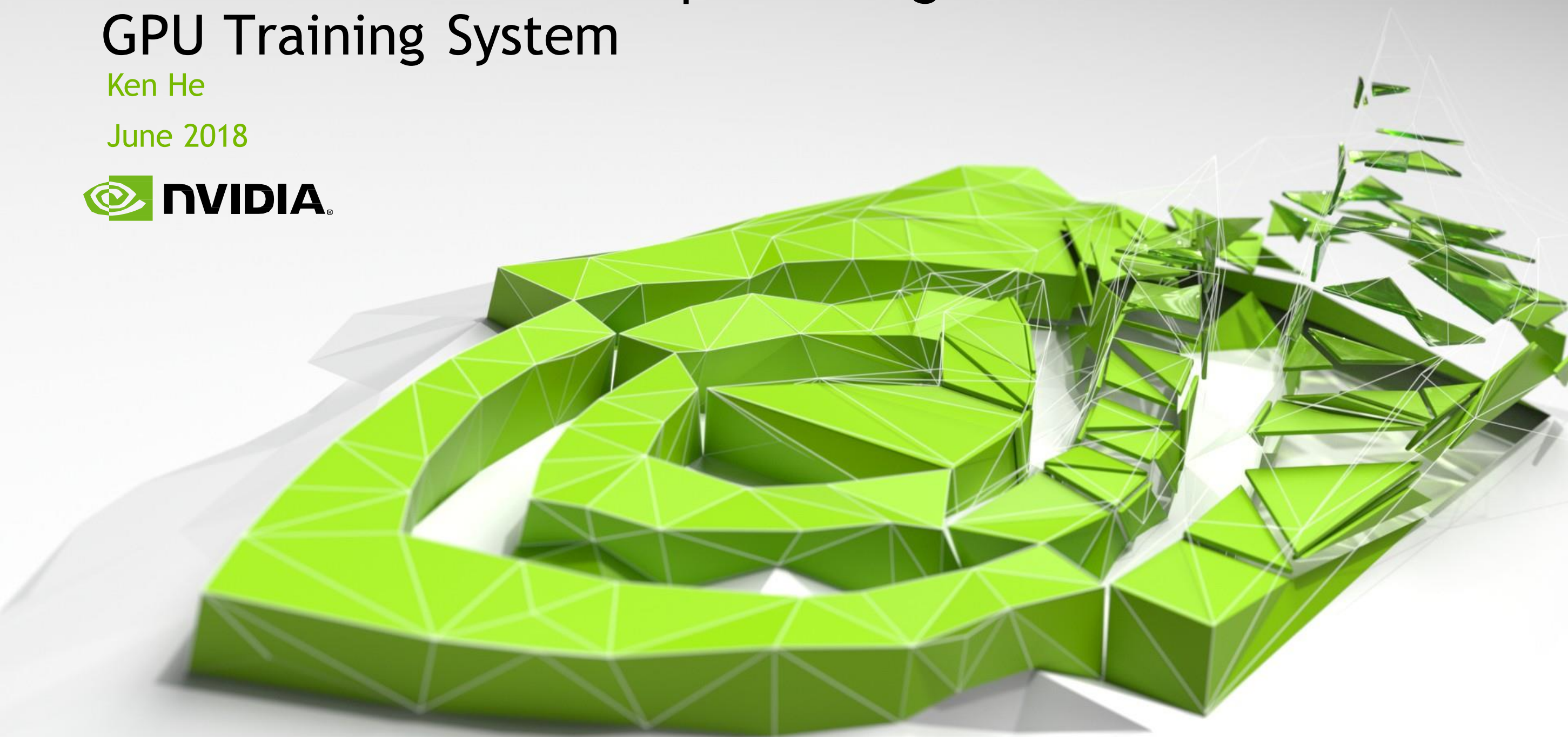


# DIGITS- Interactive Deep Learning GPU Training System

Ken He

June 2018



# Agenda

- Introduction to DIGITS
- CREATING DATASETS
- TRAIN A NETWORK
- 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

