

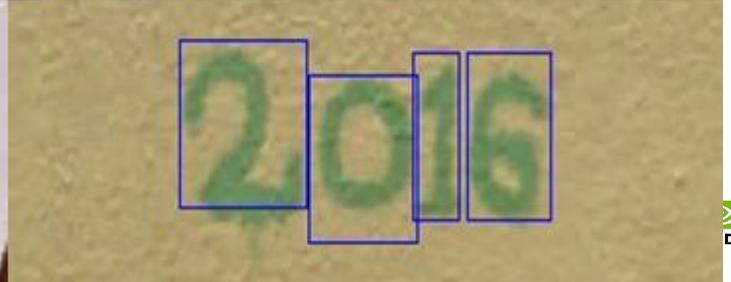
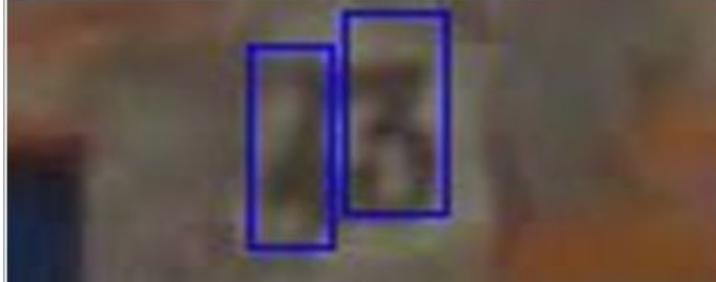
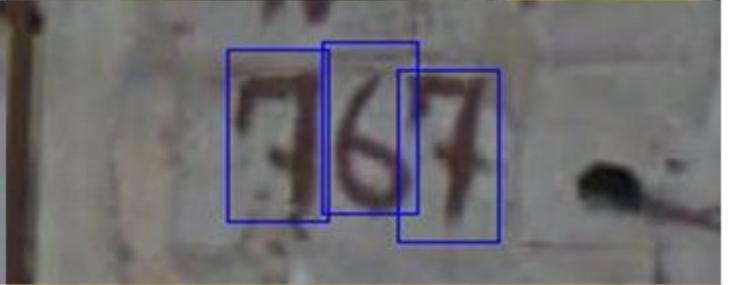
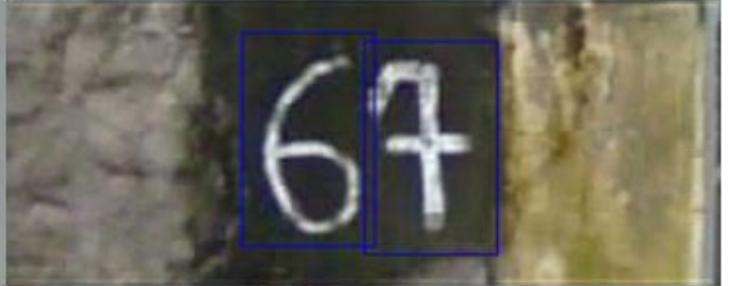
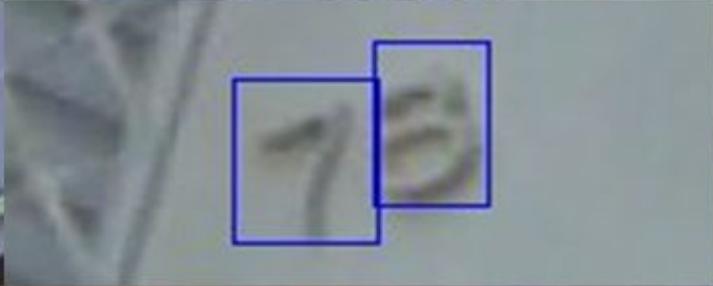
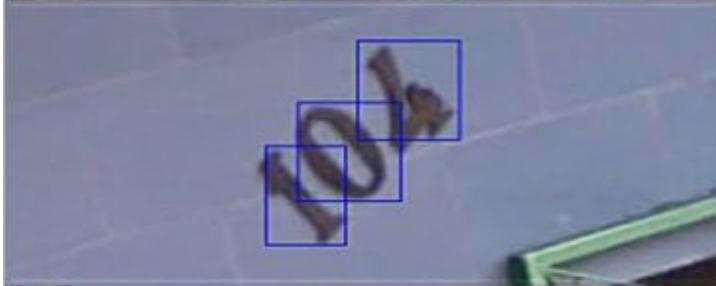
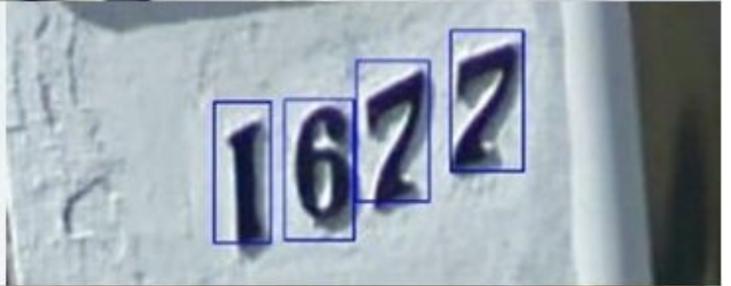
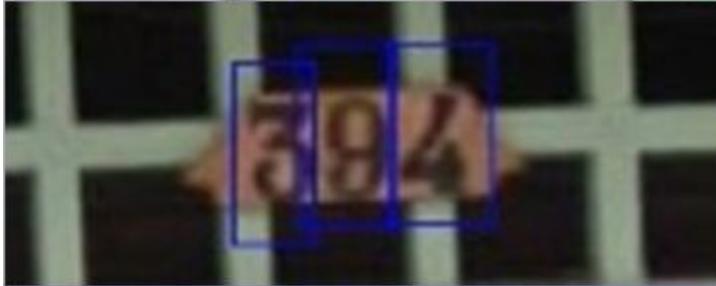


