



DEEP
LEARNING
INSTITUTE

Neural Network Deployment with DIGITS and TensorRT

Twin Karmakharm
Certified Instructor, NVIDIA Deep Learning Institute

Date



DEEP LEARNING INSTITUTE

DLI Mission

Helping people solve challenging problems using AI and deep learning.

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

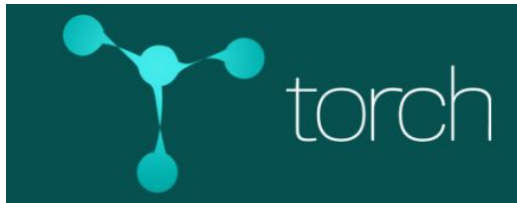
TOPICS

- Caffe
- NVIDIA'S DIGITS
- Deep Learning Approach
- NVIDIA'S TensorRT
- Lab
 - Lab Details
 - Launching the Lab Environment
- Review / Next Steps

CAFFE

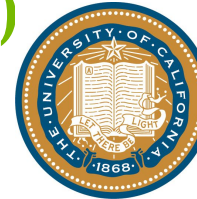
Frameworks

Many Deep Learning Tools



WHAT IS CAFFE?

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



- 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.berkeleyvision.org
<http://github.com/BVLC/caffe>

CAFFE FEATURES

Deep Learning model definition

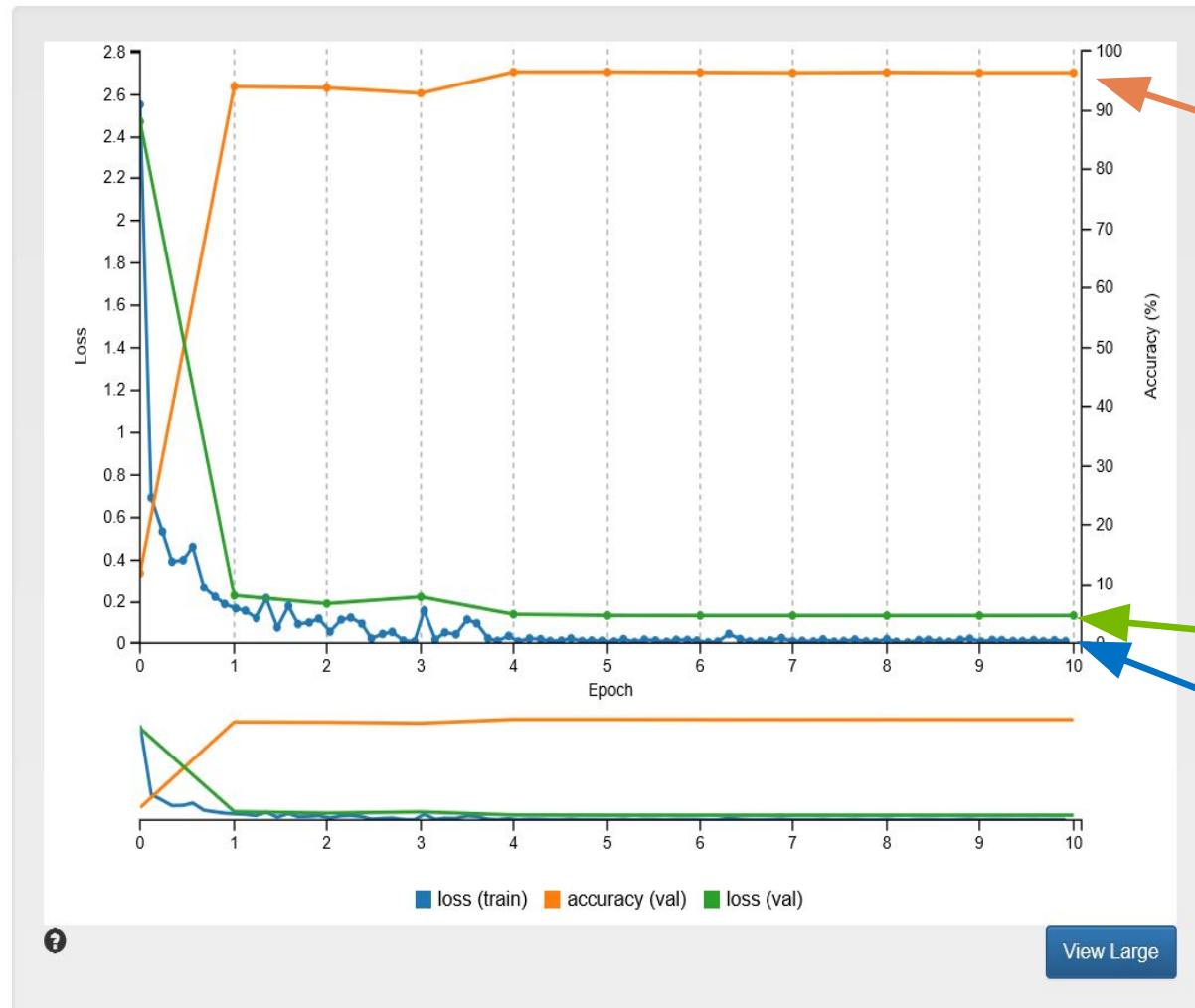
Protobuf model format

- 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"  
    }  
}
```

NVIDIA'S DIGITS

NVIDIA'S DIGITS



Accuracy
obtained from
validation dataset

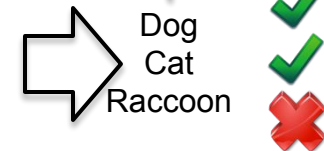
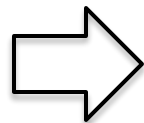
Loss function
(Validation)

Loss function
(Training)

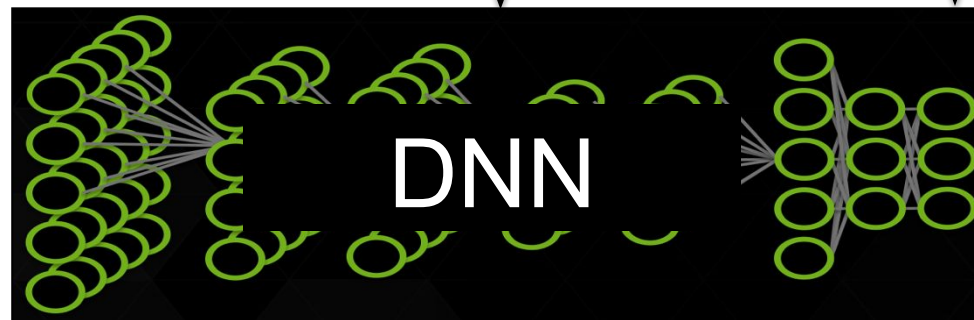
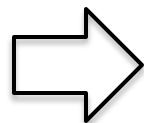
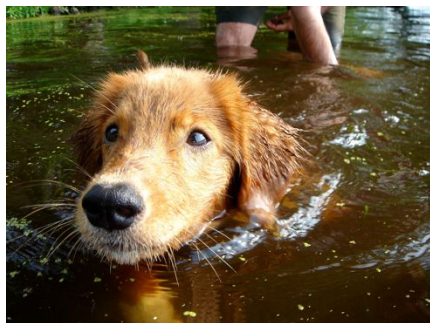
DEEP LEARNING APPROACH

Deep Learning Approach

Train:



Deploy:



Deep Learning Approach

Convolutional Neural Network

IMAGES



Conv

Pool

Conv

Pool

Conv

Pool

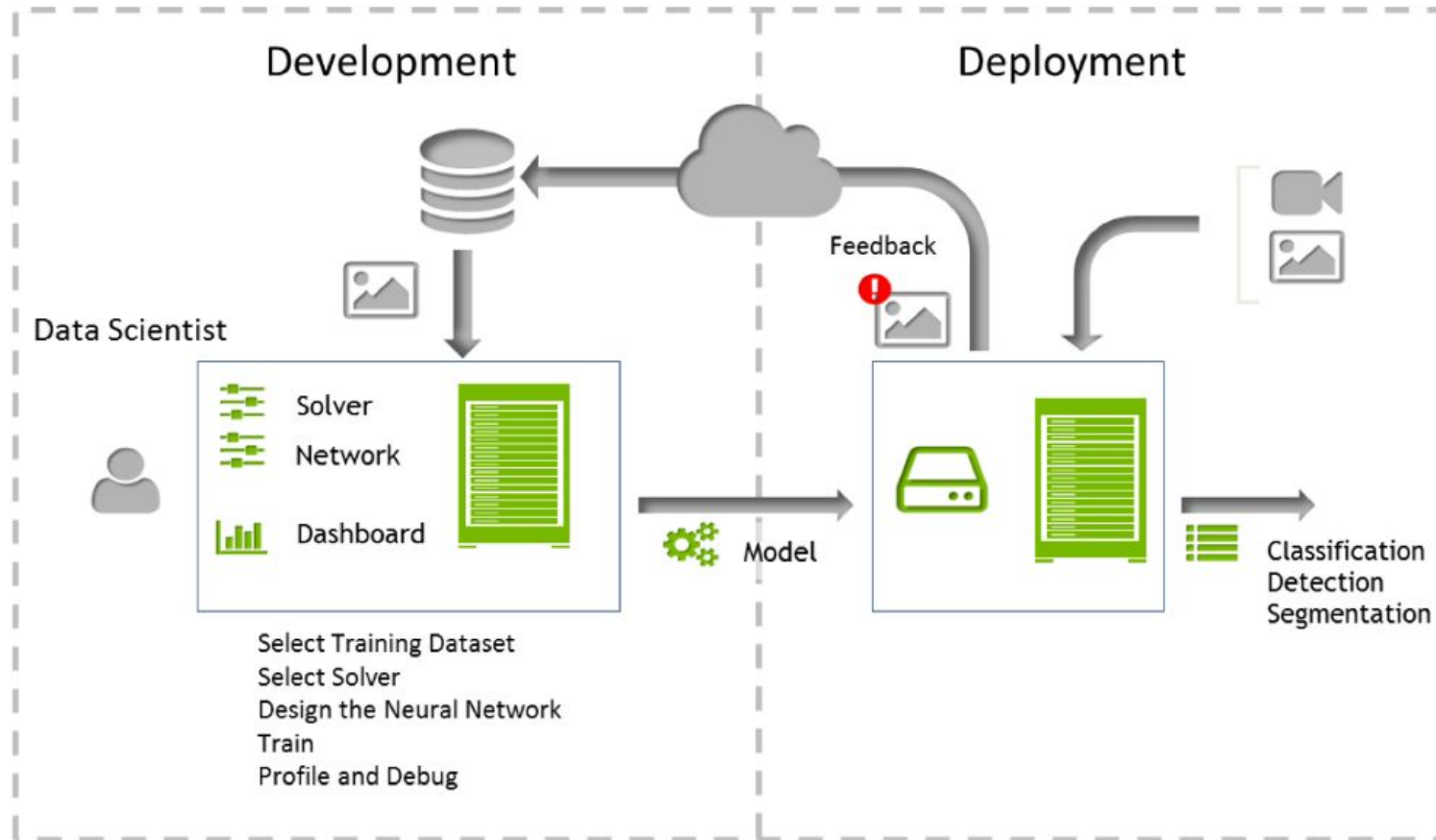
Fully connected

Fully connected



Deep Learning Approach

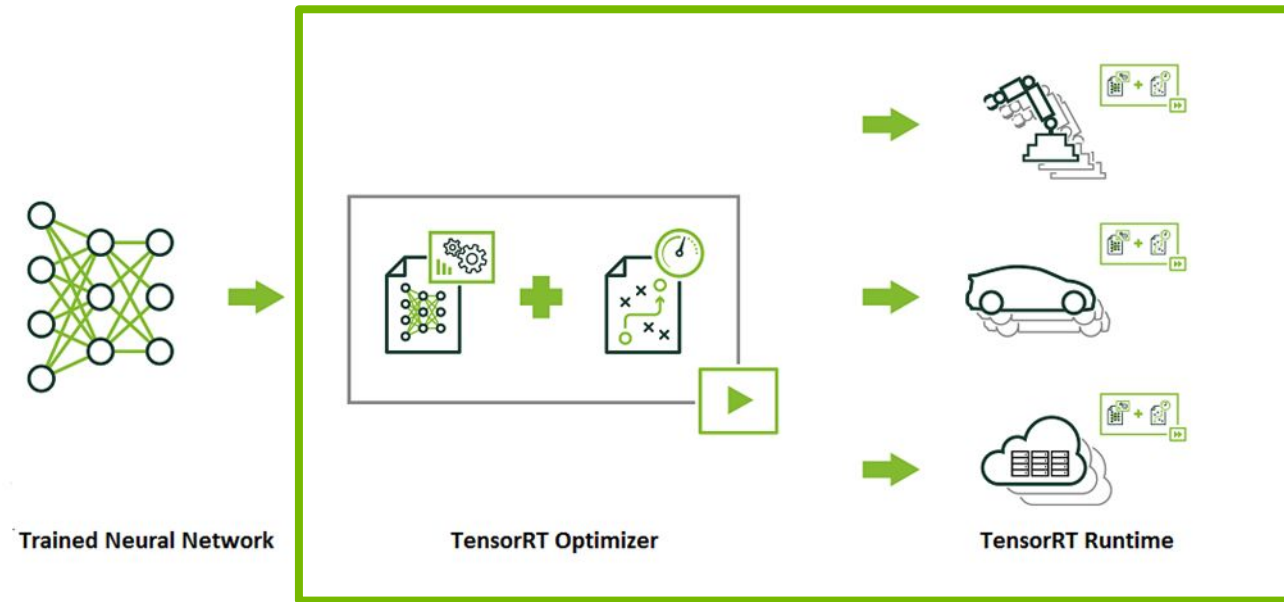
Neural network training and inference



NVIDIA'S TENSORRT

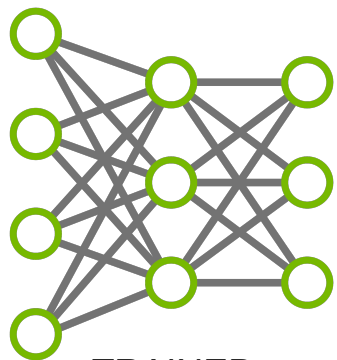
TensorRT

- Inference engine for production deployment of deep learning applications

