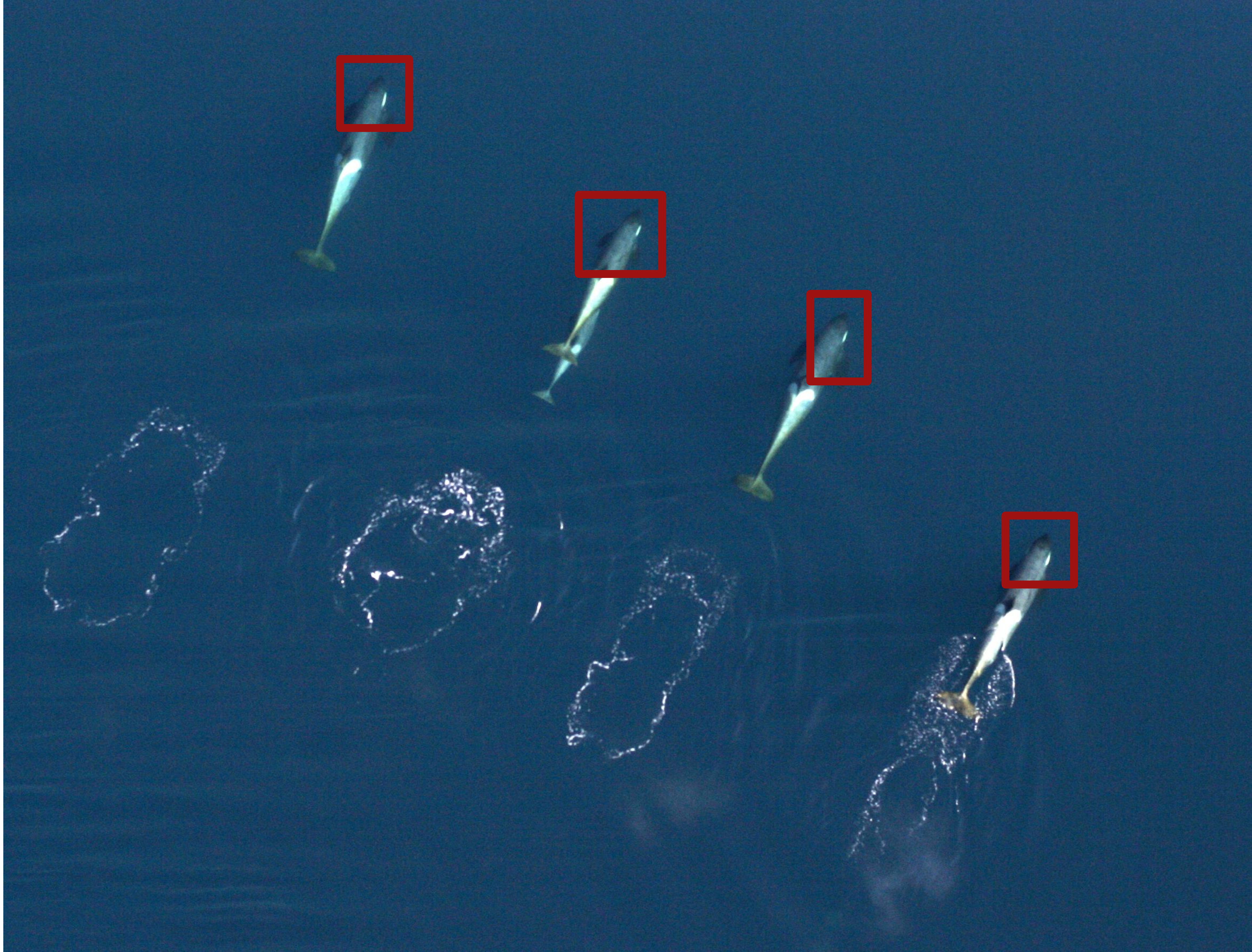
# Inputs and Outputs

| Workflow | Input | Output |
|---|---|---|
| Image Classification | Raw Pixel Values | A vector where each index corresponds with the likelihood or the image of belonging to each class |
| Object Detection | Raw Pixel Values | A vector with (X,Y) pairings for the top-left and bottom-right corner of each object present in the image |
| Image Segmentation | Raw Pixel Values | A overlay of the image for each class being segmented, where each value is the likelihood of that pixel belonging to each class |
| Text Generation | A unique vector for each 'token' (word, letter, etc.) | A vector representing the most likely next 'token'. |
| Image Rendering | Raw Pixel Values of a grainy Image | Raw pixel values of a clean image |

# Object Detection

## Finding a whale face in the ocean.

*We want to know IF there are whale faces in aerial images, and if so, where.*



Next Page:

How can we use what we know about Image Classification to detect whale faces from aerial images?

