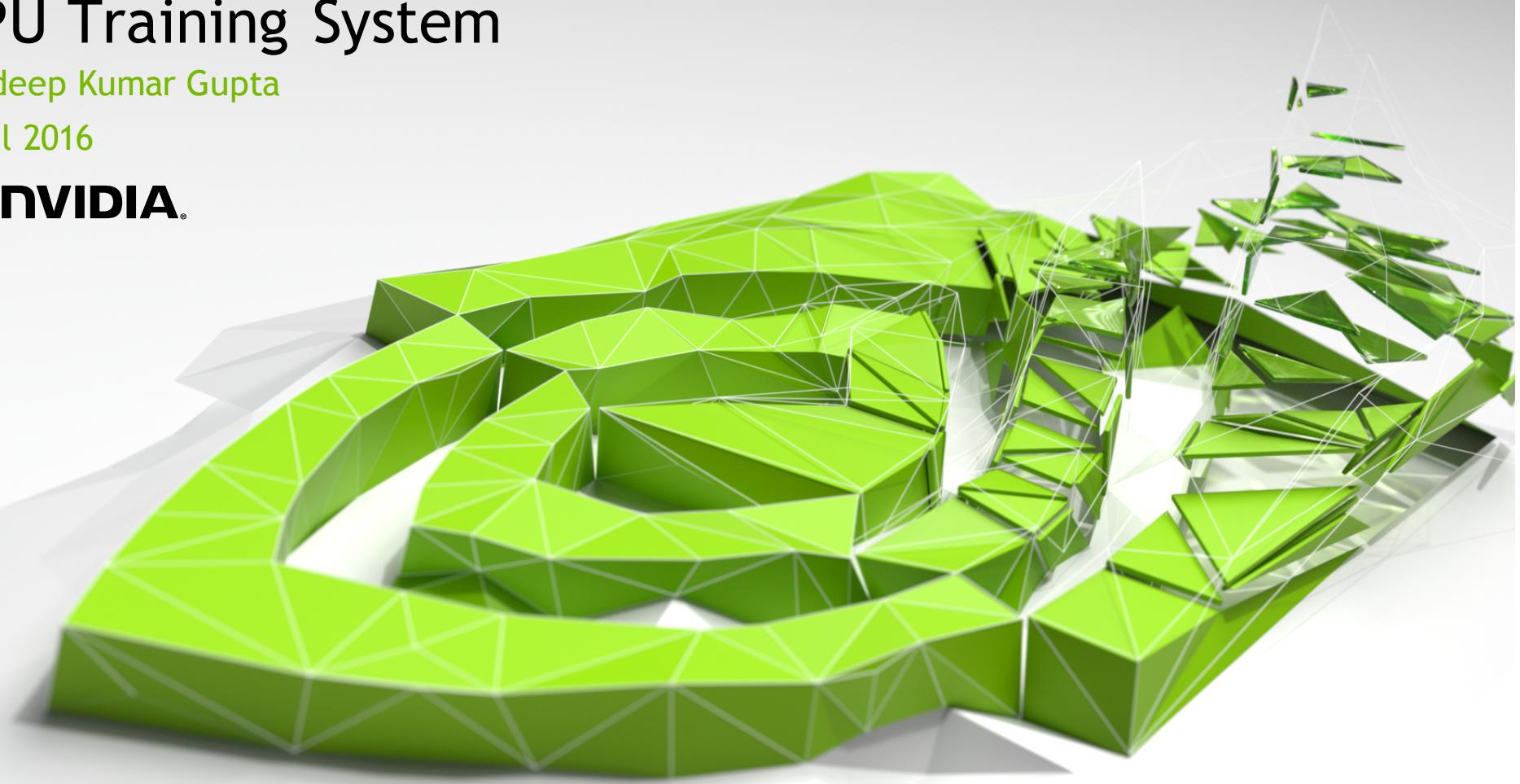


DIGITS- Interactive Deep Learning GPU Training System

Pradeep Kumar Gupta

April 2016



Agenda

- Introduction to DIGITS
- Install and start DIGITS sever
- CREATING DATASETS
- TRAIN A NETWORK
- MODIFY YOUR NETWORK
- Testing/Classification with DIGITS
- DIGITS Features at a Glance
- References

“ DIGITS makes it way easier to design the best network for the job. The DIGITS interface makes it super easy to track key diagnostics during training. The field will definitely benefit from having tools like this for configuration and introspection ”

– Simon Osindero, AI Architect at Flickr

INTRODUCTION TO DIGITS

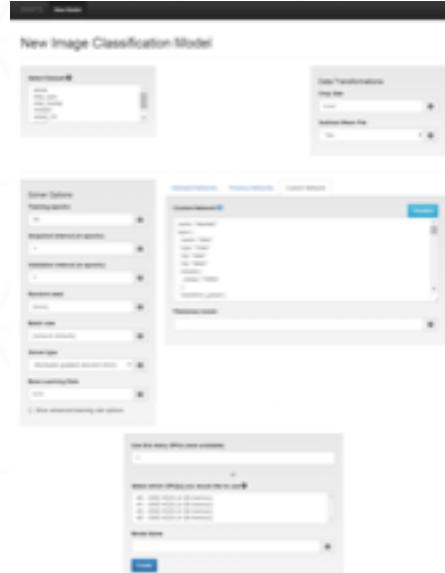
NVIDIA DIGITS

Interactive Deep Learning GPU Training System

Process Data



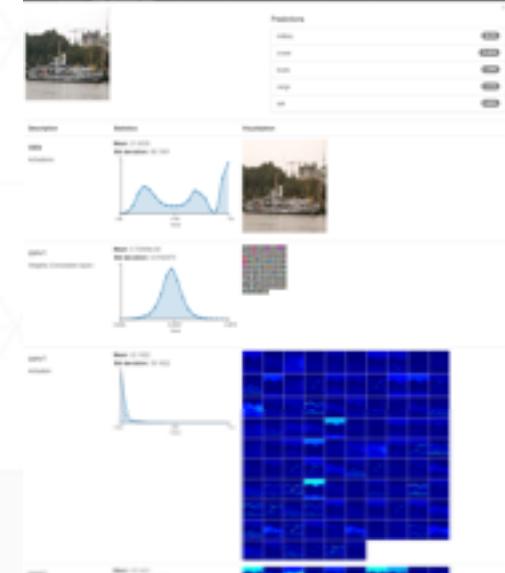
Configure DNN



Monitor Progress

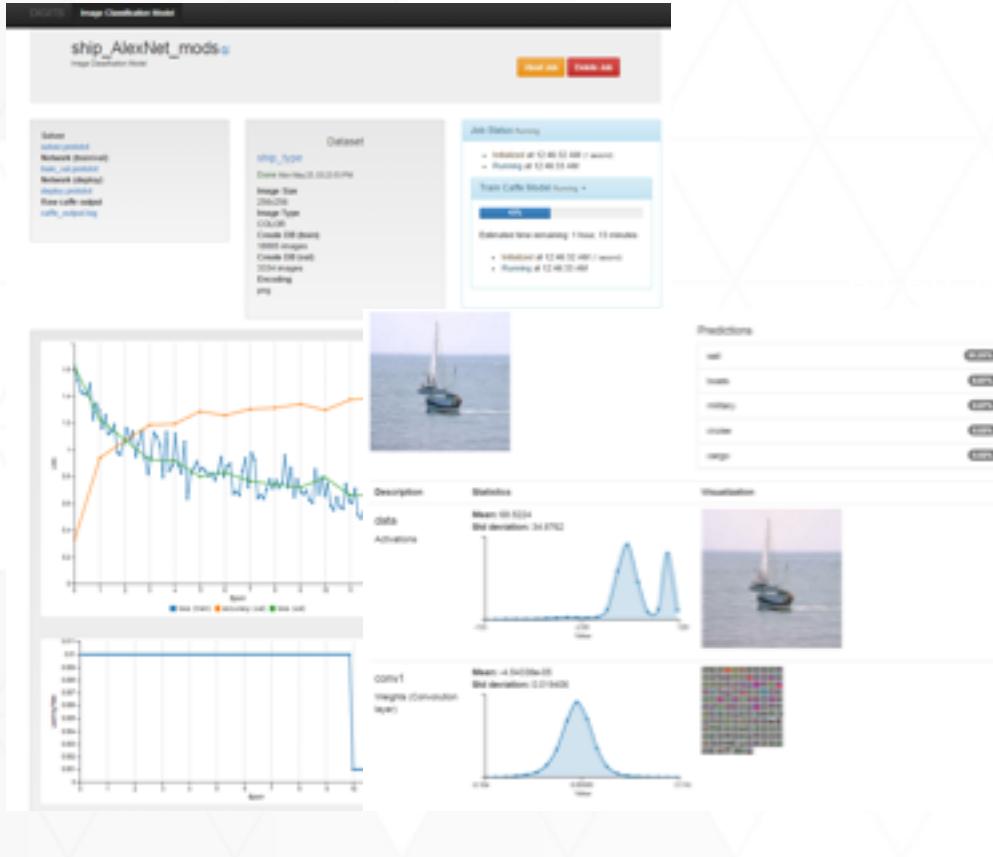


Visualization



NVIDIA DIGITS

Who is DIGITS for?



Data Scientists & Researchers:

- Quickly design the best deep neural network (DNN) for your data
- Monitor DNN training quality in real-time
- Manage training of many DNNs in parallel on multi-GPU systems, and multi-GPU training

DIGITS

Deep Learning GPU Training System

Available at <http://developer.nvidia.com/digits>

Free to use, Source Code available at Github,
latest branch v3.0

<https://github.com/NVIDIA/DIGITS>

Current release supports classification on images

Future versions: More problem types and data
formats (video, speech)

The screenshot shows the DIGITS web interface running in Mozilla Firefox. The title bar says "DIGITS - Mozilla Firefox". The main content area is titled "Home". On the left, there's a "Datasets" section with a "New Dataset" button and a dropdown menu set to "Images". It lists two entries: "dataset_imagenet@256x256" (Submitted: 05:29:57 PM (53 seconds ago), Status: Running) and "dataset_minst_10k@256x256" (Submitted: 05:21:27 PM, Status: Done after 31 seconds). On the right, there's a "Models" section with a "New Model" button and a dropdown menu set to "Images". It lists two entries: "model_mnist10k" (Submitted: 05:30:48 PM (2 seconds ago), Status: Running) and "LeNet_model_voc_cropped@256x256" (Submitted: 05:18:43 PM (12 minutes, 7 seconds ago), Status: Running). Both sections have a "Delete" button next to each entry.

DIGITS

Key Features

Visualize DNN topology and how training data activates your network

Manage training of many DNNs in parallel on multi-GPU systems

Simple setup and launch

Import a wide variety of image formats and sources

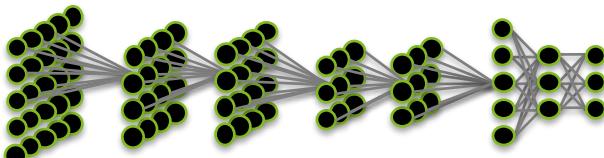
Monitor network training in real-time

Open source, so DIGITS can be customized and extended as needed

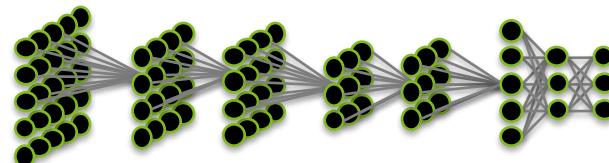
What is Deep Learning Software?



