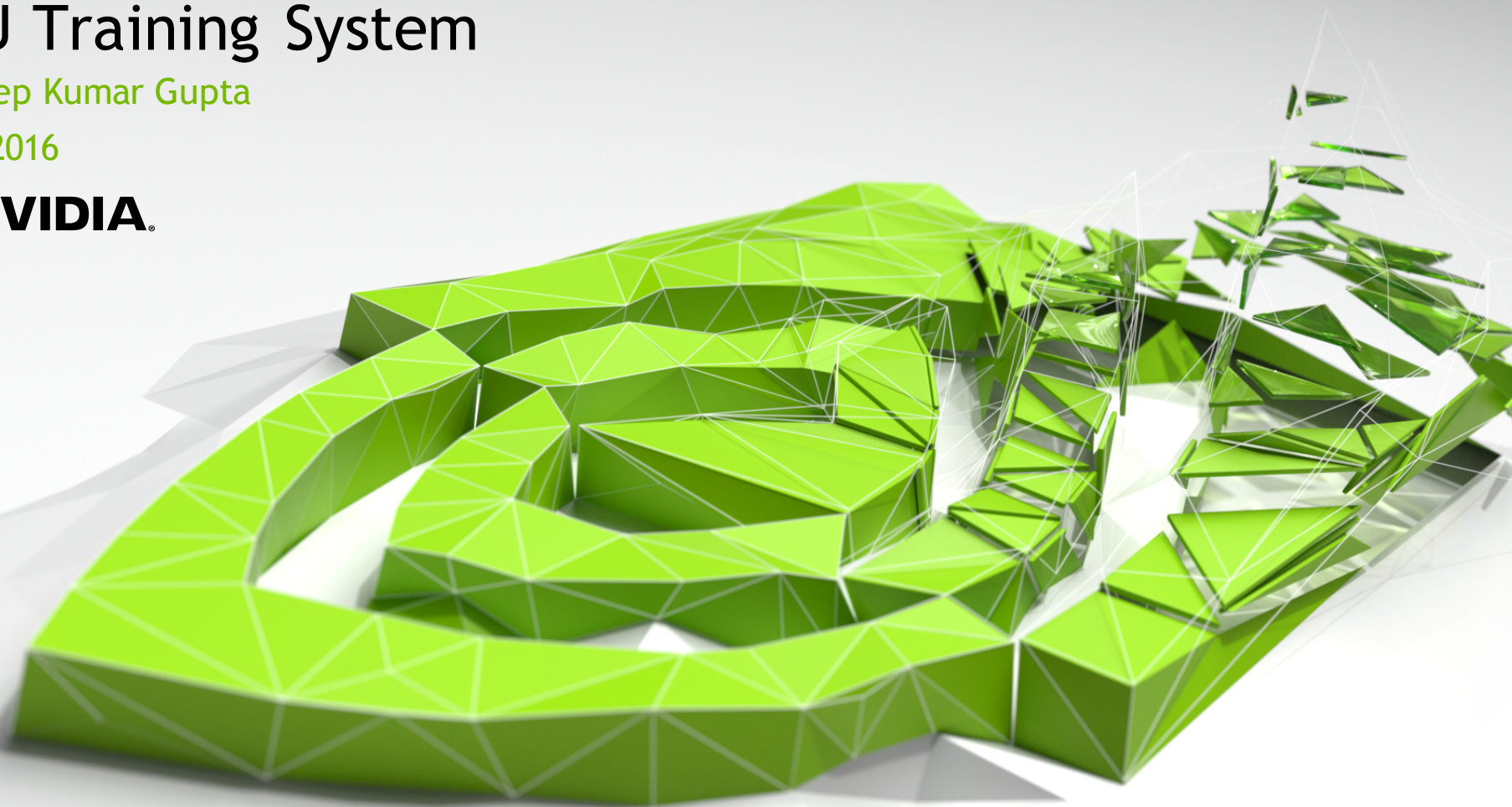


DIGITS- Interactive Deep Learning GPU Training System

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



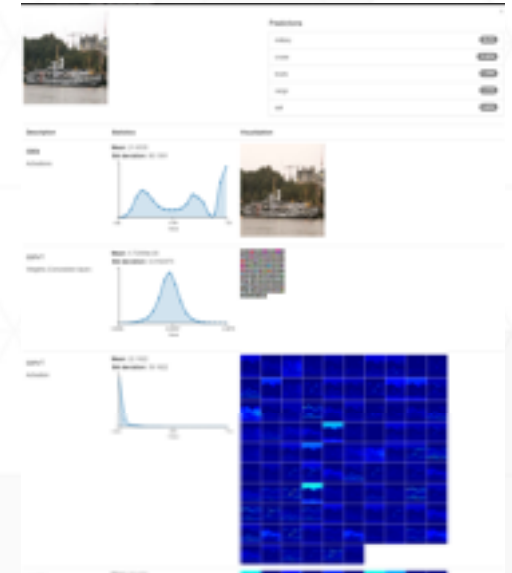
Configure DNN



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

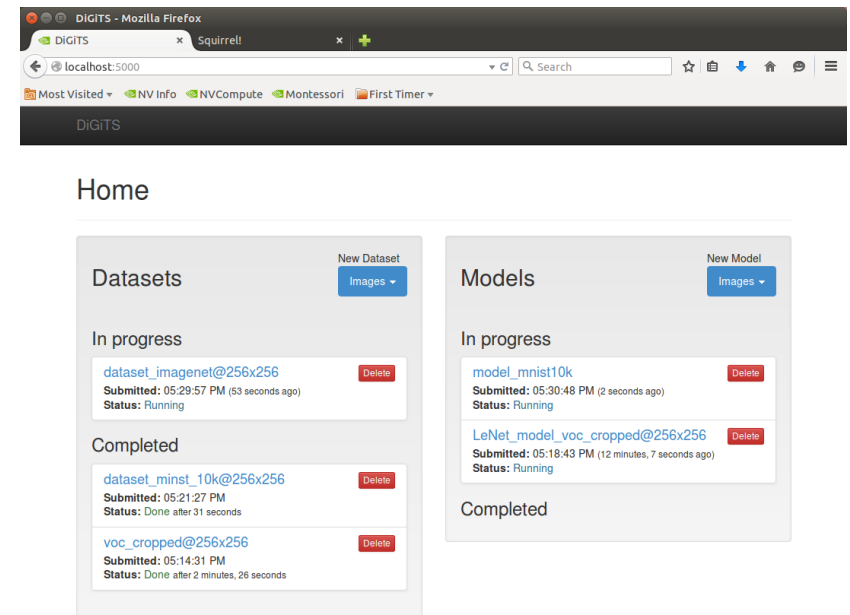
Available at <http://developer.nvidia.com/digits>

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)



DIGITS

Key Features

Visualize DNN topology and how training data activates your network

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

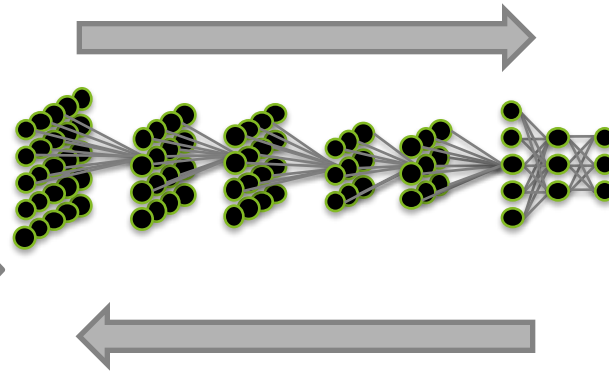
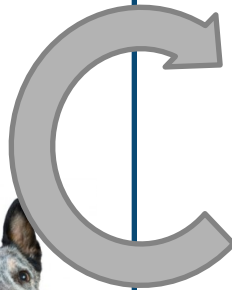
Simple setup and launch

Import a wide variety of image formats and sources

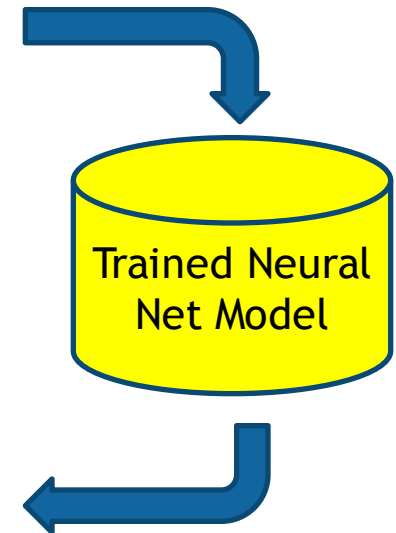
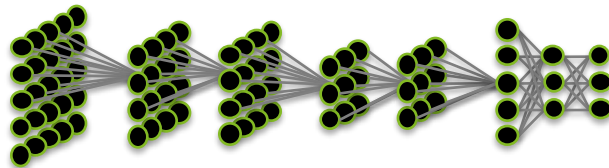
Monitor network training in real-time

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

What is Deep Learning Software?



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



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`)

Home

1/1 GPU available

No Jobs Running

Datasets

New Dataset

Images ▾

Models [View details](#)

New Model

Images ▾

NVIDIA DIGITS

LOGIN

DIGITS

Login Info ▾

Login

Username ?

