



DEEP  
LEARNING  
INSTITUTE



ODTÜ  
METU

# Neural Network Deployment with DIGITS and TensorRT

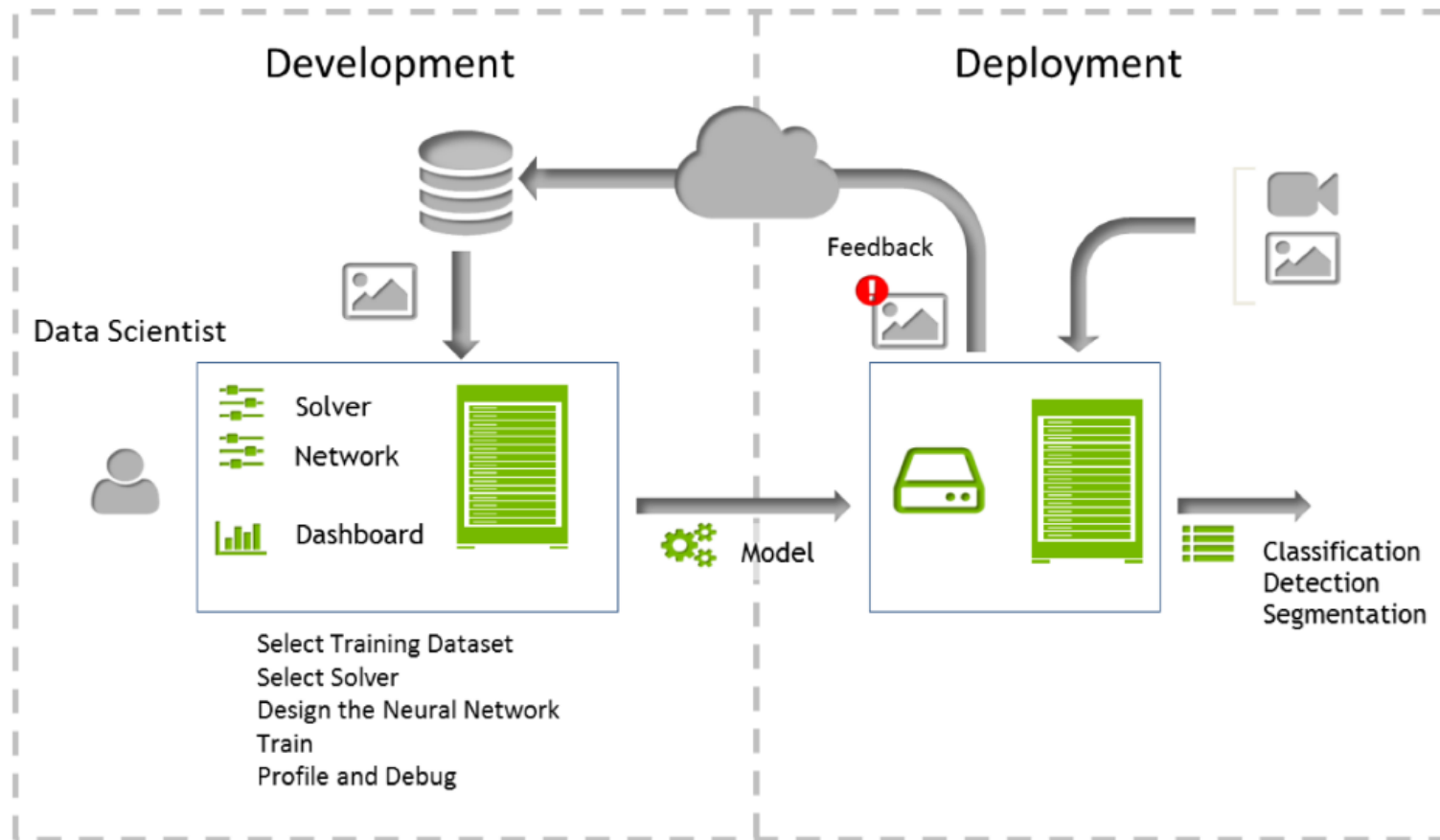
---

Dr. Alptekin Temizel  
DLI Certified Instructor  
Associate Professor, Graduate School of Informatics, METU

