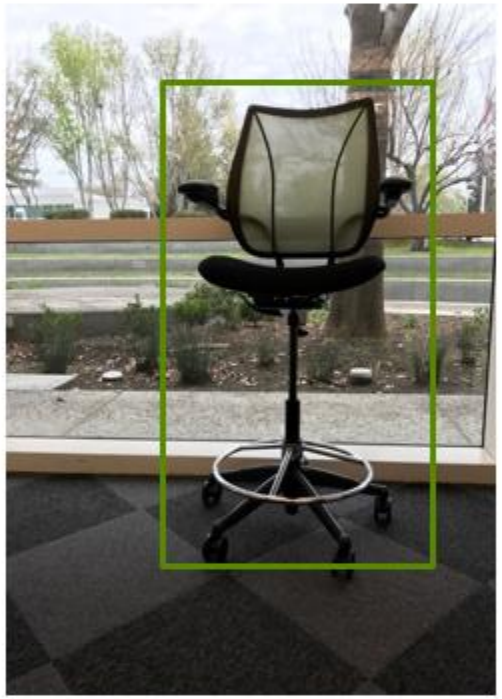
Beyond Image Classification

COMPUTER VISION TASKS

Image Classification

Image Classification + Localization



Object Detection

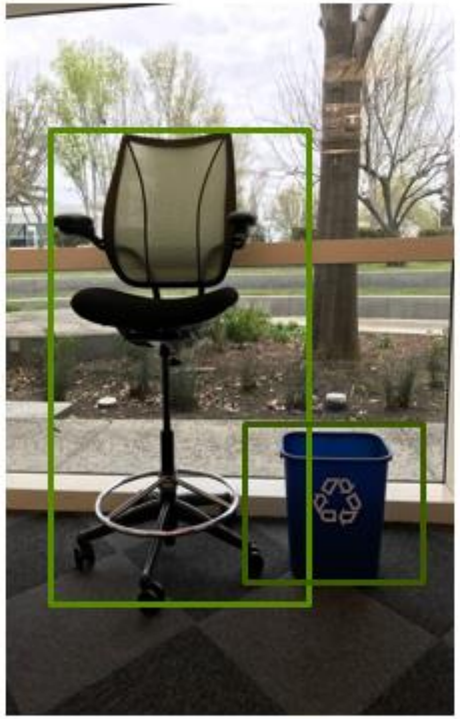
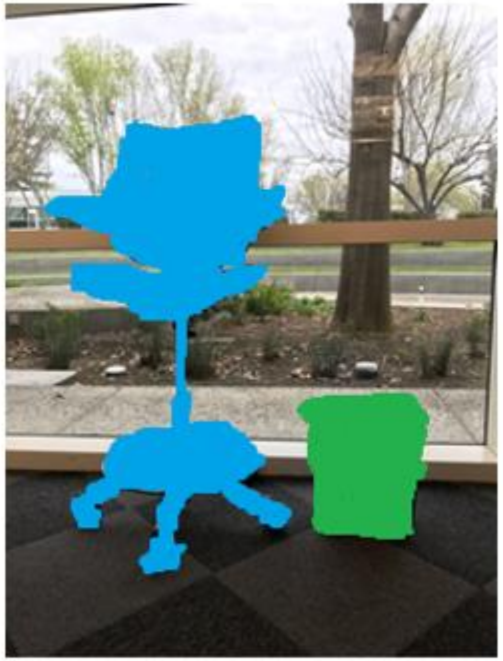
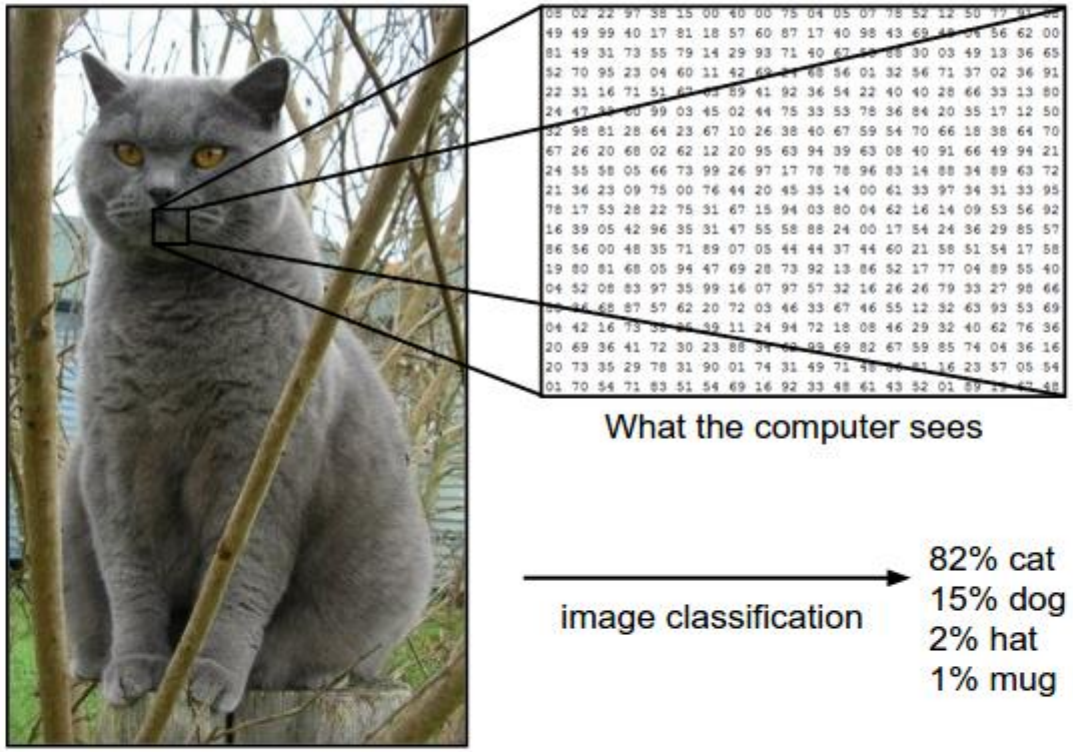


Image Segmentation



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

Inputs and Outputs



```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

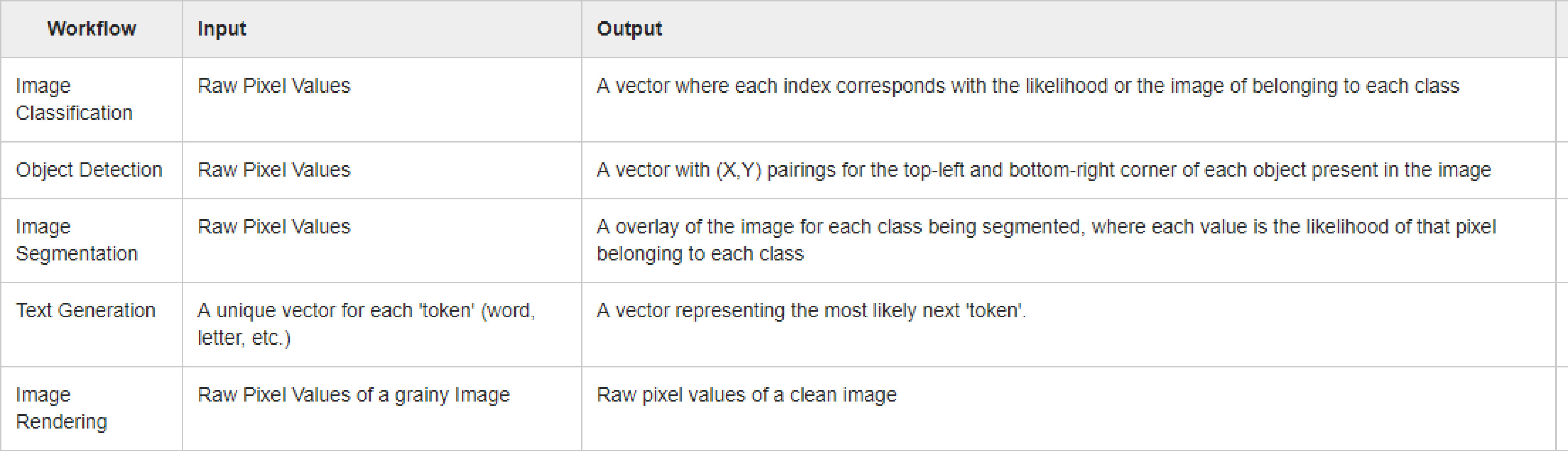
What the computer sees