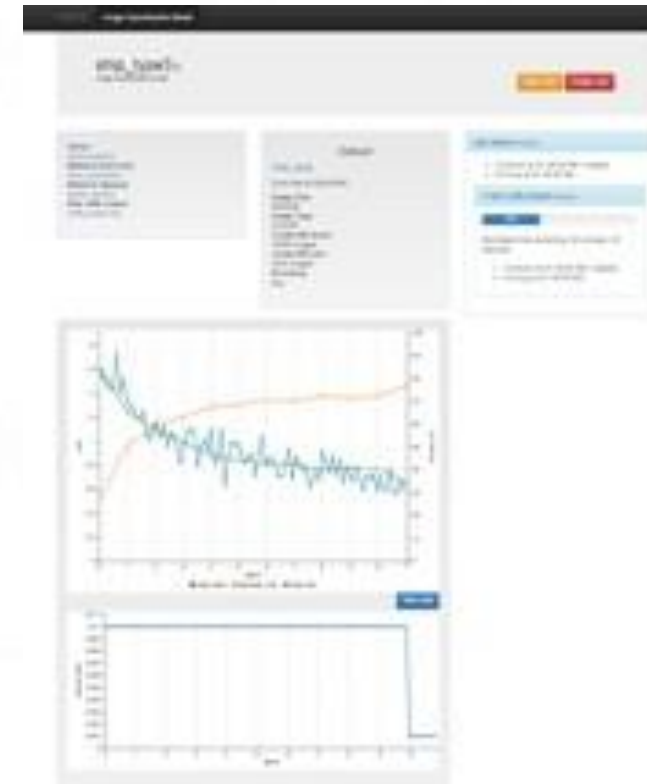
### Process Data



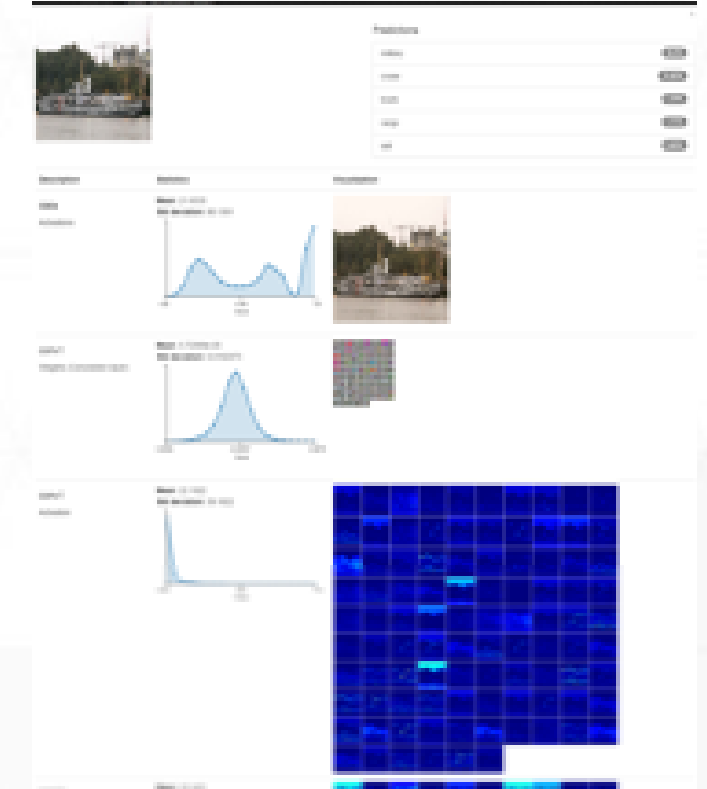
### Configure DNN

The 'Configure DNN' interface is titled 'New Image Classification Model'. It features a 'Model Name' field, a 'Model Type' dropdown menu, and a 'Model Size' dropdown menu. Below these are 'Model Options' including 'Preprocessing', 'Feature Extraction', and 'Classification'. A 'Model Summary' section on the right provides details about the model's architecture and training parameters.