DEEP
LEARNING
INSTITUTE

Object Detection using NVIDIA DIGITS

Customization and Modification

Deep Learning Institute
NVIDIA Corporation



AGENDA

Introduction to Object Detection

Detection by Combining Deep Learning with
Traditional Computer Vision

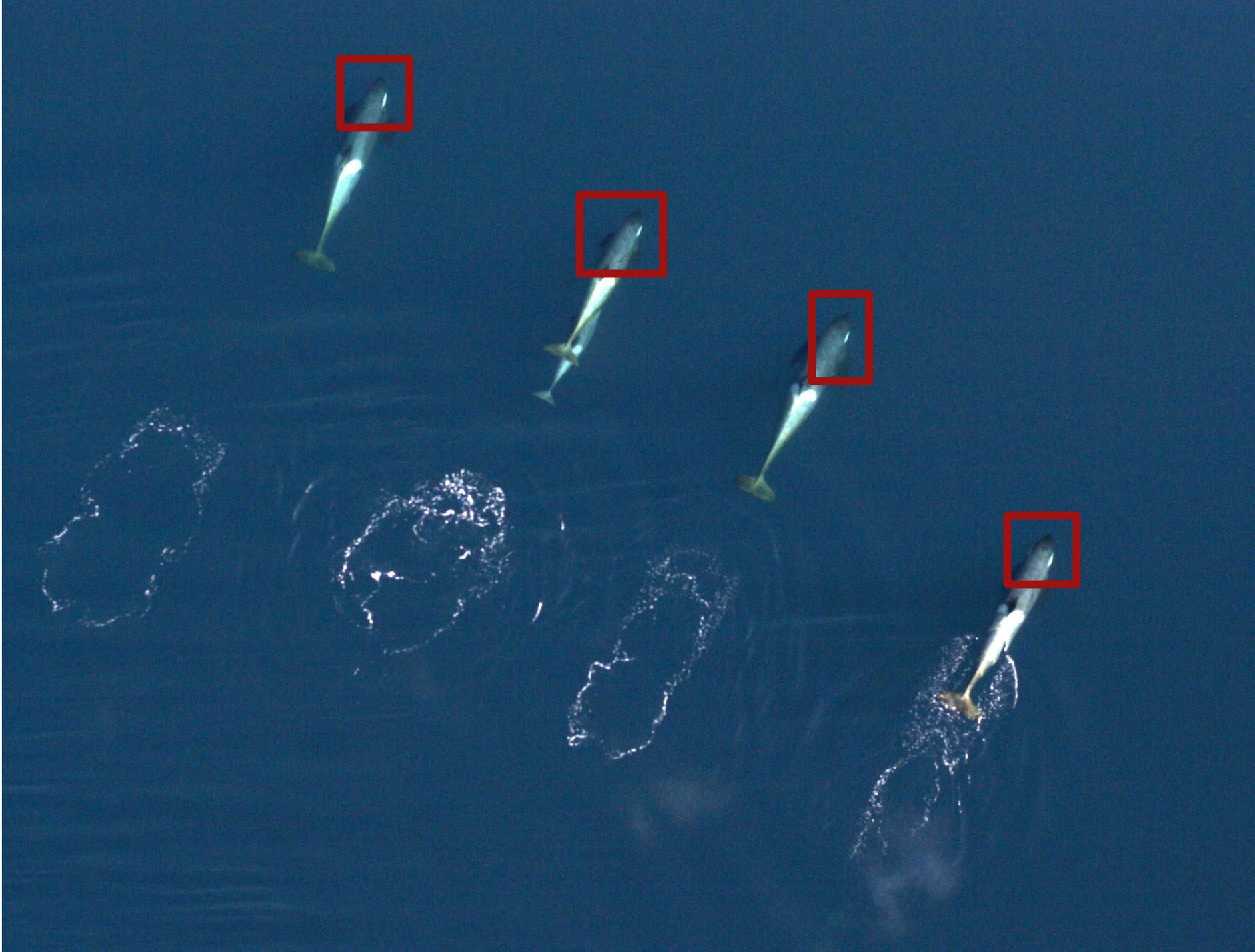
Detection by Modifying Network Architecture

State of the Art Detection

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.*



Brainstorm:

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.



