



DEEP
LEARNING
INSTITUTE

Neural Network Deployment with DIGITS and TensorRT

Twin Karmakarm

Certified Instructor, NVIDIA Deep Learning Institute



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

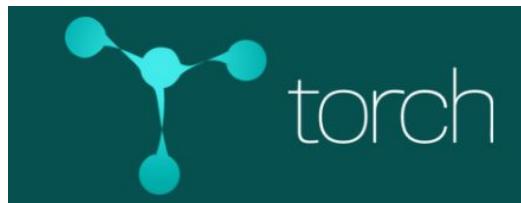
- Caffe
- NVIDIA'S DIGITS
- Deep Learning Approach
- NVIDIA'S TensorRT
- Lab
 - Lab Details
 - Launching the Lab Environment
- Review / Next Steps

CAFFE

Frameworks

Many Deep Learning Tools

Caffe



theano



▪ ▪ ▪

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

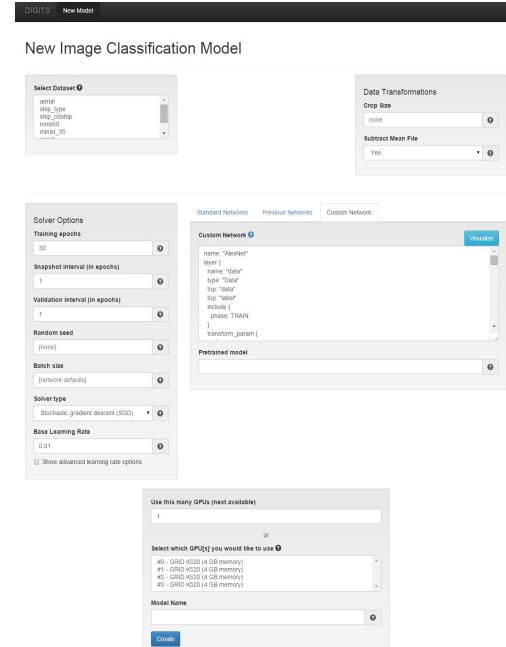
NVIDIA'S DIGITS

Interactive Deep Learning GPU Training System

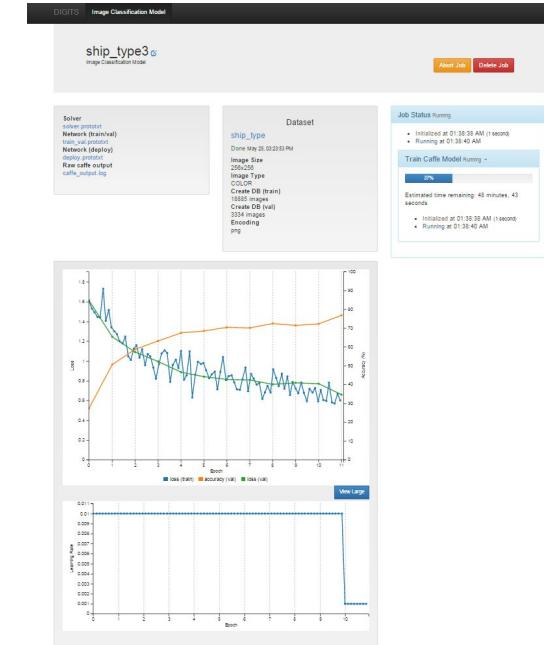
Process Data



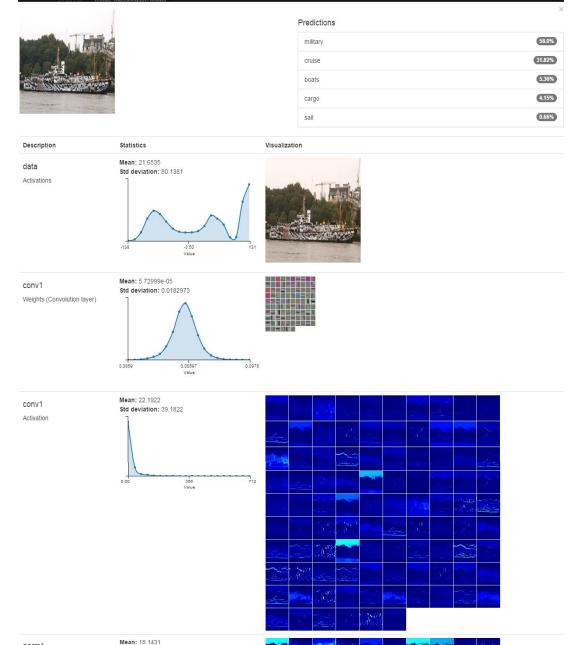
Configure DNN



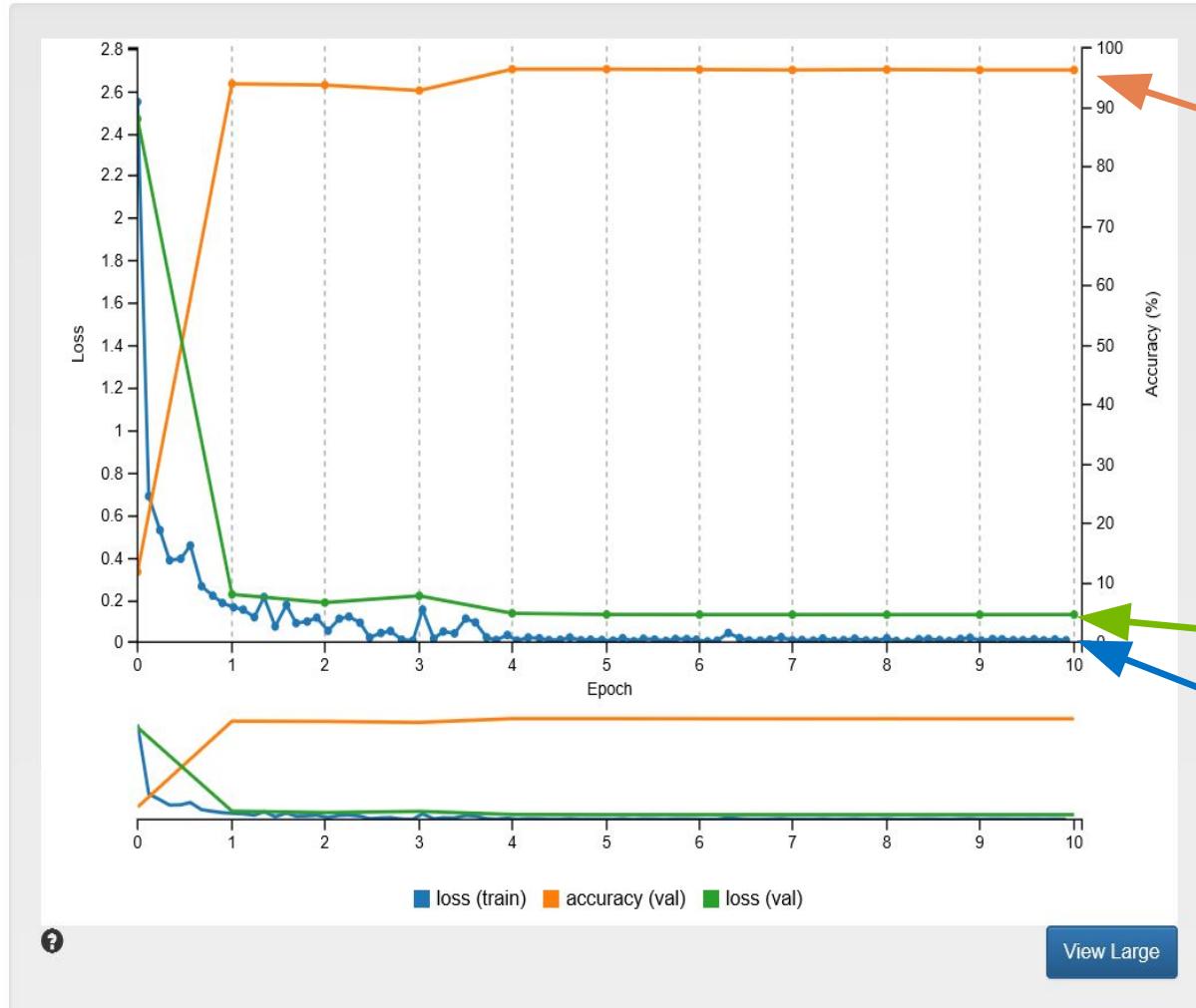
Monitor Progress



Visualization



NVIDIA'S DIGITS



Accuracy obtained from validation dataset

Loss function (Validation)

Loss function (Training)

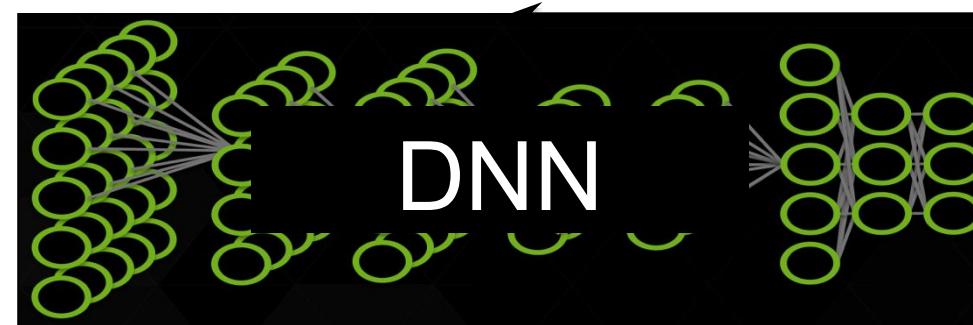
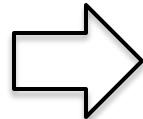
View Large