### 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

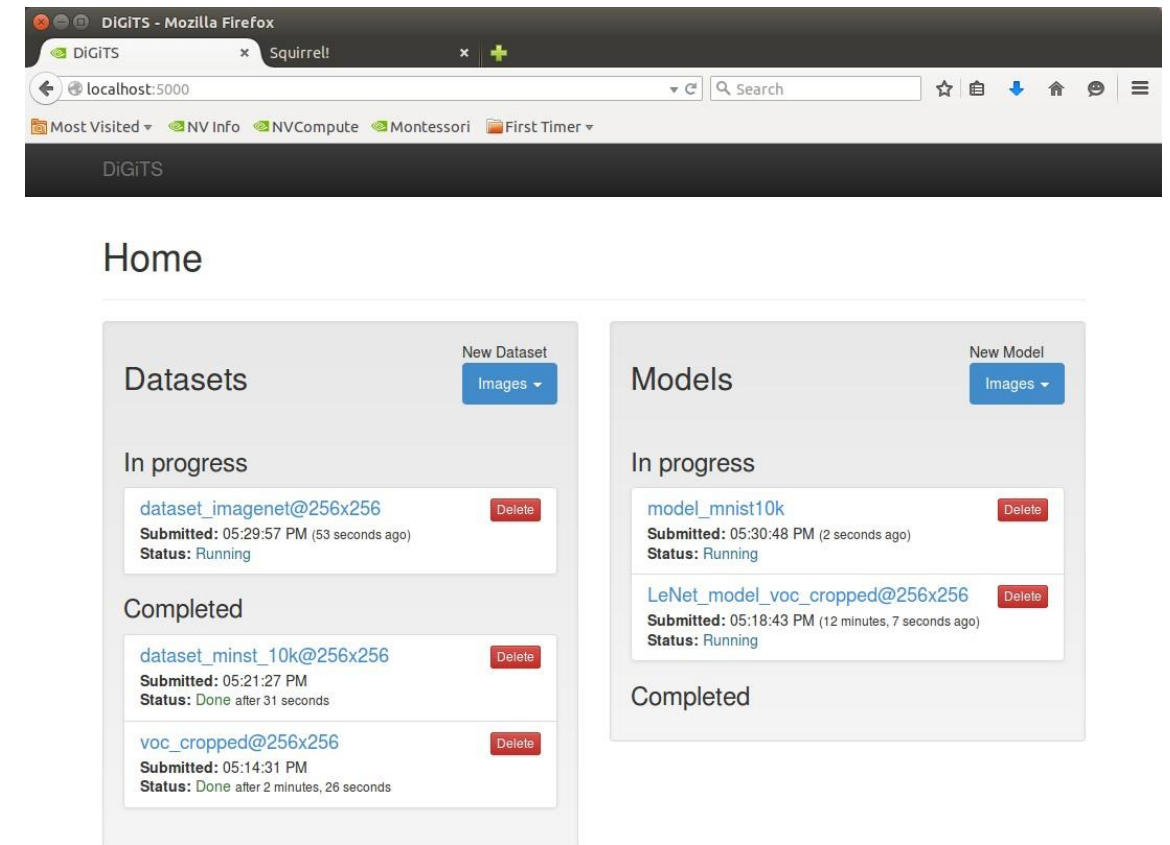
Available at <http://developer.nvidia.com/digits>

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

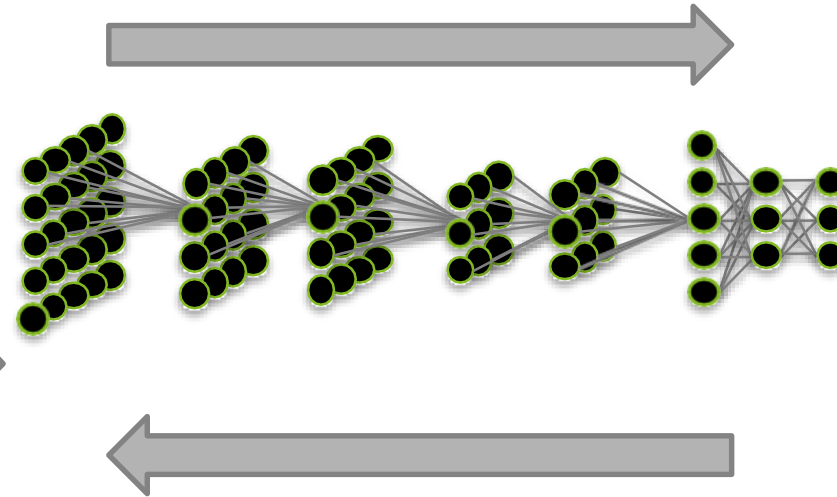
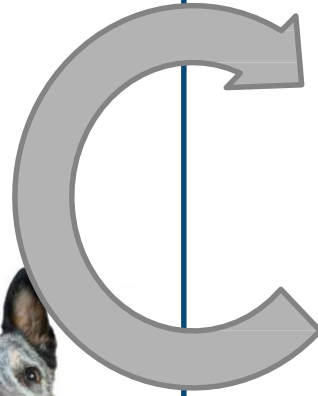
Import a wide variety of image formats and sources