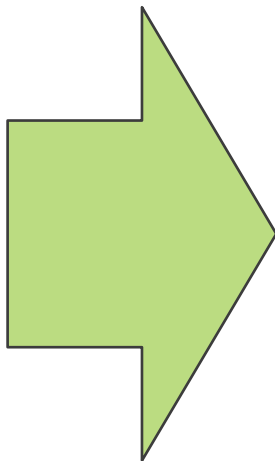


- Allows developers to focus on developing AI powered applications
 - TensorRT ensures optimal inference performance

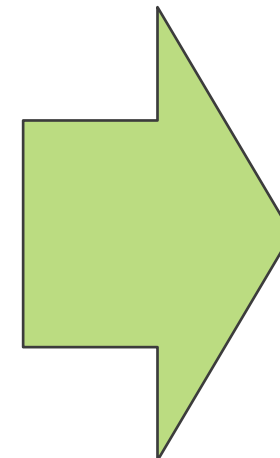
TensorRT Optimizer



TRAINED
NEURAL
NETWORK



- Fuse network layers
- Eliminate concatenation layers
- Kernel specialization
- Auto-tuning for target platform
- Select optimal tensor layout
- Batch size tuning

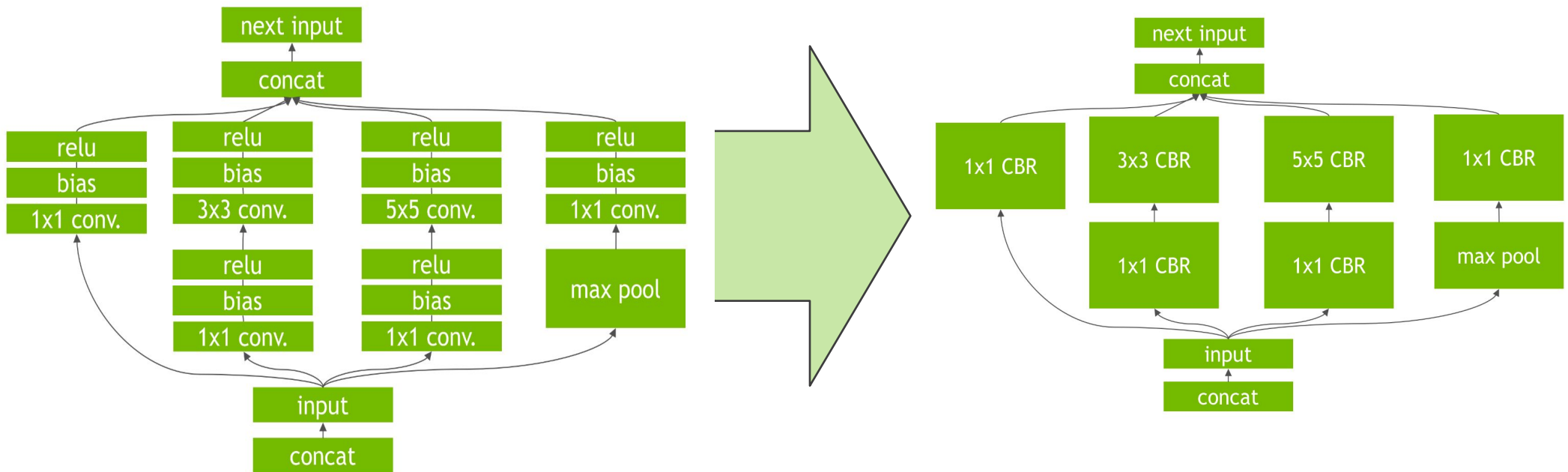


OPTIMIZED
INFERENCE
RUNTIME



TensorRT Optimizer

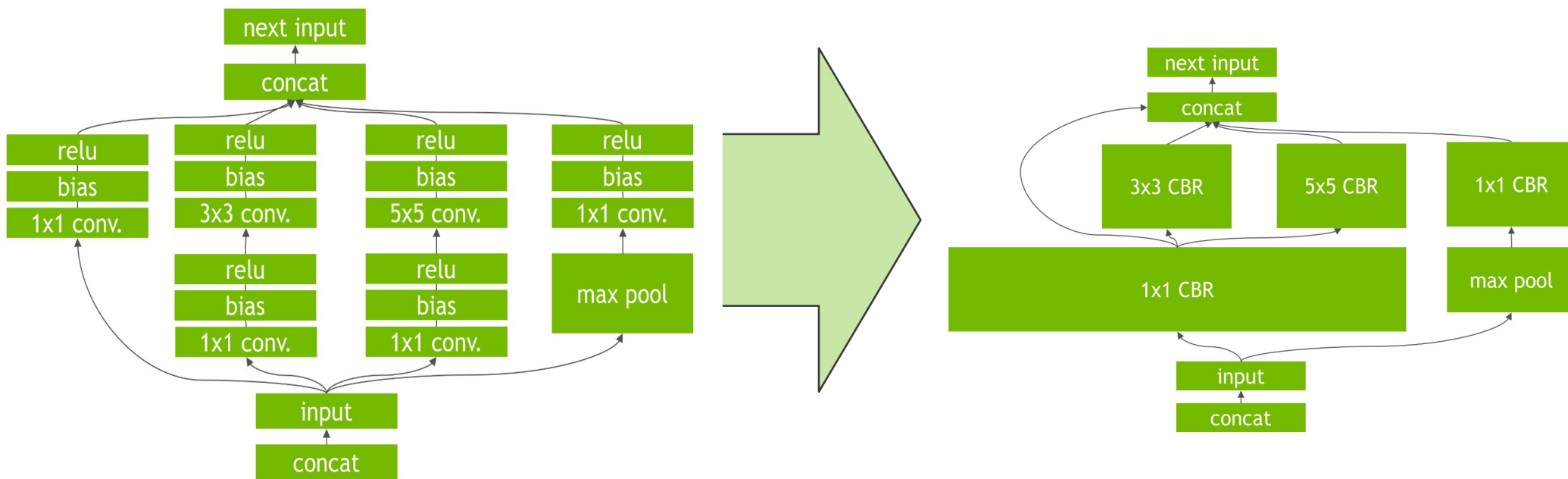
Vertical Layer Fusion



CBR = Convolution, Bias and ReLU

TensorRT Optimizer

Horizontal Layer Fusion (Layer Aggregation)



CBR = Convolution, Bias and ReLU

TensorRT Optimizer

Supported layers

- Convolution: 2D
- Activation: ReLU, tanh and sigmoid
- Pooling: max and average
- ElementWise: sum, product or max of two tensors
- LRN: cross-channel only
- Fully-connected: with or without bias
- SoftMax: cross-channel only
- Deconvolution

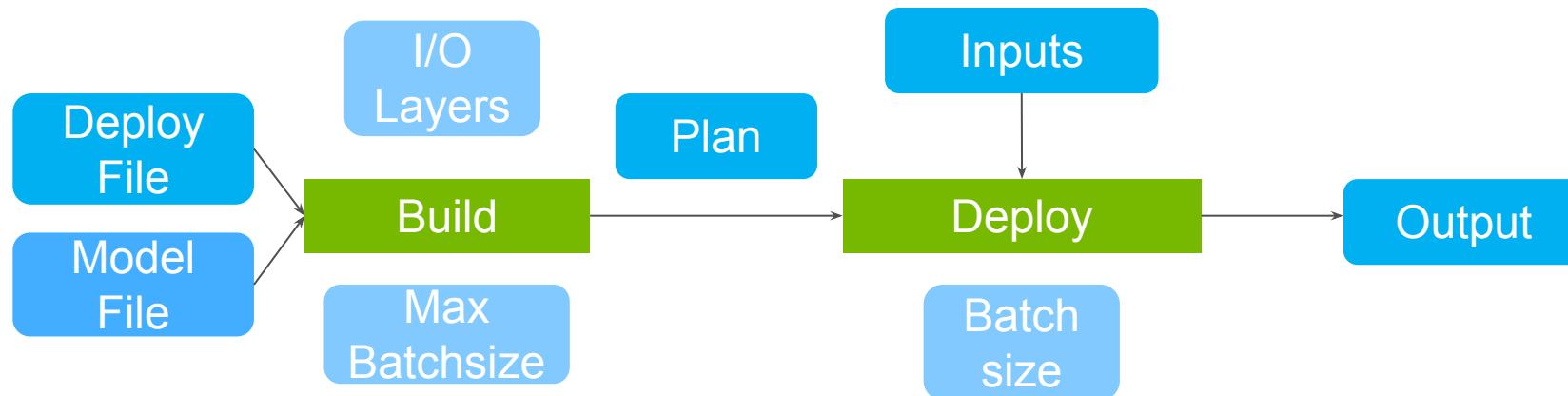
TensorRT Optimizer

- Scalability:
 - Output/Input Layers can connect with other deep learning framework directly
 - Caffe, Theano, Torch, TensorFlow
- Reduced Latency:
 - INT8 or FP16
 - INT8 delivers 3X more throughput compared to FP32
 - INT8 uses 61% less memory compared to FP32

TensorRT Runtime

Two Phases

- **Build:** optimizations on the network configuration and generates an optimized plan for computing the forward pass
- **Deploy:** Forward and output the inference result



TensorRT Runtime

- No need to install and run a deep learning framework on the deployment hardware
- Plan = runtime (serialized) object
 - Plan will be smaller than the combination of model and weights
 - Ready for immediate use