*Take 2 minutes to think through and write down (paper or computer) ideas.*

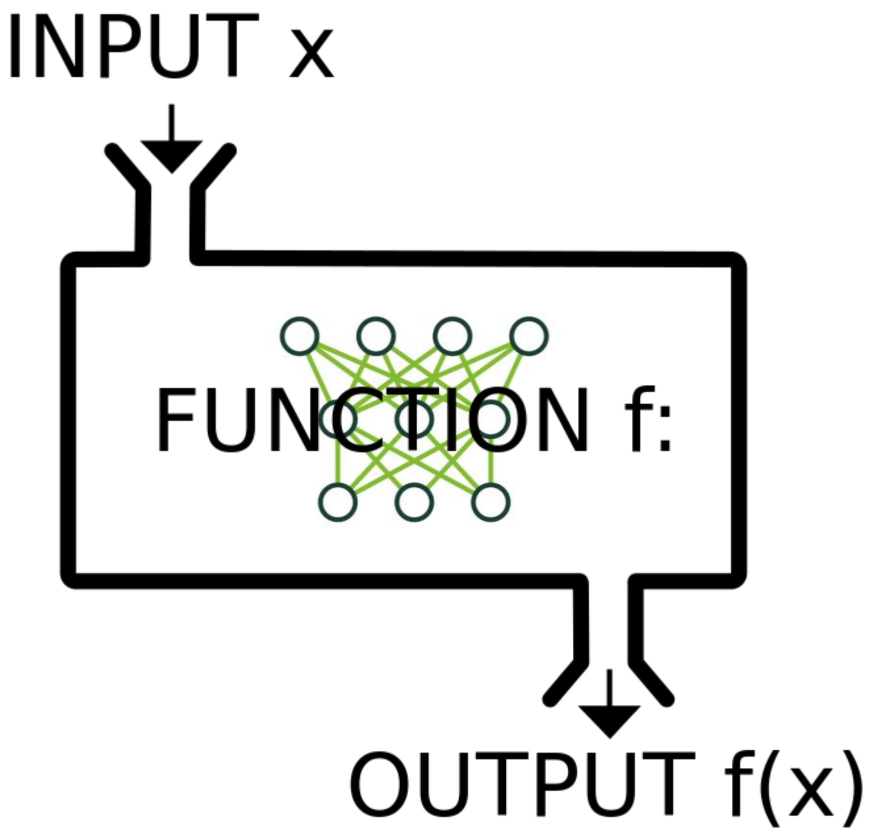
# AI at scale

Solving novel problems with code

Applications that combine trained networks with code can create new capabilities

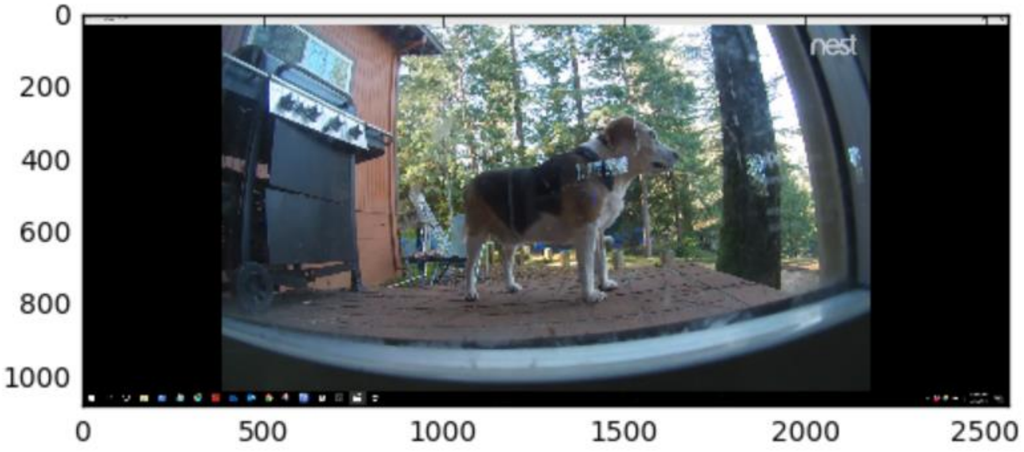
Trained networks play the role of functions