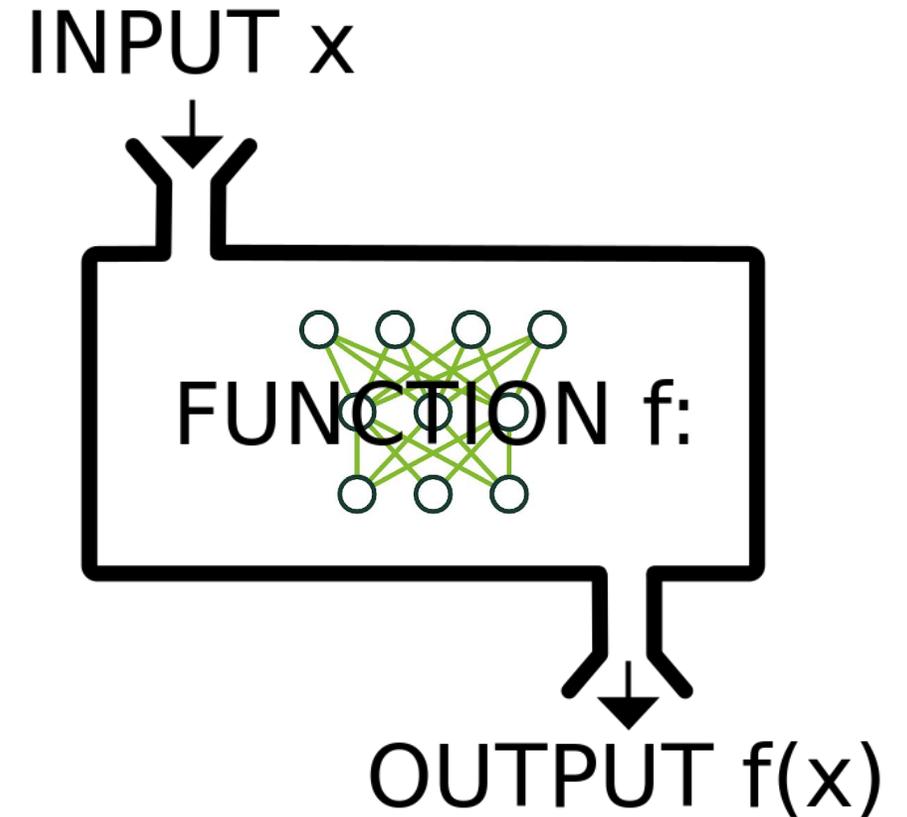
AI at scale

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**



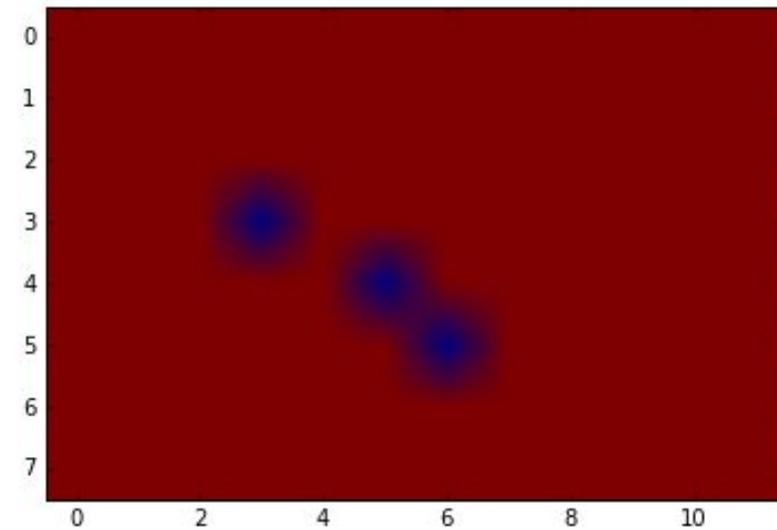
Approach 1: Sliding Window

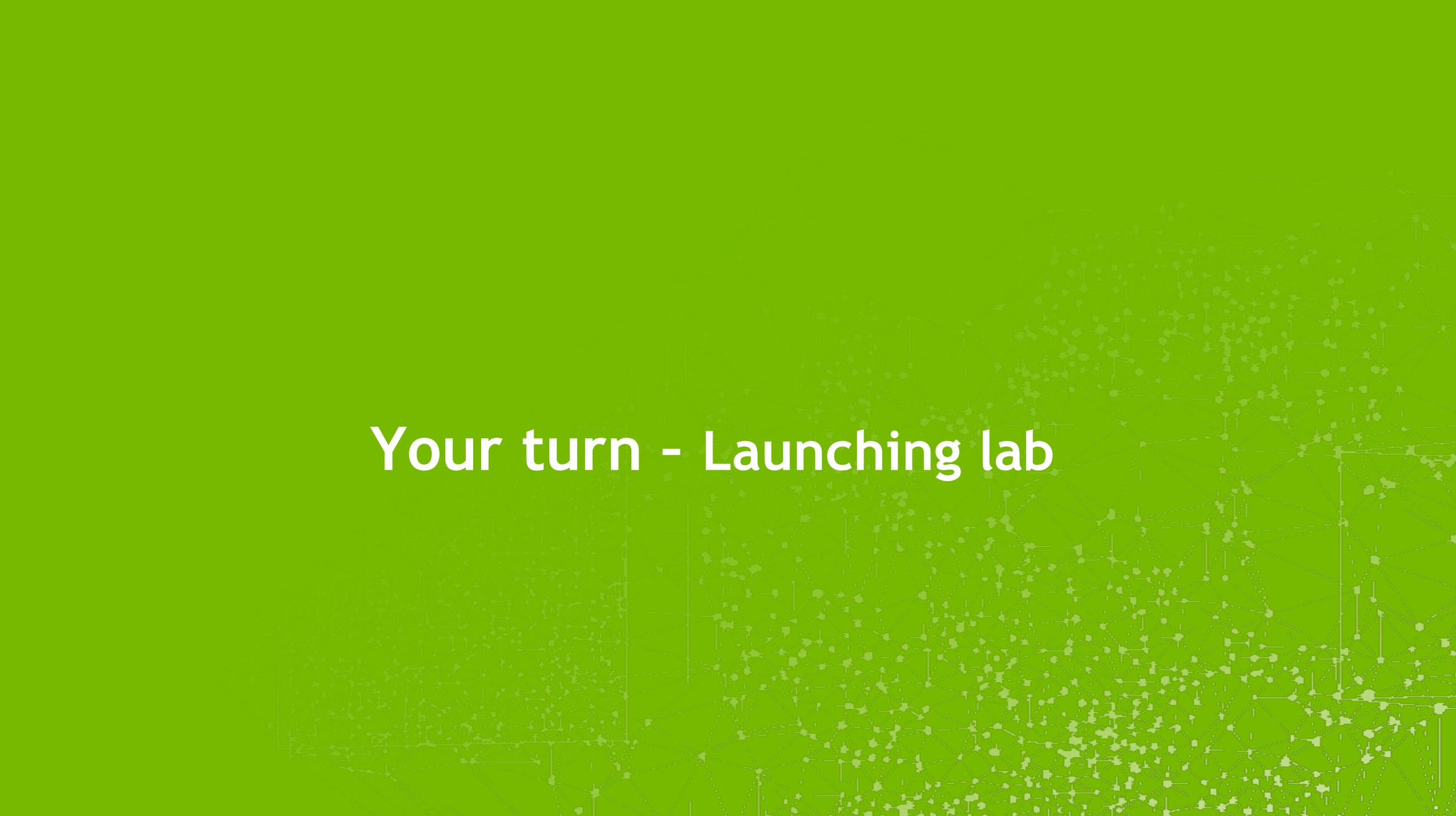
- Technique:
 - Build a whale face/not whale face classifier
 - Sliding window python application runs classifier on each 256X256 segment
 - Yes = blue, no = red



Total inference time: 10.5373151302 seconds

Total inference time: 10.5373151302 seconds





Your turn - Launching lab

Potential Confusion

Despite existing datasets and models, you will begin the lab by loading a new dataset and training a new *classification* model.

No Jobs Running

Datasets (2) Models (2) Pretrained Models (0)

Group Jobs:

New Dataset Images ▾

Classification
Object Detection
Other
Processing
Segmentation

name	refs	extension	backend	status	duration	created
▼ Ungrouped						
whale_full	1	image-object-detection		Done		
mnist	1		Imdb	Done	2m	Jul 22, 16

CONNECTING TO THE LAB ENVIRONMENT

 jupyter Object detection (unsaved changes)

File Edit View Insert Cell Kernel Help

        Run   Markdown 

Lab will take place in a Jupyter notebook



DEEP
LEARNING
INSTITUTE

Object Detection with DIGITS

In [Image Classification with DIGITS](#), you learned to successfully *train* a neural network. You saw that while traditional processing is used for classifying images, deep learning makes it not only possible, but fairly straightforward. You can now create an image classification *network and thousands of labeled images*.

JUPYTER NOTEBOOK

1. Make changes in code blocks

Copy the job directory (highlighted above) and replace `##FIXME##` in the code block below. Once you've copied the directory, execute the cell (Shift+Enter) to store it to the variable

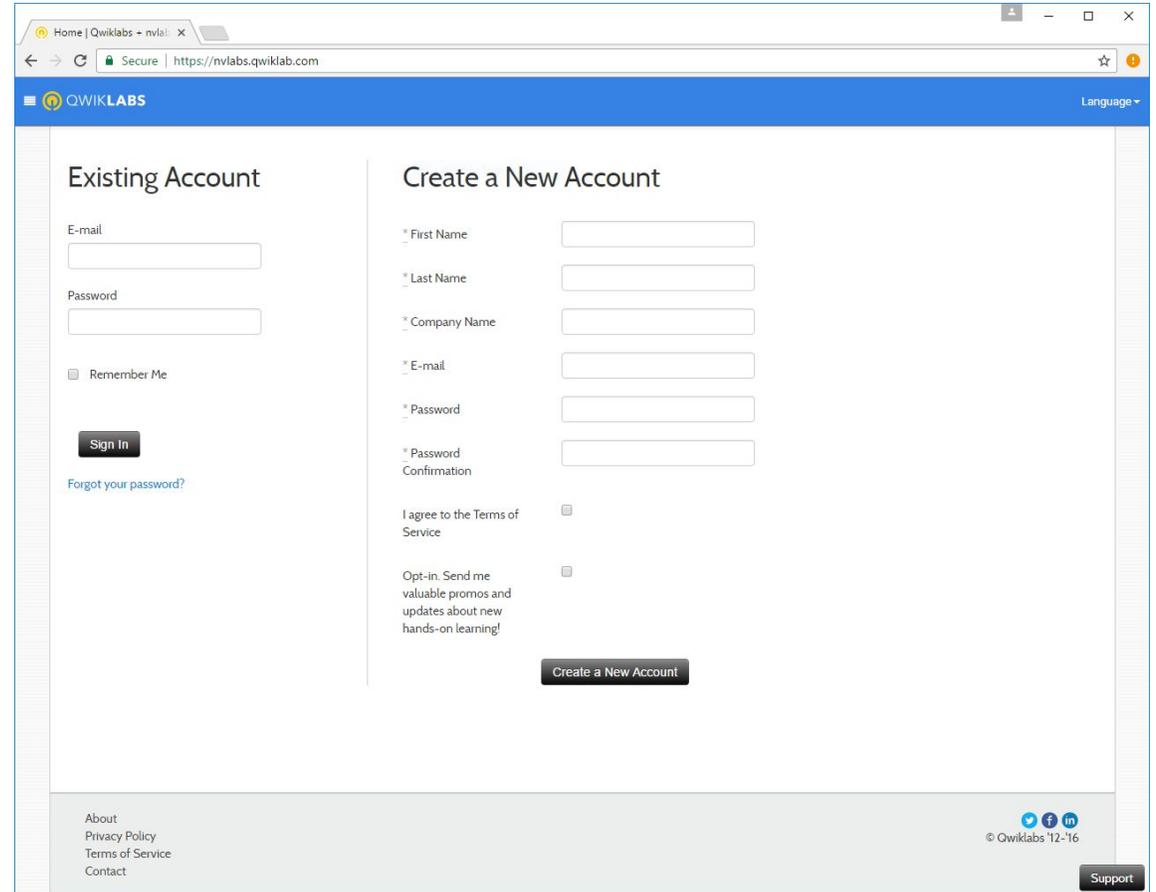
`MODEL_JOB_DIR`

```
In [ ]: MODEL_JOB_DIR = '##FIXME##' ## Remember to set this to be the job directory for  
print('Got it.')
```