Monitor network training in real-time

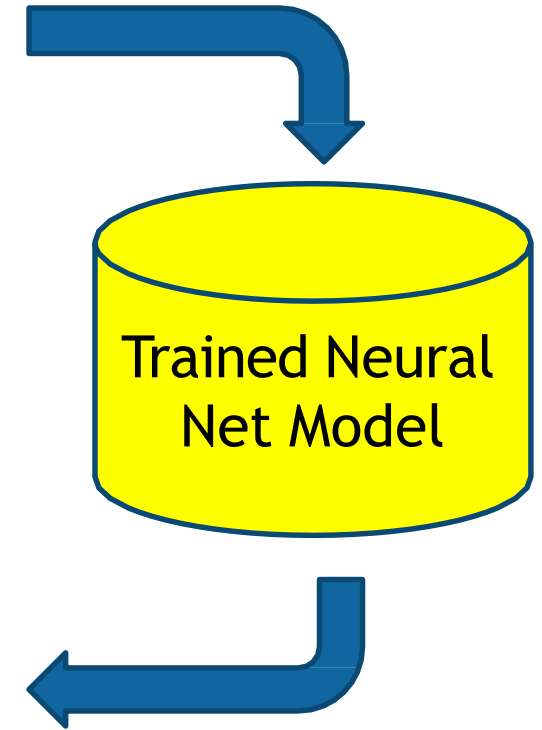
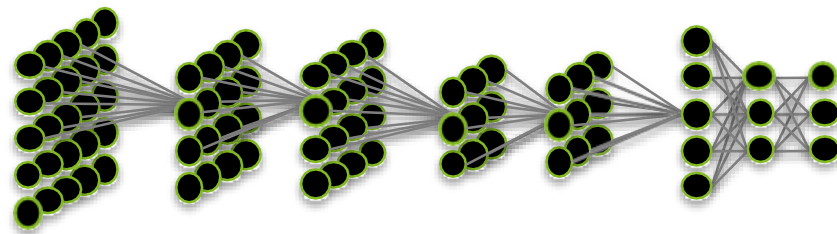
Open source, so DIGITS can be customized and extended as needed