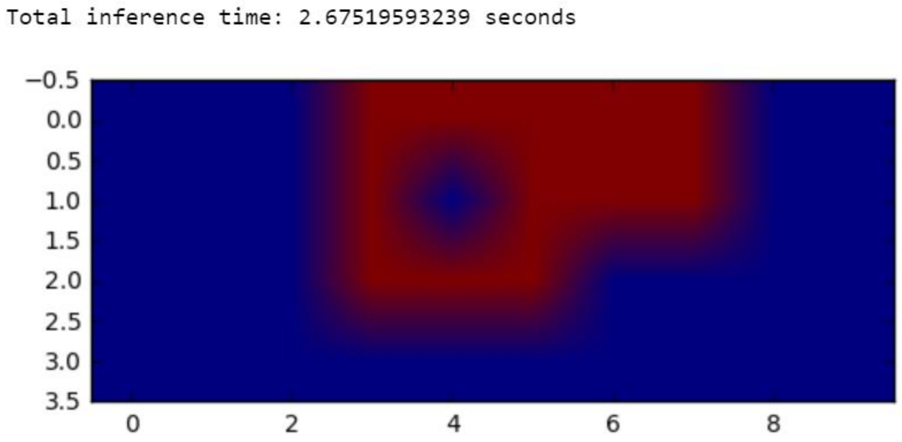
Building applications requires writing code to generate expected inputs and useful outputs



# Approach 1: Sliding Window

- Technique:
  - Build a dog/'not dog' classifier
  - Sliding window python application runs classifier on each 256X256 segment



# Next Page

## Object Detection: GPU Task 5

Bookmark this page

Launch the task below. While working through the how-to, you'll also be learning:

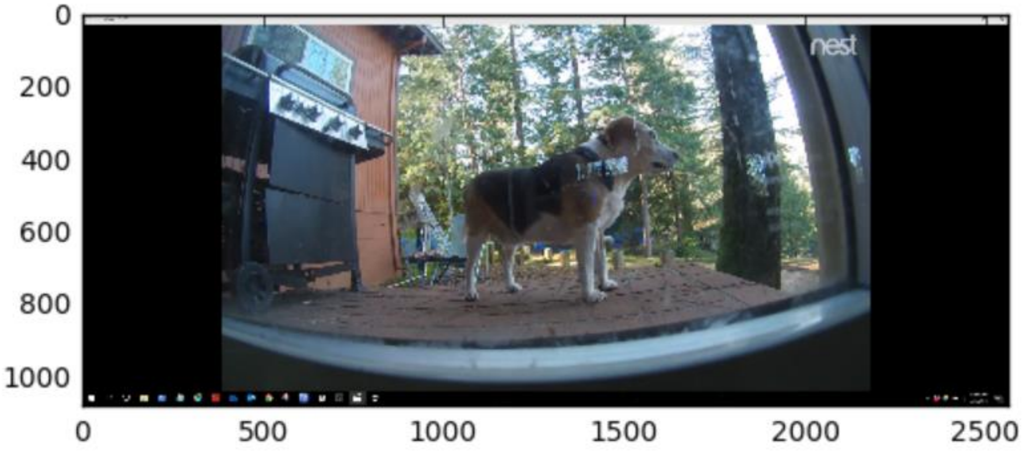
1. How to assess the inputs and outputs of a network
2. How to combine traditional computer vision techniques with deep learning.

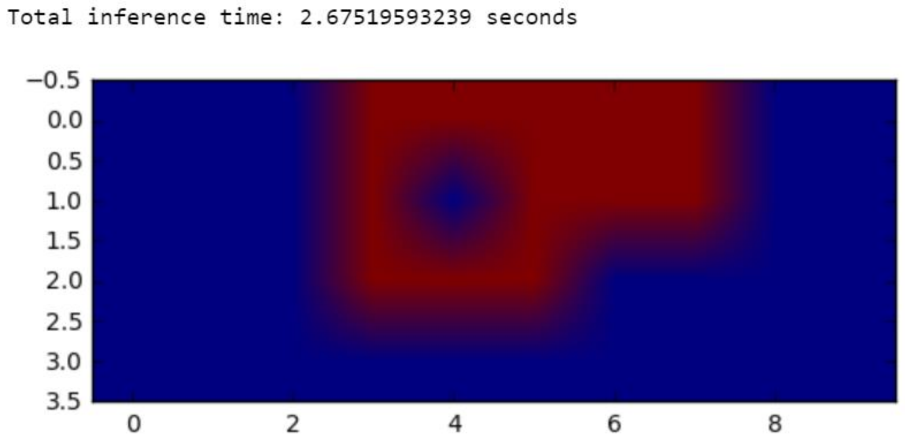
---

Press **Start** to launch GPU task 5.