8 January 2018

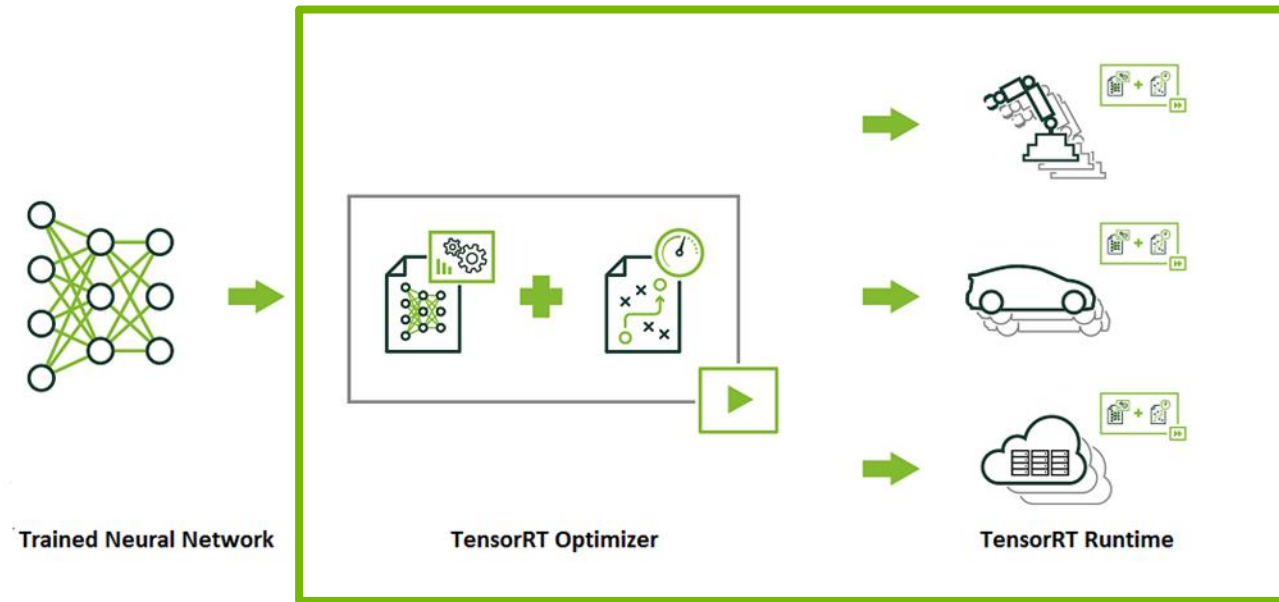
# Deep Learning Approach

## Neural network training and inference



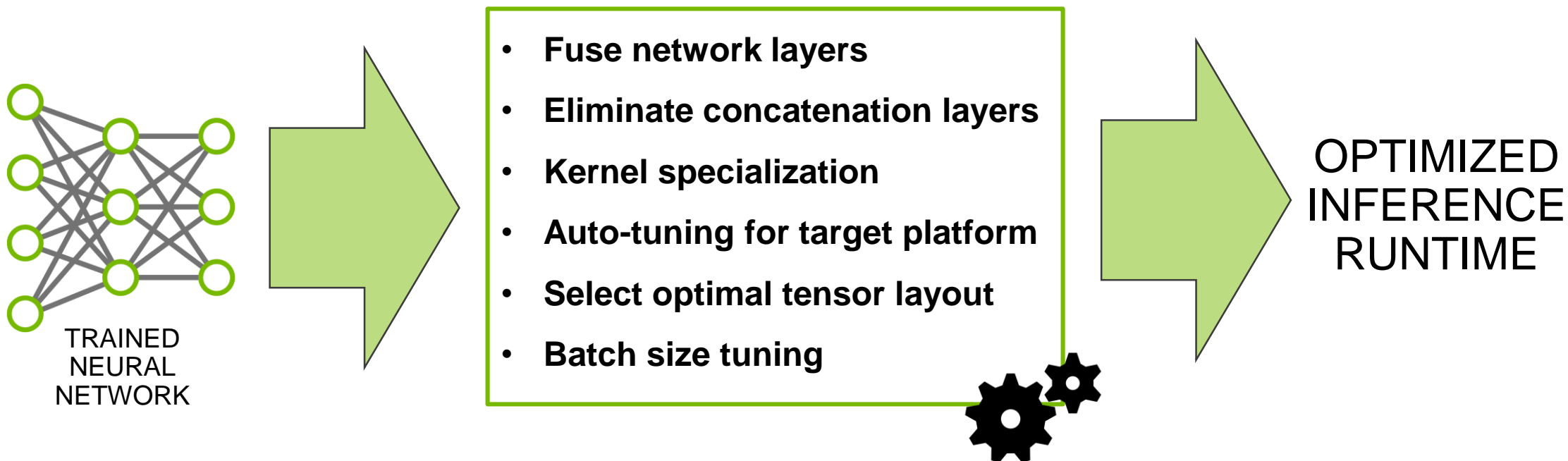
# TensorRT

- Inference engine for production