image classification

82% cat
15% dog
2% hat
1% mug

Image from the Stanford CS231 Course

Inputs and Outputs

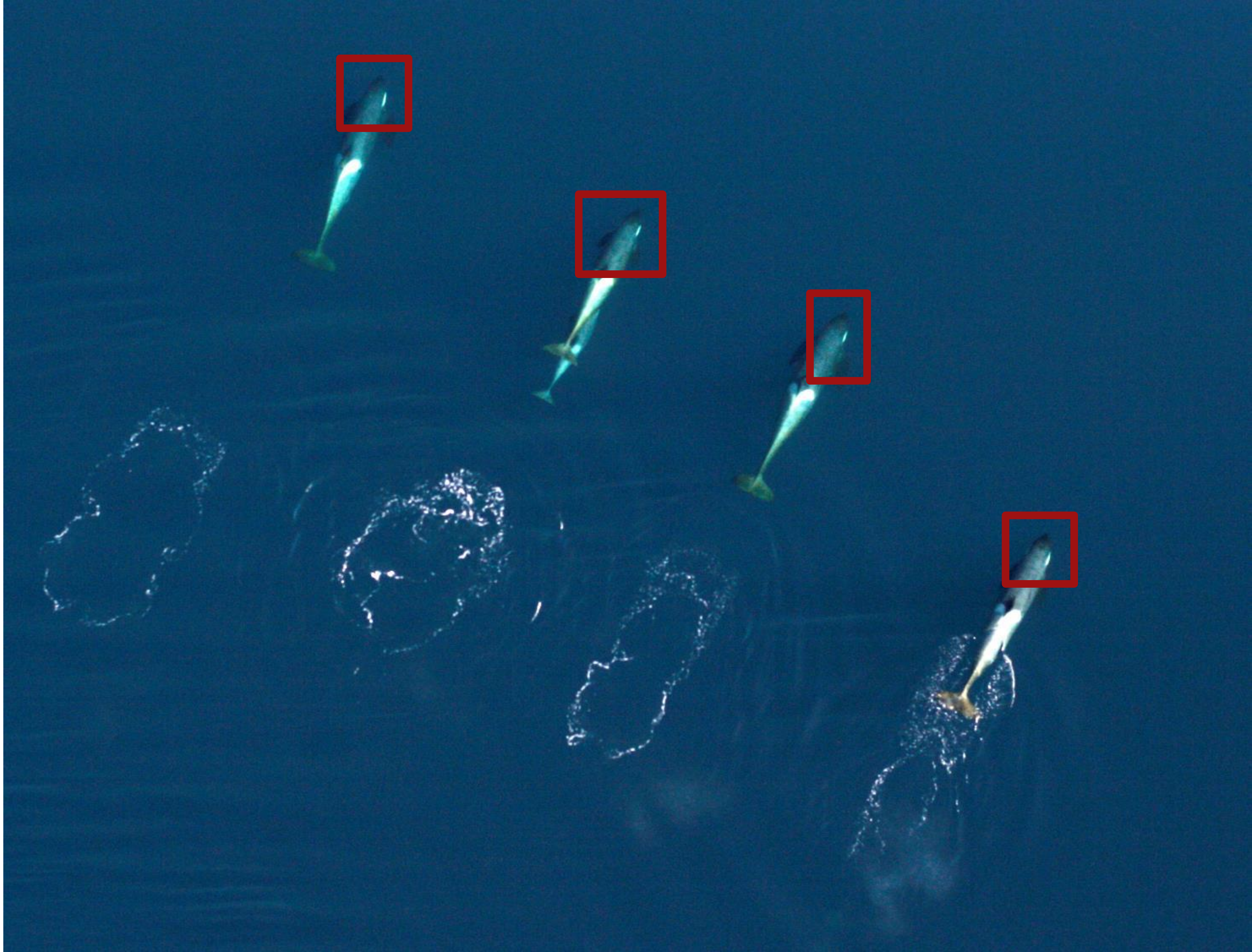


Workflow	Input	Output
Image Classification	Raw Pixel Values	A vector where each index corresponds with the likelihood or the image of belonging to each class
Object Detection	Raw Pixel Values	A vector with (X,Y) pairings for the top-left and bottom-right corner of each object present in the image
Image 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.