FC

A large grey arrow pointing from left to right, with the letters 'FC' written vertically along its path.

Compute weight update to nudge
from “turtle” towards “dog”



Trained Neural
Net Model

Deep Learning

Steps with DIGITS

Creating a Dataset

Define the Network or use existing

Choose a given Framework

Selecting a preconfigured (“standard”) network - LeNet, AlexNet, GoogleNet

Previous network

Custom network

Training a Model

Classification

Install and Start DIGITS Server

DIGITS Installation

Two Ways

Installation with Pre-Built Packages

Deb packages are provided for easy installation on Ubuntu 14.04.

Packages are provided for major releases, but not minor ones (i.e. v3.0 and v4.0, but not v3.1).

Download the web installer - <https://developer.nvidia.com/digits>

DIGITS Installation

Two Ways

Installation with Source

Get the latest branch from GitHub

Latest features but carries risk of untested features

Build NVIDIA Caffe branch - <https://github.com/NVIDIA/caffe>

Download DIGITS from github -<https://github.com/NVIDIA/DIGITS>

DIGITS Installation

HW and SW

Hardware/software recommendations

GPU(s) with compute capabilities ≥ 3.0 for cuDNN support

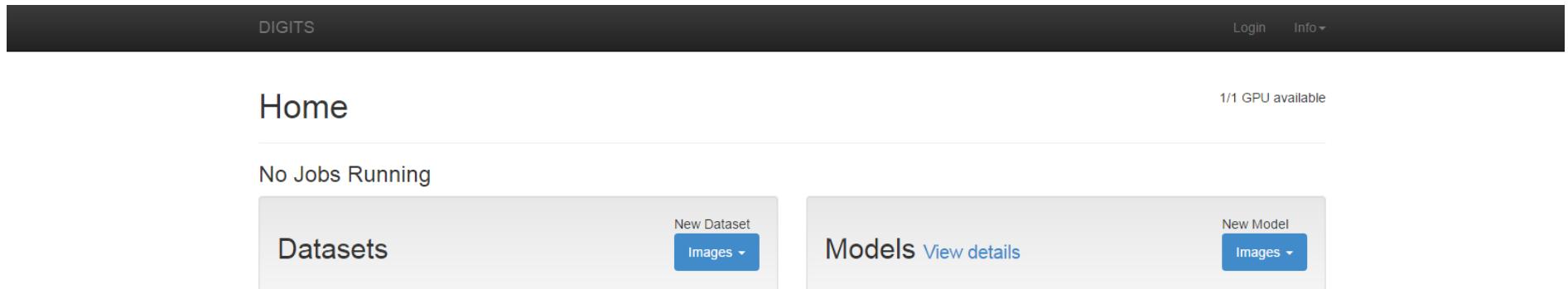
Ubuntu 14.04, CentOS

NVIDIA DIGITS

Main Console

Start DIGITS by

`http://localhost/` (if installed from Deb packages),
`http://localhost:5000/` (if using digits-devserver) or
`http://localhost:34448/` (if using digits-server)



The screenshot shows the DIGITS Main Console interface. At the top, there is a dark header bar with the word "DIGITS" on the left and "Login" and "Info" dropdown menus on the right. Below the header, the word "Home" is centered above a horizontal line. To the right of the line, it says "1/1 GPU available". The main content area has a light gray background. It displays a message "No Jobs Running" and two large rectangular cards. The left card is titled "Datasets" and has a "New Dataset" button with a dropdown menu showing "Images". The right card is titled "Models" and has a "New Model" button with a dropdown menu showing "Images". Both cards also have "View details" links.

NVIDIA DIGITS

LOGIN



Login

Username ?

Submit



Home

1/1 GPU available

No Jobs Running

Datasets

New Dataset

Images ▾

Models [View details](#)

New Model

Images ▾

CREATING DATASETS

NVIDIA DIGITS

Creating your Dataset

Main Console

The screenshot shows the NVIDIA DIGITS Main Console interface. At the top, there is a dark header bar with the word "DiGITS" in white. Below it, the title "Home" is displayed. The main content area is divided into two main sections: "Datasets" on the left and "Models" on the right.

Datasets Section:

- Header:** "Datasets"
- In progress:** "None"
- Completed:** "None"
- Action Buttons:** "New Dataset" (blue button) and "Images" (blue button with a dropdown arrow). The "Images" button is circled in green.

Models Section:

- Header:** "Models"
- In progress:** "None"
- Completed:** "None"
- Action Buttons:** "New Model" (blue button) and "Images" (blue button with a dropdown arrow).

NVIDIA DIGITS

Creating your Dataset

Main Console

Home

No Jobs Running

Datasets

New Dataset
Images ▾
Classification
Other



New Image Classification Dataset

The screenshot shows the 'New Image Classification Dataset' configuration page. It includes the following settings:

- Image Type:** Color
- Image size:** 256 x 256
- Resize Transformation:** Squash
- Training Images:** folder or URL
- Minimum samples per class:** 2
- Maximum samples per class:** 25
- % for validation:** 25
- % for testing:** 0
- Separate validation images folder:**
- Separate test images folder:**
- DB backend:** LMDB
- Image Encoding:** PNG (lossless)
- Dataset Name:** (empty field)
- Create:** (button)

In the Datasets section on the left side of the page, click on the blue Images button and select Classification which will take you to the "New Image Classification Dataset" page

NVIDIA DIGITS

Download MNIST dataset

Use the following command to download the MNIST dataset (for Deb package installations, the script is at /usr/share/digits/tools/download_data/main.py):

```
$ tools/download_data/main.py mnist ~/mnist
```

```
Downloading url=http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz ...
Downloading url=http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz ...
Downloading url=http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz ...
Downloading url=http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz ...
Uncompressing file=train-images-idx3-ubyte.gz ...
Uncompressing file=train-labels-idx1-ubyte.gz ...
Uncompressing file=t10k-images-idx3-ubyte.gz ...
Uncompressing file=t10k-labels-idx1-ubyte.gz ...
Reading labels from /home/username/mnist/train-labels.bin ...
Reading images from /home/username/mnist/train-images.bin ...
Reading labels from /home/username/mnist/test-labels.bin ...
Reading images from /home/username/mnist/test-images.bin ...
Dataset directory is created successfully at '/home/username/mnist'
Done after 16.722807169 seconds.
```

NVIDIA DIGITS

Creating your Dataset

While the model creation job is running, you should see the expected completion time on the right side

When Model creation is done, you can also see completion time and duration

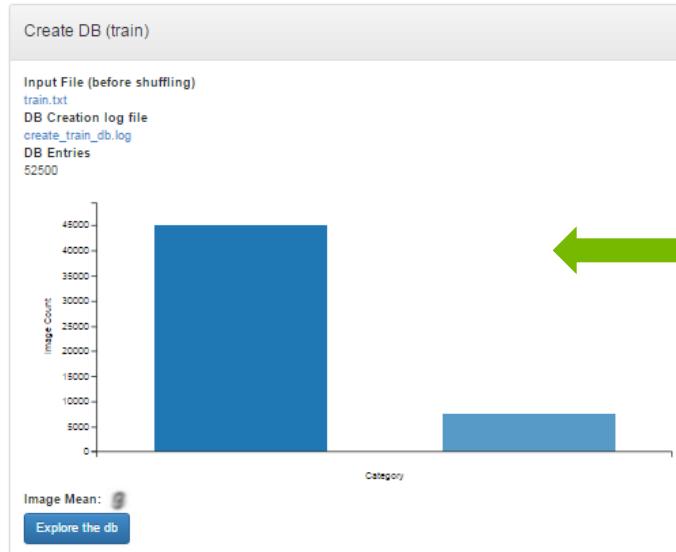
Users may download a copy of .txt files for reference

The screenshot shows the DIGITS web interface for creating datasets. At the top, there's a header with the dataset name "MNIST" and its owner "nvidia-test". Below the header are two main sections: "Job Information" and "Parse Folder (train/val)". The "Job Information" section contains details like Job Directory, Image Dimensions (28x28), Image Type (Grayscale), Resize Transformation (Half crop, half fill), DB Backend (Imdb), Image Encoding (png), DB Compression (none), and Dataset size (0 B). The "Parse Folder (train/val)" section shows the "Folder" path as "/home/pradeep/mnist". On the right side, there's a "Job Status" area with three items: "Parse Folder (train/val) Done", "Create DB (train) Done", and "Create DB (val) Done". Each item has a list of status steps: "Parse Folder (train/val) Done" includes "Initialized at 07:56:58 PM", "Running at 07:57:00 PM", and "Done at 07:59:48 PM (Total - 2 minutes, 49 seconds)"; "Create DB (train) Done" includes "Initialized at 07:56:58 PM"; "Create DB (val) Done" includes "Initialized at 07:56:58 PM", "Running at 07:57:01 PM", and "Done at 07:58:08 PM (Total - 1 minute, 9 seconds)".

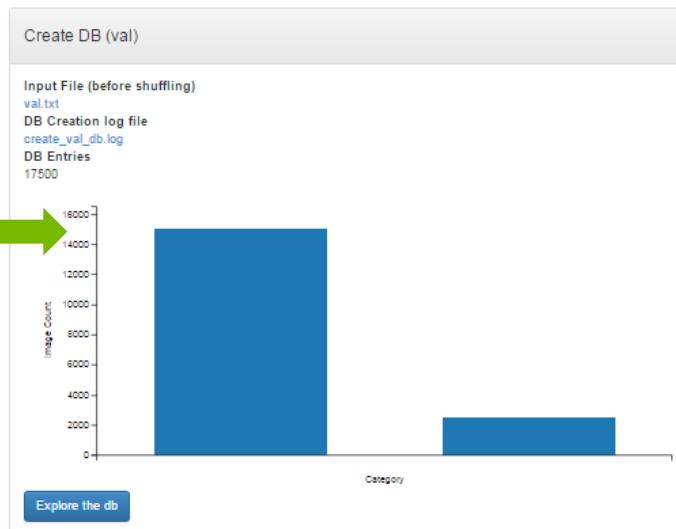
NVIDIA DIGITS

Database results

- Validation data tests the performance of the network
 - This data is only used for testing the generalization ability of the network
 - Not used to teach/train network
 - Prevents use of and identifies when network is overfit.
 - In current example 17500 images used for validation.