deployment of deep learning applications



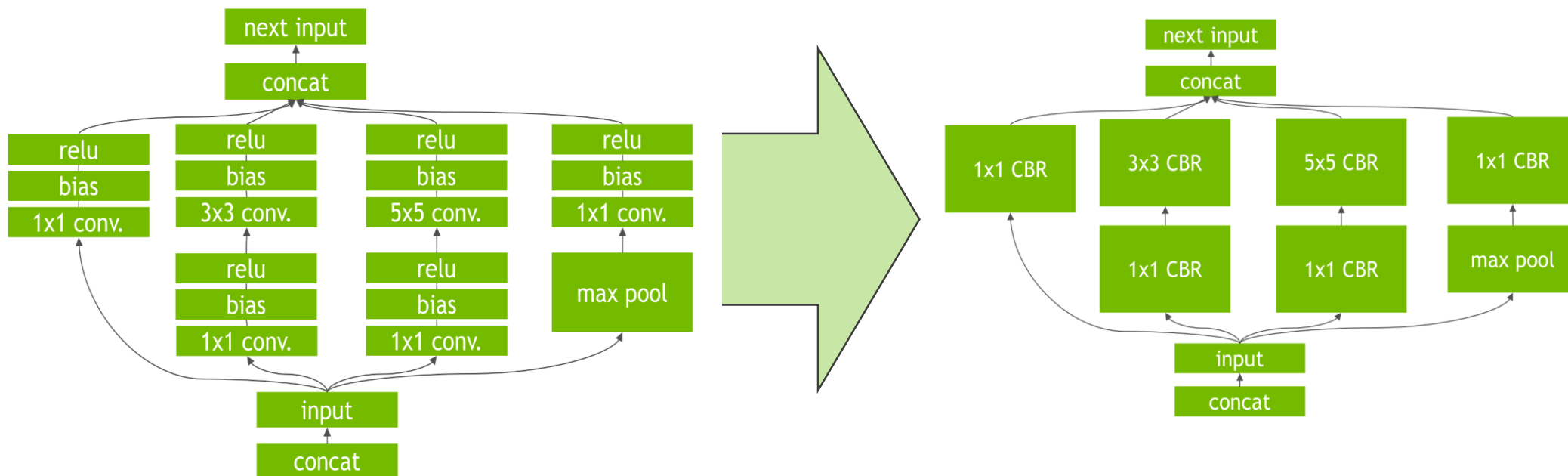
- Allows developers to focus on developing AI powered applications
  - TensorRT ensures optimal inference performance