Submit



DIGITS

nvidia-test (Logout)

Info ▾

Home

1/1 GPU available

No Jobs Running

Datasets

New Dataset

Images ▾

Models [View details](#)

New Model

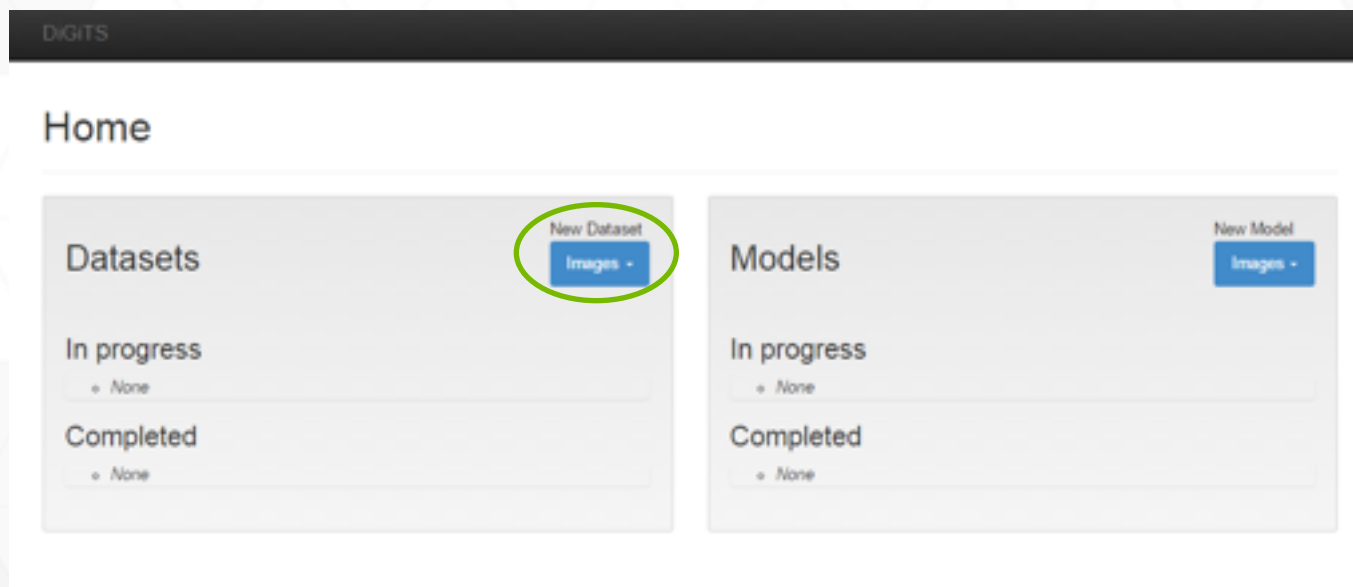
Images ▾

CREATING DATASETS

NVIDIA DIGITS

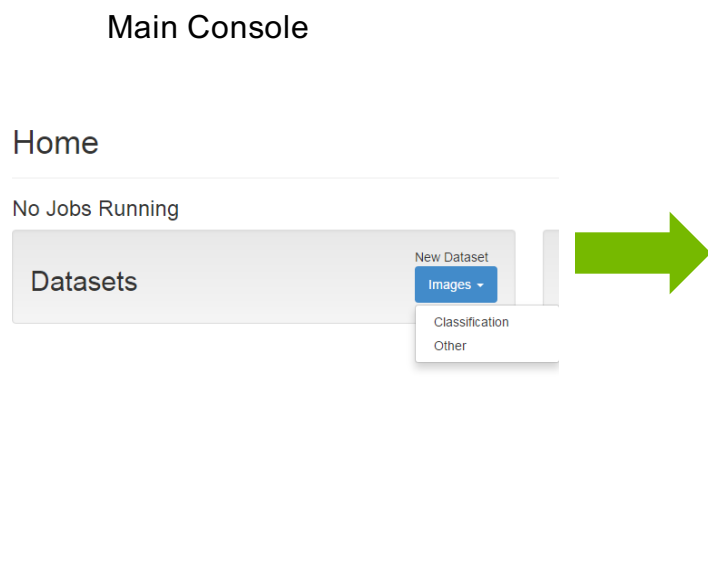
Creating your Dataset

Main Console



NVIDIA DIGITS

Creating your Dataset



New Image Classification Dataset

The form is titled 'New Image Classification Dataset'. It has two tabs at the top: 'Use Image Folder' (selected) and 'Use Text Files'. The form is divided into several sections:

- Image Type:** A dropdown menu with 'Color' selected.
- Image size:** Two input fields, both containing '256', separated by an 'x'.
- Resize Transformation:** A dropdown menu with 'Squash' selected. Below it is a 'See example' button.
- Training Images:** A text input field with the placeholder 'folder or URL'.
- Minimum samples per class:** An input field with '2'.
- Maximum samples per class:** An empty input field.
- % for validation:** An input field with '25'.
- % for testing:** An input field with '0'.
- Separate validation images folder:** An unchecked checkbox.
- Separate test images folder:** An unchecked checkbox.
- DB backend:** A dropdown menu with 'LMDB' selected.
- Image Encoding:** A dropdown menu with 'PNG (lossless)' selected.
- Dataset Name:** An empty text input field.
- Create:** A blue button at the bottom right.

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

MNIST 
Owner: nvidia-test

[Clone Job](#) [Delete Job](#)

Job Information

Job Directory
/home/pradeep/digits/digits/jobs/20160315-195658-2675

Image Dimensions
28x28

Image Type
Grayscale

Resize Transformation
Half crop, half fill

DB Backend
lmdb

Image Encoding
png

DB Compression
none

Dataset size
0 B

Parse Folder (train/val)

Folder
/home/pradeep/mnist

Job Status Done

- Initialized at 07:56:58 PM (1 second)
- Running at 07:57:00 PM (2 minutes, 48 seconds)
- Done at 07:59:48 PM
(Total - 2 minutes, 49 seconds)

Parse Folder (train/val) Done ▾

- Initialized at 07:56:58 PM

Create DB (train) Done ▾

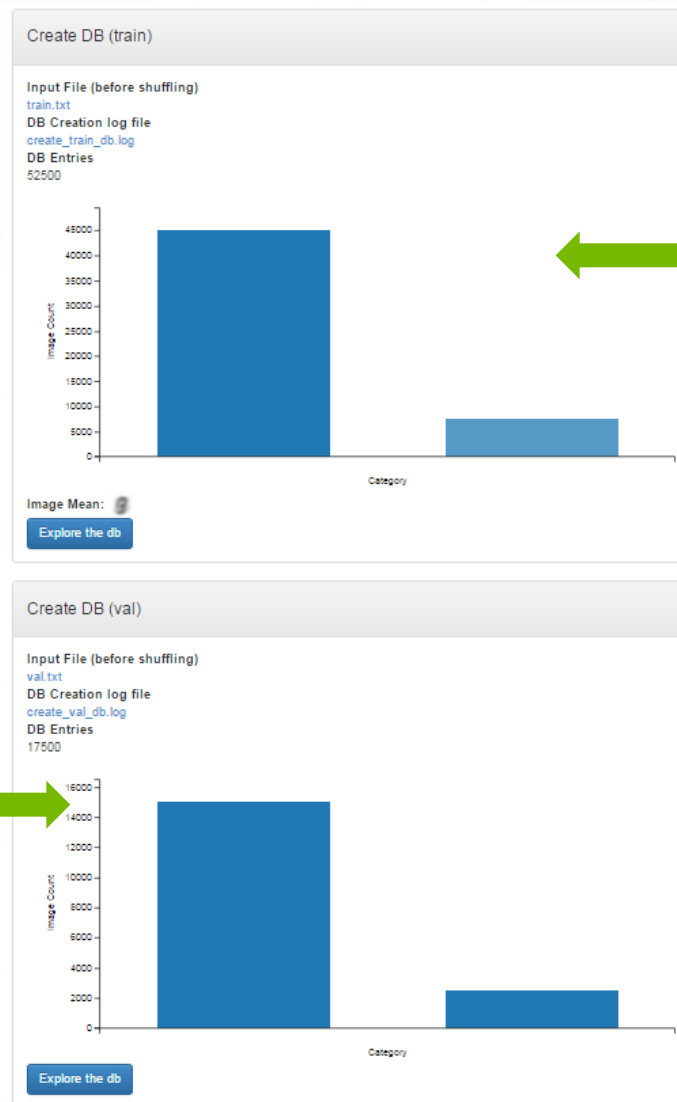
Create DB (val) Done ▾

- Initialized at 07:56:58 PM (3 seconds)
- Running at 07:57:01 PM (1 minute, 6 seconds)
- 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



Exploring MNIST (train_db) images

Show all images or filter by class: test train

Items per page: 10 - 25 - 50 - 100

« 0 1 2 3 4 5 ... 2099 »

6
train

0
test

2
train

6
train

5
train

9
test

4
train

4
train

2
train

2
train

8
train

4
train

1
train

6
train

8
test

6
test

5
train

6
train

2
train

9
train

3
train

6
test

7
train

4
train

5
train

DIGITS DEMO

Creating your Dataset

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



Home

No Jobs Running

Datasets

[New Dataset](#)
[Images ▾](#)