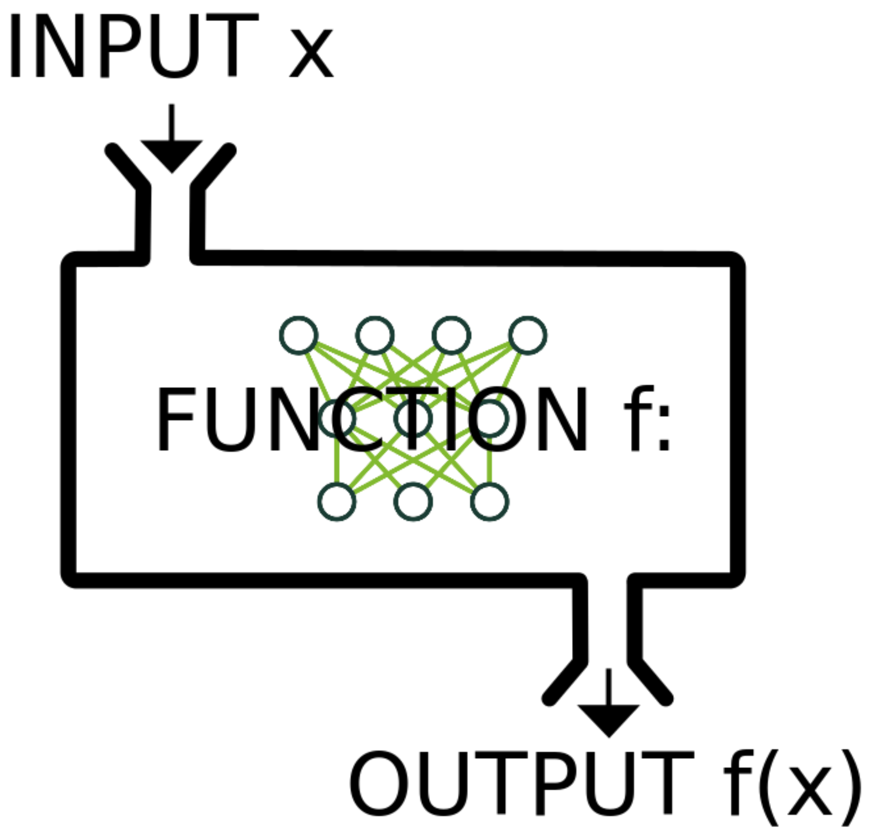
AI at scale

Solving novel problems with code

Applications that combine trained networks with code can create new capabilities

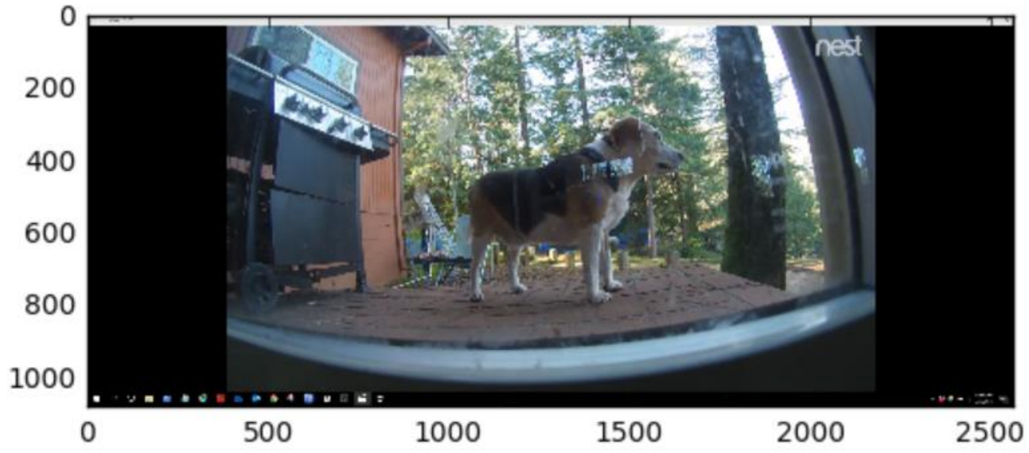
Trained networks play the role of functions

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

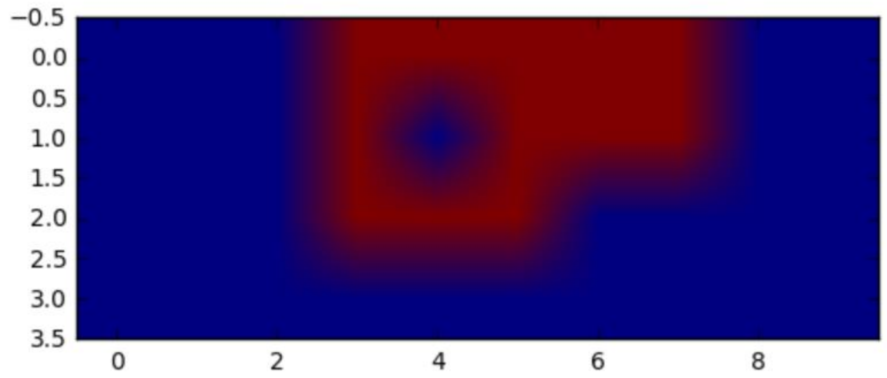


Approach 1: Sliding Window

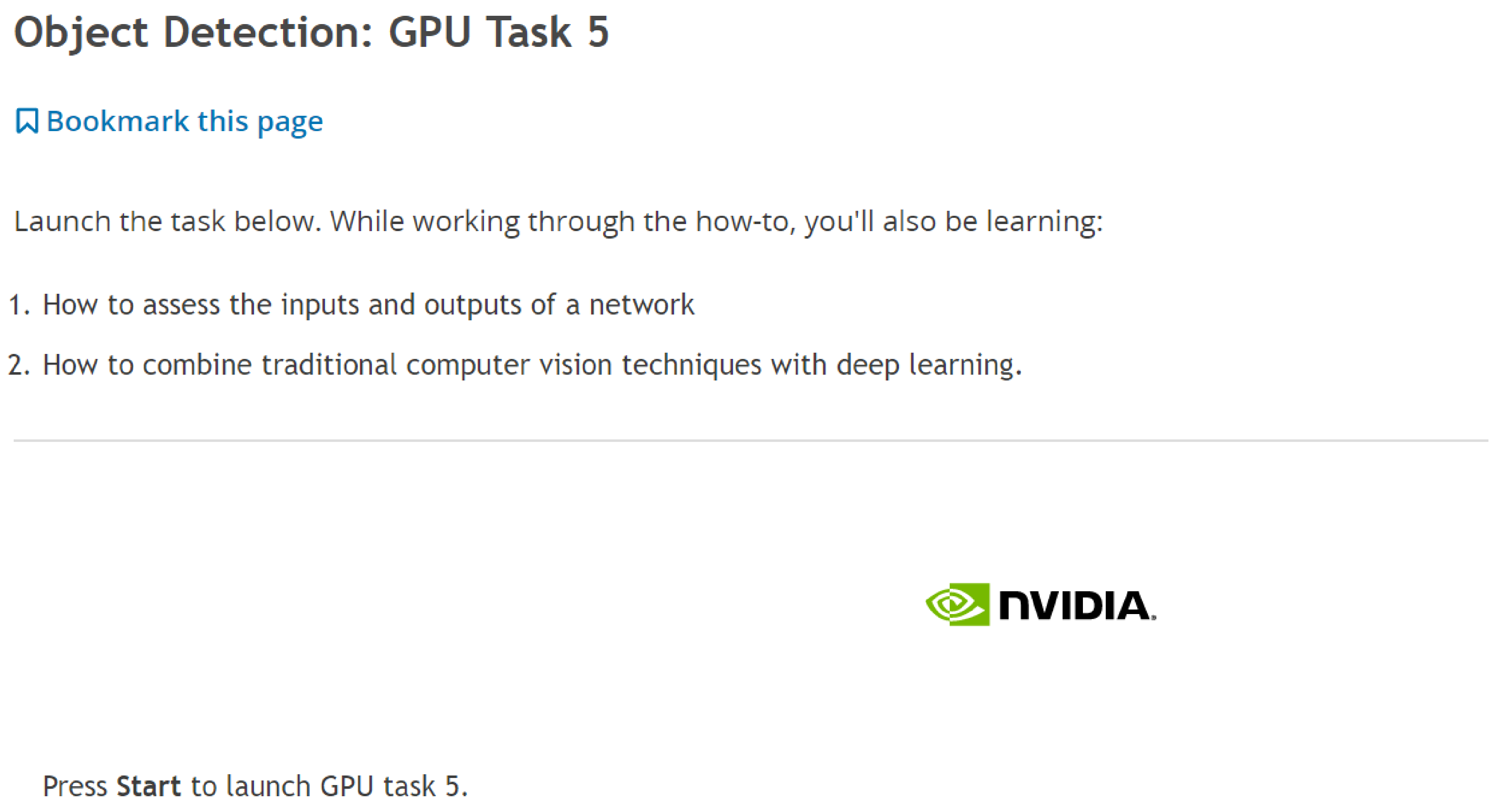
- Technique:
  - Build a dog/’not dog’ classifier