# TensorRT Optimizer



# TensorRT Optimizer

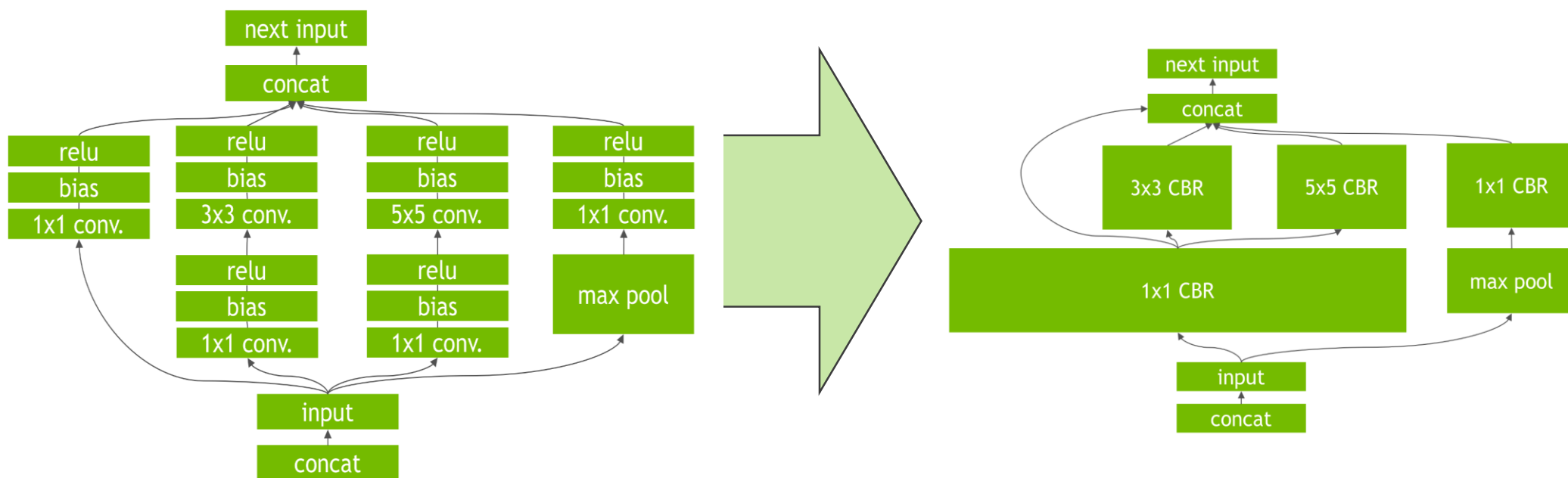
## Vertical Layer Fusion



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

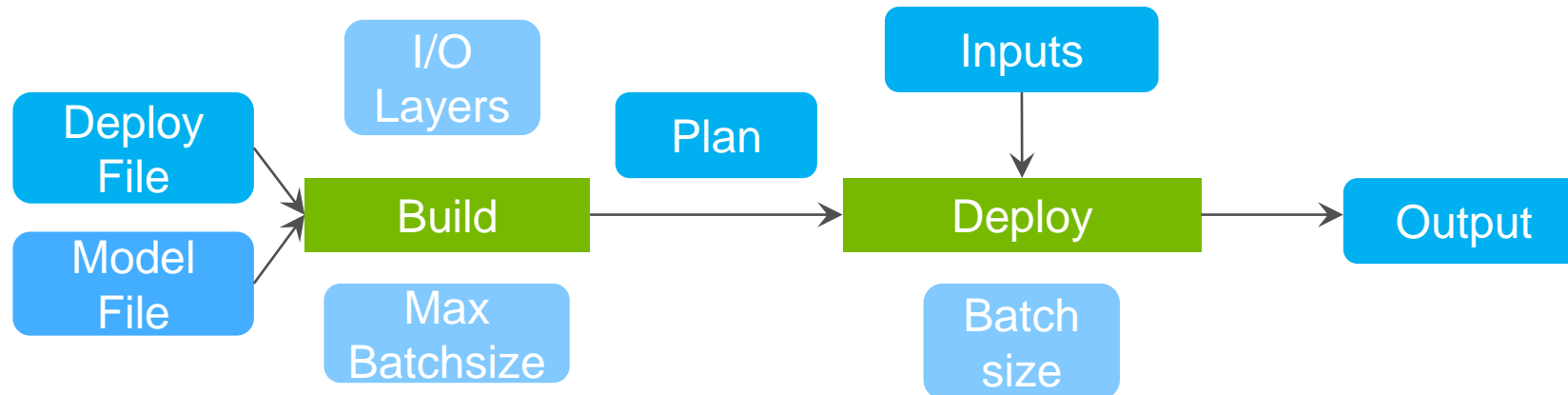
- Scalability:
  - Output/Input Layers can connect with other deep learning framework directly
    - Caffe, Theano, Torch, TensorFlow
- Reduced Latency:
  - INT8 or FP16
    - INT8 delivers 3X more throughput compared to FP32
    - INT8 uses 61% less memory compared to FP32



# 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** is a runtime (serialized) object
  - 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
- Trained for the task of pedestrian detection using a large dataset of pedestrians in a variety of indoor and outdoor scenes