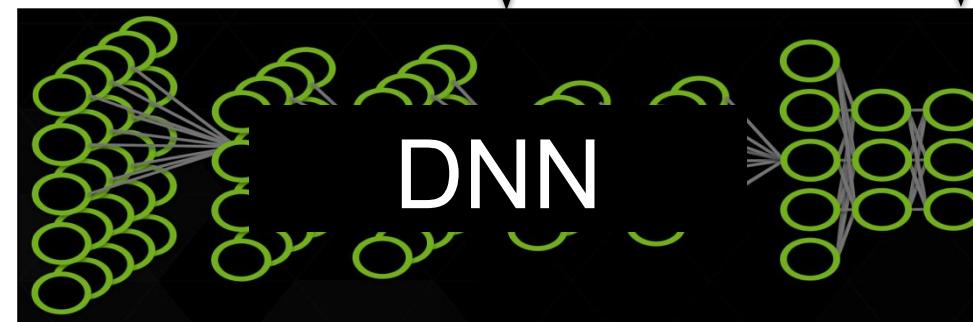
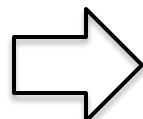
DEEP LEARNING APPROACH

Deep Learning Approach

Train:



Deploy:



Deep Learning Approach

Convolutional Neural Network

IMAGES



Conv

Pool

Conv

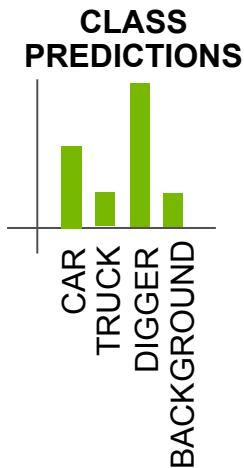
Pool

Conv

Pool

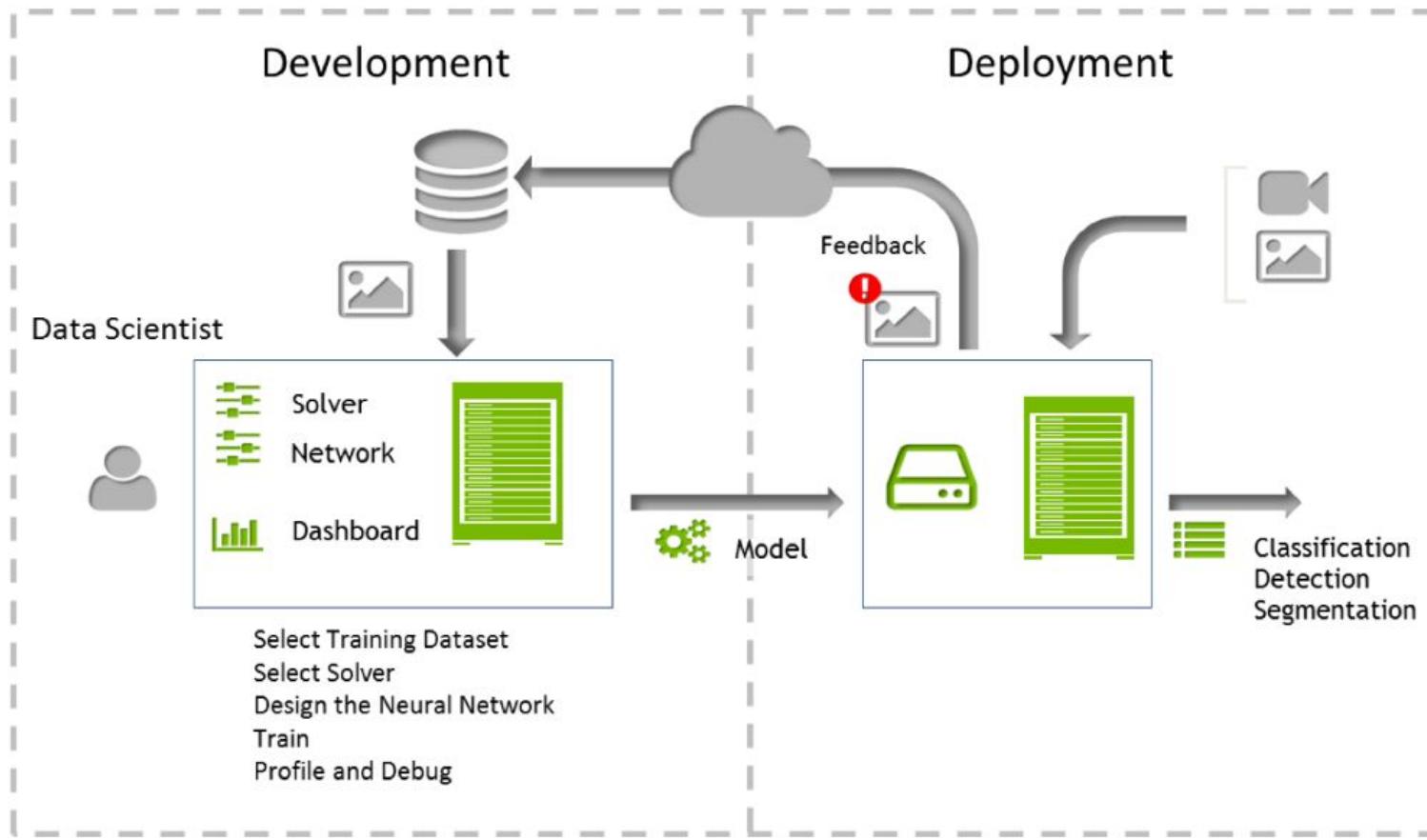
Fully connected

Fully connected



Deep Learning Approach

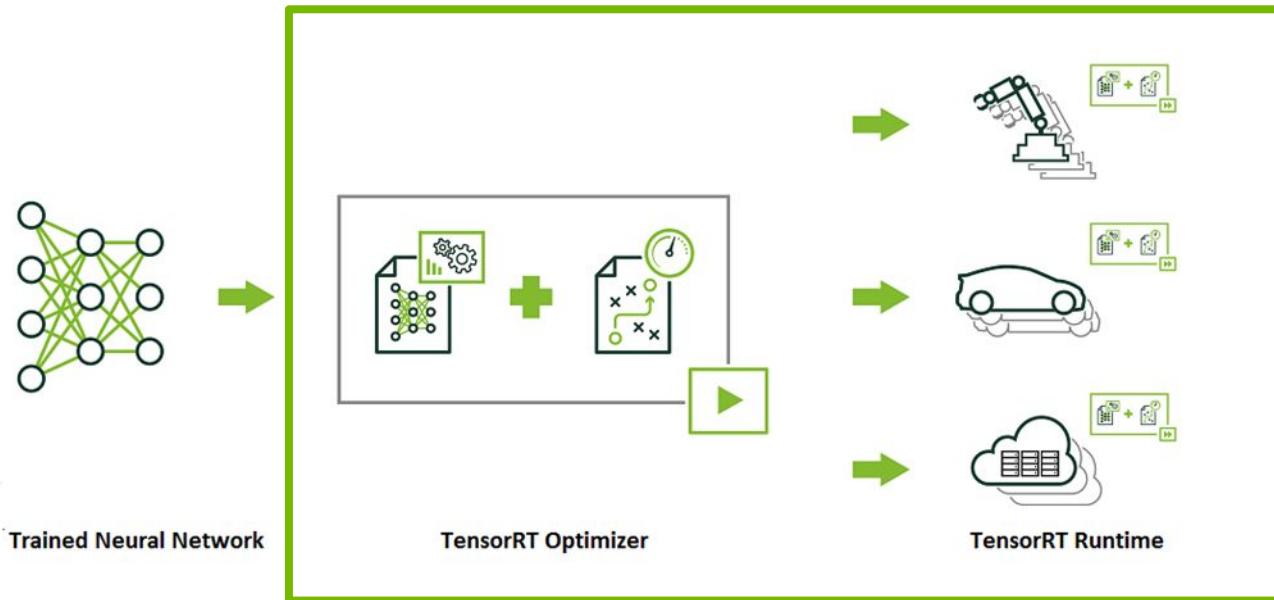