# Approach 1: Sliding Window

- Works but:
  - Needs human supervision
  - Slow – constrained by image size



# Discuss: Intro to Network architecture

# Approach 2 – Modifying Network Architecture

**Layers** are mathematical operations on tensors (Matrices, vectors, etc.)

Layers are combined to describe the **architecture** of a neural network

Modifications to network architecture impact **capability** and **performance**

Each **framework** has a different syntax for describing architectures

Regardless of framework: The **output** of each layer *must fit* the **input** of the next layer.

# CAFFE FEATURES

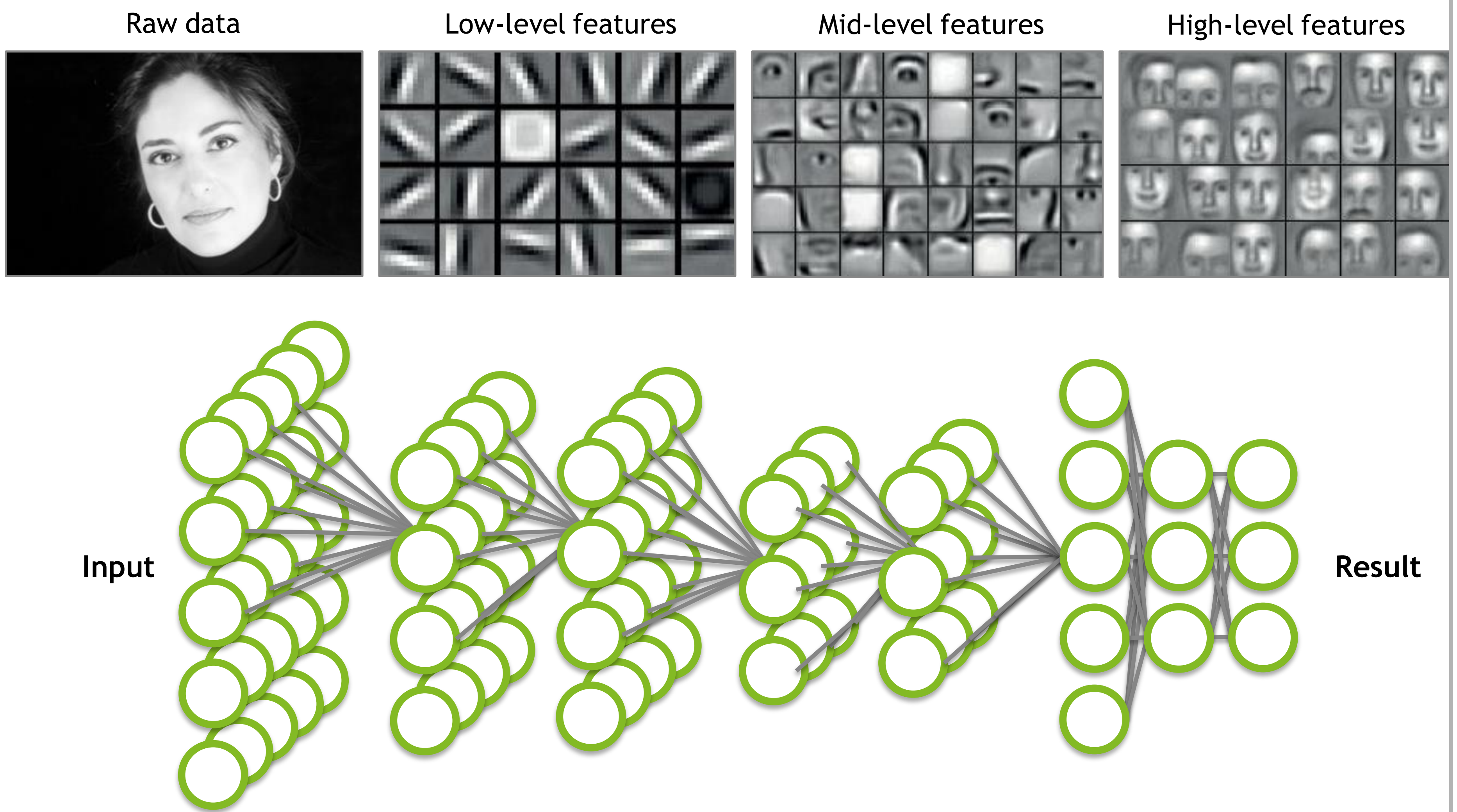
## Deep Learning model definition

Protobuf model format

- Strongly typed format
- Human readable