  - Sliding window python application runs classifier on each 256X256 segment



```
Total inference time: 2.67519593239 seconds
```



Next Page



Object Detection: GPU Task 5

Bookmark this page

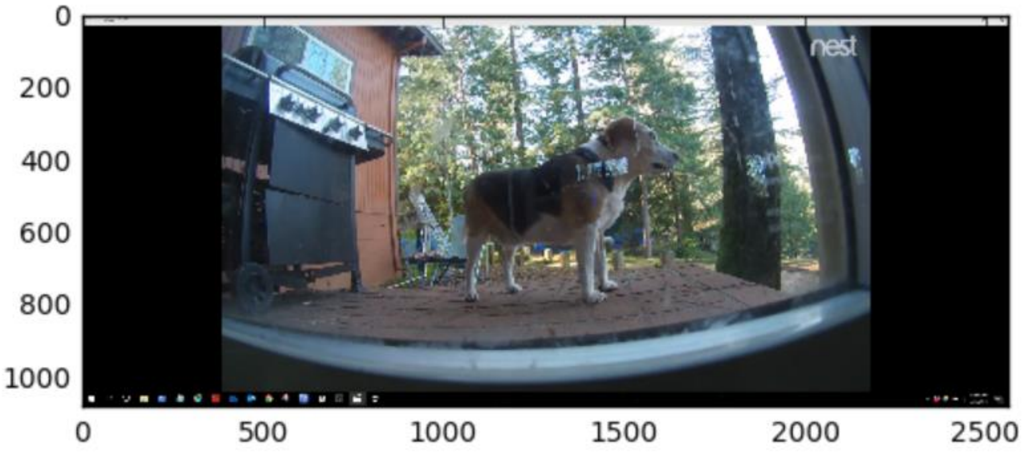
Launch the task below. While working through the how-to, you'll also be learning:

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

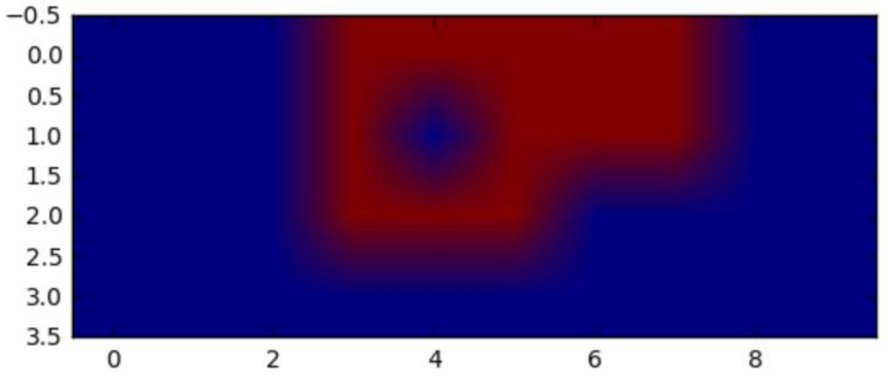
Press **Start** to launch GPU task 5.