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
<input type="radio"/> LeNet	Original paper [1998]	28x28 (gray)
<input type="radio"/> AlexNet	Original paper [2012]	256x256
<input type="radio"/> 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

Select Dataset ⓘ
MNIST

☐ Use client side file
Python Layer File (server side) ⓘ

MNIST
Done Tue Mar 15, 07:59:45 PM
Image Size
28x28
Image Type
GRAYSCALE
DB backend
lmdb
Create DB (train)
52500 images
Create DB (val)
17500 images

Data Transformations
Crop Size ⓘ

Subtract Mean ⓘ

Solver Options
Training epochs ⓘ

Snapshot interval (in epochs) ⓘ

Validation interval (in epochs) ⓘ

Random seed ⓘ

Batch size ⓘ

Solver type ⓘ

Base Learning Rate ⓘ

☐ 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)	Customize
○ AlexNet	Original paper [2012]	256x256	
○ GoogLeNet	Original paper [2014]	256x256	

Model Name ⓘ

NVIDIA DIGITS

Train a Network-Advance 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

Customize



Standard Networks Previous Networks Custom Network

Caffe Torch (experimental)

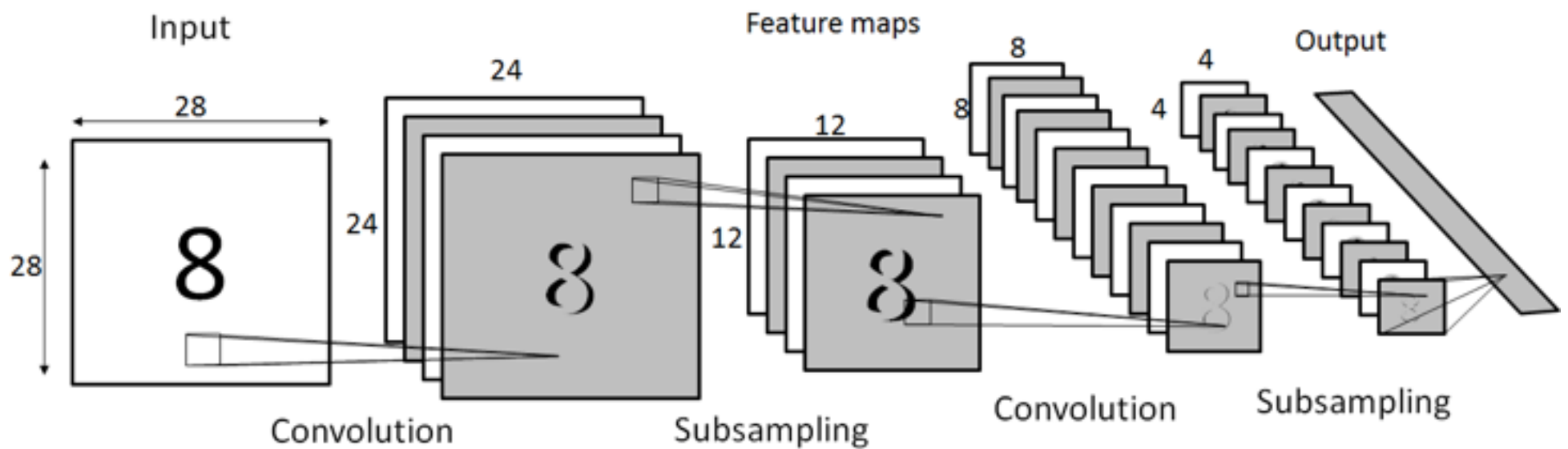
Custom Network ⓘ Visualize

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

Pretrained model(s) ⓘ

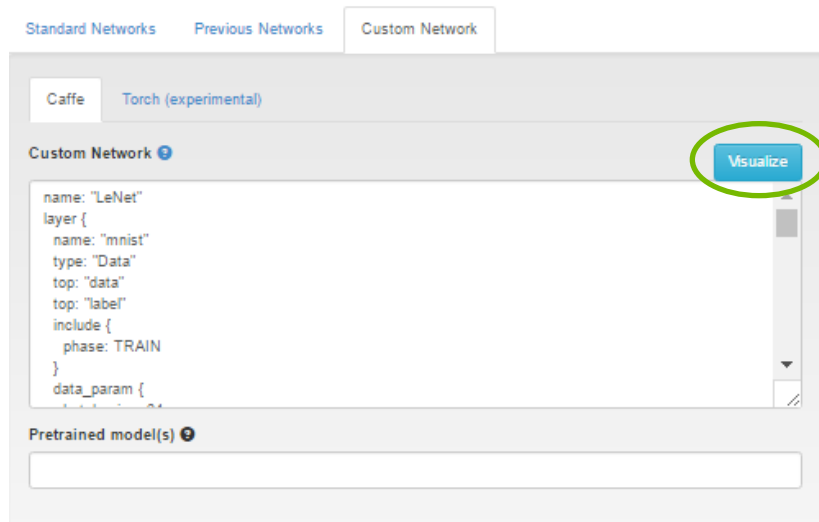
LENET

Network Configuration

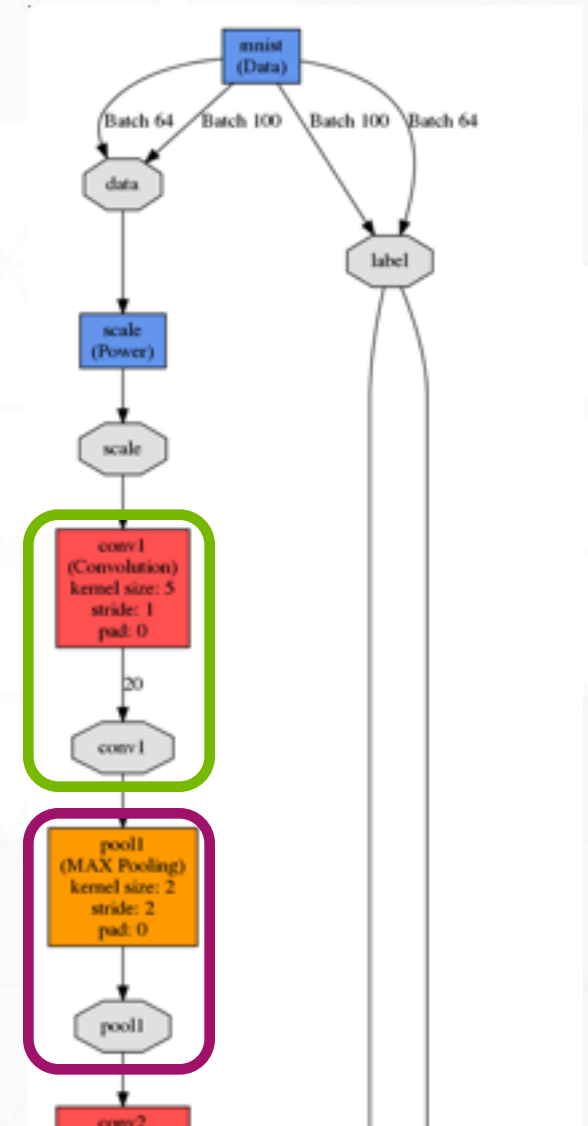
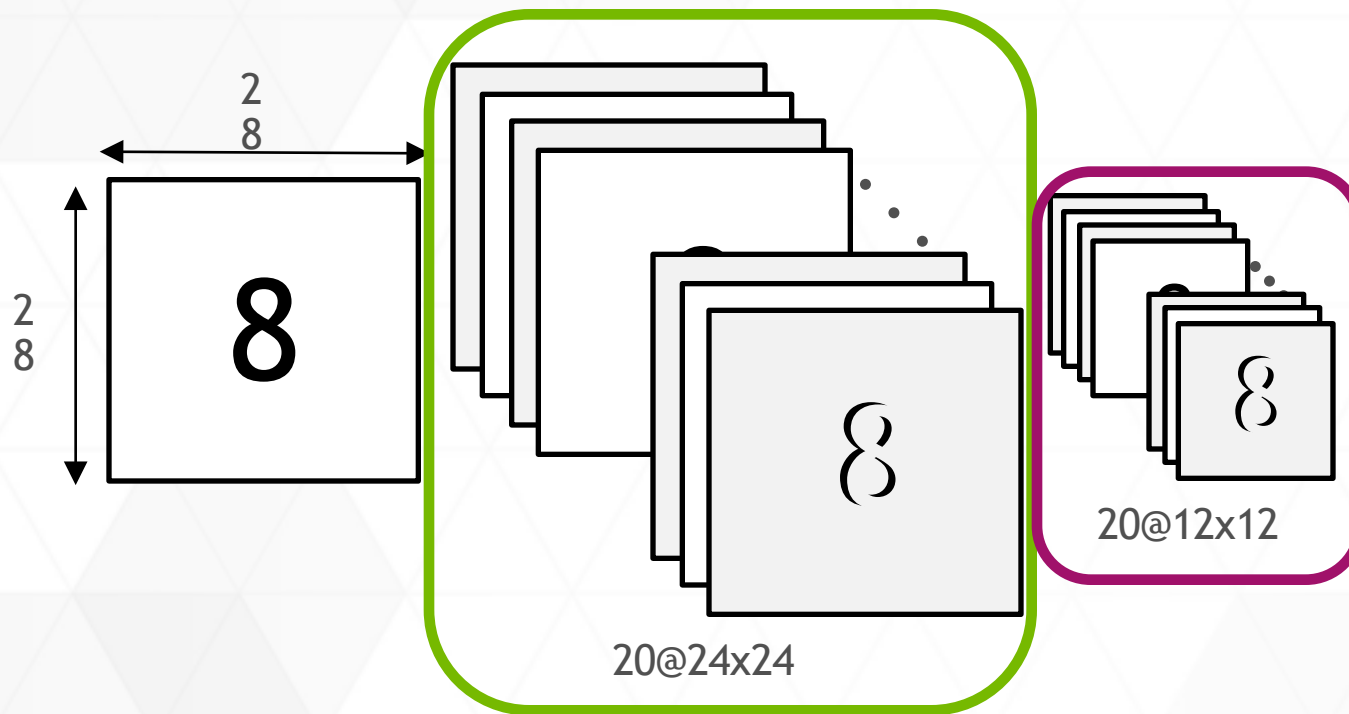