2. Simultaneous “Shift” + “Enter” while mouse is in code-block

NAVIGATING TO QWIKLABS

1. Navigate to:
<https://nvlabs.qwiklab.com>
2. Login or create a new account



The screenshot shows a web browser window displaying the Qwiklabs login and account creation page. The browser's address bar shows the URL <https://nvlabs.qwiklab.com>. The page has a blue header with the Qwiklabs logo and a language selection dropdown. The main content area is divided into two columns: "Existing Account" and "Create a New Account".

Existing Account:

- E-mail:
- Password:
- Remember Me
-
- [Forgot your password?](#)

Create a New Account:

- * First Name:
- * Last Name:
- * Company Name:
- * E-mail:
- * Password:
- * Password Confirmation:
- I agree to the Terms of Service
- Opt-in. Send me valuable promos and updates about new hands-on learning!
-

At the bottom of the page, there are links for "About", "Privacy Policy", "Terms of Service", and "Contact". On the right side, there are social media icons for Facebook, Twitter, and LinkedIn, along with the text "© Qwiklabs 12-'16" and a "Support" button.

ACCESSING LAB ENVIRONMENT

3. Select the event “Fundamentals of Deep Learning” in the upper left

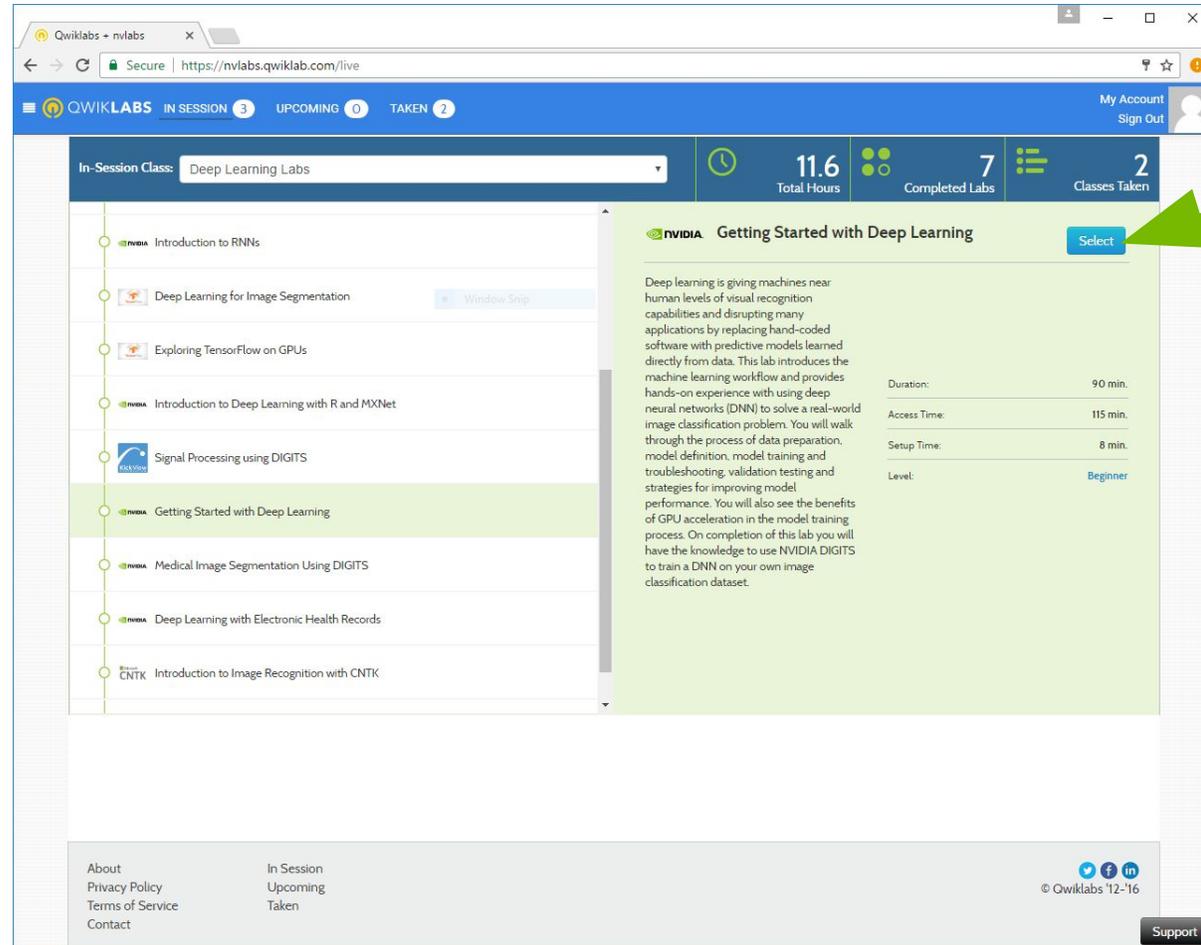
4. Click the “Object Detection with DIGITS” Class from the list

The screenshot shows the Qwiklabs interface. At the top, there is a navigation bar with 'QWIKLABS' and status indicators for 'IN SESSION' (3), 'UPCOMING' (0), and 'TAKEN' (2). A user profile is visible in the top right corner. Below the navigation bar, there is a header section with 'In-Session Class: Deep Learning Labs', 'Total Hours: 11.6', 'Completed Labs: 7', and 'Classes Taken: 2'. The main content area displays a list of classes on the left and a detailed view of the selected class on the right. The list includes:

- Introduction to RNNs
- Deep Learning for Image Segmentation
- Exploring TensorFlow on GPUs
- Introduction to Deep Learning with R and MXNet
- Signal Processing using DIGITS
- Getting Started with Deep Learning (highlighted)
- Medical Image Segmentation Using DIGITS
- Deep Learning with Electronic Health Records
- Introduction to Image Recognition with CNTK

The detailed view for 'Getting Started with Deep Learning' shows a description: 'Deep learning is giving machines near human levels of visual recognition capabilities and disrupting many applications by replacing hand-coded software with predictive models learned directly from data. This lab introduces the machine learning workflow and provides hands-on experience with using deep neural networks (DNN) to solve a real-world image classification problem. You will walk through the process of data preparation, model definition, model training and troubleshooting, validation testing and strategies for improving model performance. You will also see the benefits of GPU acceleration in the model training process. On completion of this lab you will have the knowledge to use NVIDIA DIGITS to train a DNN on your own image classification dataset.' The details include: Duration: 90 min, Access Time: 115 min, Setup Time: 8 min, and Level: Beginner. A 'Select' button is visible in the top right of the detailed view.