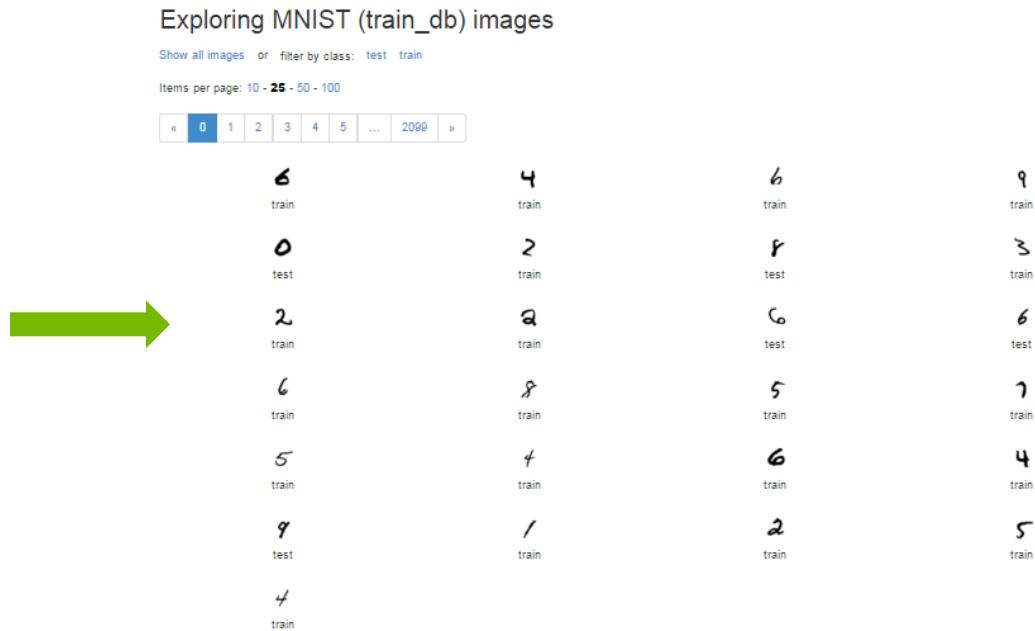
- Training data is used to train our neural network.
- Teaches the network to classify object categories
- Training Data Set, Current example uses 52500 Images for training



NVIDIA DIGITS

Database Results

Database of Images can be explored via Exploring “Explore DB” tab and you can see test images used for training/validation



The screenshot shows the 'Exploring MNIST (train_db) images' page. At the top, there are links to 'Show all images' or 'filter by class: test train'. Below that, it says 'Items per page: 10 - 25 - 50 - 100'. A navigation bar shows pages 0 through 2099. The main area displays a grid of 25 handwritten digit images, each with its value and category (train or test). The first row contains digits 6, 4, 6, 9 (train). The second row contains digits 0, 2, 8, 3 (test). The third row contains digits 2, 2, 6, 6 (test). The fourth row contains digits 6, 8, 5, 7 (train). The fifth row contains digits 5, 4, 6, 4 (train). The sixth row contains digits 9, /, 2, 5 (train). The seventh row contains digits 4 (train).

Value	Category
6	train
4	train
6	train
9	train
0	test
2	train
8	train
3	train
2	train
2	train
6	test
6	test
6	train
5	train
7	train
5	train
4	train
9	test
/	train
2	train
5	train
4	train

DIGITS DEMO

Creating your Dataset

A new database is
Created as visible on
Home Page of DIGITS



Home

No Jobs Running

New Dataset

Images ▾

Datasets

MNIST	Imdb
Submitted:	07:56:58 PM
Status:	Done after 2 minutes, 49 seconds

Delete

TRAIN A NETWORK

TRAINING

Choose Framework

With DIGITS 3.0, two frameworks are integrated into DIGITS

- Caffe
- Torch

NVIDIA DIGITS

Train A network

Training a network
interface, please note

- Database selection
- Data Transformations
- Solver Options
- Different Network configurations
 - Caffe
 - Torch(experimental)

New Image Classification Model

Select Dataset

MNIST

Use client side file

Python Layer File (server side)

Data Transformations

Crop Size

none

Subtract Mean

Image

Solver Options

Training epochs

30

Snapshot interval (in epochs)

1

Validation interval (in epochs)

1

Random seed

[none]

Batch size

[network defaults]

Solver type

Stochastic gradient descent (SGD)

Base Learning Rate

0.01

Show advanced learning rate options

Standard Networks

Previous Networks

Custom Network

Caffe

Torch (experimental)

Network	Details	Intended image size
LeNet	Original paper [1998]	28x28 (gray)
AlexNet	Original paper [2012]	256x256
GoogLeNet	Original paper [2014]	256x256

NVIDIA DIGITS

Train a Network

Select the Database

Provide a Model Name

Do any changes in Solver options

Start with a default Network like LeNet (Framework can be anyone Caffe/Torch)

Click on Create Button

New Image Classification Model

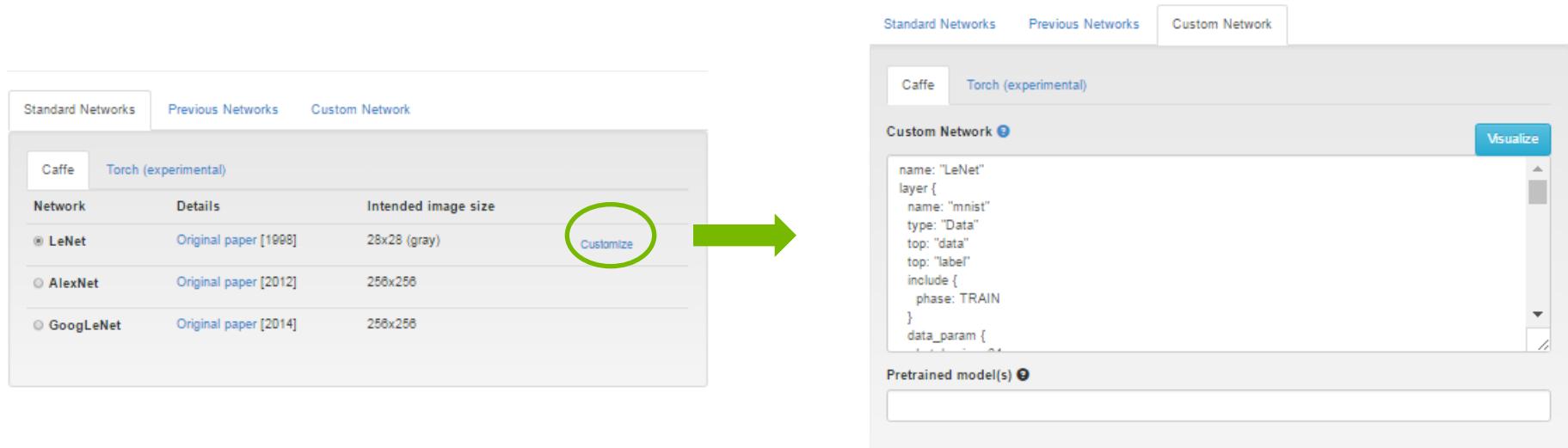
The screenshot shows the DIGITS web interface for creating a new image classification model. It is divided into several sections:

- Dataset Selection:** A dropdown menu shows "MNIST" selected. Below it are checkboxes for "Use client side file" and "Python Layer File (server side)".
- MNIST Dataset Details:** Information about the dataset is displayed, including "Done Tue Mar 15, 07:59:48 PM", "Image Size 28x28", "Image Type GRayscale", "DB backend lmdb", "Create DB (train) 52500 images", "Create DB (val) 17500 images", and "Data Transformations".
- Solver Options:** Settings for training include "Training epochs" (30), "Snapshot interval (in epochs)" (1), "Validation interval (in epochs)" (1), "Random seed" ([none]), "Batch size" ([network defaults]), "Solver type" (Stochastic gradient descent (SGD)), and "Base Learning Rate" (0.01). There is also a checkbox for "Show advanced learning rate options".
- Network Selection:** A table lists three pre-defined networks:

Network	Details	Intended image size	Customize
LeNet	Original paper [1998]	28x28 (gray)	Customize
AlexNet	Original paper [2012]	256x256	
GoogLeNet	Original paper [2014]	256x256	
- Model Name:** A text input field contains "MNIST-Model".
- Create Button:** A blue button labeled "Create" is located at the bottom right of the model name input field.

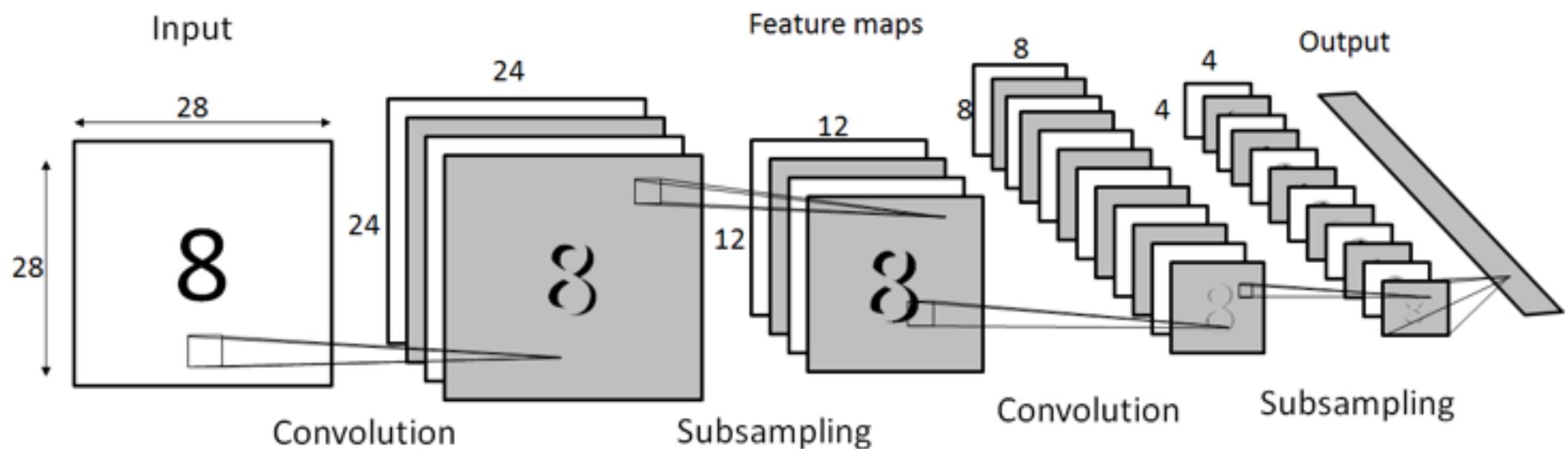
NVIDIA DIGITS

Train a Network-Advance Options



LENET

Network Configuration



NVIDIA DIGITS

Train a Network-Advance Options

Standard Networks Previous Networks Custom Network

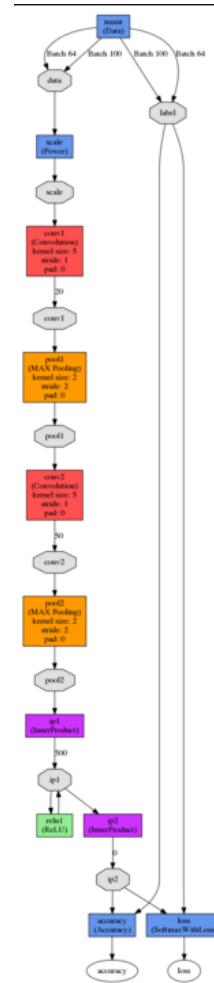
Caffe Torch (experimental)

Custom Network (LeNet)