# 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

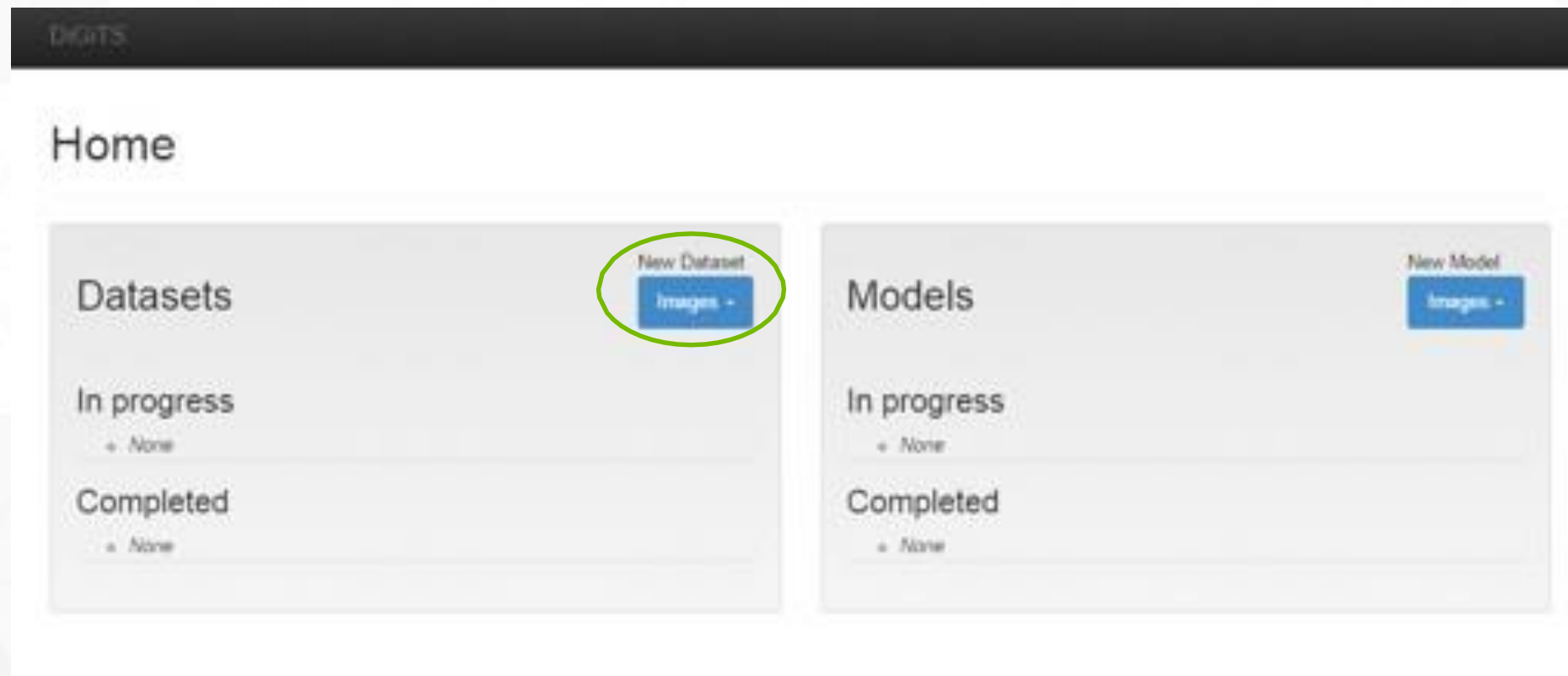
Classification

# CREATING DATASETS

# NVIDIA DIGITS

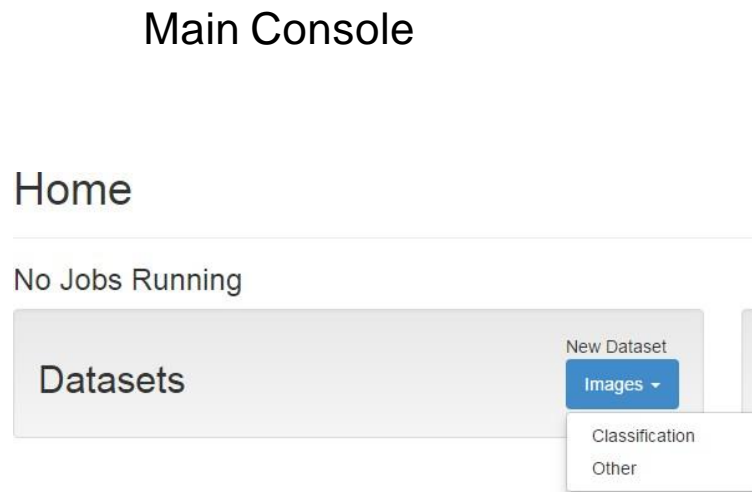
## Creating your Dataset

Main Console



# NVIDIA DIGITS

## Creating your Dataset



### New Image Classification Dataset

The screenshot shows the 'New Image Classification Dataset' form. It has two tabs: '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 for width and height, both set to '256', with an 'x' separator between them.
- Resize Transformation**: A dropdown menu with 'Squash' selected. Below it is a 'See example' button.
- Training Images**: A text input field for '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**: A checkbox that is unchecked.
- Separate test images folder**: A checkbox that is unchecked.
- 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.

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 JobDelete 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