- Auto-generates and checks Caffe code
- Developed by Google, currently managed by Facebook
- 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: 3
        stride: 1
        weight_filler {
                type: "xavier"
        }
}
```

# Image Classification Network (CNN)




**Application components:**

**Task objective**
e.g. Identify face

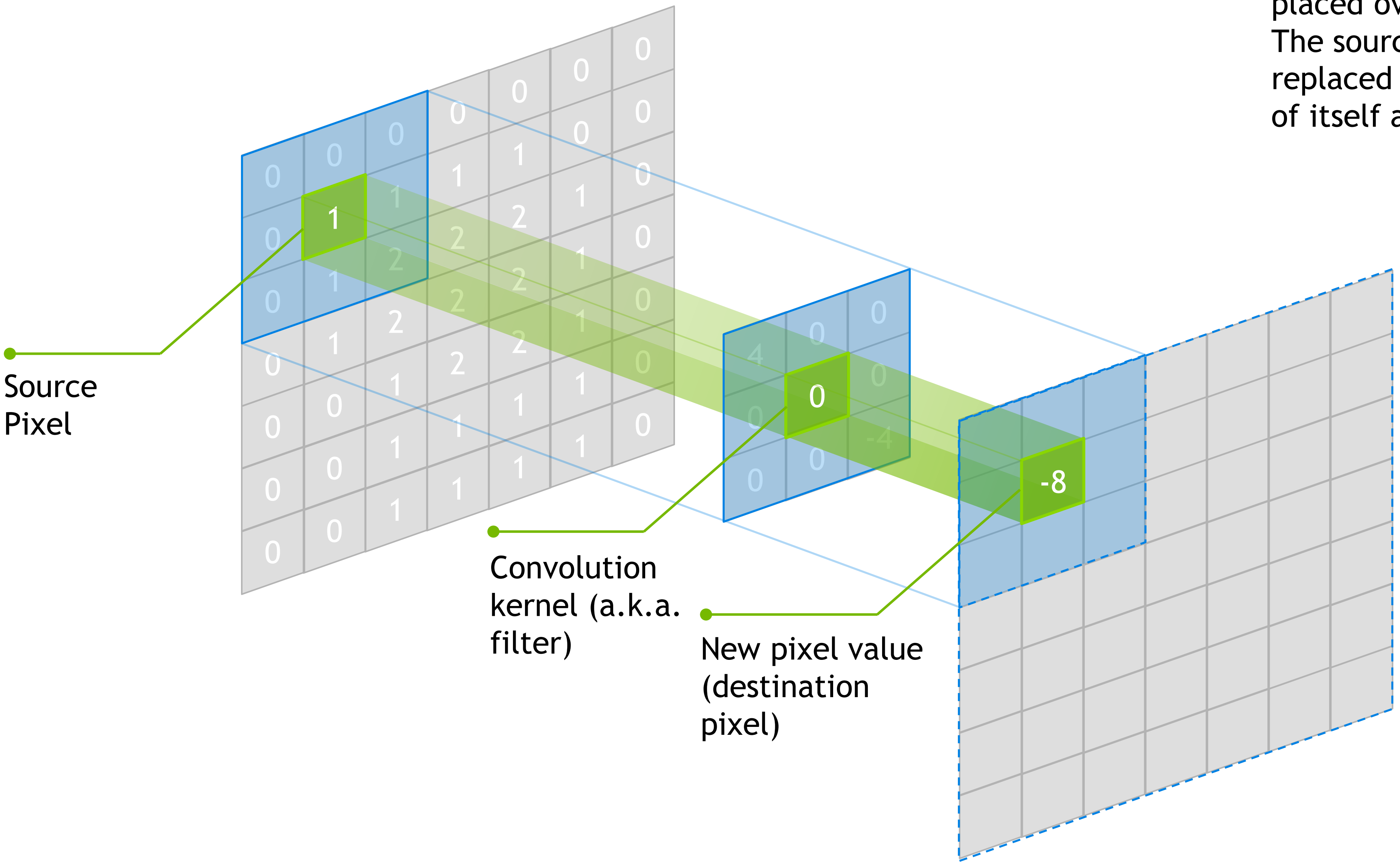
**Training data**
10-100M images

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

**Learning algorithm**
~30 Exaflops
1-30 GPU days

# 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.



# APPROACH 2 - Network Modification

- Approach :
  - Change the structure of the network
  - FC = Fully Connected = Matrix Multiplication = Size Constraint