NVIDIA DIGITS

Train a Network-Advance Options



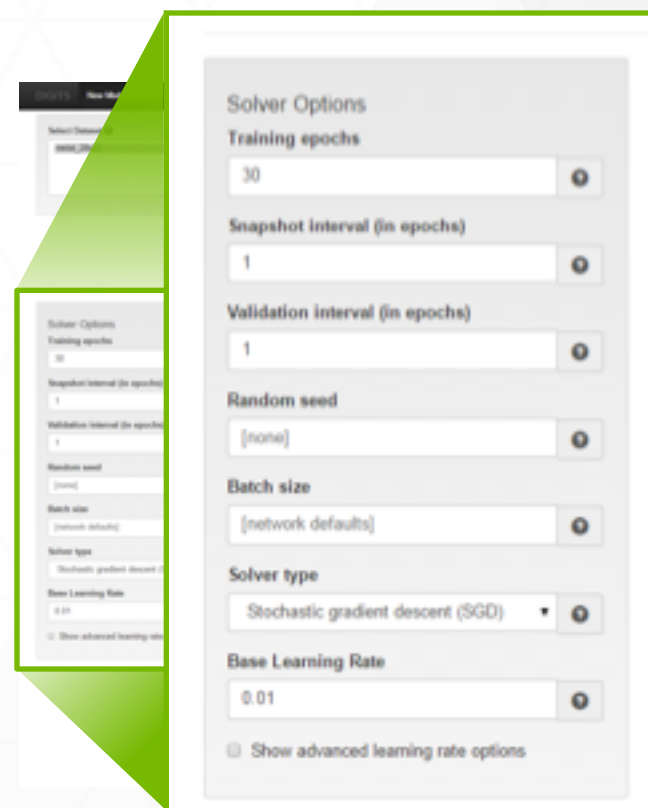
NETWORK PARAMETERS



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



The screenshot displays the 'Train a Network' interface in NVIDIA DIGITS, specifically the 'Solver Options' panel. The panel contains the following settings:

- 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

NVIDIA DIGITS

Single and multi-GPU training is easy

Single GPU system

The screenshot shows the 'New Image Classification Model' interface. On the left, the 'Server Options' panel includes fields for 'Training epochs' (10), 'Resnet interval (in epochs)' (1), 'Validation interval (in epochs)' (1), 'Random seed' (0), 'Batch size' (32), 'Network type' (Resnet50), and 'Base Learning Rate' (0.01). In the center, the 'Standard Networks' table lists three options:

Network	Details	Intended image size
LeNet	Original paper (1998)	28x28 pixel
ResNet	Original paper (2015)	224x224
SqueezeNet	Original paper (2016)	224x224

On the right, the 'Data Transformations' panel has 'Crop Size' (224) and 'Subtract Mean File' (Yes). At the bottom, there is a 'Model Name' input field and a 'Create' button.

Multiple GPU system

This screenshot shows the same interface as the single GPU system, but with additional options for multi-GPU training. The 'Server Options' panel includes a 'Number of GPUs' field set to 1. The 'Custom Networks' panel on the right is expanded, showing a list of custom networks. A green box highlights the 'Use how many GPUs (if available)' dropdown menu, which is currently set to 1. Below this, a list of GPU configurations is visible, including '4x 1080i 4GB', '4x 1080i 8GB', and '4x 1080i 16GB'. The 'Model Name' field and 'Create' button are at the bottom.

Select the number of GPUs you want to use.