Neural network training and inference



NVIDIA'S TENSORRT

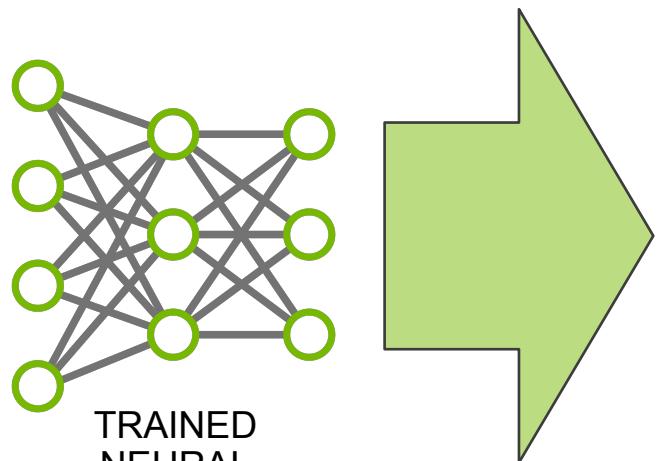
TensorRT

- 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



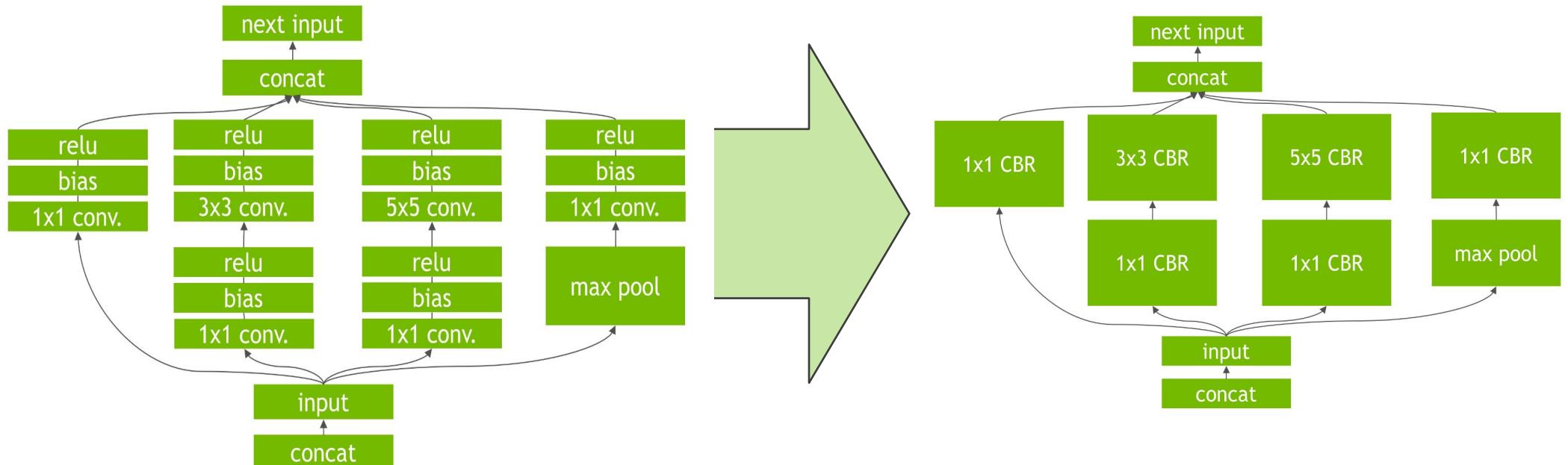
- **Fuse network layers**
- **Eliminate concatenation layers**
- **Kernel specialization**
- **Auto-tuning for target platform**
- **Select optimal tensor layout**
- **Batch size tuning**