```
}
layer {
  name: "pool5"
  type: "Pooling"
  bottom: "conv5"
  top: "pool5"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "fc6"
  type: "InnerProduct"
  bottom: "pool5"
  top: "fc6"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 4096
    weight_filler {
      type: "gaussian"
      std: 0.005
    }
    bias_filler {
      type: "constant"
      value: 0.1
    }
  }
}
layer {
  name: "relu6"
  type: "ReLU"
  bottom: "fc6"
  top: "fc6"
}
```

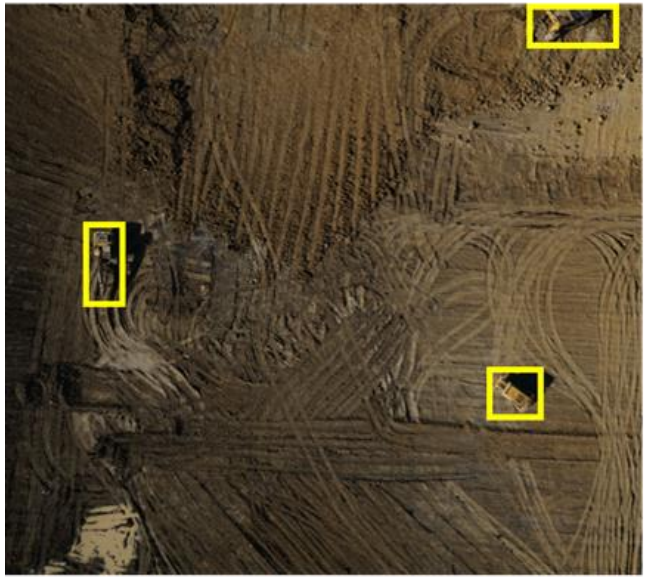
```
layer {
  name: "conv6"
  type: "Convolution"
  bottom: "pool5"
  top: "conv6"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 4096
    pad: 0
    kernel_size: 6
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0.1
    }
  }
}
layer {
  name: "relu6"
  type: "ReLU"
  bottom: "conv6"
  top: "conv6"
}
```

# Back to lab

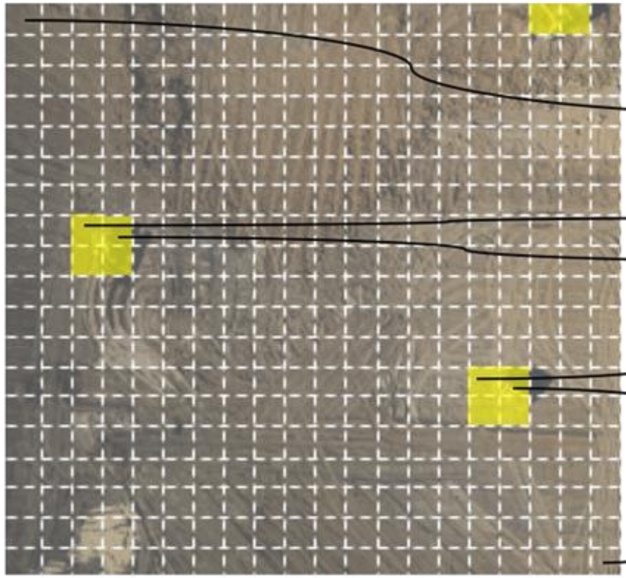
- Replace layers by reading carefully
- Ask for help if you need
- Continue through end of lab
- We’ll discuss “Detectnet” post-lab

# Approach 3: End-to-End Solution

## Need dataset with inputs and corresponding (often complex) output



Training image with bounding box annotations



Bounding boxes mapped to grid squares

Bounding box coordinates in pixels relative to center of grid square

| class | $x_1$ | $y_1$ | $x_2$ | $y_2$ | coverage |
|---|---|---|---|---|---|
| dontcare | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... |