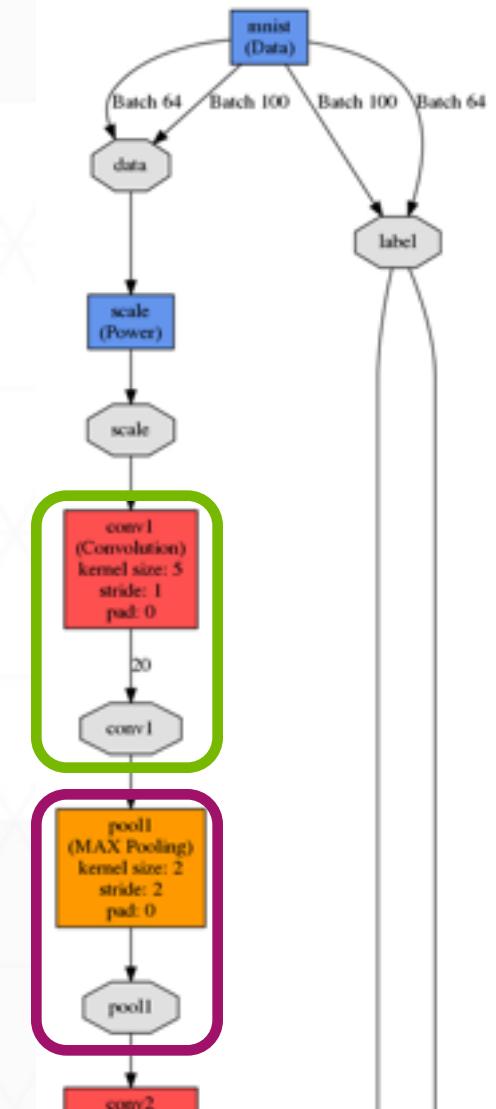
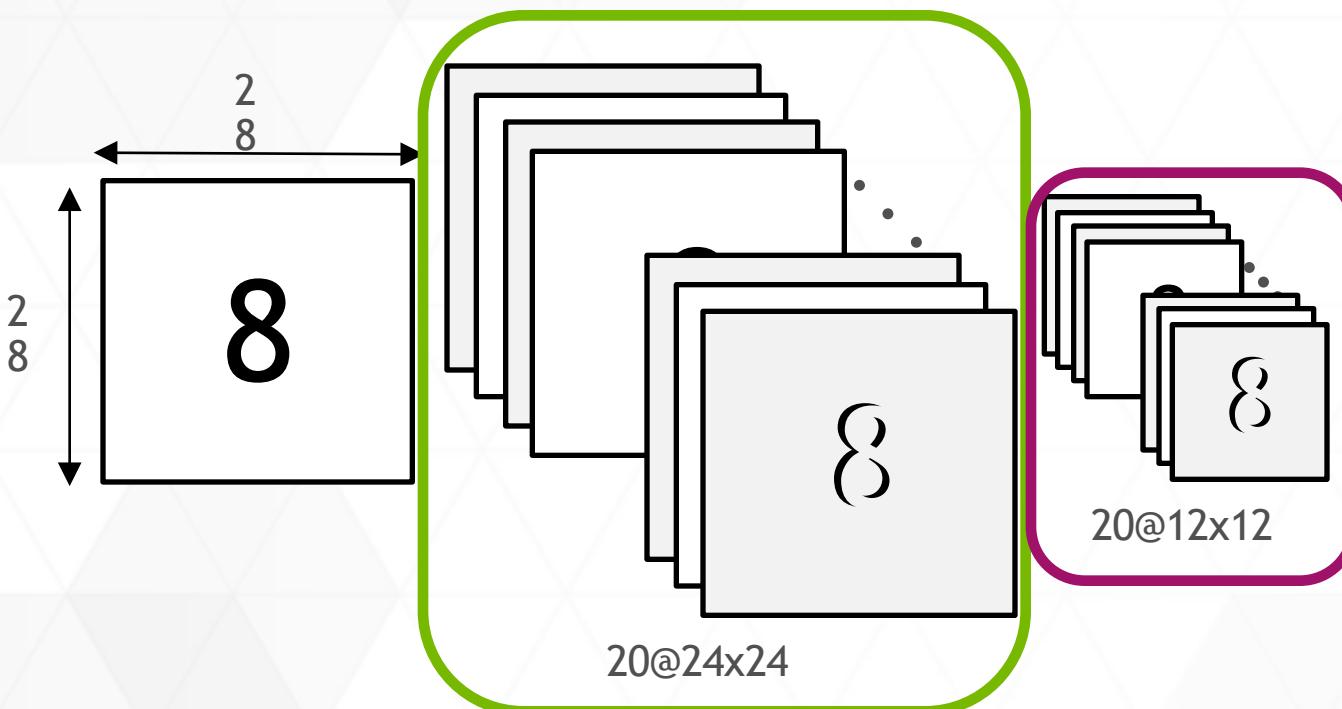
```
name: "LeNet"
layer {
  name: "mnist"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TRAIN
  }
  data_param {
```

Visualize

Pretrained model(s) (LeNet)



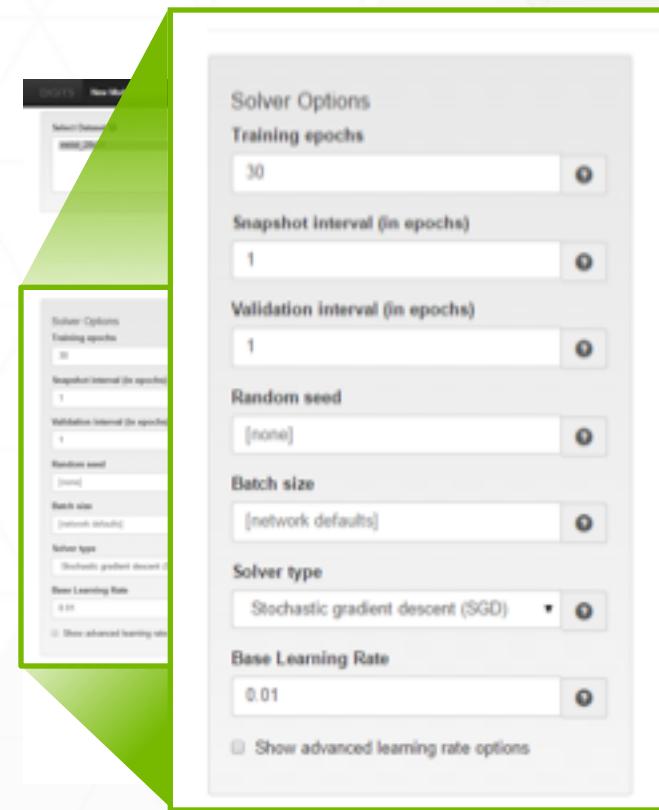
NETWORK PARAMETERS



NVIDIA DIGITS

Train a Network-Advance Options

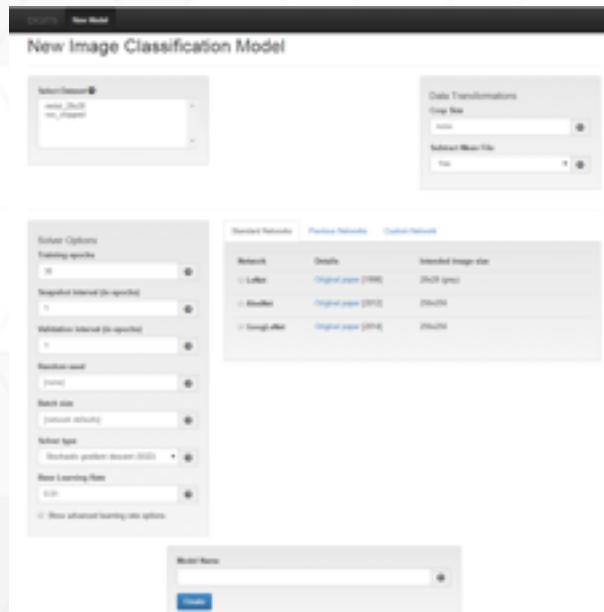
- Training epochs - processing of all data
- Snapshot interval - saving trained network
- Validation interval - DNN test with the validation data
- Batch size - number of images processed together
- Solver type - SGD, ADAGRAD, NAG
- Learning rate and policy



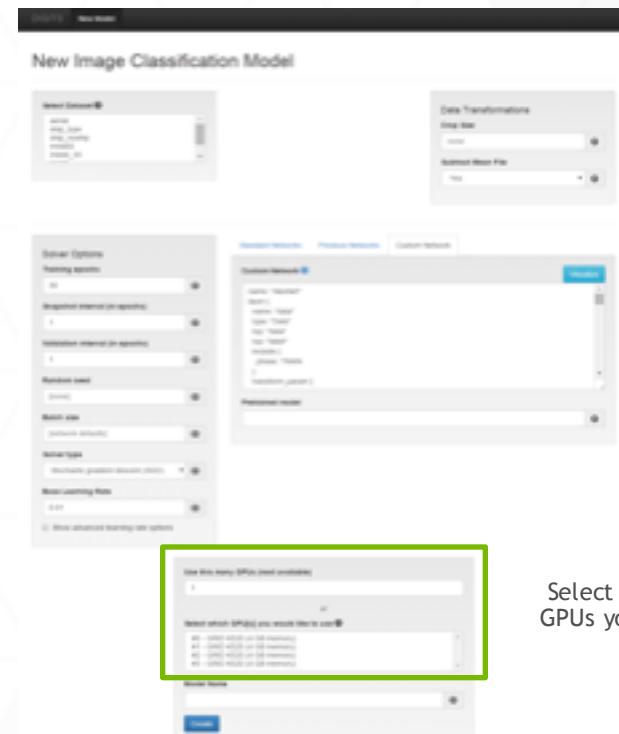
NVIDIA DIGITS

Single and multi-GPU training is easy

Single GPU system



Multiple GPU system



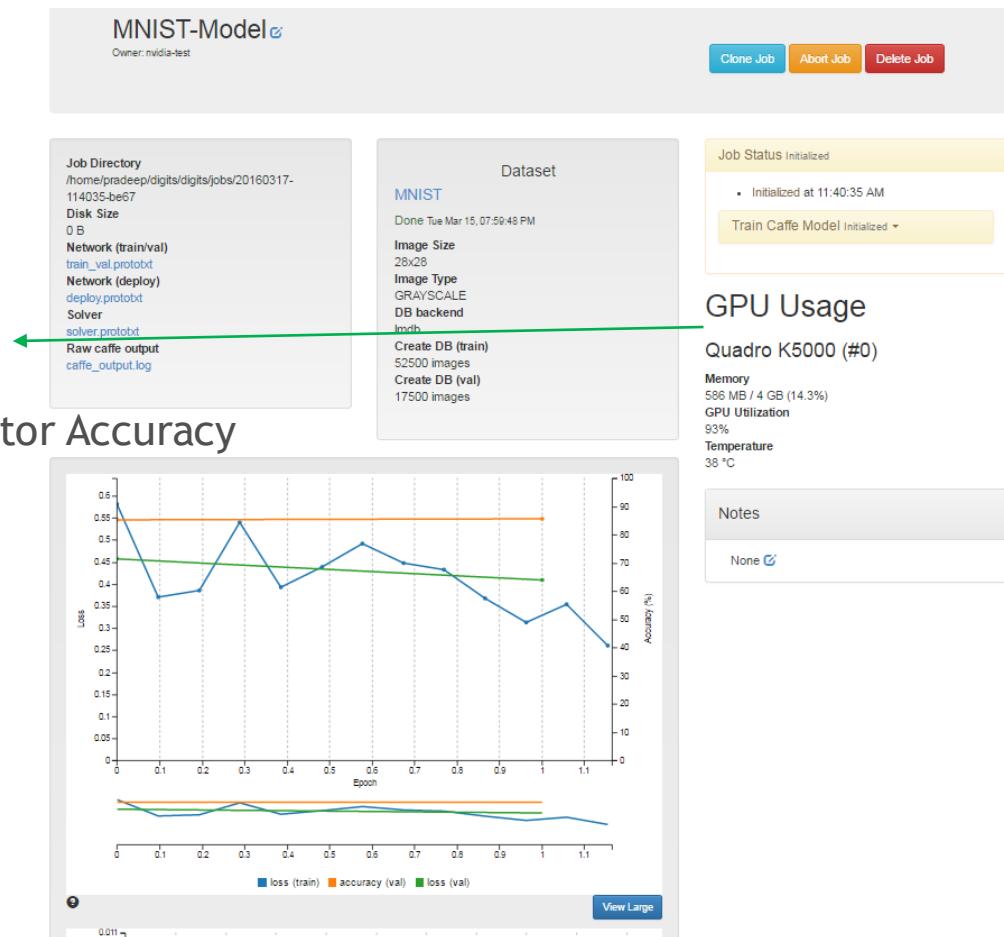
Select the number of GPUs you want to use.