 - Alternatively, state can be serialized and saved to disk or to an object store for distribution
- Three files needed to deploy a classification neural network:
 - Network architecture file (deploy.prototxt)
 - Trained weights (net.caffemodel)
 - Label file to provide a name for each output class

LAB DETAILS

Lab Architectures / Datasets

- *GoogleNet*
 - CNN architecture trained for image classification using the [ilsvrc12](#) [Imagenet](#) dataset
 - 1000 class labels to an entire image based on the dominant object present
- *pedestrian_detectNet*
 - CNN architecture able to assign a global classification to an image and detect multiple objects within the image and draw bounding boxes around them
 - Pre-trained model provided has been trained for the task of pedestrian detection using a large dataset of pedestrians in a variety of indoor and outdoor scenes

Lab Tasks

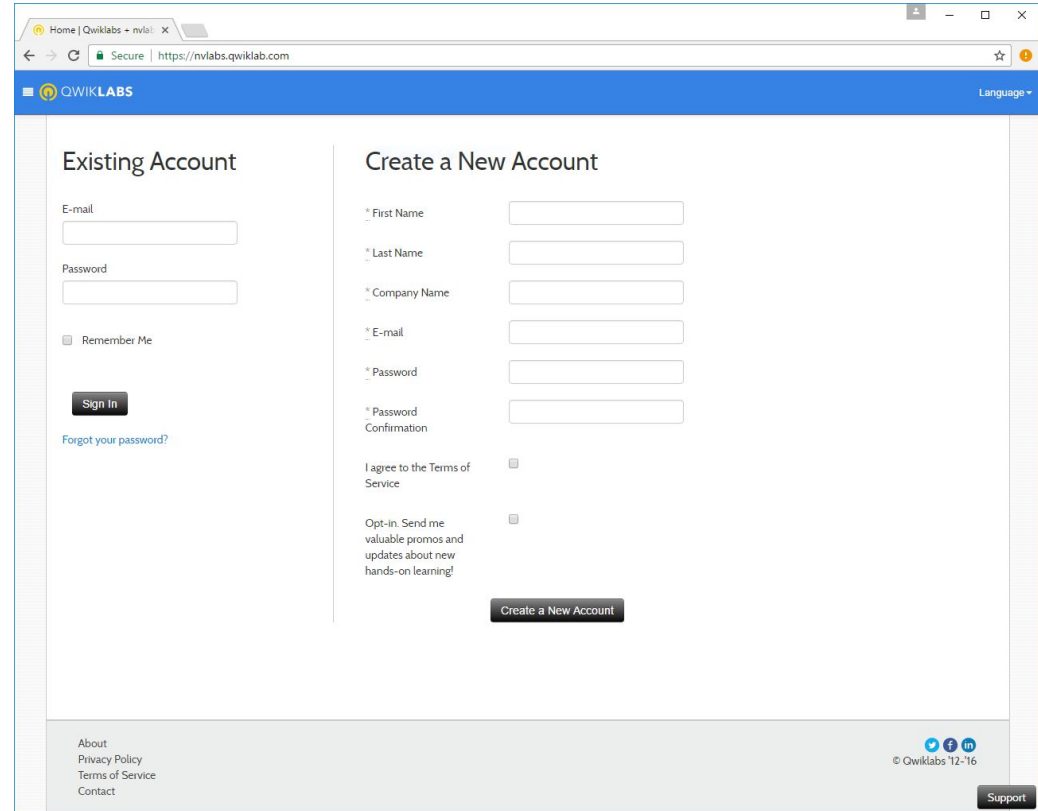
- GPU Inference Engine (GIE) = TensorRT
- Part 1: Inference using DIGITS
 - Will use existing model in DIGITS to perform inference on a single image
- Part 2: Inference using Pycaffe
 - Programming production-like deployable inference code
- Part 3: NVIDIA TensorRT
 - Will run TensorRT Optimizer to build a plan
 - Deploy the plan using TensorRT Runtime

LAUNCHING THE LAB ENVIRONMENT

NAVIGATING TO QWIKLABS

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

Please use the email address used to register for session



The screenshot shows the Qwiklabs website interface. The browser address bar displays "https://nvlabs.qwiklab.com". The page features two main sections: "Existing Account" and "Create a New Account".

Existing Account: Includes input fields for "E-mail" and "Password", a "Remember Me" checkbox, a "Sign In" button, and a "Forgot your password?" link.