| digger | -2 | -8 | 18 | 24 | 1 |
| digger | -18 | -8 | 2 | 24 | 1 |
| ... | ... | ... | ... | ... | ... |
| digger | -6 | -8 | 22 | 24 | 1 |
| digger | -24 | -8 | 8 | 24 | 1 |
| ... | ... | ... | ... | ... | ... |
| dontcare | 0 | 0 | 0 | 0 | 0 |

DetectNet input data representation

# Approach 3 – End to end solution

High-performing neural network architectures require deep **experimentation**

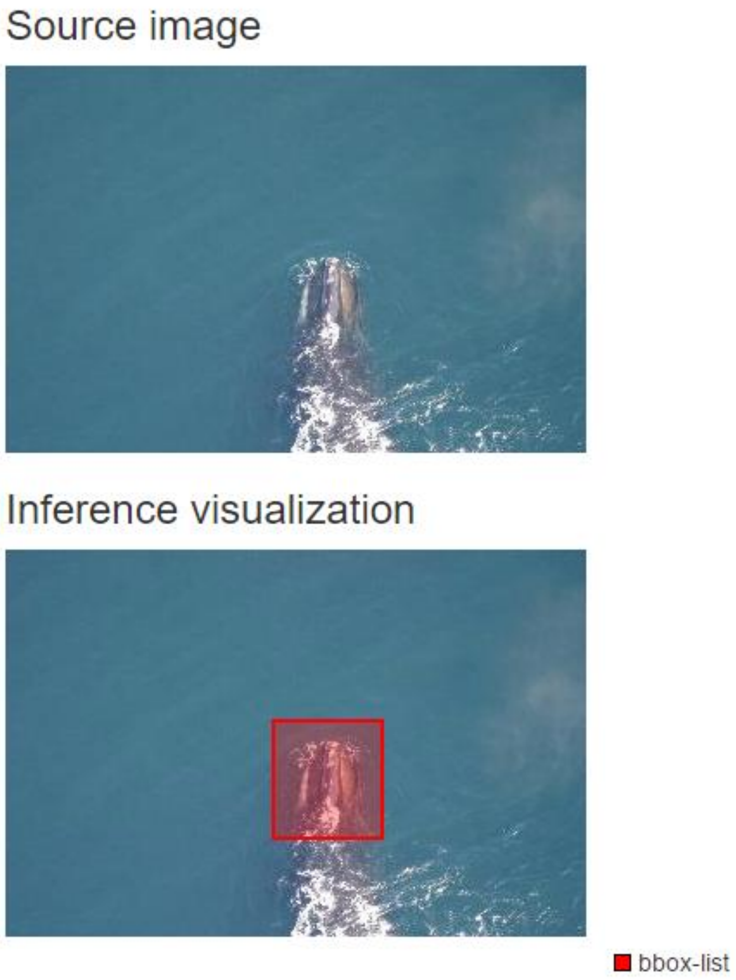
You can benefit from the work of the **community** through the **modelzoo** of each framework

Implementing a new network requires an understanding of data and training **expectations.**

Find projects **similar to your project** as starting points.

# Approach 3: End-to-End Solution

- DetectNet:
  - Like AlexNet, DetectNet is optimized for a certain type of learning, but is generalizable to multiple contexts.



# Closing thoughts - Creating new functionality

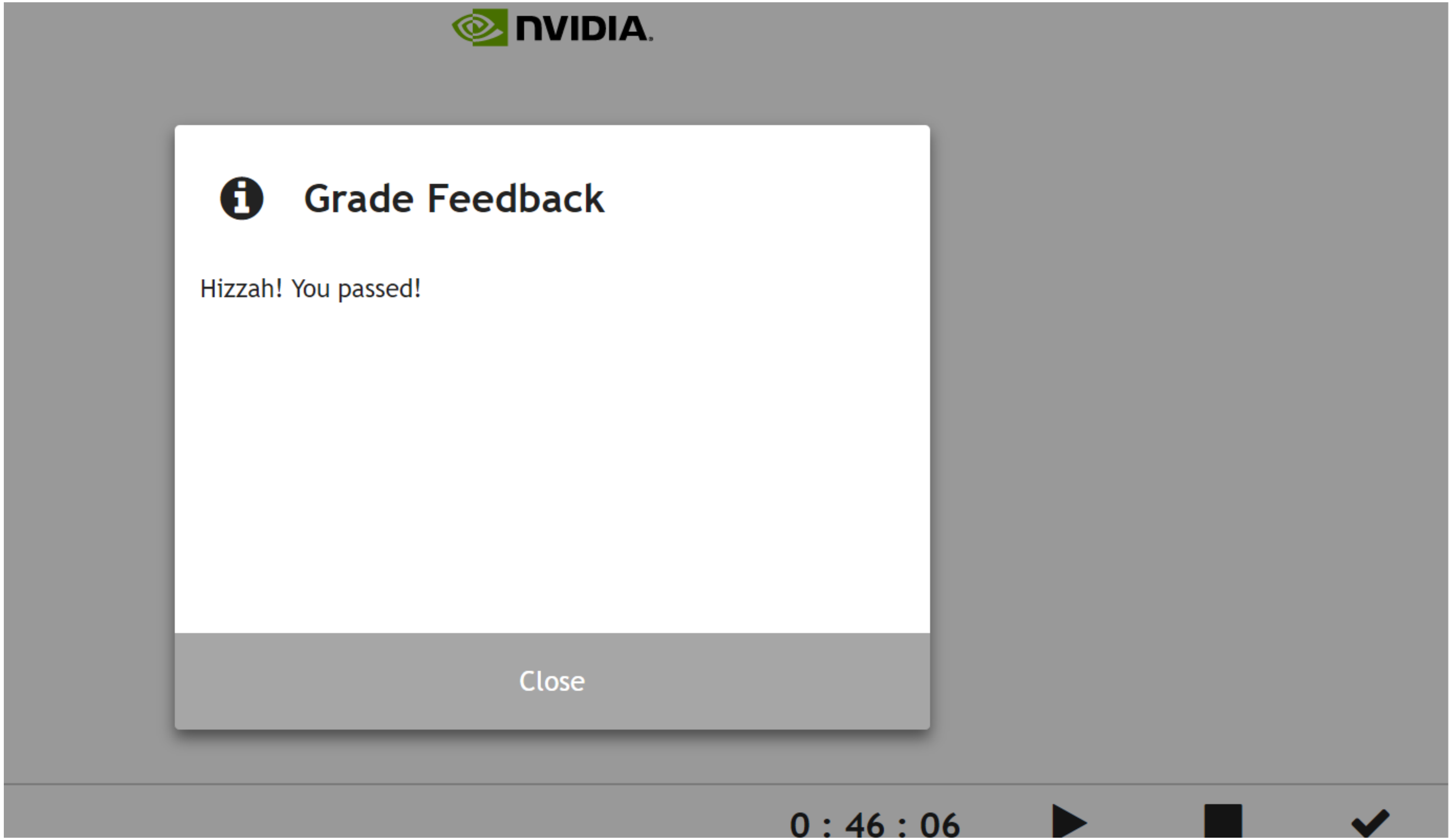
- Approach 1: Combining DL with programming
  - Scaling models programmatically to create new functionality
- Approach 2: Experiment with network architecture
  - Study the math of neural networks to create new functionality
- Approach 3: Identify similar solutions
  - Study existing solutions to implement new functionality

In [3]:
```
!python submission.py '/dli/data/whale/data/train/not_face/w_1.jpg' #This should return "not whale" at the very bottom of the out
```