NVIDIA DIGITS

Training Results

Monitor GPU, Memory usage and Temperature

If performance is poor, abort, modify, and retrain.

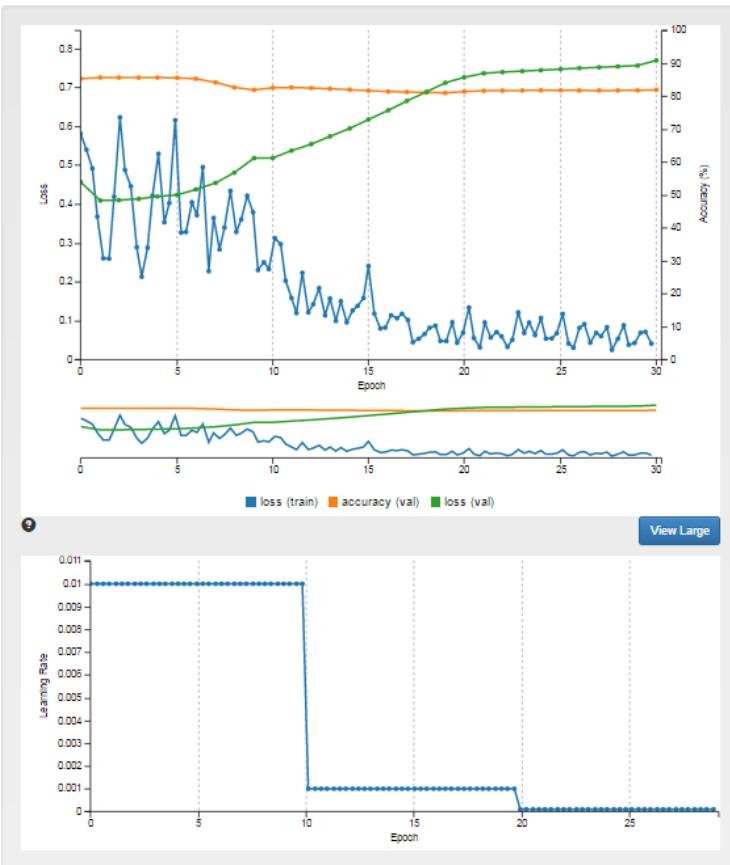


NVIDIA DIGITS

Training Results

First Graph is about the Loss and Accuracy graphs

Second Graph is about the Learning rate, as the training progresses, learning rate goes down as model is getting more mature.



OVERFITTING AND UNDERFITTING

How can I use DIGITS to tell me this is happening?

Overfitting



Validation data helps you identify when/if this occurs!

Underfitting



Loss is not increasing
but the accuracy is not
improving

Modifying your Network & Classification

NVIDIA DIGITS

Single Image Classification

Select Model
Epoch #3

Download

Image URL

Upload image
 No file chosen

Show visualizations and statistics ⓘ

Classify One Image ⓘ

Upload Image List

No file chosen
Accepts a list of filenames or urls (you can use your .txt file)

Classify Many Images ⓘ

Number of Images use from the file
 100
Leave blank to use all

Number of images to show per category
 9

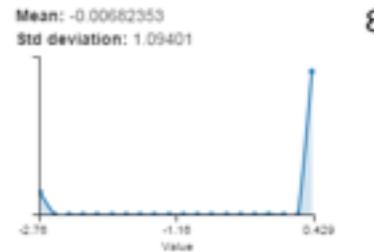
Top N Predictions per Category ⓘ



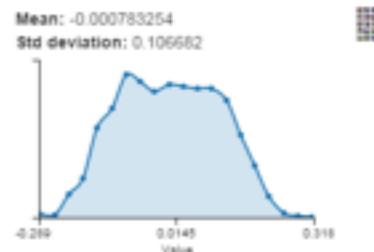
* Lab tasks

SINGLE IMAGE CLASSIFICATION RESULTS

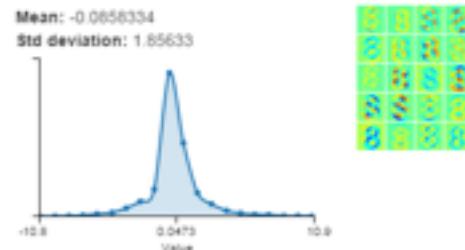
scale
Activation



conv1
Weights (Convolution layer)



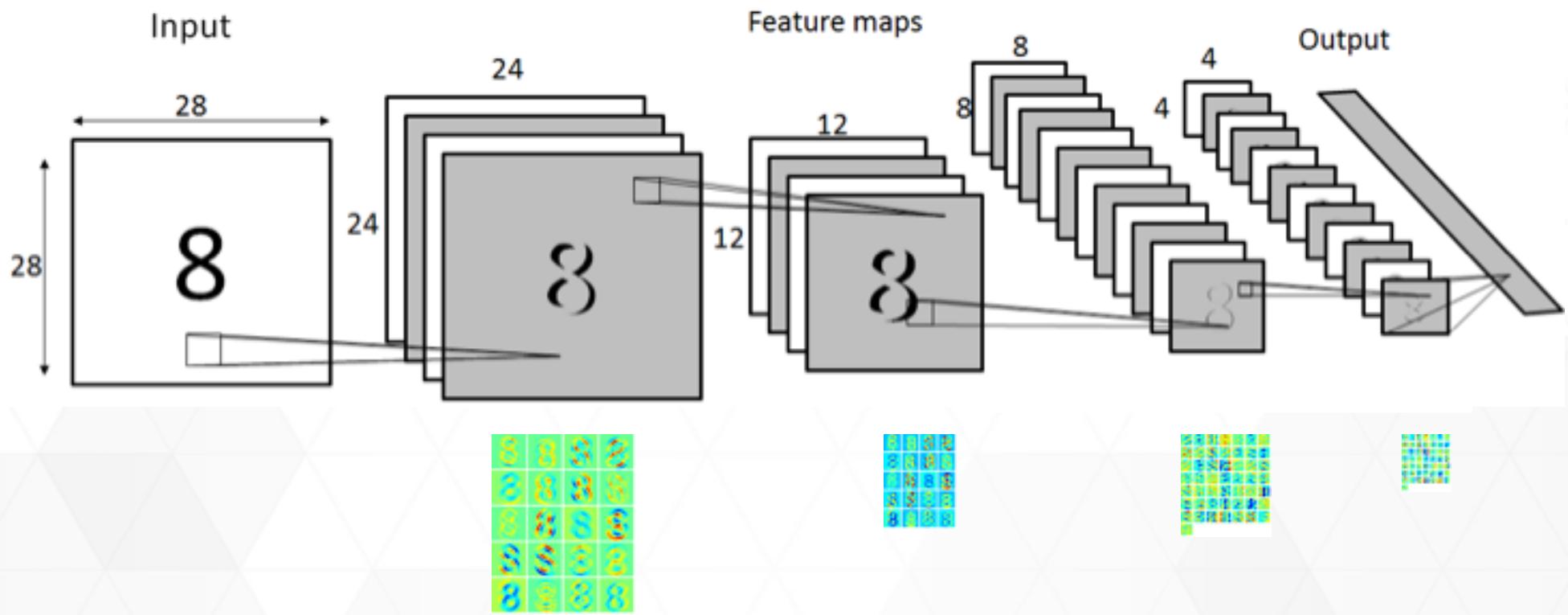
conv1
Activation



- Network response at each layer will display
- Visualize responses from different inputs