Approach 1: Sliding Window

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



```
Total inference time: 2.67519593239 seconds
```



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

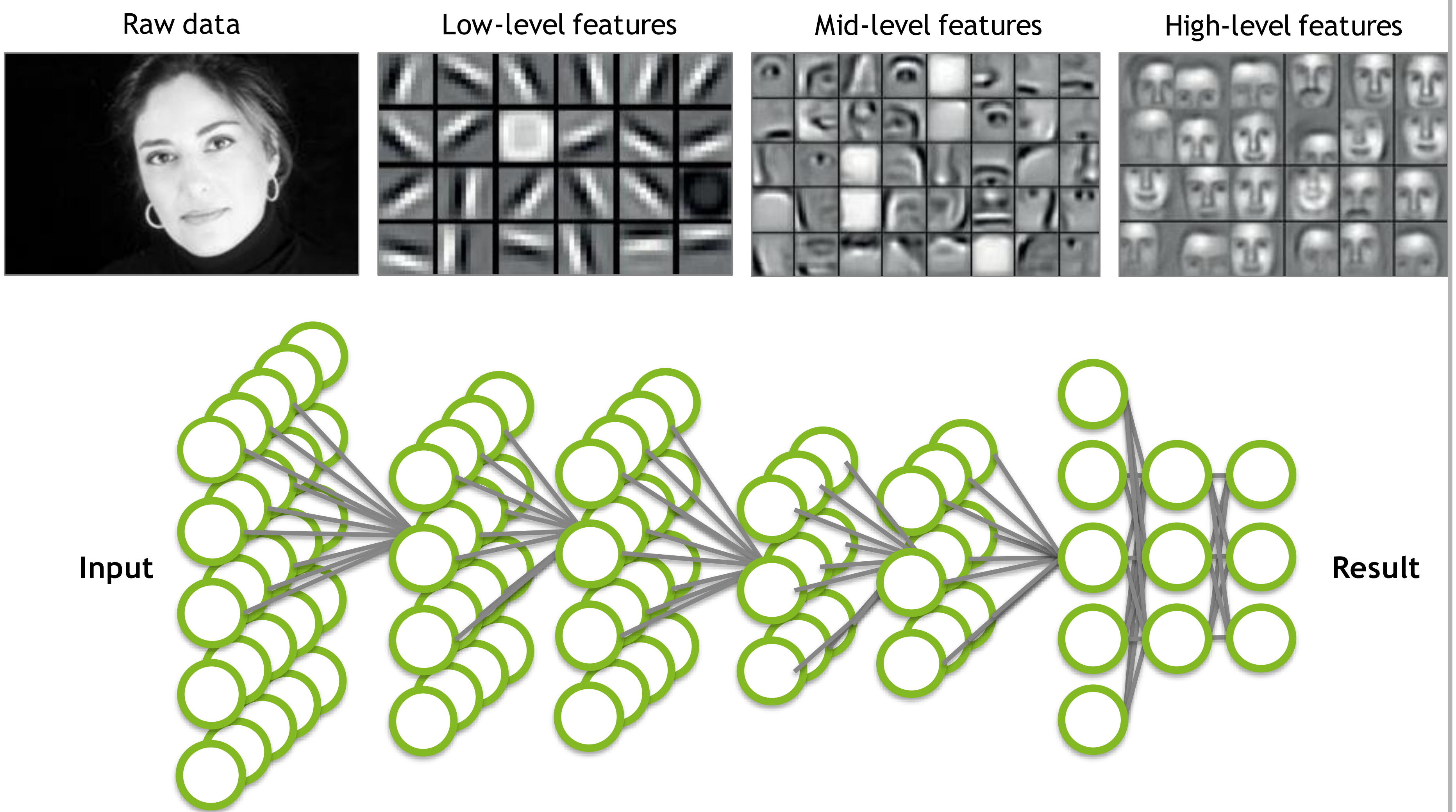
Deep Learning model definition

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

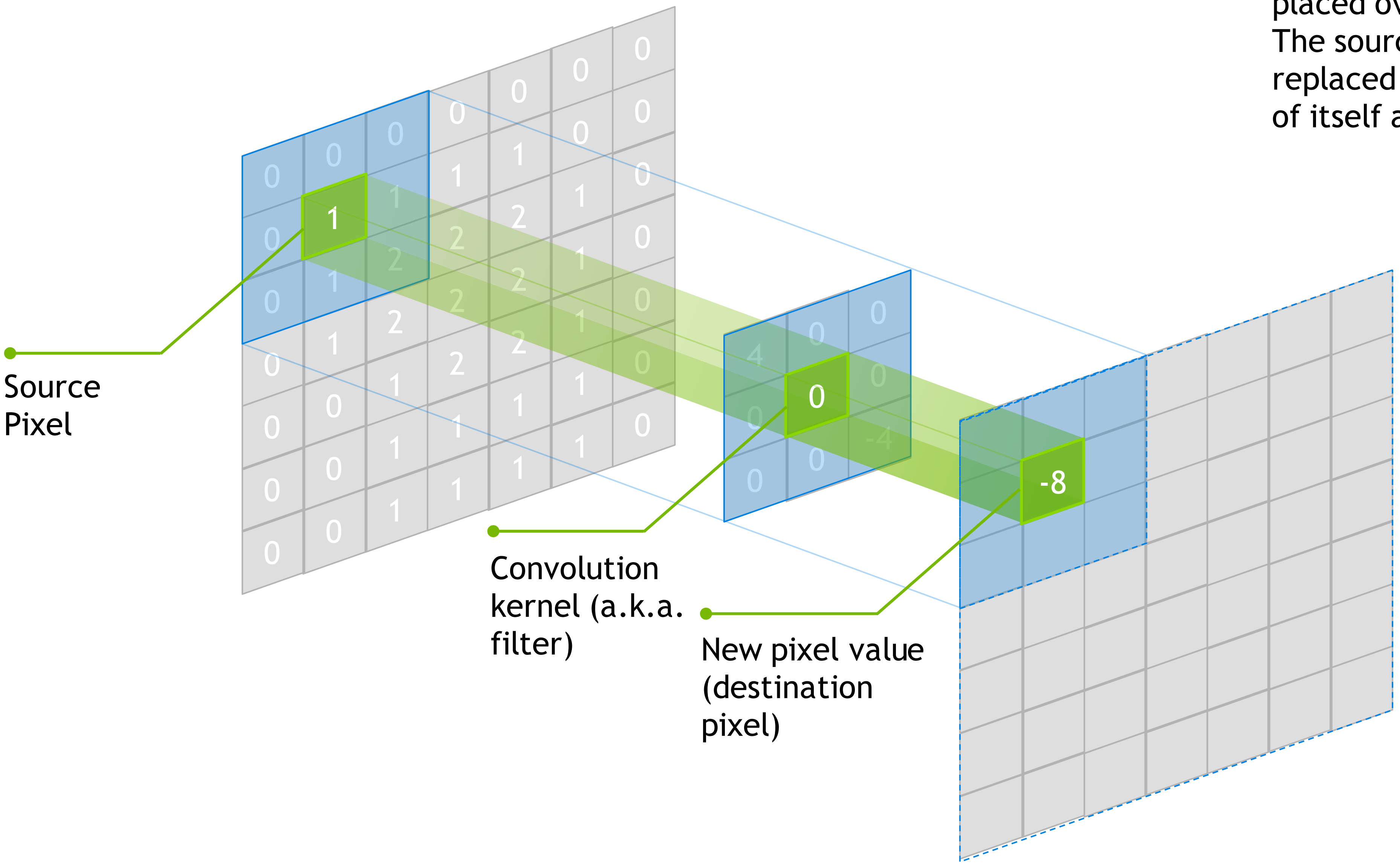
Training data
10-100M images

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.