Create a New Account: Includes input fields for "First Name", "Last Name", "Company Name", "E-mail", "Password", and "Password Confirmation". It also features checkboxes for "I agree to the Terms of Service" and "Opt-in. Send me valuable promos and updates about new hands-on learning!". A "Create a New Account" button is located at the bottom of this section.

The footer contains links for "About", "Privacy Policy", "Terms of Service", and "Contact", along with social media icons and a "Support" button.

ACCESSING LAB ENVIRONMENT

3. Select the event specific In-Session Class in the upper left

4. Click the “Deep Learning Network Deployment” Class from the list

The screenshot displays the NVIDIA Deep Learning Labs interface. At the top, there is a navigation bar with the following information: 'In-Session Class: Deep Learning Labs' (with a dropdown arrow), '36.5 Total Hours', '21 Completed Labs', and '4 Classes Taken'. Below this is a 'Class Details' section with a list of classes. The class 'Deep Learning Network Deployment' is highlighted in green. To the right of the list, a detailed view for 'Deep Learning Network Deployment' is shown, including a description, a 'Select' button, and a table of metrics.

Metric	Value
Duration:	90 min.
Access Time:	115 min.
Setup Time:	6 min.
Level:	Beginner

LAUNCHING THE LAB ENVIRONMENT

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

The screenshot shows the NVIDIA Deep Learning Labs interface. At the top, there is a navigation bar with the following information: 'In-Session Class: Deep Learning Labs', '36.5 Total Hours', '21 Completed Labs', and '4 Classes Taken'. Below this is a 'Class Details' sidebar on the left with a list of classes. The main content area displays the details for the 'Deep Learning Network