NETWORK CONFIGURATION

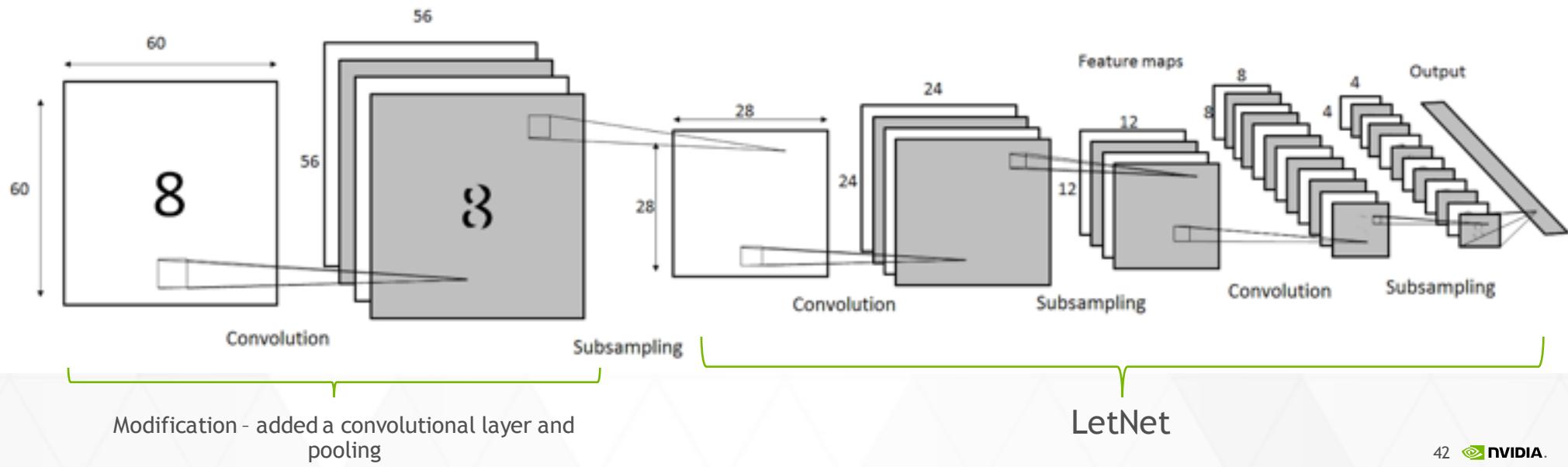
Visualizing LetNet Network Responses



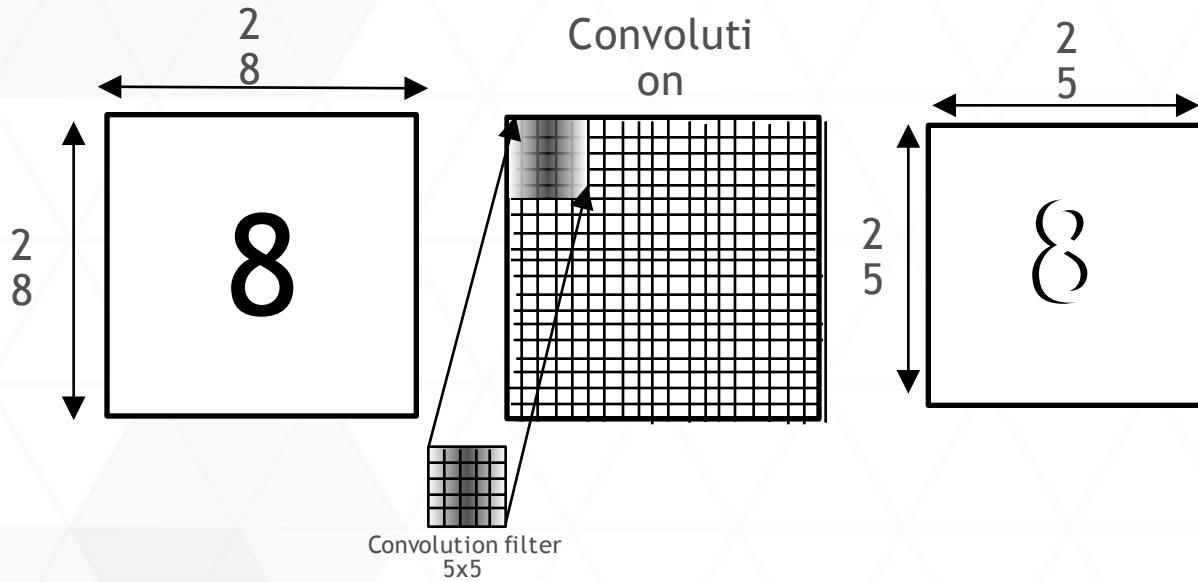
NETWORK CONFIGURATIONS

Modifying a Network

- Modifying a network can improve performance
- There are many parameters - add or remove a layer, pooling, activation function, zero padding, increasing outputs



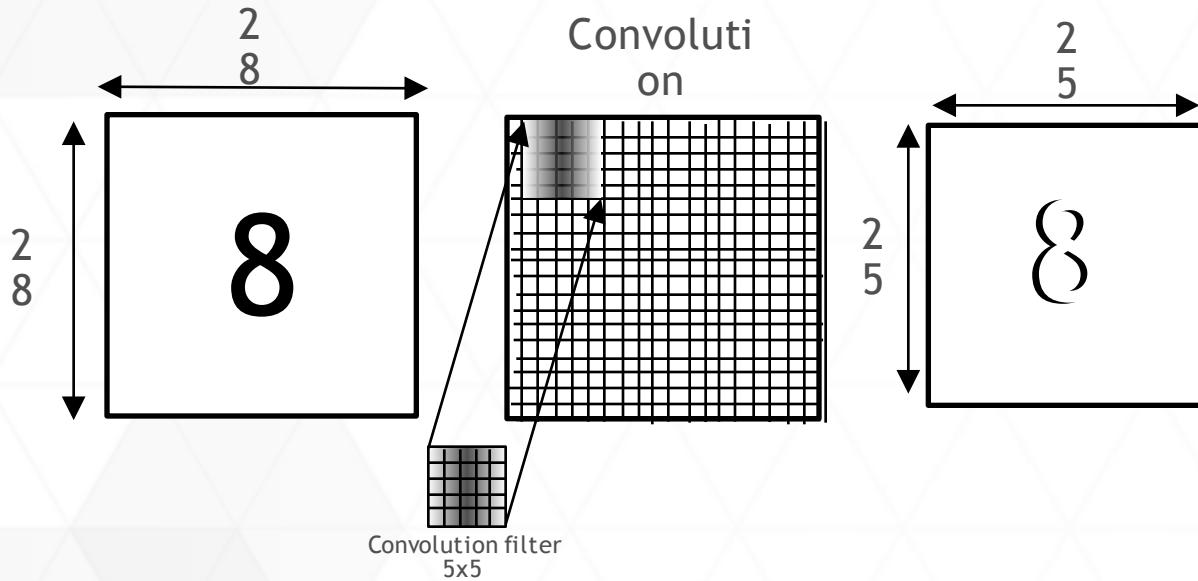
MODIFYING YOUR NETWORK



Zero padding, input is reduced by the 2*radius of the kernel
Input[28x28]*filter[5x5]=FeatureMap[25x25]

```
layer {
    name: "conv1"
    type: "Convolution"
    bottom: "scale"
    top: "conv1"
    param {
        lr_mult: 1
    }
    param {
        lr_mult: 2
    }
    convolution_param {
        num_output: 20
        kernel_size: 5
        stride: 1
        weight_filler {
            type: "xavier"
        }
        bias_filler {
            type: "constant"
        }
    }
}
```

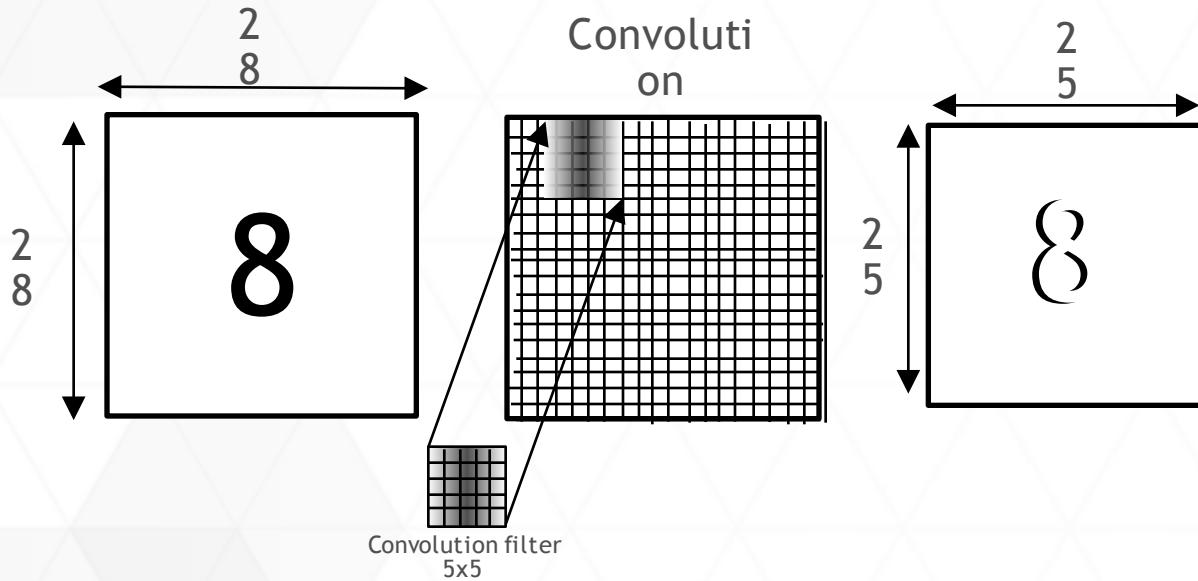
MODIFYING YOUR NETWORK



Zero padding, input is reduced by the $2 \times$ radius of the kernel
Input[28x28]*filter[5x5]=FeatureMap[25x25]

```
layer {
    name: "conv1"
    type: "Convolution"
    bottom: "scale"
    top: "conv1"
    param {
        lr_mult: 1
    }
    param {
        lr_mult: 2
    }
    convolution_param {
        num_output: 20
        kernel_size: 5
        stride: 1
        weight_filler {
            type: "xavier"
        }
        bias_filler {
            type: "constant"
        }
    }
}
```

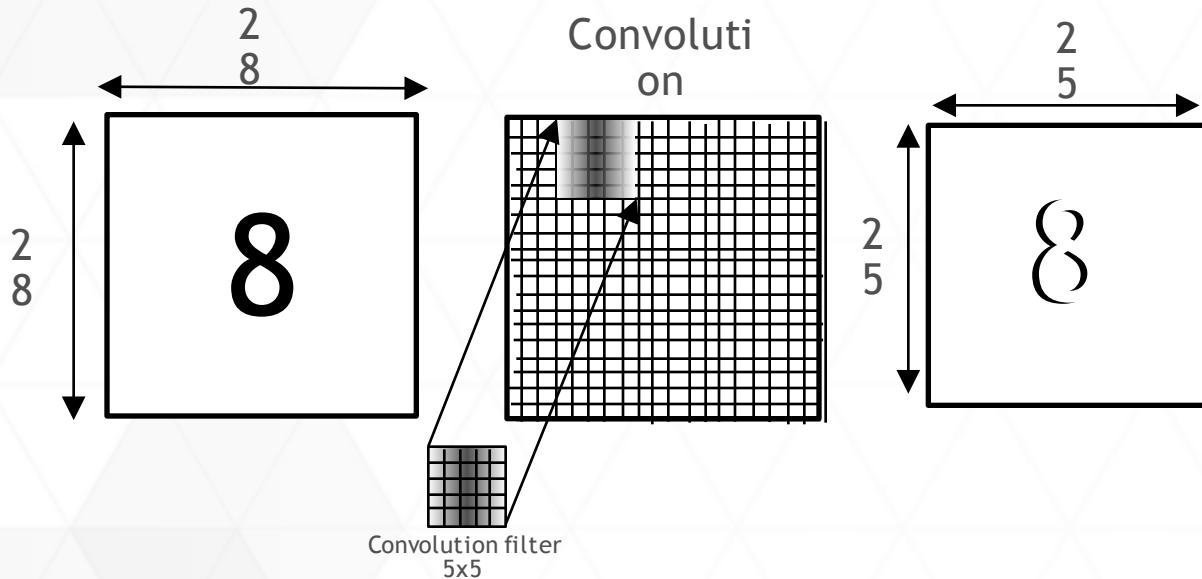
MODIFYING YOUR NETWORK



Zero padding, input is reduced by the $2 \times$ radius of the kernel
Input[28x28]*filter[5x5]=FeatureMap[25x25]

```
layer {
    name: "conv1"
    type: "Convolution"
    bottom: "scale"
    top: "conv1"
    param {
        lr_mult: 1
    }
    param {
        lr_mult: 2
    }
    convolution_param {
        num_output: 20
        kernel_size: 5
        stride: 1
        weight_filler {
            type: "xavier"
        }
        bias_filler {
            type: "constant"
        }
    }
}
```

MODIFYING YOUR NETWORK



Zero padding, input is reduced by the $2 \times$ radius of the kernel
Input[28x28]*filter[5x5]=FeatureMap[25x25]

```
layer {
    name: "conv1"
    type: "Convolution"
    bottom: "scale"
    top: "conv1"
    param {
        lr_mult: 1
    }
    param {
        lr_mult: 2
    }
    convolution_param {
        num_output: 20
        kernel_size: 5
        stride: 1
        weight_filler {
            type: "xavier"
        }
        bias_filler {
            type: "constant"
        }
    }
}
```