0	0	0	0	0	0	0
0	1	1	1	1	0	0
0	1	2	2	2	1	0
0	1	2	2	2	1	0
0	0	1	2	2	1	0
0	0	1	1	1	1	0
0	0	1	1	1	1	0

4	0	0
0	0	0
0	0	-4

-8

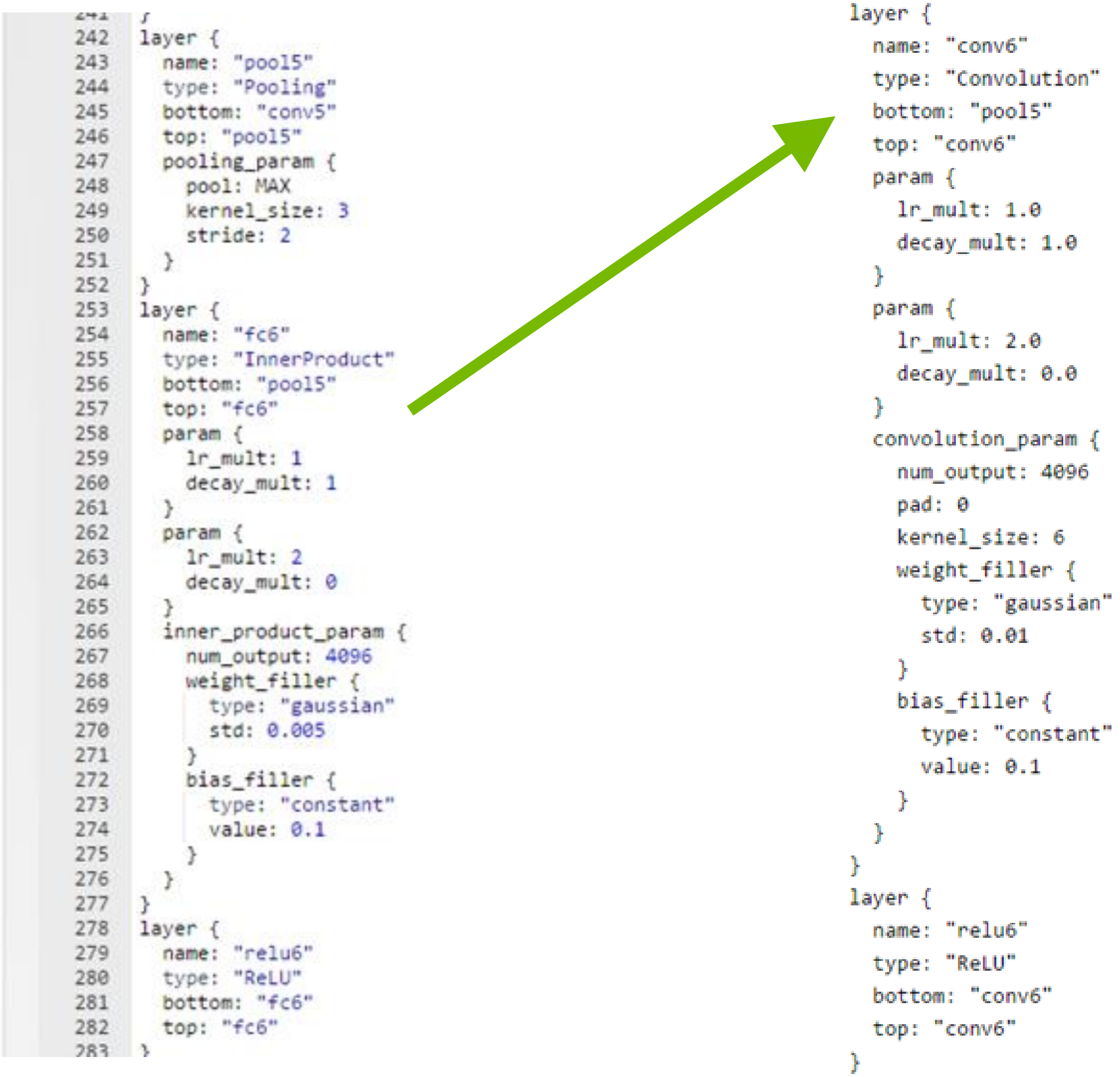
Source Pixel

Convolution kernel (a.k.a. filter)

New pixel value (destination pixel)

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"
}
```

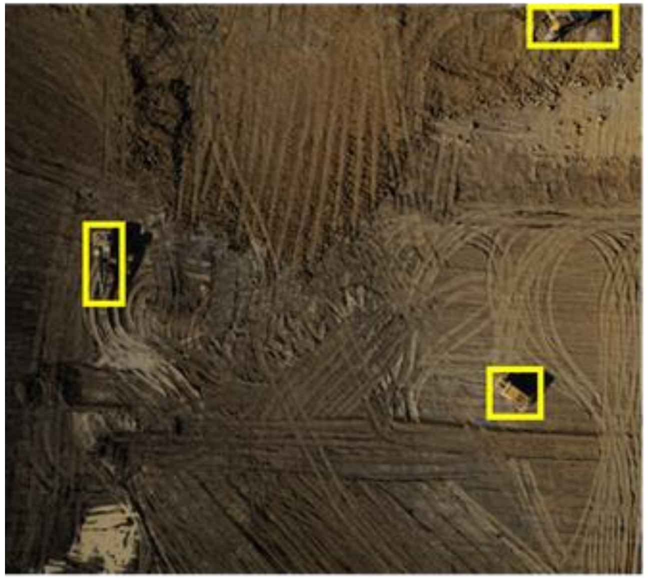
```
layer {
  name: "conv6"
  type: "Convolution"
  bottom: "pool5"
  top: "conv6"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 4096
    pad: 0
    kernel_size: 6
    weight_filler {