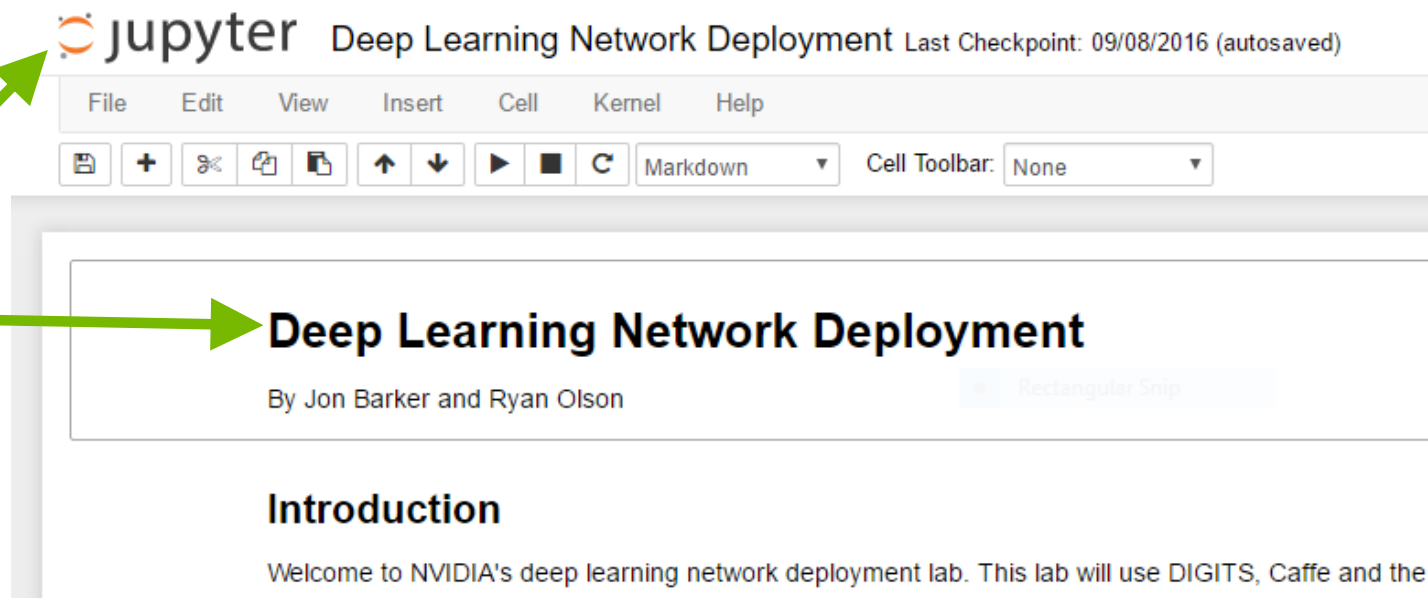
# Lab Tasks

- 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

# CONNECTING TO THE LAB ENVIRONMENT

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



The screenshot shows a Jupyter Notebook 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 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. Under the menu bar is a toolbar with icons for saving, adding, undo, redo, copy, paste, up, down, run, 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 of the notebook displays the title 'Deep Learning Network Deployment' in a large, bold font. Below the title, it says 'By Jon Barker and Ryan Olson'. Further down, the section 'Introduction' is visible, with the text 'Welcome to NVIDIA's deep learning network deployment lab. This lab will use DIGITS, Caffe and the'.

**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](http://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](http://WWW.GPUTECHCONF.COM)

# 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](http://websocketstest.com)

- Look for this result:

WebSockets (Port 80)	
Connected	Yes ✓
Data Receive	Yes ✓
Data Send	Yes ✓
Echo Test	Yes ✓
Server time	2015/04/02 02:42:20