LAUNCHING THE LAB ENVIRONMENT



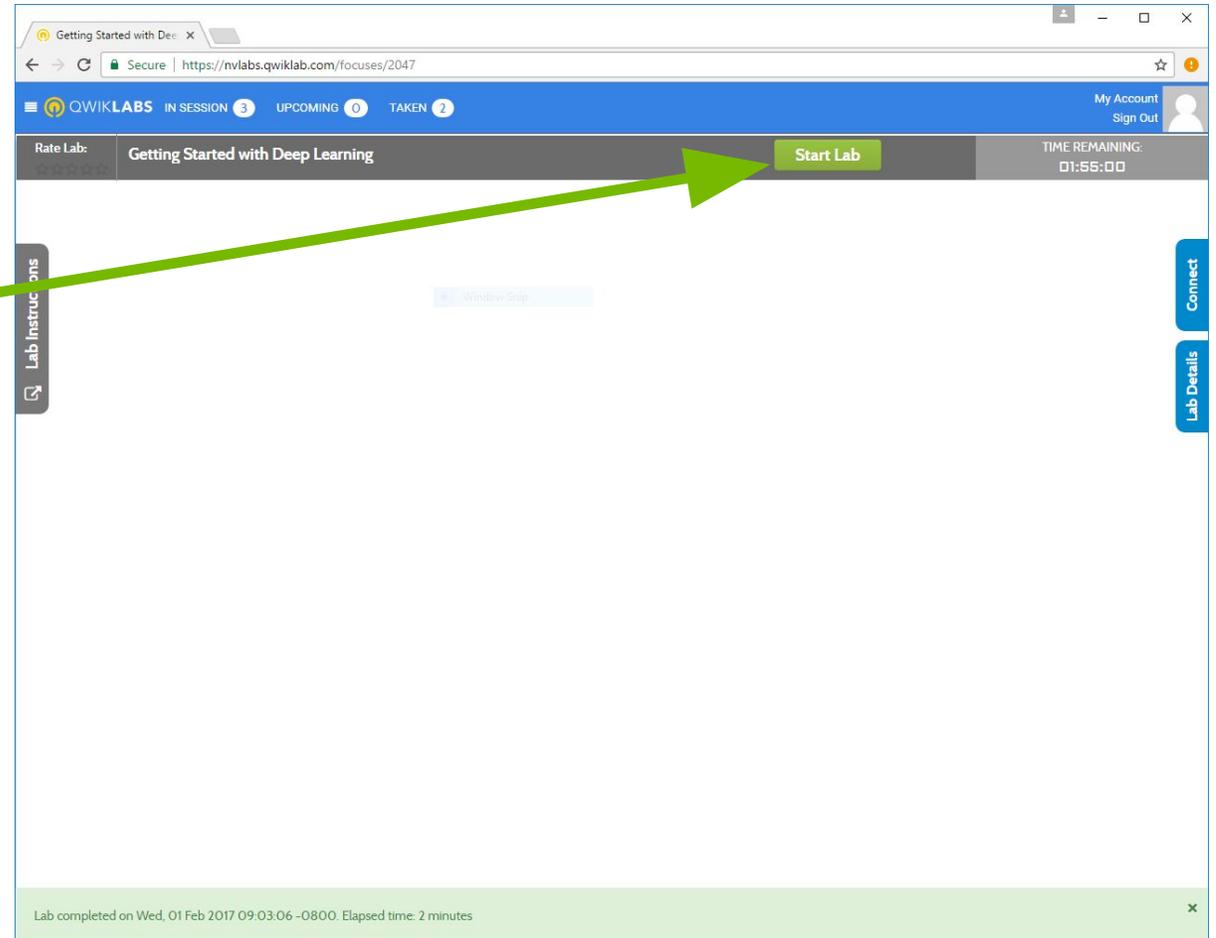
The screenshot shows the Qwiklabs web interface. At the top, there's a navigation bar with 'QWIKLABS' and status indicators for 'IN SESSION' (3), 'UPCOMING' (0), and 'TAKEN' (2). A user profile icon is visible on the right. Below the navigation bar, there's a section for 'In-Session Class: Deep Learning Labs' with statistics: '11.6 Total Hours', '7 Completed Labs', and '2 Classes Taken'. A list of labs is shown on the left, with 'Getting Started with Deep Learning' highlighted in green. A detailed view of this lab is shown on the right, including a description, duration (90 min), access time (115 min), setup time (8 min), and level (Beginner). A blue 'Select' button is visible next to the lab title, and a green arrow points to it from the text on the right.

5. Click on the Select button to launch the lab environment

- After a short wait, lab Connection information will be shown
- Please ask Lab Assistants for help!