Deployment' class, which is highlighted in green. A blue 'Select' button is located in the top right corner of this class detail card, with a green arrow pointing to it from the text '5. Click on the Select button to launch the lab environment'. The class details include a description, duration (90 min), access time (115 min), setup time (6 min), and level (Beginner).

Class	Duration	Access Time	Setup Time	Level
Introduction to Deep Learning				
Approaches to Object Detection using DIGITS				
Identifying Whale Sounds with Audio Classification				
Deep Learning Network Deployment	90 min.	115 min.	6 min.	Beginner
Introduction to RNNs				
Exploring TensorFlow on GPUs				
Introduction to Deep Learning with R and MXNet				

- 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



The screenshot shows the QwikLabs interface. At the top, there is a blue header with the QwikLabs logo, navigation links for 'IN SESSION 2', 'UPCOMING 1', and 'TAKEN 4', and a 'Standard View' button. Below the header, the main content area is divided into sections. On the left, there is a 'Rate Lab:' section with a star rating and the lab title 'Deep Learning Network Deployment'. In the center, there is a status indicator consisting of a small icon and the text '**Launching**'. On the right, there is a 'TIME REMAINING:' section showing '01:55:00'. A green arrow points from the text below to the '**Launching**' status indicator.

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

CONNECTING TO THE LAB ENVIRONMENT

LABS IN SESSION 2 UPCOMING 1 TAKEN 4 Standard View

Deep Learning Network Deployment End TIME REMAINING: 01:54:21

Rectangular Snip

Connect

Lab Connection
Please follow the lab instructions to connect to your lab

Warning: Do not transmit data into the AWS Console that is not related to C or the lab you are taking.