- Execute cells with ctrl+enter or pressing play button

# 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.

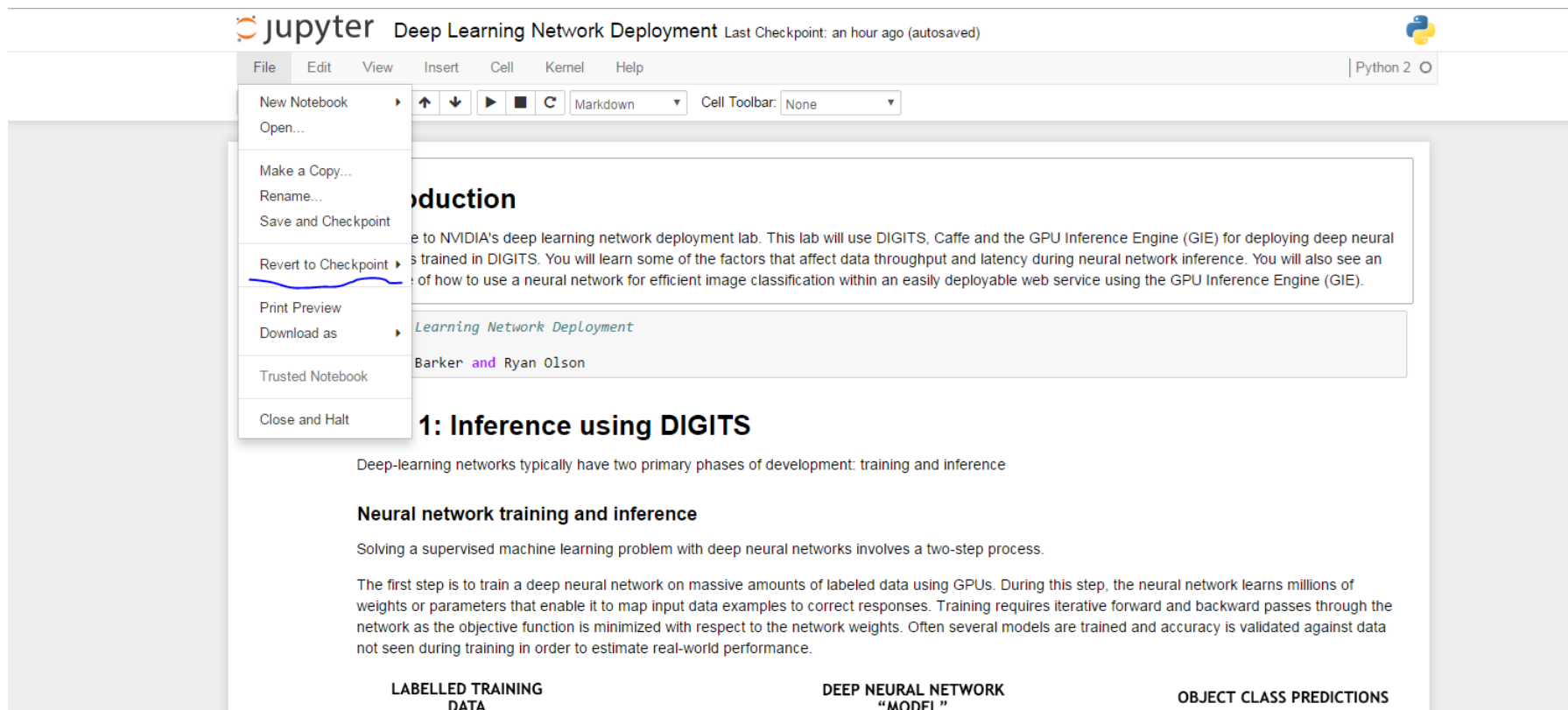
- Click Kernel -> Restart. Try again.

- 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

## Revert to some checkpoint



The screenshot shows a JupyterLab interface titled "Deep Learning Network Deployment" with a status bar indicating "Last Checkpoint: an hour ago (autosaved)". The "File" menu is open, and the "Revert to Checkpoint" option is highlighted with a blue underline. The menu also includes options like "New Notebook", "Open...", "Make a Copy...", "Rename...", "Save and Checkpoint", "Print Preview", "Download as", "Trusted Notebook", and "Close and Halt".

The background content is a notebook page titled "Introduction" by "Barker and Ryan Olson". It describes the use of NVIDIA's deep learning network deployment lab, DIGITS, and the GPU Inference Engine (GIE) for deploying deep neural networks. The page includes a section titled "1: Inference using DIGITS" and a diagram illustrating the workflow from "LABELLED TRAINING DATA" to "DEEP NEURAL NETWORK 'MODEL'" to "OBJECT CLASS PREDICTIONS".

**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**