      type: "gaussian"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

Back to lab

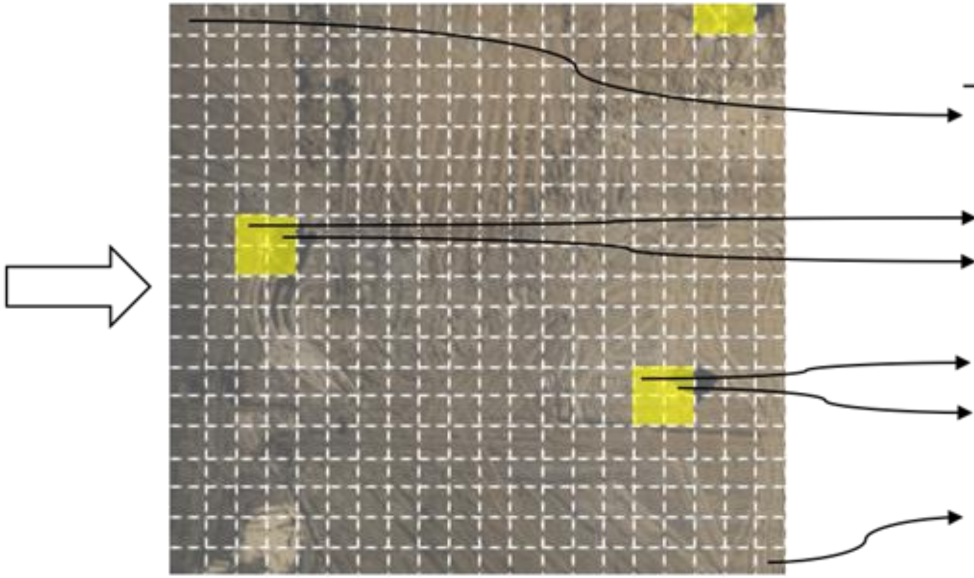
- Replace layers by reading carefully
- Ask for help if you need
- Continue through end of lab
- We’ll discuss “Detectnet” post-lab

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**

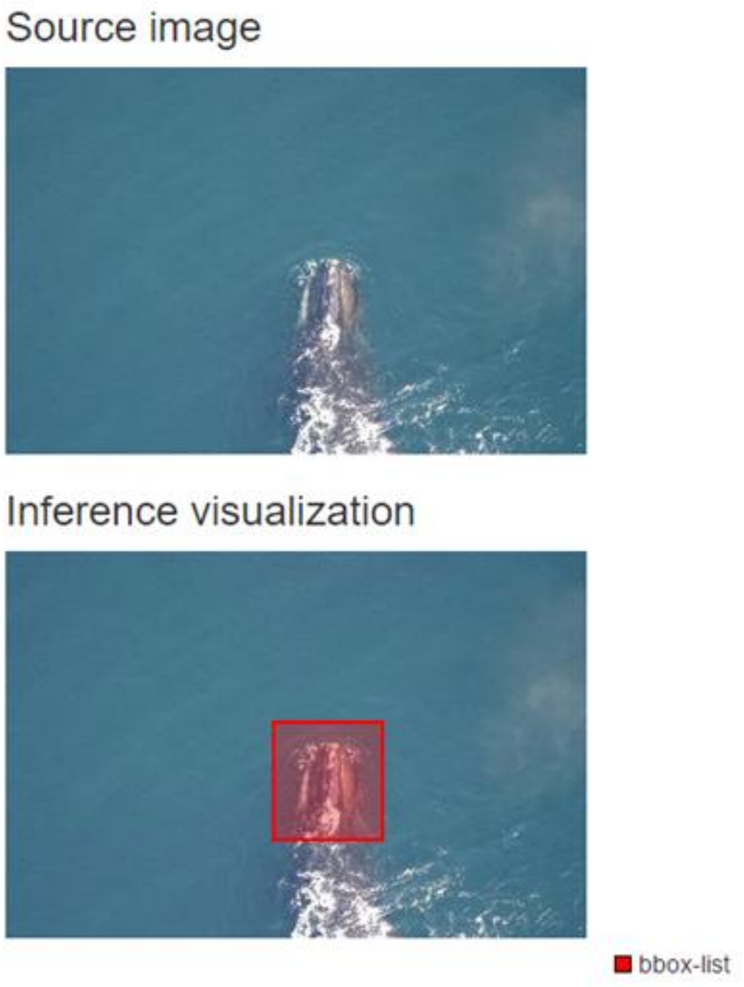
You can benefit from the work of the **community** through the **modelzoo** of each framework

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

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

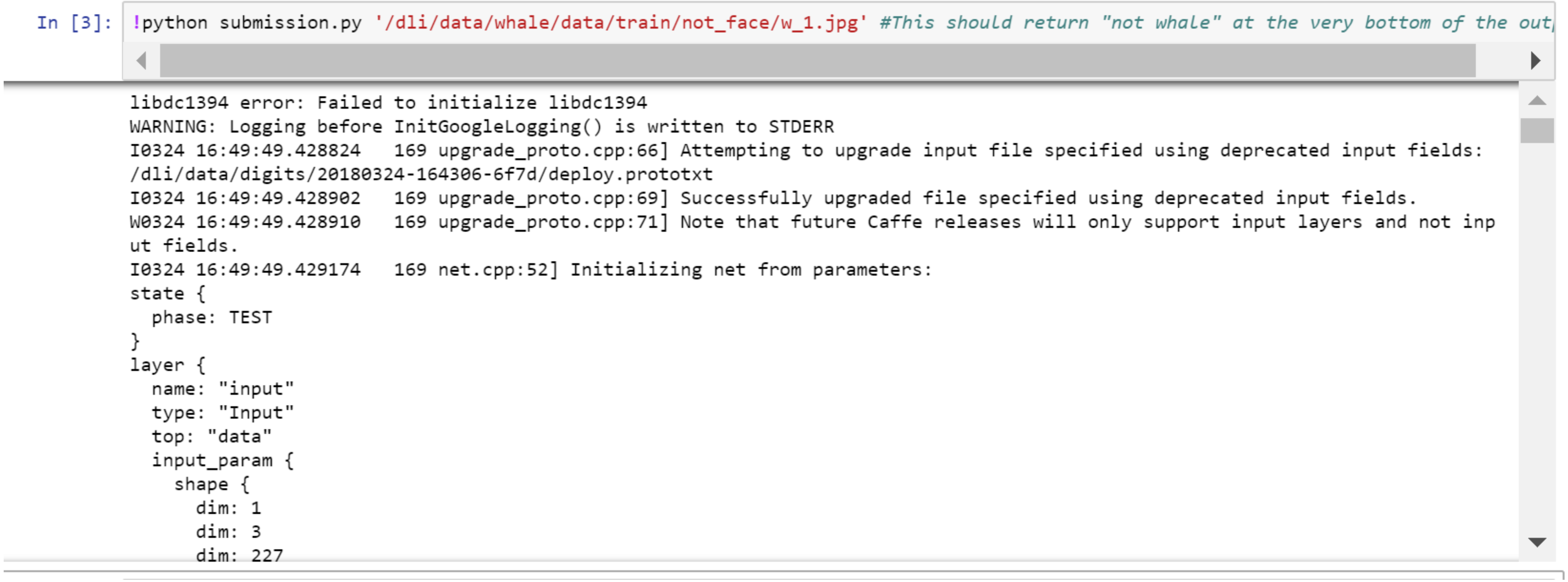