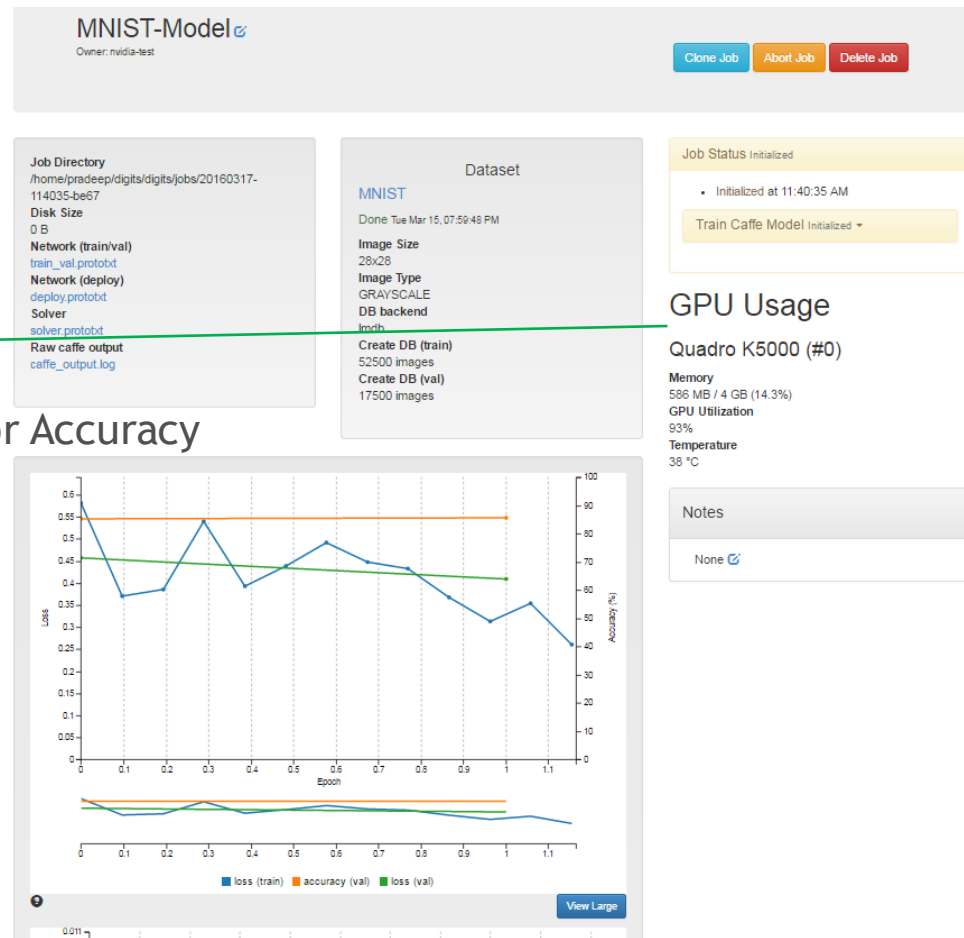
NVIDIA DIGITS

Training Results

Monitor GPU, Memory usage and
Temperature

Monitor Accuracy

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

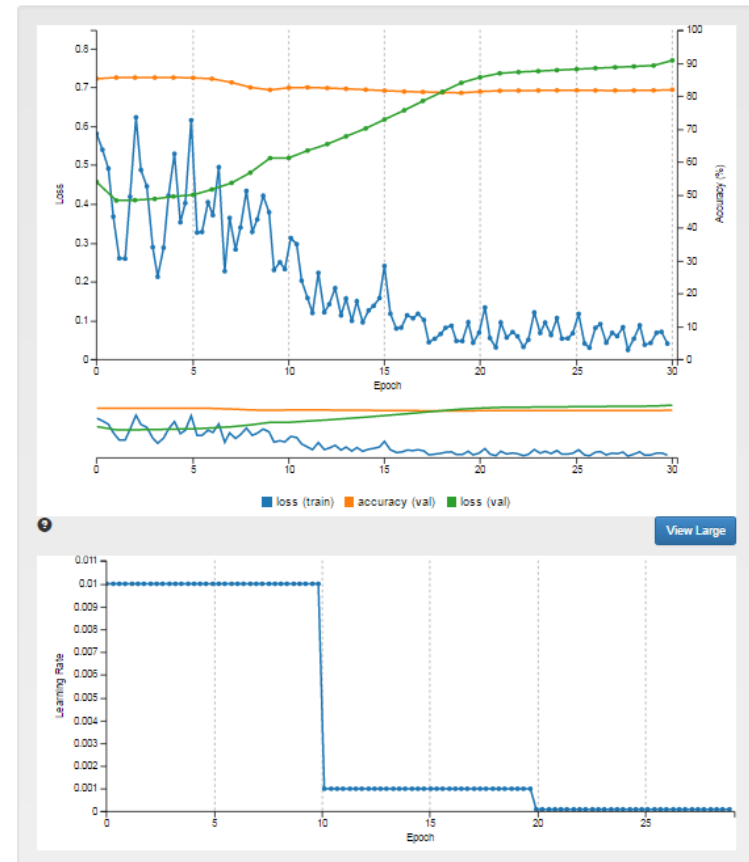


NVIDIA DIGITS

Training Results

First Graph is about the Loss and Accuracy graphs

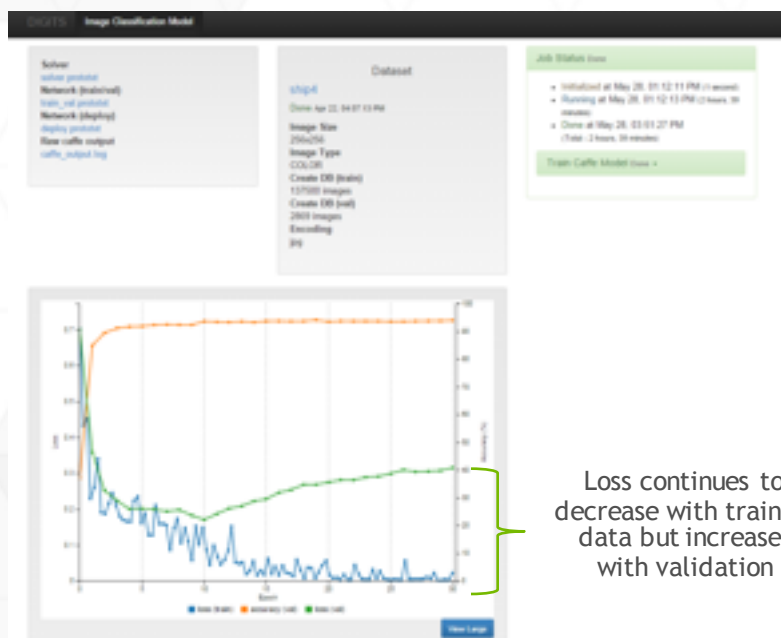
Second Graph is about the Learning rate, as the training progresses, learning rate goes down as model is getting more mature.



OVERFITTING AND UNDERFITTING

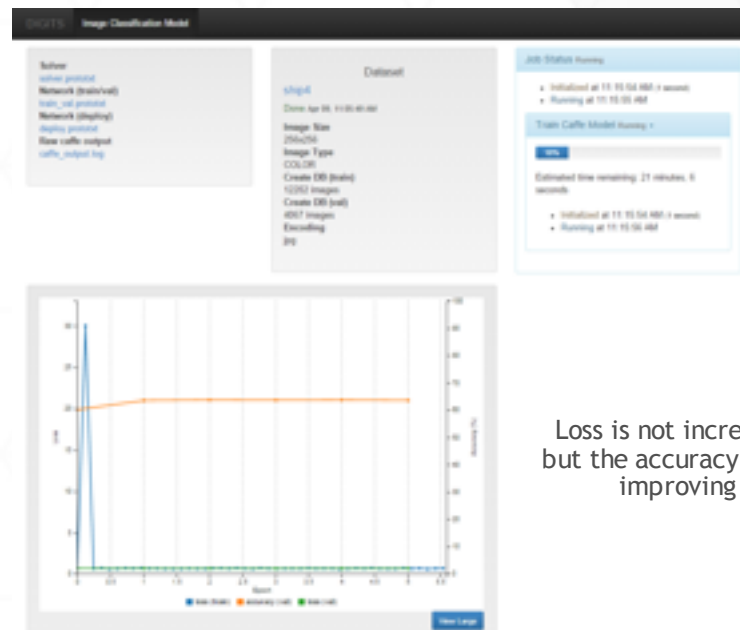
How can I use DIGITS to tell me this is happening?

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
[Text Field]

Upload image
Choose File No file chosen

☐ Show visualizations and statistics ⓘ

Classify One Image ⓘ

Upload Image List
Choose File No file chosen
Accepts a list of filenames or urls (you can use your rat.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