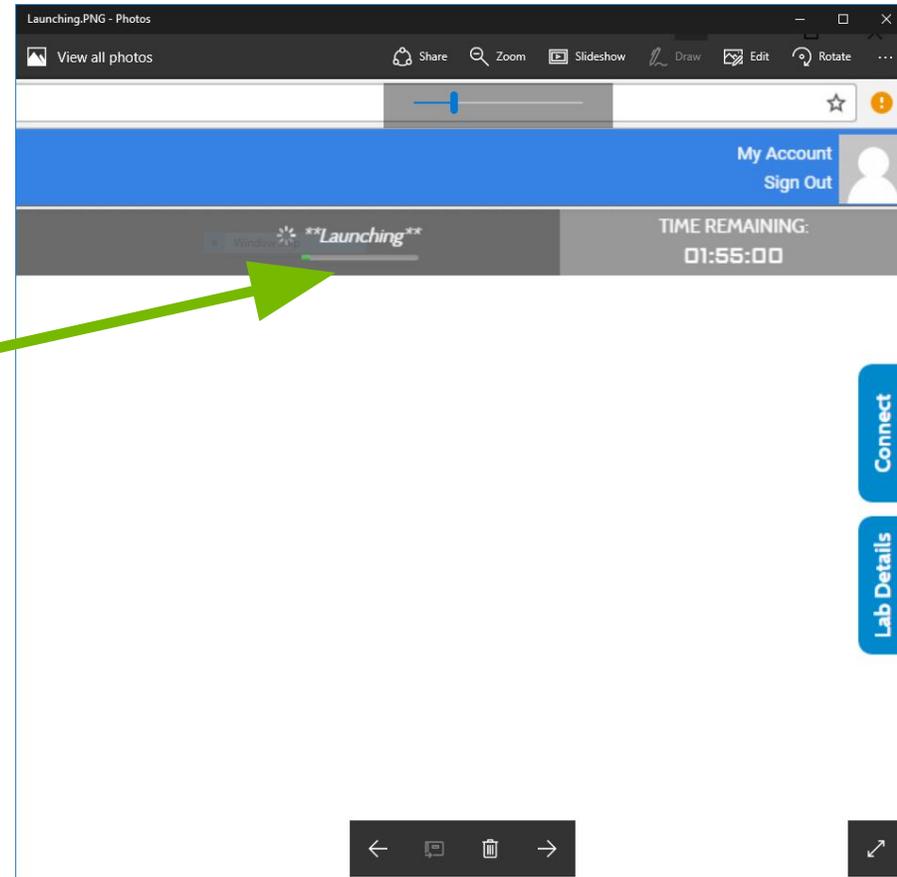
LAUNCHING THE LAB ENVIRONMENT

6. Click on the Start Lab button



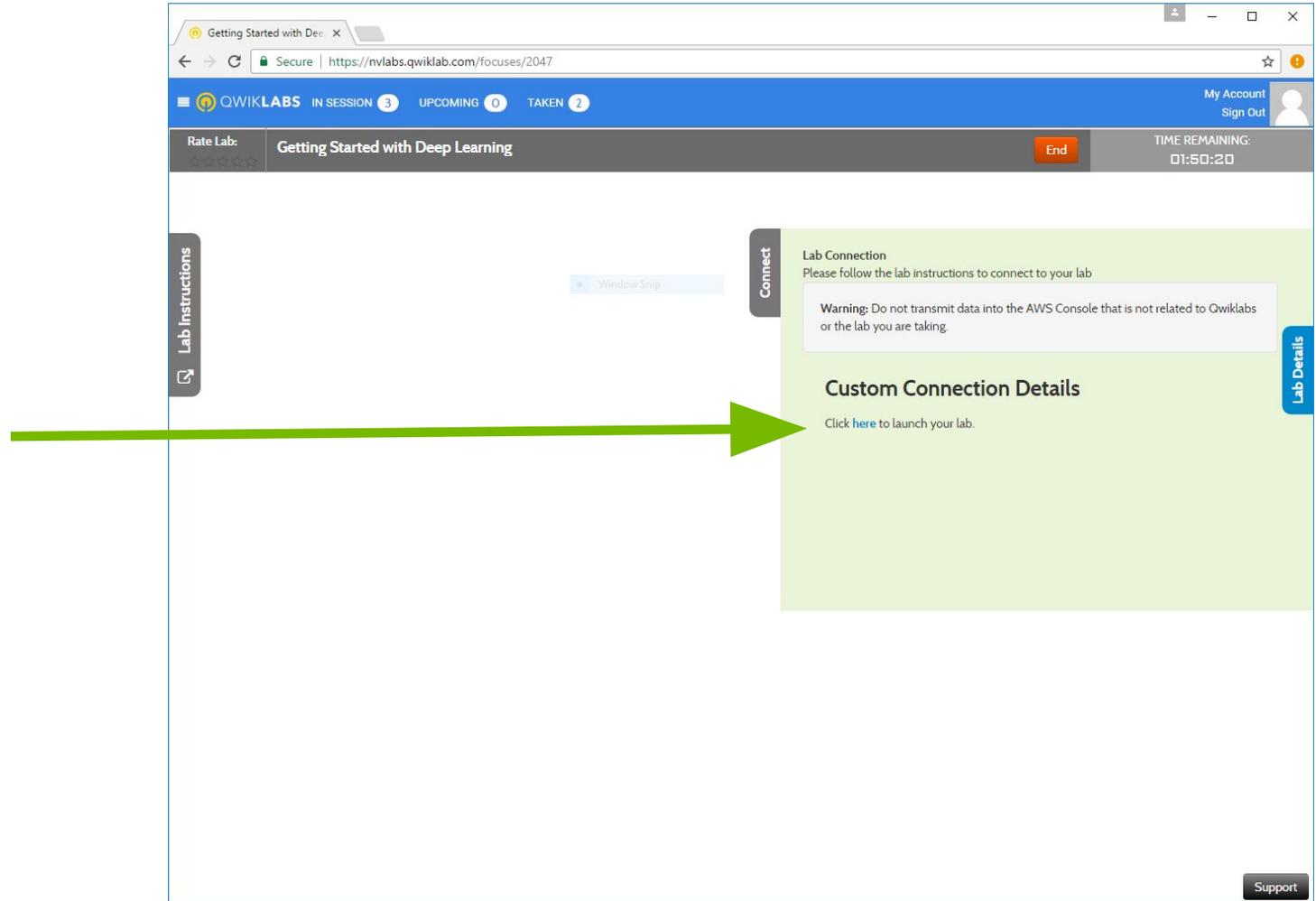
LAUNCHING THE LAB ENVIRONMENT

You should see that the lab environment is “launching” towards the upper-right corner



CONNECTING TO THE LAB ENVIRONMENT

7. Click on “here” to access your lab environment / Jupyter notebook



**Follow lab instructions through end of
Approach 1**

Discuss: Intro to Network Architecture

The background of the slide is a solid green color. Overlaid on this background is a faint, light-colored network diagram. This diagram consists of numerous small, interconnected nodes and lines, forming a complex web that represents a network architecture. The nodes are distributed across the entire frame, with a higher density in the lower right quadrant.

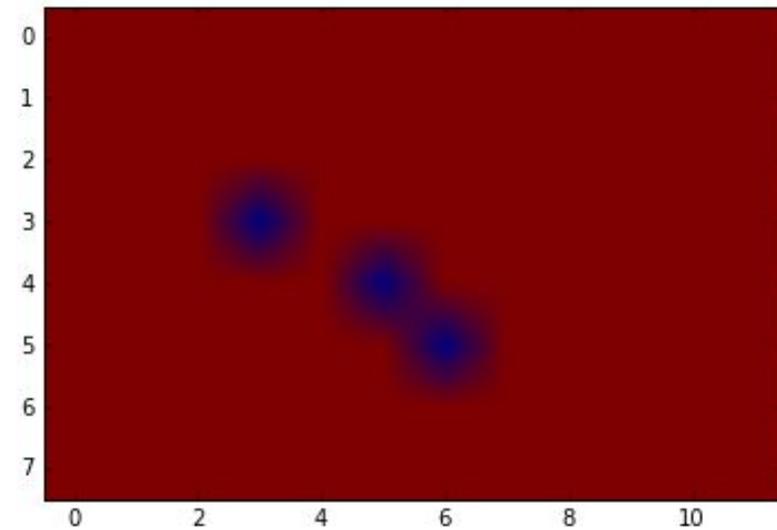
Approach 1: Sliding Window

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



Total inference time: 10.5373151302 seconds

Total inference time: 10.5373151302 seconds



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.

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.