OPTIMIZED
INFERENCE
RUNTIME

TensorRT Optimizer

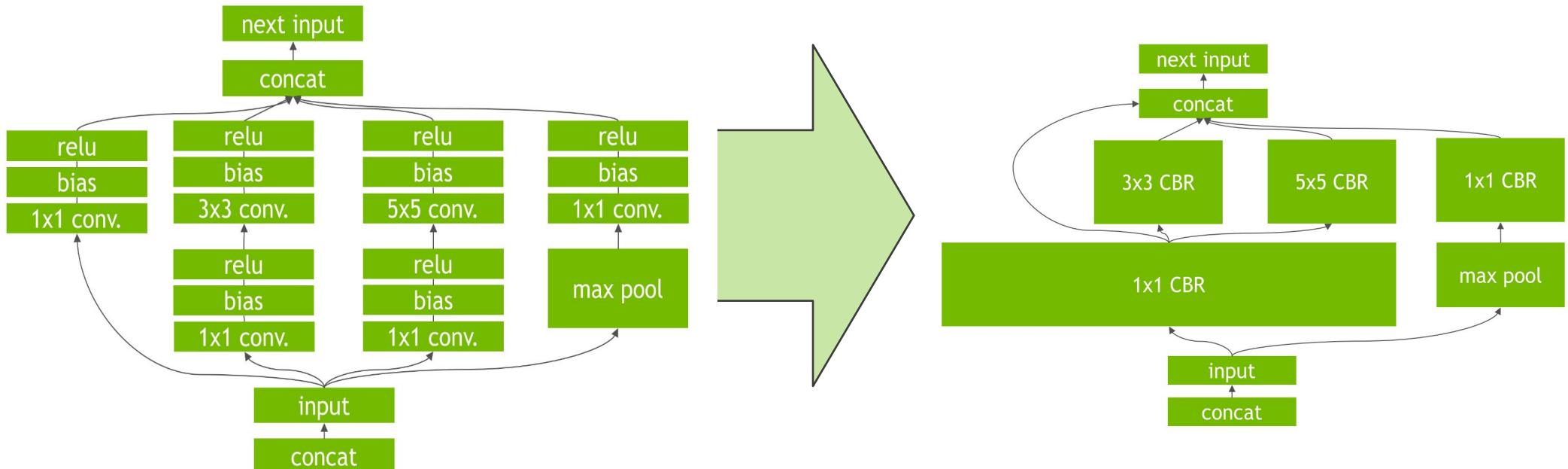
Vertical Layer Fusion



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

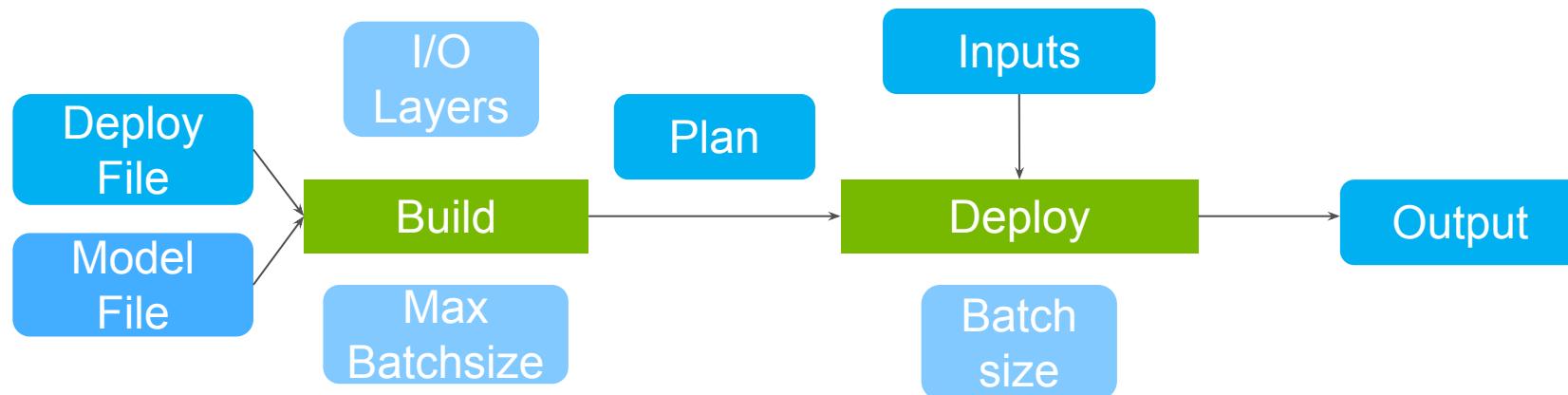
TensorRT Optimizer

- **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

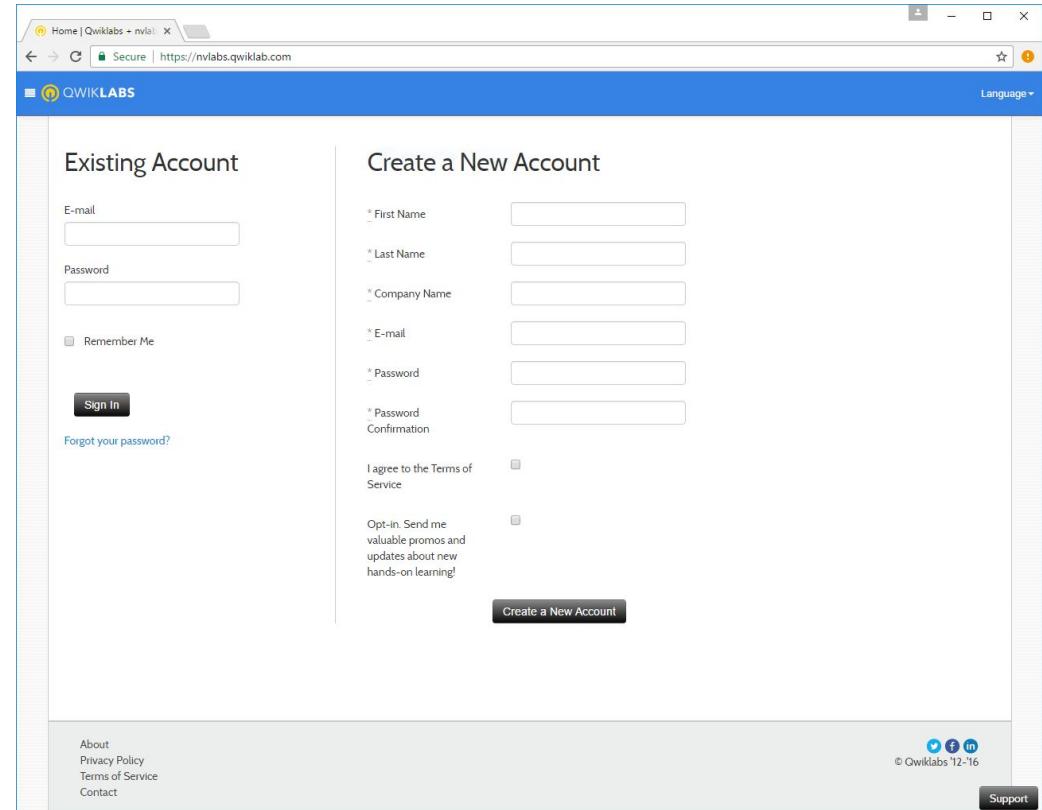
- GPU Inference Engine (GIE) = TensorRT
- Part 1: Inference using DIGITS
 - Will use existing model in DIGITS to perform inference on a single image
- Part 2: Inference using Pycaffe
 - Programming production-like deployable inference code
- Part 3: NVIDIA TensorRT
 - Will run TensorRT Optimizer to build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



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 shows a user interface for a learning platform. At the top, there is a header with the following information:

- In-Session Class: Deep Learning Labs
- 36.5 Total Hours
- 21 Completed Labs
- 4 Classes Taken

Below the header, there is a section titled "Class Details" which lists several classes:

- Introduction to Deep Learning
- Approaches to Object Detection using DIGITS
- Identifying Whale Sounds with Audio Classification
- Deep Learning Network Deployment** (This class is highlighted with a green background)
- Introduction to RNNs
- Exploring TensorFlow on GPUs
- Introduction to Deep Learning with R and MXNet

To the right of the class list, there is a detailed description of the selected class, "Deep Learning Network Deployment":

nvidia Deep Learning Network Deployment Select

Deep learning software frameworks leverage GPU acceleration to train deep neural networks (DNNs). But what do you do with a DNN once you have trained it? The process of applying a trained DNN to new test data is often referred to as ‘inference’ or ‘deployment’. In this lab you will test three different approaches to deploying a trained DNN for inference. The first approach is to directly use inference functionality within a deep learning framework, in this case DIGITS and Caffe. The second approach is to integrate inference within a custom application by using a deep learning framework API, again using Caffe but this time through its Python API. The final approach is to use the NVIDIA High Performance GPU Inference Engine (TensorRTGE) which will automatically create an optimized inference run-time from a trained Caffe model and network description file. You will learn about the role of each of these approaches and how they can be used to build a production system.