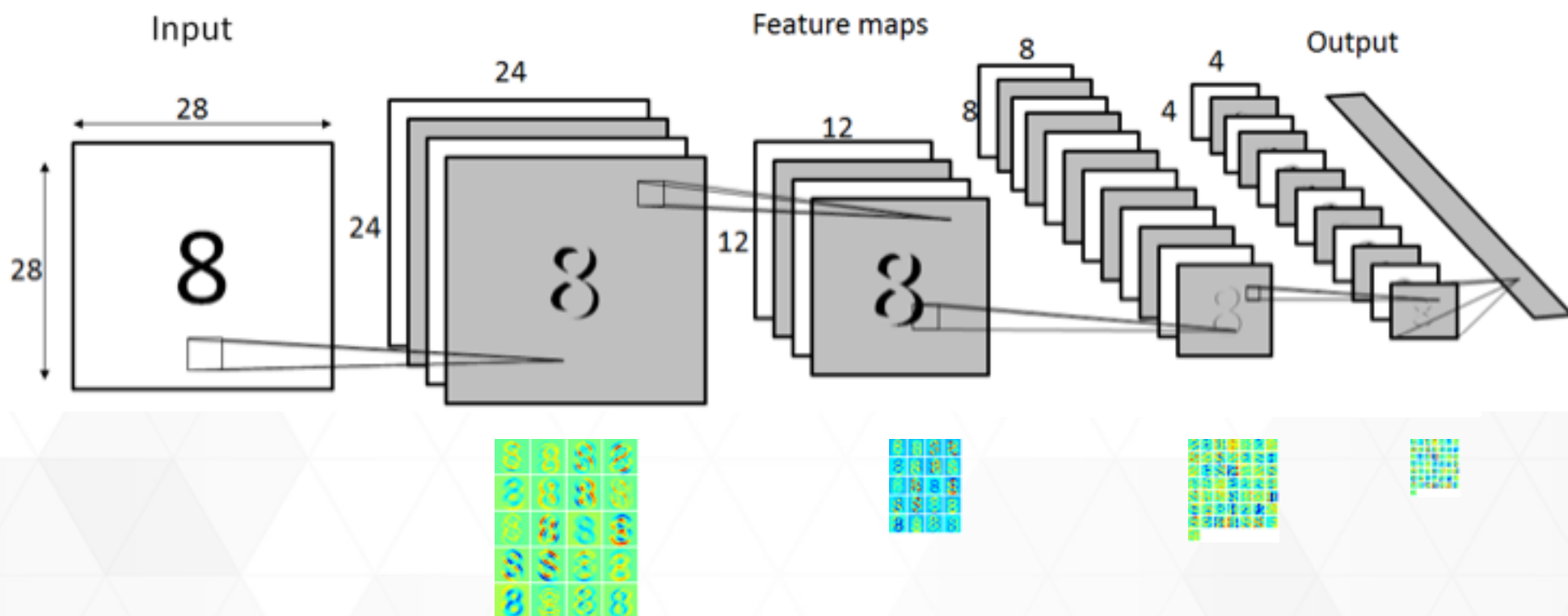
SINGLE IMAGE CLASSIFICATION RESULTS

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



NETWORK CONFIGURATION

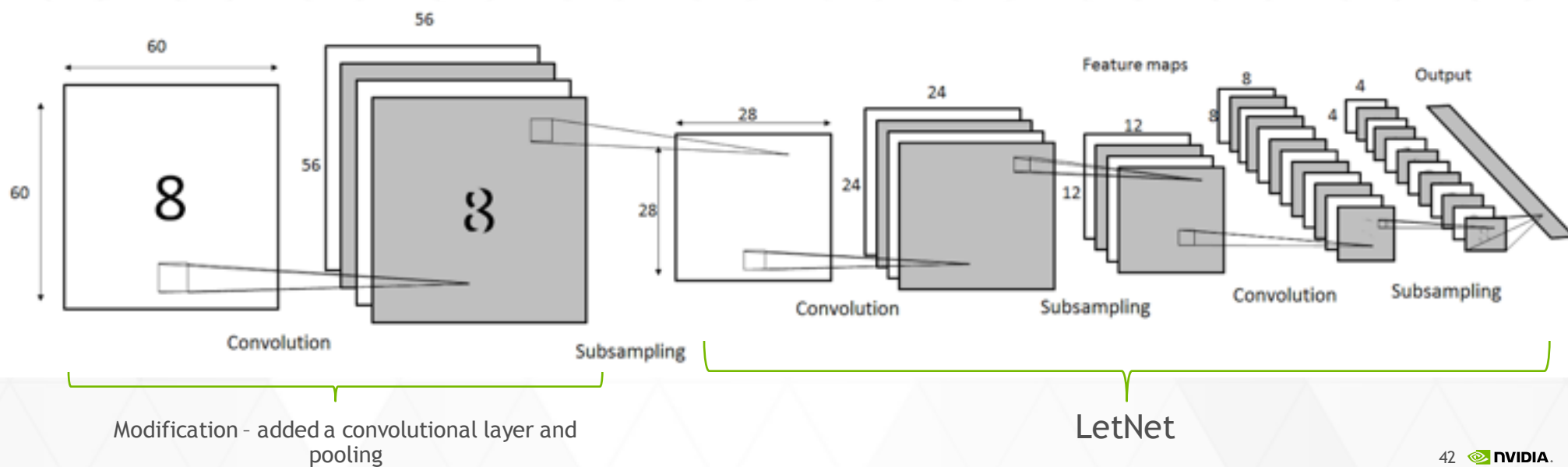
Visualizing LetNet Network Responses



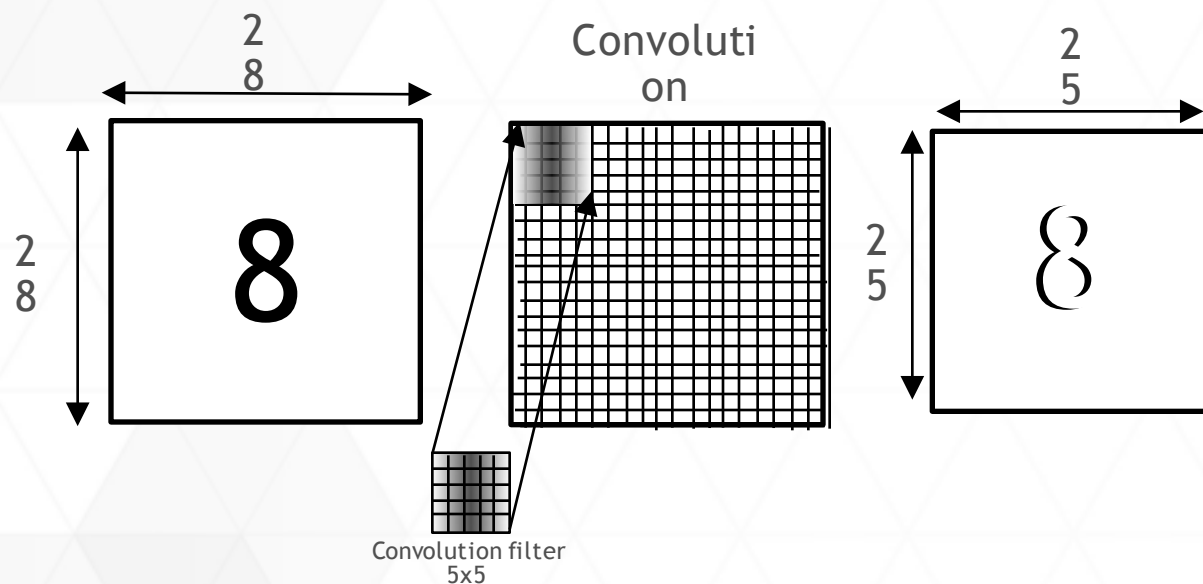
NETWORK CONFIGURATIONS

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



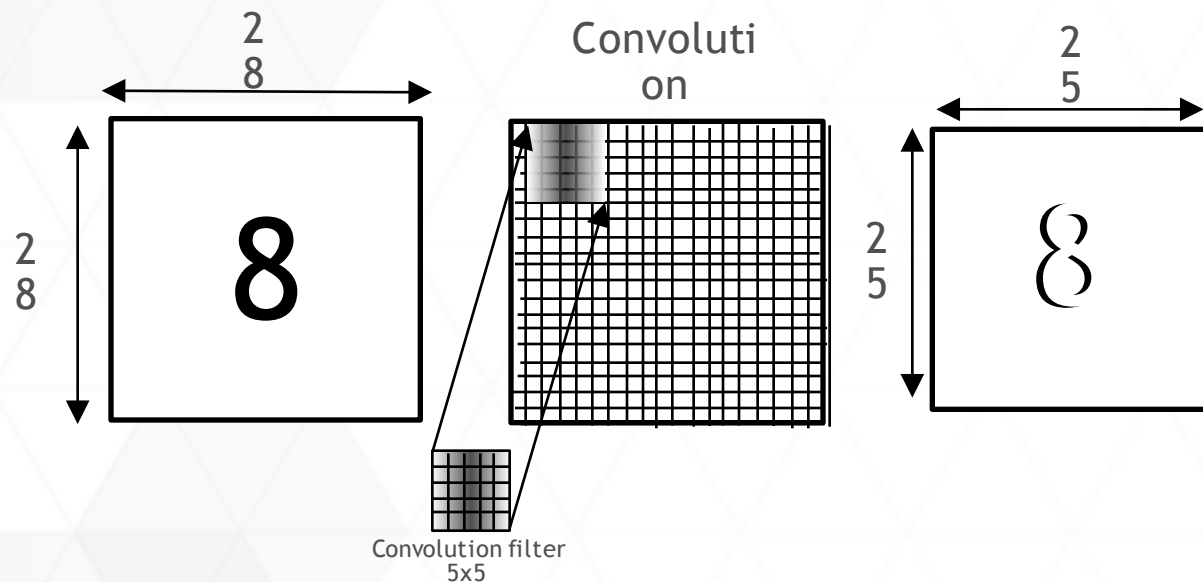
MODIFYING YOUR NETWORK



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

```
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"
    }
  }
}
```

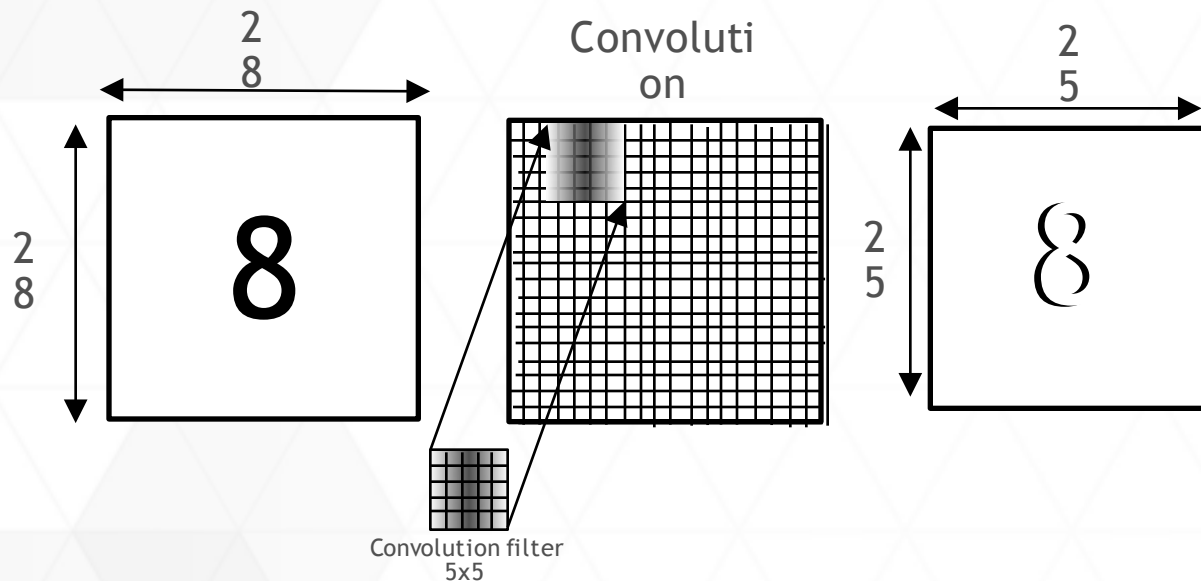
MODIFYING YOUR NETWORK



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

```
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"
    }
  }
}
```

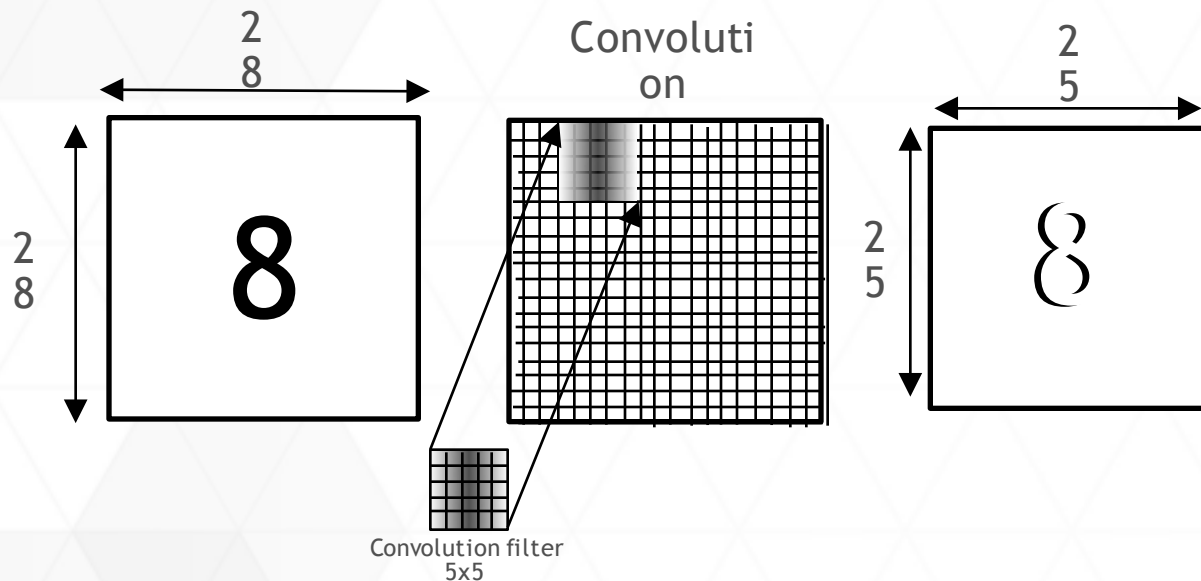
MODIFYING YOUR NETWORK



Zero padding, input is reduced by the $2 \times \text{radius}$ of the kernel
 $\text{Input}[28 \times 28] \times \text{filter}[5 \times 5] = \text{FeatureMap}[25 \times 25]$

```
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 $2 \times \text{radius}$ of the kernel
 $\text{Input}[28 \times 28] * \text{filter}[5 \times 5] = \text{FeatureMap}[25 \times 25]$