Approach 3: End-to-End Solution

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



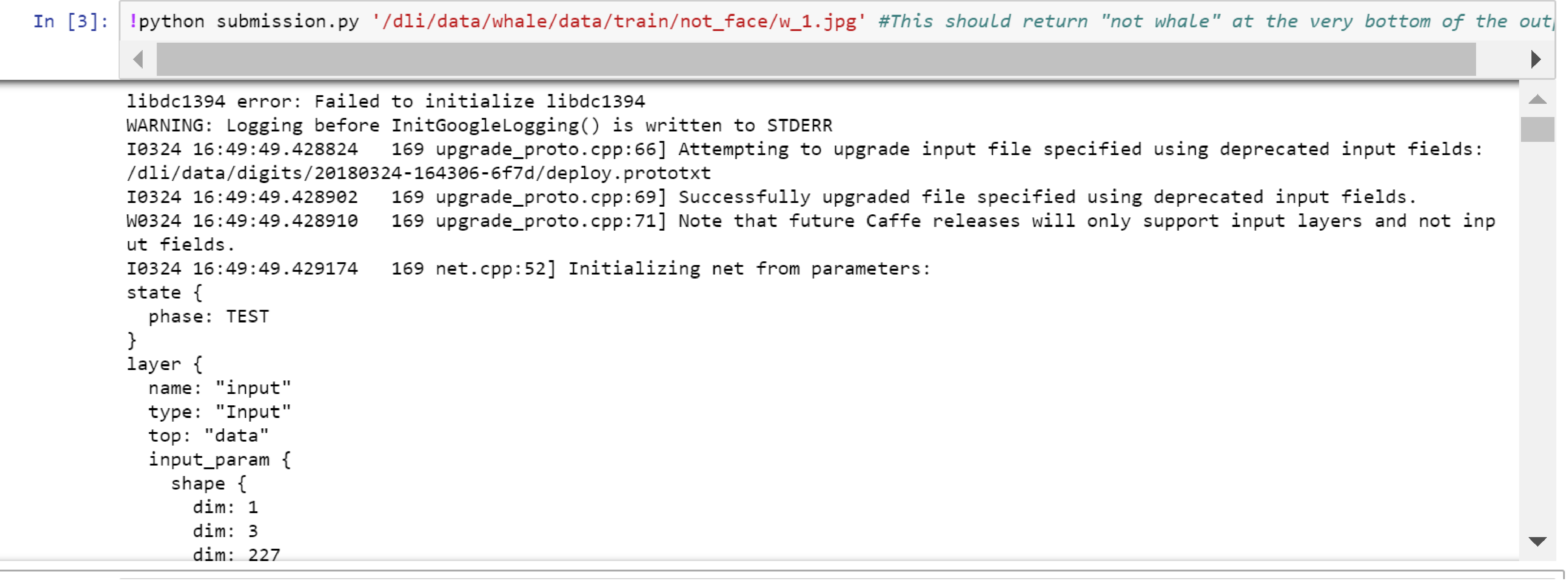
Closing thoughts – Creating new functionality

- 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



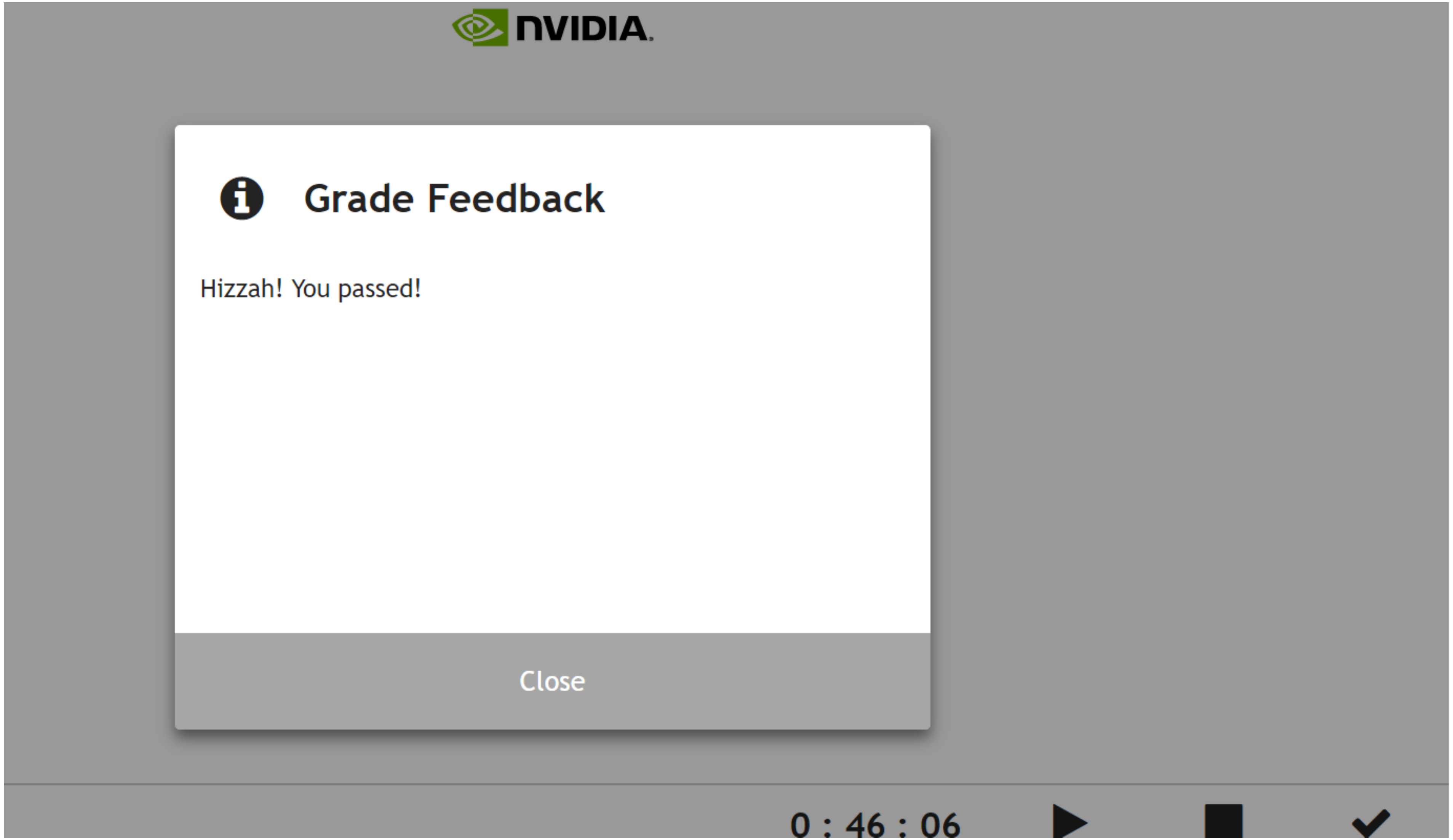
```
In [3]: !python submission.py '/dli/data/whale/data/train/not_face/w_1.jpg' #This should return "not whale" at the very bottom of the out
```

```
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
```



Grade Feedback

Hizzah! You passed!

NVIDIA

DEEP
LEARNING
INSTITUTE

www.nvidia.com/dli