Duration: 90 min.

Access Time: 115 min.

Setup Time: 6 min.

Level: Beginner

LAUNCHING THE LAB ENVIRONMENT

The screenshot shows a web-based interface for launching a lab environment. At the top, there's a header with the text "In-Session Class: Deep Learning Labs", a clock icon, "36.5 Total Hours", a completed labs icon with the number "21", and a classes taken icon with the number "4". Below this is a green bar labeled "Class Details". A list of lab topics is shown, with the first one, "Deep Learning Network Deployment", highlighted in green. To the right of this list is a detailed description of the selected lab, titled "nVIDIA Deep Learning Network Deployment". The description text explains that the lab covers deploying trained DNNs for inference using DIGITS and Caffe, or integrating inference into a custom application via a Python API, or using the NVIDIA High Performance GPU Inference Engine (TensorRTGE). It also specifies a duration of 90 min., access time of 115 min., setup time of 6 min., and a level of Beginner. A blue "Select" button is located at the top right of this description box.

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

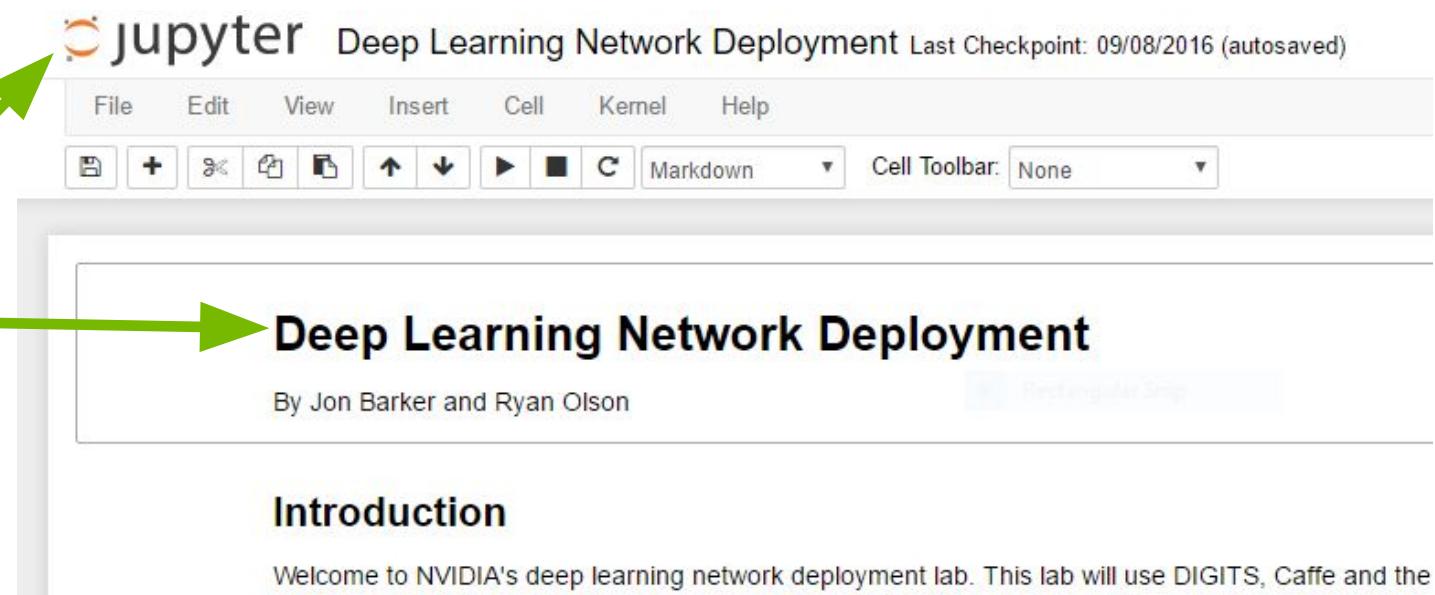


7. Click on “here” to access your lab environment / Jupyter notebook
-
- A green arrow originates from the word "here" in the step 7 list and points to the word "here" in the "Custom Connection Details" section of the screenshot.

A screenshot of the 'Custom Connection Details' section. It contains a 'Connect' button, a 'Lab Connection' section with instructions, a warning message about data transmission, and a 'Custom Connection Details' section with a 'Click here to launch your lab.' link. A green arrow points to this 'here' link.

CONNECTING TO THE LAB ENVIRONMENT

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



Jupyter Notebook Introduction

Interface: Run

The screenshot shows a Jupyter Notebook interface running on a web browser. The title bar indicates it's an IPython Notebook titled "Deep Learning Network Deployment". The toolbar includes standard options like File, Edit, View, Insert, Cell, Kernel, Help, and a Python 2 kernel selector. A "Cell Toolbar" dropdown is open, showing options like "Code" (selected), "Text", "Markdown", "Raw", and "None". A "Cell" icon in the toolbar is circled in blue.