```
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"
    }
  }
}
```

```
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

SELECT DATASET

mini_256x256

mini_256x256

Done 01:01:02 PM

Image Size
256x256

Image Type
COLOR

Create DB (train)
40962 images

Create DB (val)
16398 images

Encoding
png

Data Transformations

Crop Size
none

Subtract Mean File
Yes

Solver Options

Training epochs
30

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

Standard Networks **Previous Networks** **Custom Network**

Custom Network

name: "LeNet"
layer [
 name: "input"
 type: "Data"
 top: "data"
 include: [
 phase: "TRAIN"
]
 data_param: {

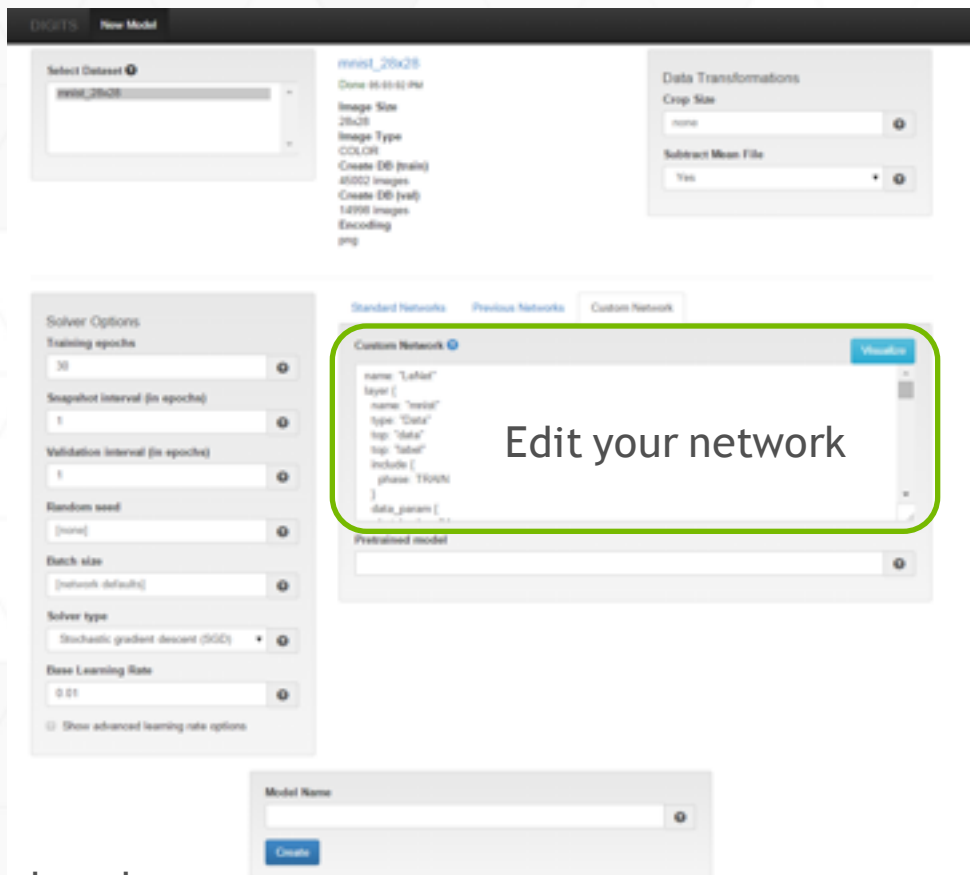
Edit your network

Pretrained model

Model Name

Create

MODIFYING YOUR NETWORK



* Lab tasks

```
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"
    }
  }
}
```

} Input and output to the layer

MODIFYING YOUR NETWORK

The screenshot shows the Digits web interface with the 'New Model' tab selected. The 'Select Dataset' dropdown is set to 'mnist_28x28'. The 'Data Transformations' section shows 'Crop Size' set to 'none' and 'Subtract Mean File' set to 'Yes'. The 'Solver Options' section includes fields for 'Training epochs' (30), 'Snapshot interval (in epochs)' (1), 'Validation interval (in epochs)' (1), 'Random seed' (none), 'Batch size' (network default), 'Solver type' (Stochastic gradient descent (SGD)), and 'Base Learning Rate' (0.01). The 'Custom Network' tab is active, displaying a JSON configuration for a layer named 'relu1'. A green box highlights the 'Edit your network' text in the JSON editor.

```
{
  "name": "relu1",
  "layer": {
    "name": "relu1",
    "type": "Data",
    "top": "data",
    "bottom": "conv1",
    "include": {
      "phase": "TRAIN"
    },
    "data_params": {
    }
  }
}
```

Pretrained model

Model Name

Create

Edit your network

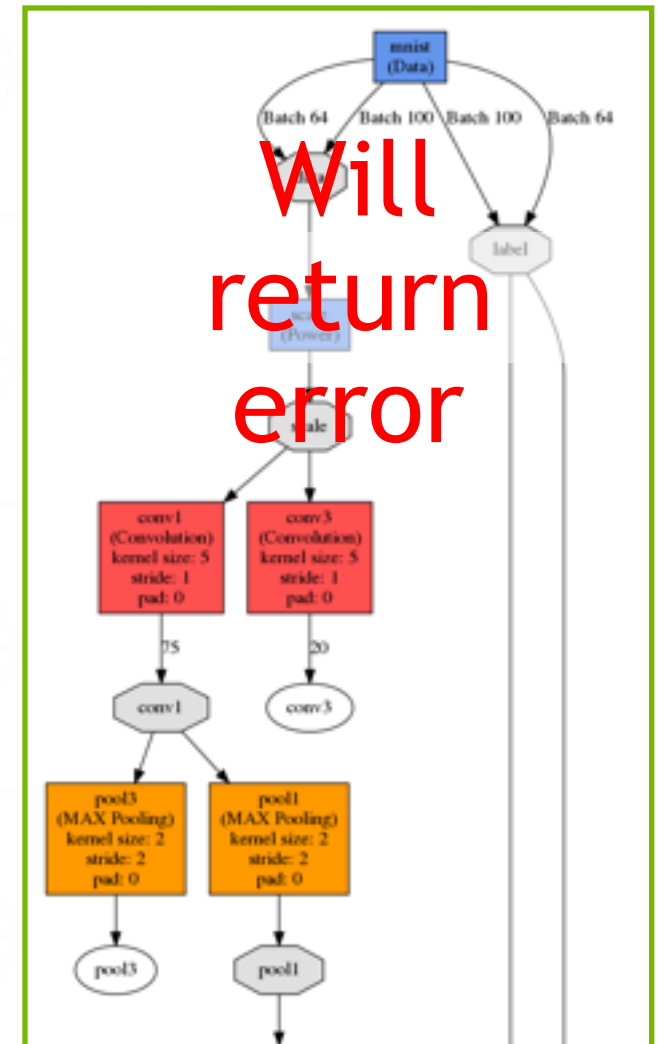
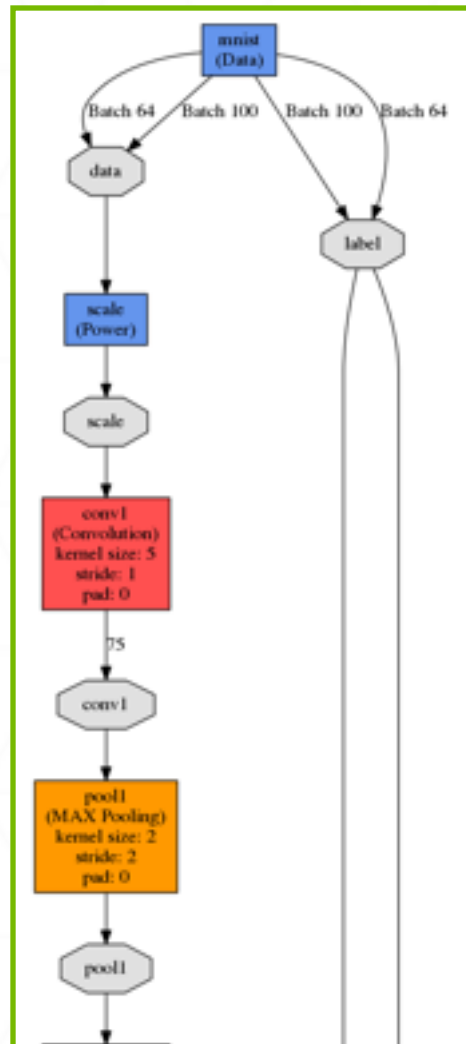
```
* layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}
```

} Input and output to the layer

* Lab tasks

MODIFYING YOUR NETWORK

Visualize Configuration Changes



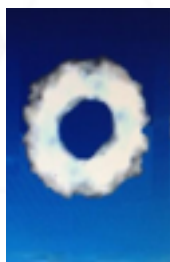
ANOTHER WAY TO IMPROVE PERFORMANCE

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 to be classified correctly when the network is trained with this digit data?