CAFFE FEATURES

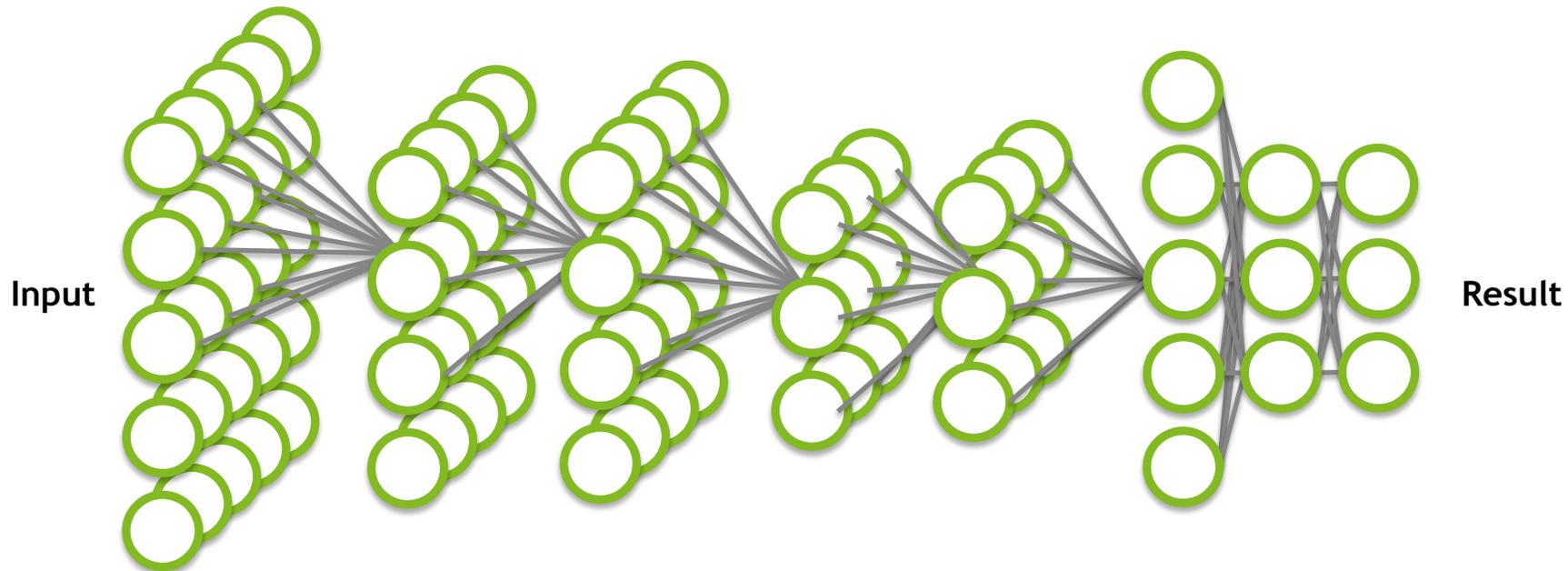
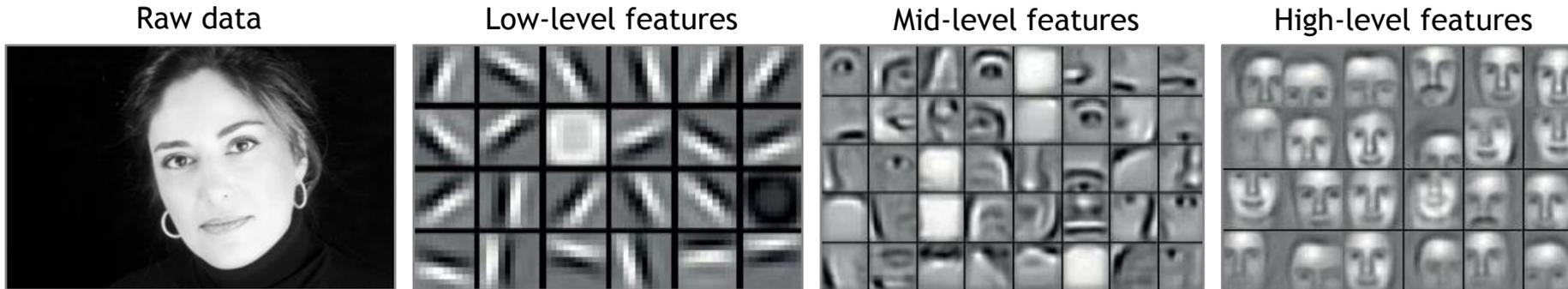
Deep Learning model definition

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: 16  
    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

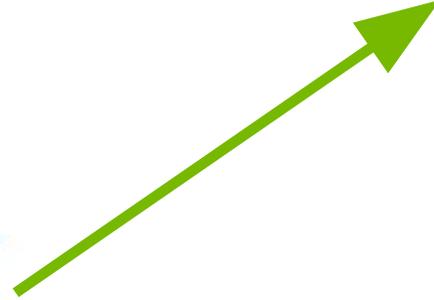
Network architecture
~ 10s-100s of layers
1B parameters

Learning algorithm
~ 30 Exaflops
1-30 GPU days

APPROACH 2 - Network Modification

- Modify **AlexNet** by using **Caffe** in **DIGITS**
- Replace **layers** by **reading carefully**

```
241 }
242 layer {
243   name: "pool5"
244   type: "Pooling"
245   bottom: "conv5"
246   top: "pool5"
247   pooling_param {
248     pool: MAX
249     kernel_size: 3
250     stride: 2
251   }
252 }
253 layer {
254   name: "fc6"
255   type: "InnerProduct"
256   bottom: "pool5"
257   top: "fc6"
258   param {
259     lr_mult: 1
260     decay_mult: 1
261   }
262   param {
263     lr_mult: 2
264     decay_mult: 0
265   }
266   inner_product_param {
267     num_output: 4096
268     weight_filler {
269       type: "gaussian"
270       std: 0.005
271     }
272     bias_filler {
273       type: "constant"
274       value: 0.1
275     }
276   }
277 }
278 layer {
279   name: "relu6"
280   type: "ReLU"
281   bottom: "fc6"
282   top: "fc6"
283 }
```



```
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"
}
```