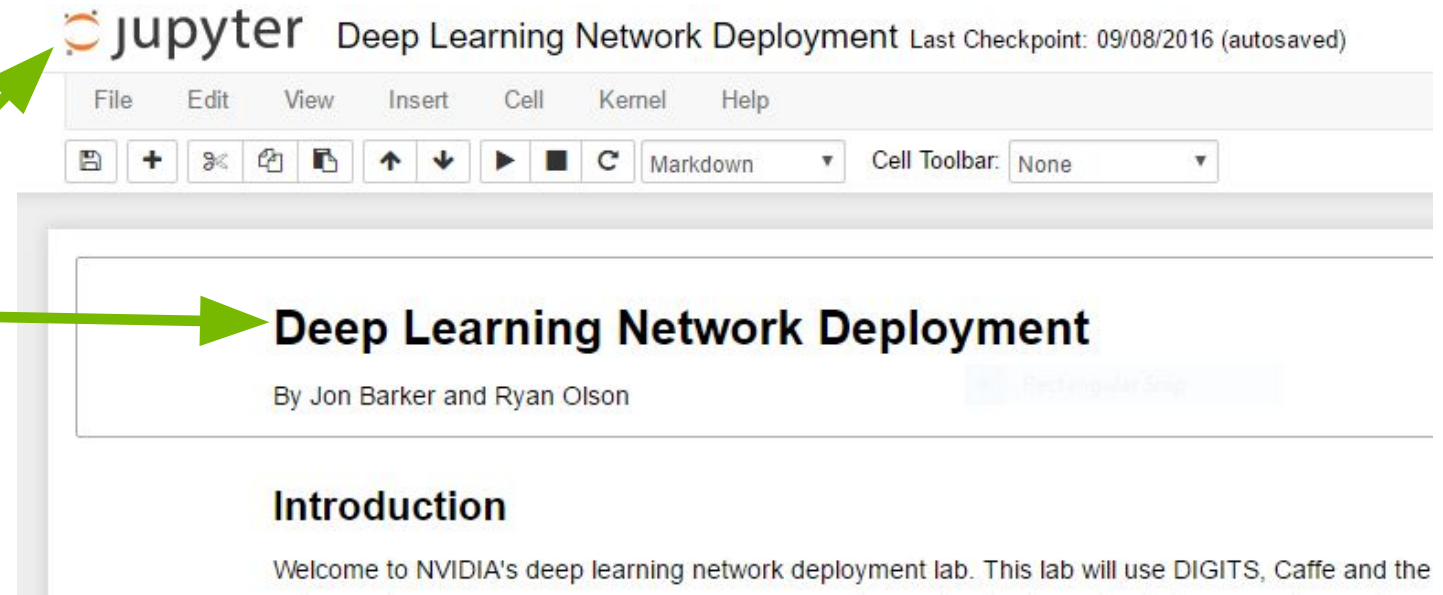
Custom Connection Details

Click [here](#) to launch your lab.

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

CONNECTING TO THE LAB ENVIRONMENT

You should see your
“Deep Learning
Network
Deployment”
Jupyter notebook



The screenshot displays the JupyterLab interface. At the top, the Jupyter logo is followed by the text "jupyter Deep Learning Network Deployment Last Checkpoint: 09/08/2016 (autosaved)". Below this is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, and Help. A toolbar contains icons for file operations (save, new, open, save as), navigation (up, down), execution (run, stop, refresh), and a dropdown menu currently set to "Markdown". To the right of the toolbar is a "Cell Toolbar" dropdown set to "None". The main content area shows the notebook title "Deep Learning Network Deployment" in a large, bold font, followed by the authors "By Jon Barker and Ryan Olson" and a "Restart and Save" button. Below the title is a section header "Introduction" and the beginning of a paragraph: "Welcome to NVIDIA's deep learning network deployment lab. This lab will use DIGITS, Caffe and the".

Jupyter Notebook Introduction

Interface: Run

The screenshot shows a Jupyter Notebook interface in a browser. The browser address bar shows the URL: `ec2-54-226-175-192.compute-1.amazonaws.com/KDD2nSSYWy5F/notebooks/Deep%20Learning%20Network%20Deployment.ipynb#`. The notebook title is "Deep Learning Network Deployment" and it indicates "Last Checkpoint: an hour ago (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with various icons. The "Run" button (a play icon) in the toolbar is circled in blue. Below the toolbar, there is a code cell with the following content:

```
solver.prototxt
Raw caffe output
caffe_output.log
Pretrained Model
/home/moriarty/src/caffe/models/bvlc_googlenet/b
vic_googlenet.caffemodel
```

Feature shape (3, 512, 1024)
Label shape (1, 107, 16)

Similarly if you look at the corresponding dataset in DIGITS you can find the **Job Directory** containing the dataset mean image.

Exercise: Find the job directory for the pedestrian_detectnet model and the pedestrian_dummy_dataset and use them as the MODEL_JOB_DIR and DATA_JOB_DIR parameters in the code cells below to apply DetectNet to a sequence of test frames from the video melbourne.mp4 using pycaffe.

In [2]:

```
# Import required Python Libraries
%pylab inline
pylab.rcParams['figure.figsize'] = (15, 9)
import caffe
import numpy as np
import time
import os
import cv2
from IPython.display import clear_output
```

Populating the interactive namespace from numpy and matplotlib

In [*]:

```
# Configure Caffe to use the GPU for inference
caffe.set_mode_gpu()
```

In []:

```
# Set the model job directory from DIGITS here
MODEL_JOB_DIR='/home/ubuntu/digits/digits/jobs/20160905-143028-2f08'
# Set the data job directory from DIGITS here
DATA_JOB_DIR='/home/ubuntu/digits/digits/jobs/20160905-135347-01d5'
```

STARTING DIGITS

Instruction in
Jupyter notebook
will link you to
DIGITS



Using DIGITS, anyone can easily get started and interactively train their NVIDIA, located here: <https://github.com/NVIDIA/DIGITS>. However, DIGITS

Inference using DIGITS

Now click [here](#) to open DIGITS in a separate tab. If at any time DIGITS a
The DIGITS server you will see running contains two neural networks list

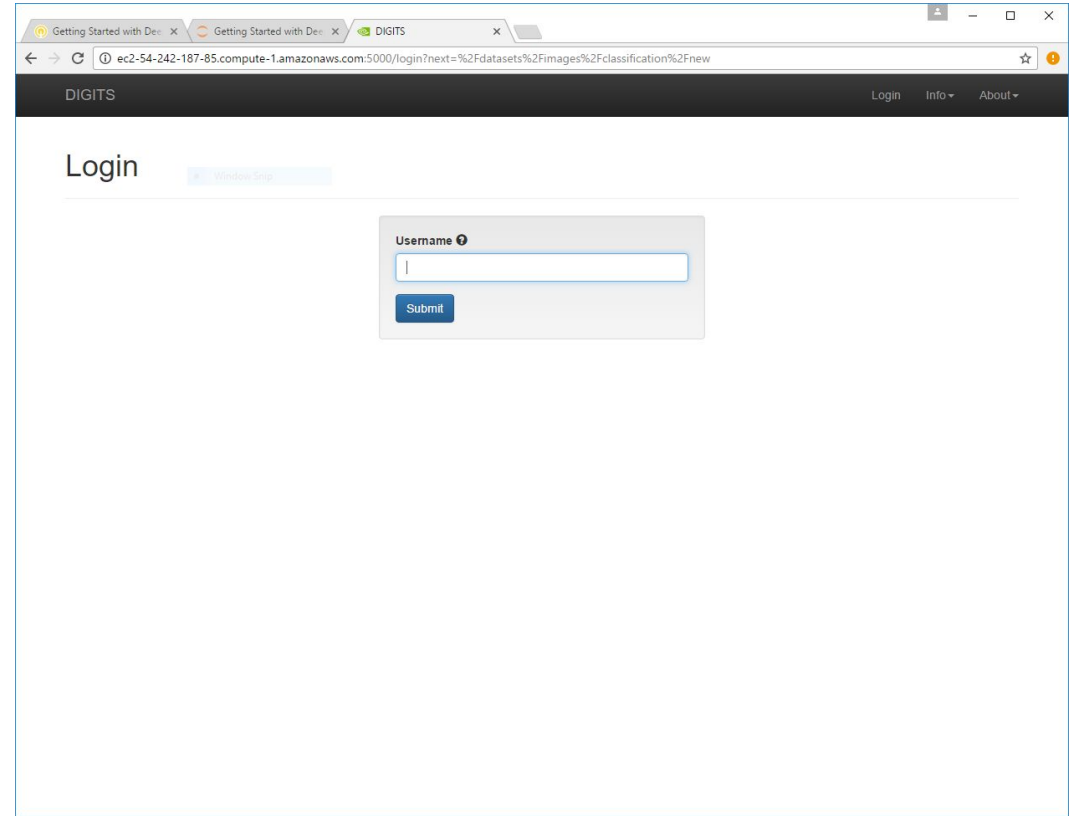
Home

Group Jobs: 

No Jobs Running

ACCESSING DIGITS

- Will be prompted to enter a username to access DIGITS
 - Can enter any username
 - Use lower case letters



REVIEW / NEXT STEPS

WHAT'S NEXT

- Use / practice what you learned
- Discuss with peers practical applications of DNN
- Reach out to NVIDIA and the Deep Learning Institute
- Look for local meetups
- Follow people like Andrej Karpathy and Andrew Ng

WHAT'S NEXT

TAKE SURVEY

...for the chance to win an NVIDIA SHIELD TV.

Check your email for a link.

ACCESS ONLINE LABS

Check your email for details to access more DLI training online.

ATTEND WORKSHOP

Visit www.nvidia.com/dli for workshops in your area.

JOIN DEVELOPER PROGRAM

Visit <https://developer.nvidia.com/join> for more.

GTC AROUND THE WORLD

GTC CHINA

BEIJING

SEPTEMBER 25 -27, 2017

GTC EUROPE

MUNICH

OCTOBER 10 - 12, 2017

GTC ISRAEL

TEL AVIV

OCTOBER 18, 2017

GTC DC

WASHINGTON, DC

NOVEMBER 1 - 2, 2017

GTC JAPAN

TOKYO

DECEMBER 12 - 13, 2017

GTC 2018

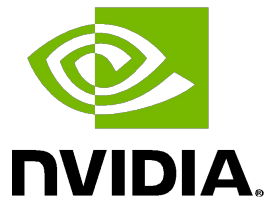
SILICON VALLEY

MARCH 26 - 29, 2018

WWW.GPUTECHCONF.COM

Instructor: Charles Killam, LP.D.

TAs: Will Ramey



DEEP
LEARNING