NETWORK PARAMETERS

Convolution

```
layer {
  name: "conv0"
  type: "Convolution"
  bottom: "data"
  top: "conv0"  
  param {
    lr_mult: 1.0
  }
  param {
    lr_mult: 2.0
  }
  convolution_param {
    num_output: 20
    kernel_size: 5
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
      value: 0.9
    }
  }
}
```

Pooling/Subsampling

```
layer {
  name: "pool0"
  type: "Pooling"
  bottom: "conv0"  
  top: "pool0"  
  pooling_param {
    pool: MAX
    kernel_size: 2
    stride: 2
  }
}
```

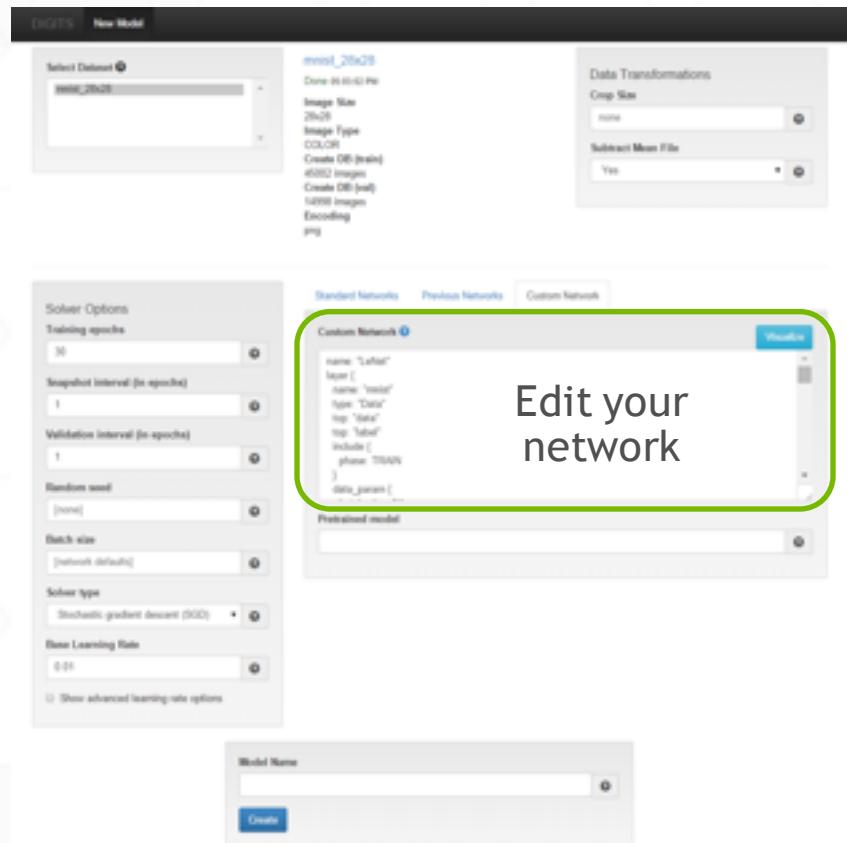
Activation

```
layer {
  name: "relu0"
  type: "ReLU"
  bottom: "pool0"  
  top: "pool0"
}
```

* Lab tasks

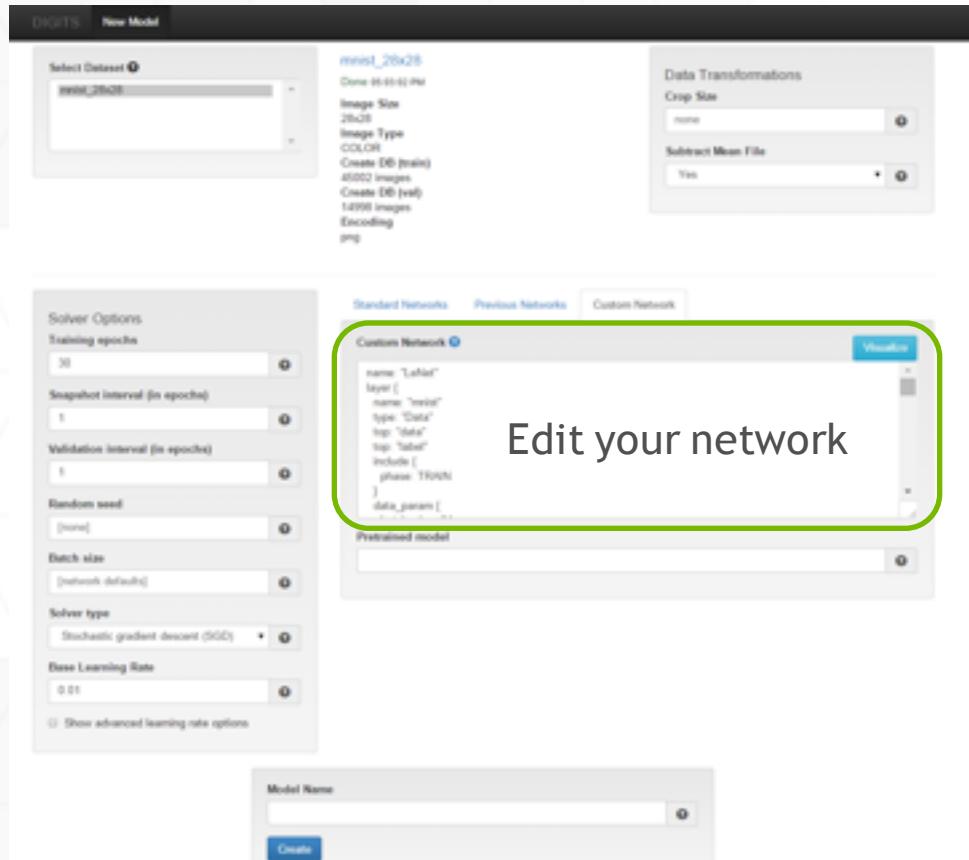
NVIDIA DIGITS

Modifying your Network



Edit your
network

MODIFYING YOUR NETWORK



* Lab tasks

```
layer {  
    name: "conv1"  
    type: "Convolution"  
    bottom: "scale" } Input and output to  
    top: "conv1" the layer  
    param {  
        lr_mult: 1  
    }  
    param {  
        lr_mult: 2  
    }  
    convolution_param {  
        num_output: 20 *  
        kernel_size: 5  
        stride: 1  
        weight_filler {  
            type: "xavier"  
        }  
        bias_filler {  
            type: "constant"  
        }  
    }
```

MODIFYING YOUR NETWORK

DIGITS New Model

Select Dataset: mnist_28x28

mnist_28x28
Done 01-01-01 PM
Image Size: 28x28
Image Type: COLOR
Create DB (train): 45502 images
Create DB (val): 14998 images
Encoding: png

Data Transformations

Crop Size: none
Subtract Mean File: Yes

Solver Options

Training epochs: 50
Snapshot interval (in epochs): 1
Validation interval (in epochs): 1
Random seed: [none]
Batch size: [network default]
Solver type: Stochastic gradient descent (SGD)
Base Learning Rate: 0.01
Show advanced learning rate options

Custom Network

Custom Network

name: "TfNet"
layer [
 name: "input"
 type: "Data"
 top: "data"
 top: "label"
 include [
 phase: TRAIN
]
 data_param [
]
]
Input and output to the layer

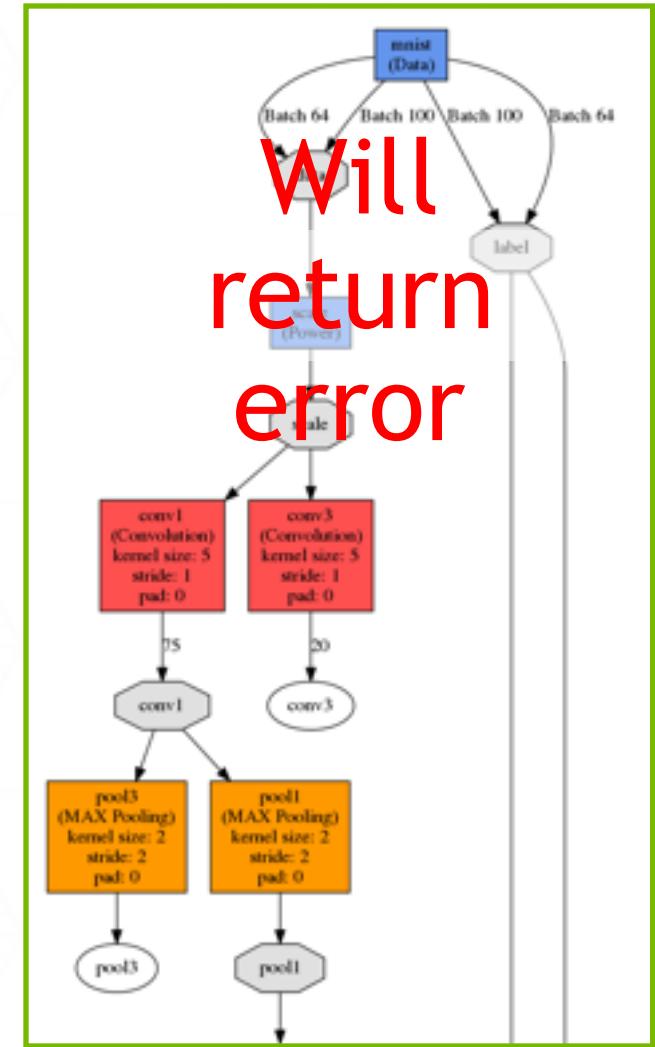
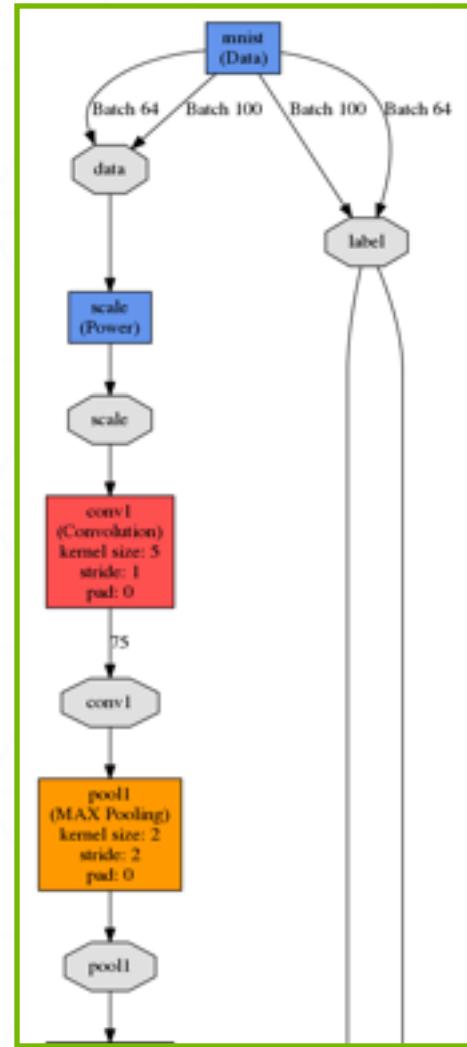
Model Name:
Create

* Lab tasks

* layer {
 name: "relu1"
 type: "ReLU"
 bottom: "conv1"
 top: "conv1" } Input and output to the layer

MODIFYING YOUR NETWORK

Visualize Configuration Changes



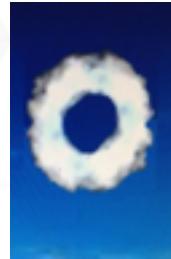
ANOTHER WAY TO IMPROVE PERFORMANCE

Data Augmentation

- Sometimes training data is not a great representation of the field data
 - MNIST data is grayscale, black text with white background

0 1 2 3 4 5 6 7 8 9

- Will these images be classified correctly when the network is trained with this digit data?