Below the toolbar, there are two code cells:

- 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'
```

On the right side of the notebook, there are two status boxes:

- A left box contains file paths: "solver.prototxt", "Raw caffe output", "caffe_output.log", "Pretrained Model", and "Pretrained Model" again, pointing to "/home/morlarity/src/caffe/models/bvlc_googlenet/bvlc_googlenet.caffemodel".
- A right box shows memory usage: "Feature shape (3, 512, 1024)" and "Label shape (1, 107, 16)".

Handwritten annotations include a blue circle around the "Cell" icon in the toolbar, another blue circle around the first code cell, and a blue circle around the first cell's output area.

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, DIGIT

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

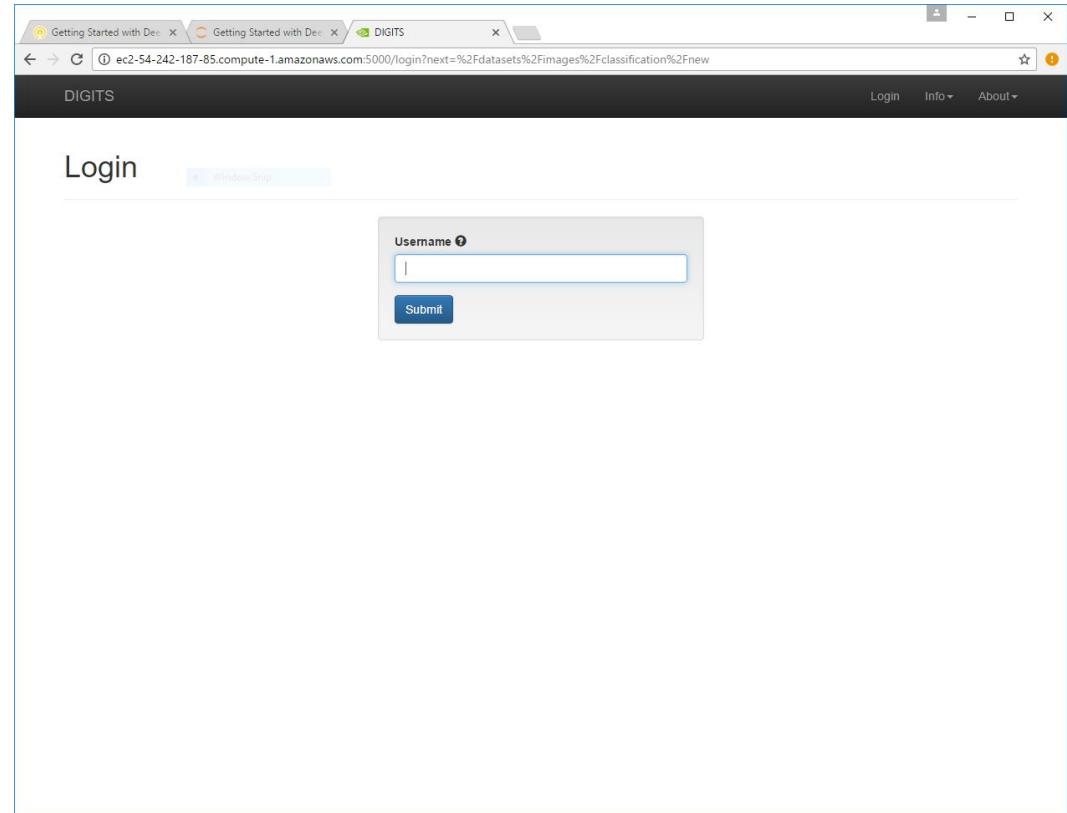
Home

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



www.nvidia.com/dli

DEEP
LEARNING
INSTITUTE

Instructor: Twin Karmakharan

Join the Conversation
#GTC18



CONNECT

Connect with technology experts from NVIDIA and other leading organisations.



LEARN

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



DISCOVER

Discover the latest breakthroughs in fields such as autonomous vehicles, HPC, smart cities, VR, robotics, and more.



INNOVATE

Hear about disruptive innovations as startups and researchers present their work.

USE CODE NVMDIERINGER TO SAVE 25% | REGISTER AT WWW.GPUTECHCONF.EU

Join us at Europe's premier conference on artificial intelligence.

9-11 October 2018 at the International Congress Centre, Munich.

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	2016/02/04 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 interface with the title "jupyter Deep Learning Network Deployment". The "File" menu is open, and the "Revert to Checkpoint" option is highlighted with a blue underline. The main content area displays a section titled "Introduction" which describes the lab's purpose and objectives. Below this, there is a section titled "1: Inference using DIGITS" with a sub-section titled "Neural network training and inference". At the bottom, there are three labels: "LABELLED TRAINING DATA", "DEEP NEURAL NETWORK MODEL", and "OBJECT CLASS PREDICTIONS".

jupyter Deep Learning Network Deployment Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Help Python 2

New Notebook Open... Make a Copy... Rename... Save and Checkpoint Revert to Checkpoint Print Preview Download as Trusted Notebook Close and Halt

Introduction

This lab will use 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