ANOTHER WAY TO IMPROVE PERFORMANCE

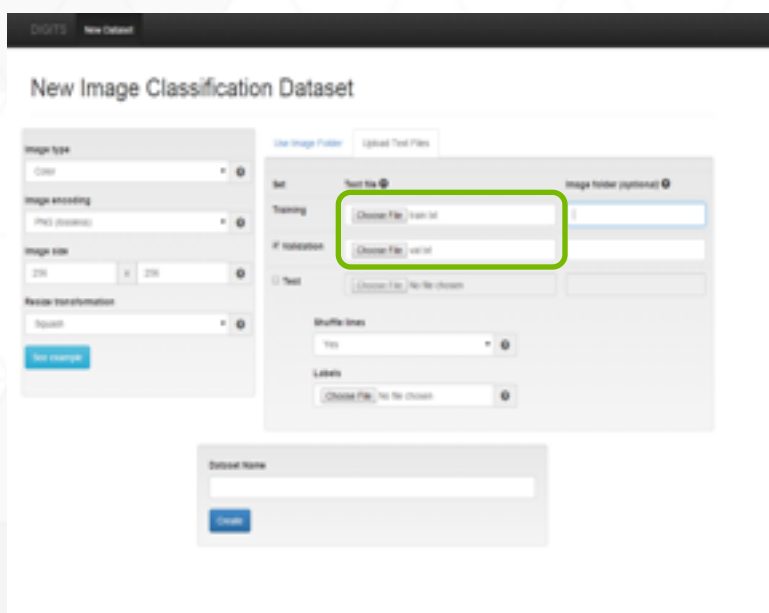
Data Augmentation

- 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

USING AN AUGMENTED DATA SET



The screenshot shows a web interface for creating a new image classification dataset. The form is titled 'New Image Classification Dataset'. It has several sections: 'Image type' with a dropdown set to 'Color'; 'Image encoding' with a dropdown set to 'PNG (preferred)'; 'Image size' with width and height set to 256; 'Resize transformation' with a dropdown set to 'Square'; and a 'New Image' button. Below this is a 'Dataset Name' field with a 'Create' button. To the right, there are tabs for 'Use Image Folder' and 'Upload Test Files'. The 'Use Image Folder' tab is active, showing fields for 'Set', 'Training', 'If Validation', and 'Test'. The 'Training' field is highlighted with a green box and contains the text '/home/user/train/0/0_1.jpg'. The 'If Validation' field contains '/home/user/train/0/0_1_invert.jpg'. The 'Test' field contains '/home/user/train/5/5_1.jpg'. Below these are fields for 'Shuffle lines' (set to 'Yes') and 'Labels' (set to '0').

- 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



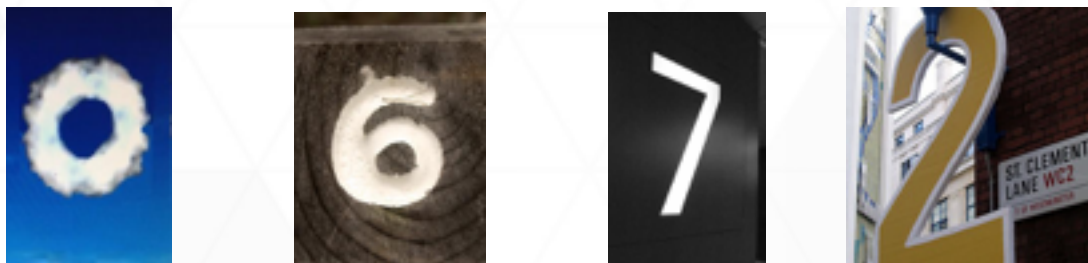
ANOTHER WAY TO IMPROVE PERFORMANCE

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



Tegra
a

Select Model

Epoch #30

Download the model you want to use

Download

Image URL

Upload Image

Choose File No file chosen

☐ Show visualizations and statistics

Classify One Image

Upload Image List

Choose File 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

DEPLOYMENT WITH TEGRA

Rapid Classification Anywhere

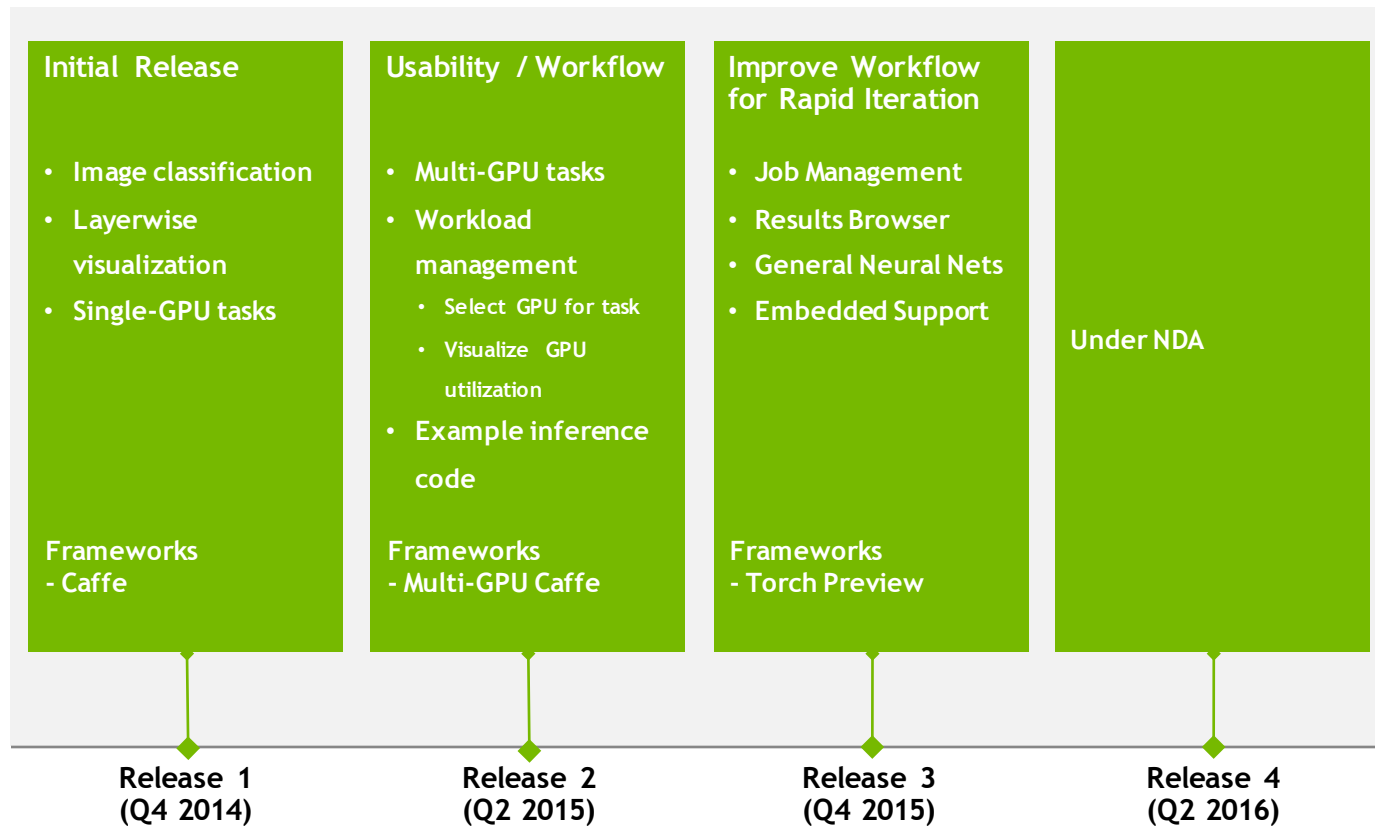
- Flexible
- Low Power
- Easy to use
- GPU accelerated



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
<input type="text" value="aerial"/>	<input type="text" value="Status"/>	<input type="text" value="Runtime"/>	<input type="text" value="Loss"/>
5layer_aerial	Aborted	00:00:11	
aerial_2layer	Aborted	00:03:20	87.3365
aerial_5layer	Running	00:00:01	1.26419
aerial_5layer	Aborted	00:02:12	
aerial_5layer	Aborted	00:07:35	
aerial_5layer_steprate	Running	00:00:01	0.825354099274
aerial_alexnet	Aborted	00:05:42	1.06914
aerial_deepnetwork	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.gwlab.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