```
libdc1394 error: Failed to initialize libdc1394
WARNING: Logging before InitGoogleLogging() is written to STDERR
I0324 16:49:49.428824   169 upgrade_proto.cpp:66] Attempting to upgrade input file specified using deprecated input fields:
/dli/data/digits/20180324-164306-6f7d/deploy.prototxt
I0324 16:49:49.428902   169 upgrade_proto.cpp:69] Successfully upgraded file specified using deprecated input fields.
W0324 16:49:49.428910   169 upgrade_proto.cpp:71] Note that future Caffe releases will only support input layers and not inp
ut fields.
I0324 16:49:49.429174   169 net.cpp:52] Initializing net from parameters:
state {
  phase: TEST
}
layer {
  name: "input"
  type: "Input"
  top: "data"
  input_param {
    shape {
      dim: 1
      dim: 3
      dim: 227
```

In [3]:
```
!python submission.py '/dli/data/whale/data/train/not_face/w_1.jpg' #This should return "not whale" at the very bottom of the out
```

```
libdc1394 error: Failed to initialize libdc1394
WARNING: Logging before InitGoogleLogging() is written to STDERR
I0324 16:49:49.428824   169 upgrade_proto.cpp:66] Attempting to upgrade input file specified using deprecated input fields:
/dli/data/digits/20180324-164306-6f7d/deploy.prototxt
I0324 16:49:49.428902   169 upgrade_proto.cpp:69] Successfully upgraded file specified using deprecated input fields.
W0324 16:49:49.428910   169 upgrade_proto.cpp:71] Note that future Caffe releases will only support input layers and not inp
ut fields.
I0324 16:49:49.429174   169 net.cpp:52] Initializing net from parameters:
state {
  phase: TEST
}
layer {
  name: "input"
  type: "Input"
  top: "data"
  input_param {
    shape {
      dim: 1
      dim: 3
      dim: 227
```


NVIDIA
Grade Feedback
Hizzah! You passed!
Close
0 : 46 : 06

Instructor: [ ]

[email]

DEEP
LEARNING
INSTITUTE

www.nvidia.com/dli