INSTITUTE

www.nvidia.com/dli

APPENDIX

Lab Debug

Can't display Ipython Notebook?

IPython Notebook

- Chrome/Firefox/Safari recommended. IE will work but not as well
- Websockets are required - you can test at websocketstest.com
 - Look for this result:
- Execute cells with ctrl+enter or pressing play button
-

WebSockets (Port 80)	
Connected	Yes ✓
Data Receive	Yes ✓
Data Send	Yes ✓
Echo Test	Yes ✓
Server time	2016/04 02:42:20

Lab Debug

Don't know if cell is running??

You should see In[*] and not In[] or In[<some number>].

Solid grey circle in the top-right of the browser window

If you only see #1 and not #2, then you need to try the following in order:

Press the stop button on the toolbar. Try again.

Click Kernel -> Restart. Try again.

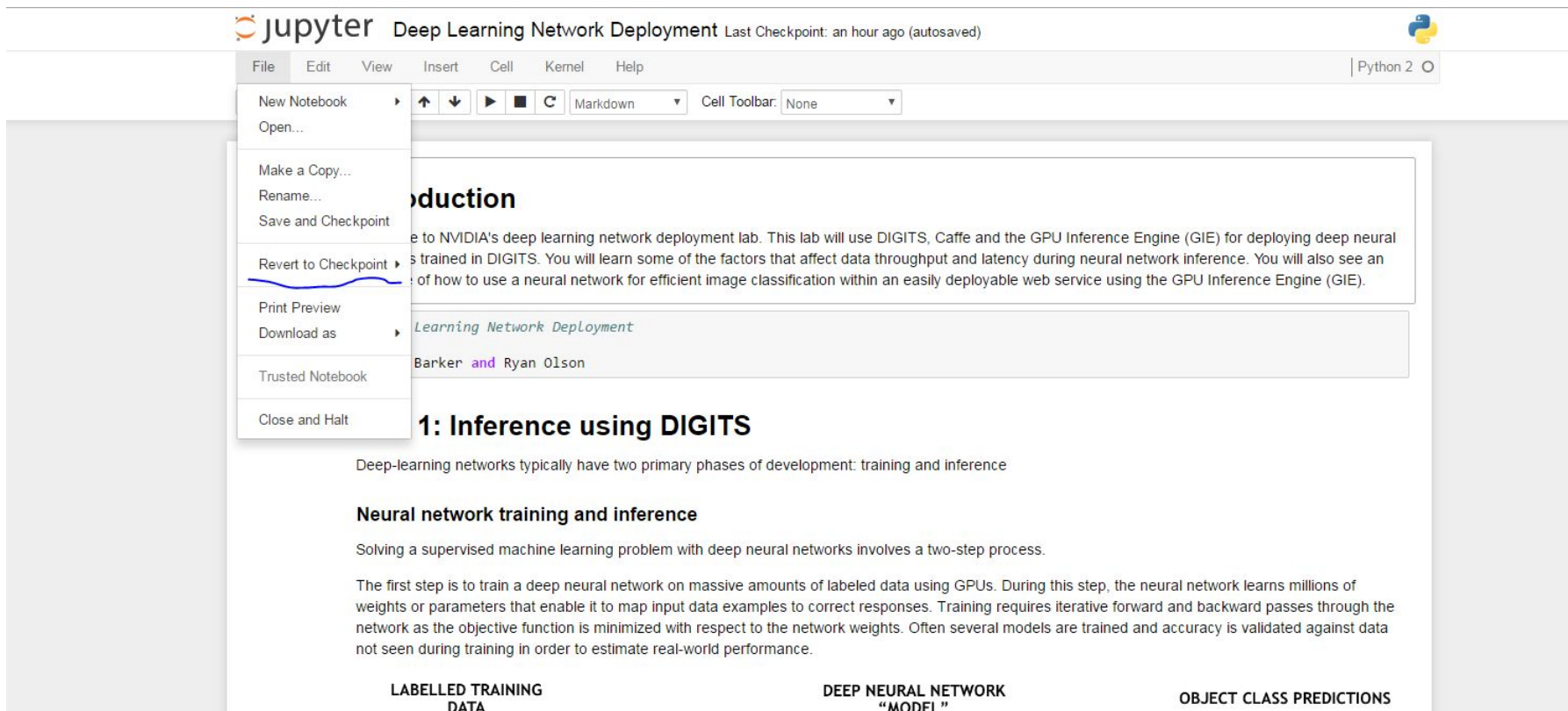
Save the Notebook and refresh the page. Try again.

End the lab from the qwikLABS page and start a new instance. All work will be lost.

(Please let me know before you do this)

Lab Debug

Reverse to some checkpoint



The screenshot shows a Jupyter Notebook titled "Deep Learning Network Deployment" with a "Last Checkpoint: an hour ago (autosaved)" status. The "File" menu is open, and the "Revert to Checkpoint" option is highlighted with a blue underline. The notebook content includes an introduction to NVIDIA's deep learning network deployment lab, a section titled "1: Inference using DIGITS", and a diagram illustrating the inference process.

Introduction

...to NVIDIA's deep learning network deployment lab. This lab will use DIGITS, Caffe and the GPU Inference Engine (GIE) for deploying deep neural networks trained in DIGITS. You will learn some of the factors that affect data throughput and latency during neural network inference. You will also see an example of how to use a neural network for efficient image classification within an easily deployable web service using the GPU Inference Engine (GIE).

Learning Network Deployment

Barker and Ryan Olson

1: Inference using DIGITS

Deep-learning networks typically have two primary phases of development: training and inference

Neural network training and inference

Solving a supervised machine learning problem with deep neural networks involves a two-step process.

The first step is to train a deep neural network on massive amounts of labeled data using GPUs. During this step, the neural network learns millions of weights or parameters that enable it to map input data examples to correct responses. Training requires iterative forward and backward passes through the network as the objective function is minimized with respect to the network weights. Often several models are trained and accuracy is validated against data not seen during training in order to estimate real-world performance.

LABELLED TRAINING DATA DEEP NEURAL NETWORK "MODEL" OBJECT CLASS PREDICTIONS