ANOTHER WAY TO IMPROVE PERFORMANCE

Data Augmentation

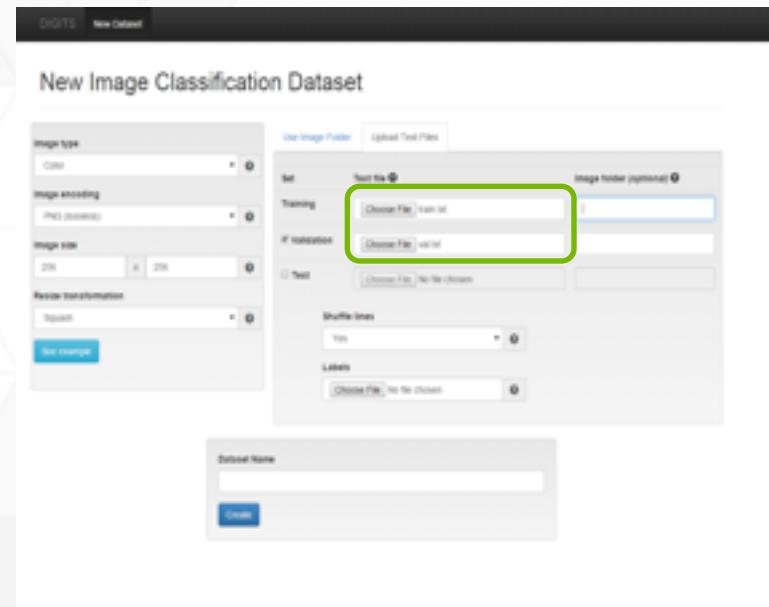
- Depending on the deployment scenario, simple modifications can be made to the training data to improve performance



- There are many ways to augment data
 - Rotations, noise, color distortions, stretching, etc. *
- Many ways to modify images - ImageMagick, Pillow, OpenCV

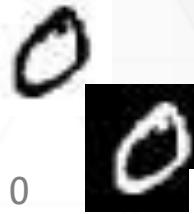
* Lab tasks

USING AN AUGMENTED DATA SET



- **train.txt**

/home/user/train/0/0_1.jpg 0



/home/user/train/0/0_1_invert.jpg 0



/home/user/train/5/5_1.jpg 5

/home/user/train/5/5_1_invert.jpg 5

- **val.txt**

/home/user/mnist/val/7_1.jpg 7



/home/user/mnist/val/7_1_invert.jpg 7

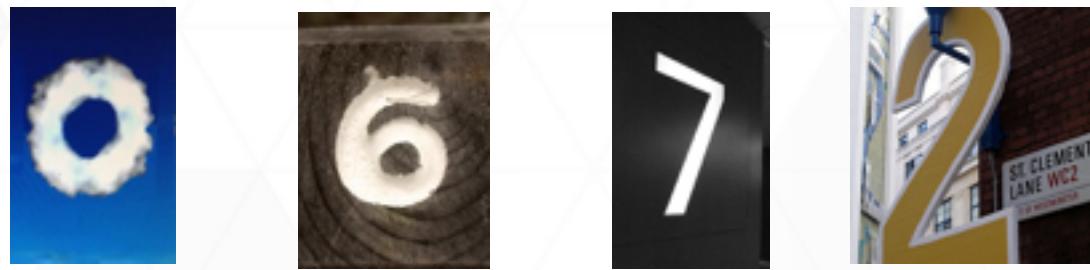
ANOTHER WAY TO IMPROVE PERFORMANCE

Data Augmentation

- Example augmentation - inverted copies of the input data



- Would a network trained with this data augmented, accurately classify these images?



DEPLOYING YOUR NETWORK

- Deploy in the cloud



- Deploy on a mobile device



Select Model
Epoch #30 Download the model you want to use

Download

Image URL

Upload image

No file chosen

Show visualizations and statistics

Classify One Image

Upload Image List

No file chosen

Accepts a list of filenames or urls (you can use your val.txt file)

Classify Many Images

Number of images use from the file
100

Leave blank to use all

Number of images to show per category
9

Top N Predictions per Category

This form allows users to select a pre-trained model (Epoch #30) and download it. It also provides options for classifying individual images or many images from a list, specifying the number of images to use and the number to show per category, and enabling visualization and statistics.

DEPLOYMENT WITH TEGRA

Rapid Classification Anywhere

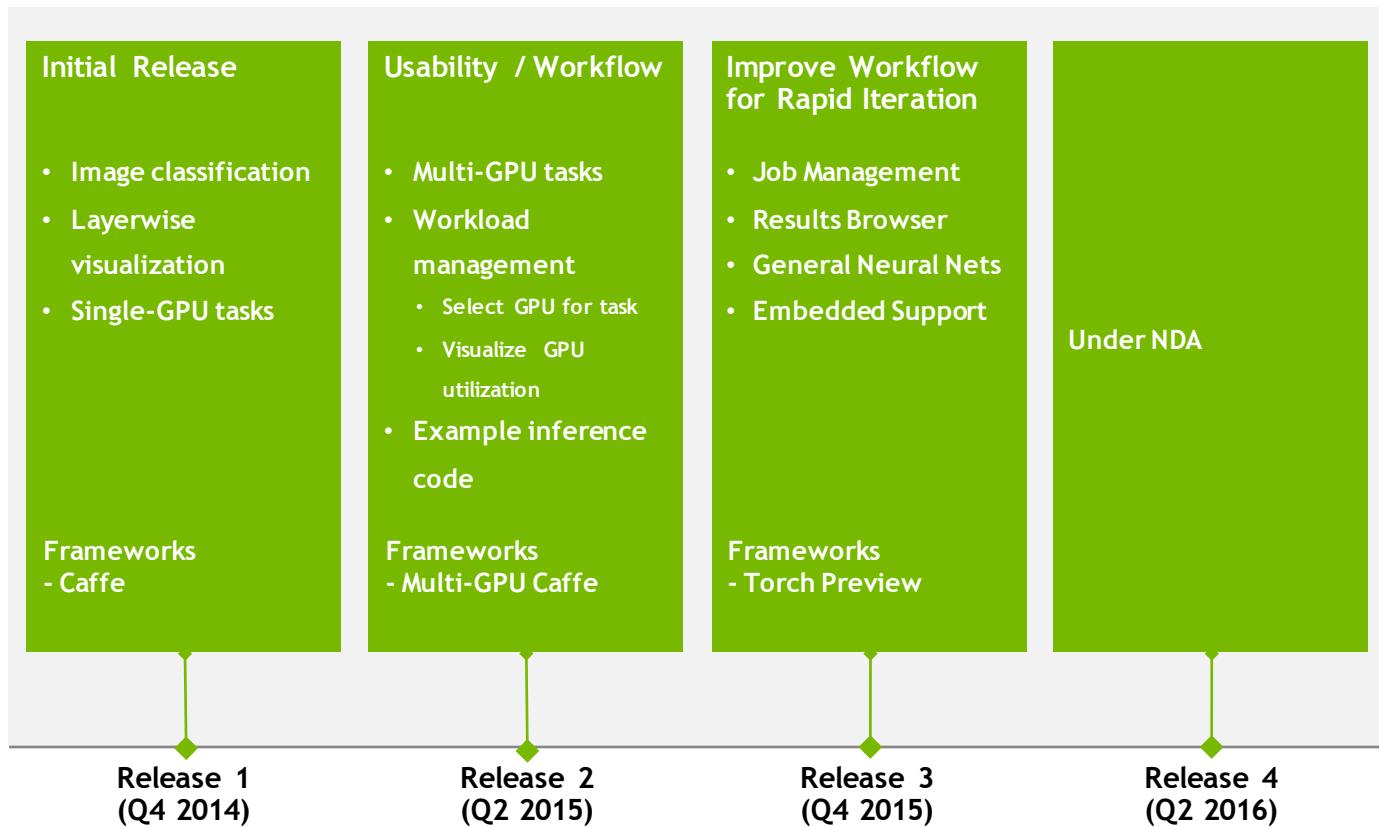
- Flexible
- Low Power
- Easy to use
- GPU accelerated



Build Caffe on your portable platform
Download your trained network from DIGITS

DIGITS Features at a Glance

NVIDIA DIGITS ROADMAP



WHAT'S NEW IN DIGITS 3?

DIGITS 3 Improves Training Productivity with Enhanced Workflows

- ▶ Train neural network models with Torch support (preview)
- ▶ Save time by quickly iterating to identify the best model
- ▶ Easily manage multiple jobs to optimize use of system resources
- ▶ Active open source project with valuable community contributions

Name	Status	Runtime	Loss
Slayer_aerial	Aborted	00:00:11	
aerial_2layer	Aborted	00:03:20	87.3365
aerial_Slayer	Running	00:00:01	1.26419
aerial_5layer	Aborted	00:02:12	
aerial_Slayer	Aborted	00:07:35	
aerial_Slayer_steprate	Running	00:00:01	0.825354099274
aerial_alexnet	Aborted	00:05:42	1.06914
aerial_deeppnetwork	Running	00:00:01	1.61509923935
Name	Status	Runtime	Loss

New Results Browser!

developer.nvidia.com/digits

NVIDIA DIGITS

Resources

- Where to get DIGITS
 - Easy to use web installer <https://developer.nvidia.com/digits>
 - github - <https://github.com/NVIDIA/DIGITS>
 - Remember to install NVIDIA's Caffe branch - <https://github.com/NVIDIA/caffe>
- User support
 - DIGITS Users Google group - <https://groups.google.com/forum/#!forum/digits-users>
- For more information on getting started with DIGITS
 - Parallel forall - <http://devblogs.nvidia.com/parallelforall/easy-multi-gpu-deep-learning-digits-2/>
 - Getting started guide - <https://github.com/NVIDIA/DIGITS/blob/master/docs/GettingStarted.md>

HANDS-ON LAB

1. Create an account at nvidia.qwiklab.com
 2. Go to “Getting Started with DIGITS” lab at bit.ly/dlnvlab2
 3. Start the lab and enjoy!
-
- Only requires a supported browser, no NVIDIA GPU necessary!
 - Lab is free until end of this Deep Learning Lab series

DEEP LEARNING SERIES

- Review the other seminars in series
- Seminar #3 - Getting Started with the Caffe Framework
- Seminar #4 - Getting Started with the Theano Framework
- Seminar #5 - Getting Started with the Torch Framework
- More information available at developer.nvidia.com/deep-learning-courses