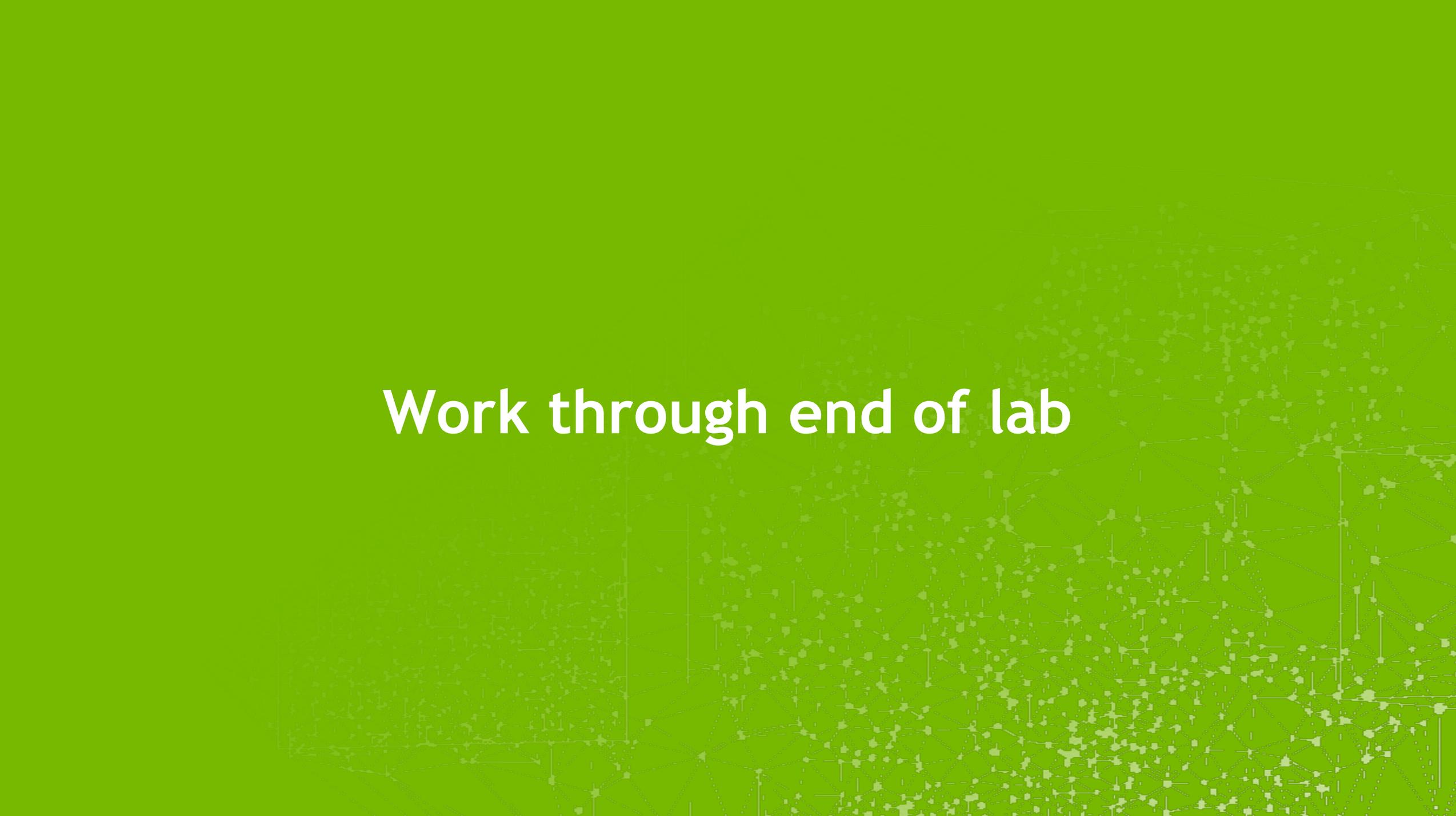
RETURN TO THE LAB

Work through the end

We will debrief “Approach 3” post-lab

Ask for help if needed

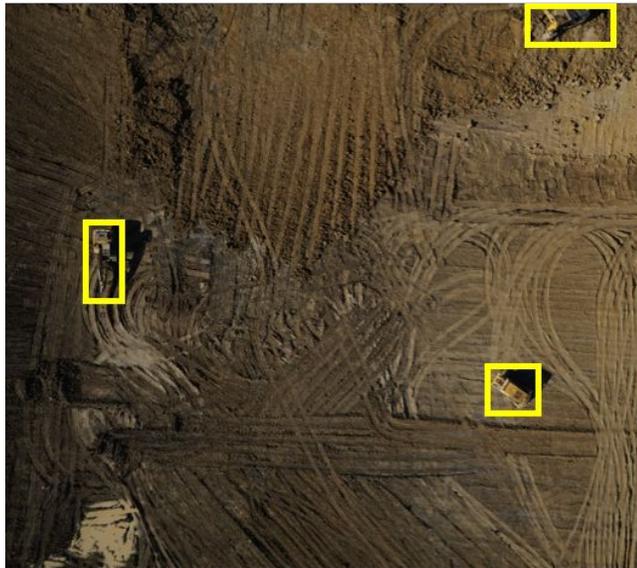
If at any point you get stuck, seek out solutions



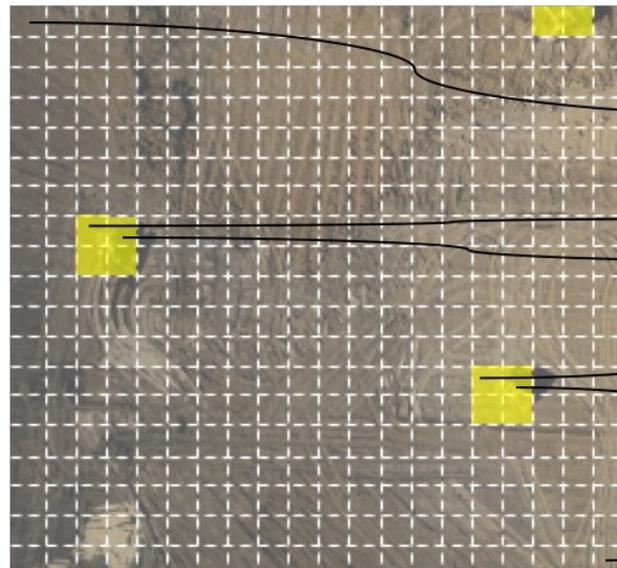
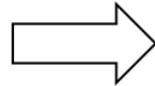
Work through end of lab

Approach 3: End-to-End Solution

Need dataset with inputs and corresponding (often complex) output



Training image with bounding box annotations

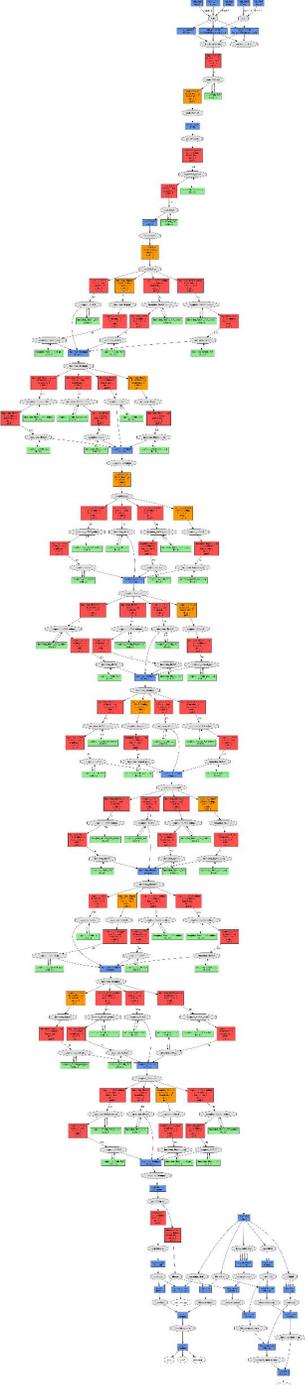


Bounding boxes mapped to grid squares

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 requires **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:
 - Architecture designed for detecting **anything**
 - Dataset is **whale-face specific**
 - DetectNet is **efficient** and **accurate**

Source image



Inference visualization



Source image



Inference visualization



■ bbox-IIST

ADDITIONAL APPROACHES TO OBJECT DETECTION ARCHITECTURE

- R-CNN = Region CNN
- Fast R-CNN
- Faster R-CNN Region Proposal Network
- RoI-Pooling = Region of Interest Pooling

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

GPU TECHNOLOGY CONFERENCE

March 26-29, 2018 | Silicon Valley | #GTC18
www.gputechconf.com



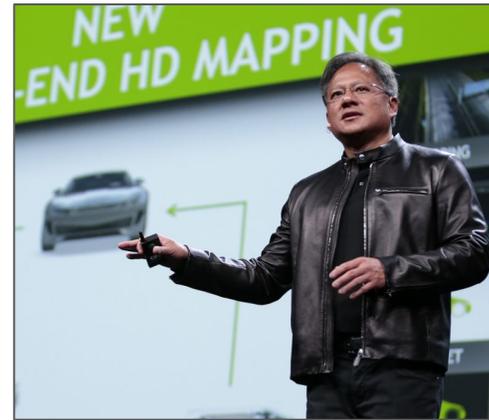
CONNECT

Connect with technology experts from NVIDIA and other leading organizations



LEARN

Gain insight and valuable hands-on training through hundreds of sessions and research posters



DISCOVER

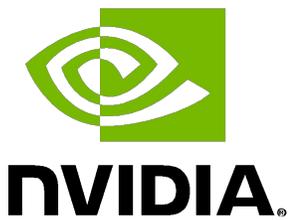
See how GPUs are creating amazing breakthroughs in important fields such as deep learning and AI



INNOVATE

Hear about disruptive innovations from startups

Enjoy the world's most important event for GPU developers
March 26-29, 2018 in Silicon Valley



DEEP
LEARNING
INSTITUTE

www.nvidia.com/dli