Imdb

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

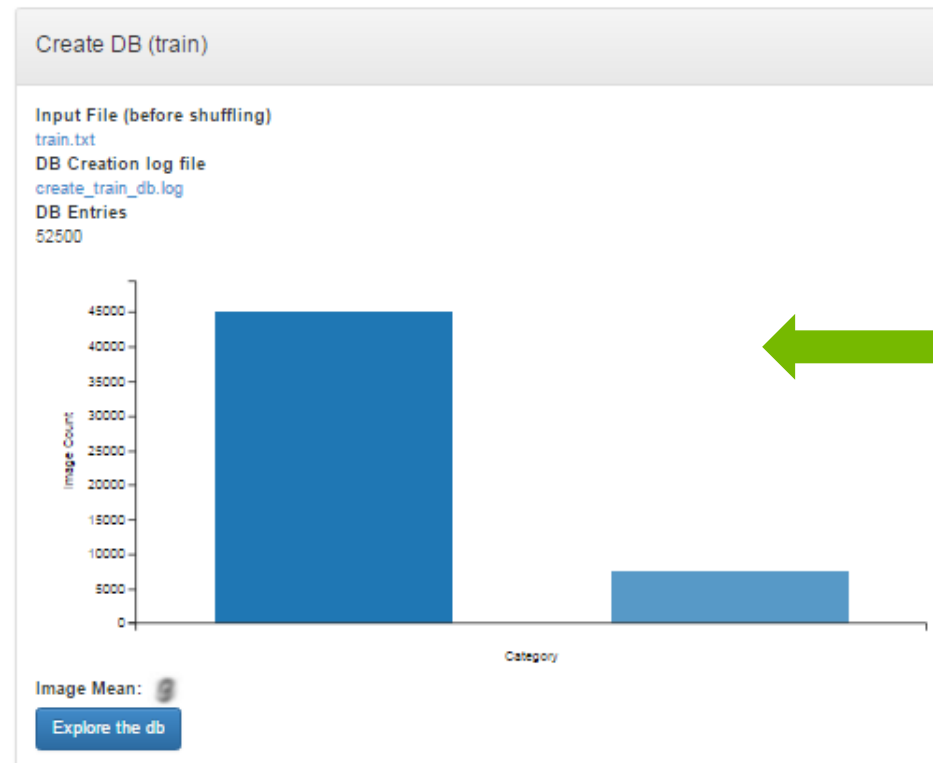
Total - 1 minute, 9 seconds



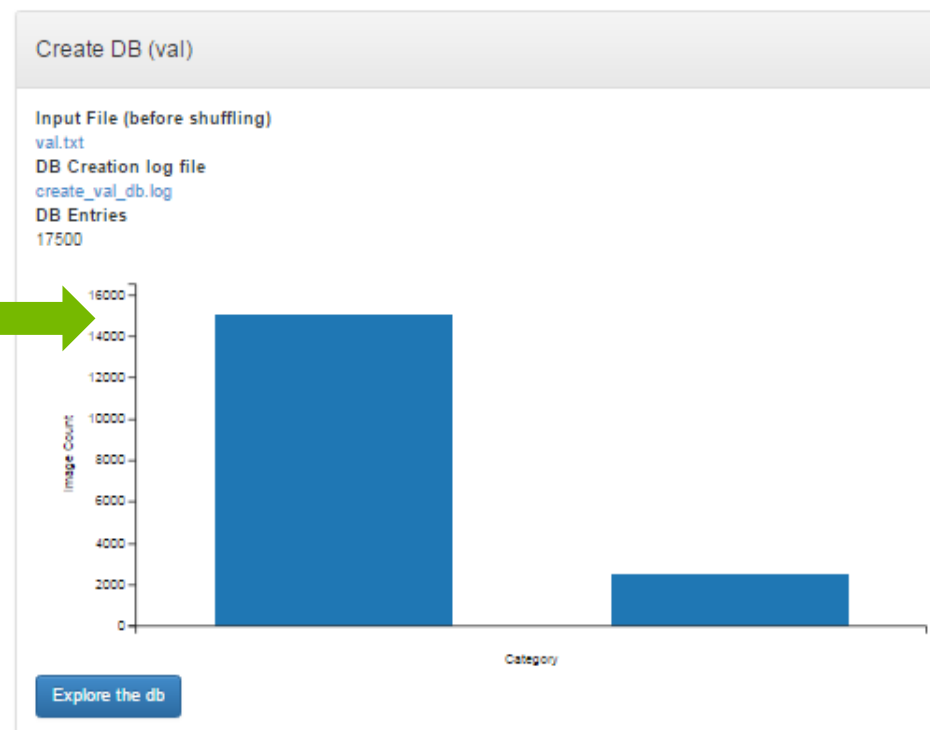
# NVIDIA DIGITS

## 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 ... 2000 »

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

Delete

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

# TRAIN A NETWORK

# TRAINING

## Choose Framework

With DIGITS 6.0, three frameworks are integrated into DIGITS

- Caffe
- Torch
- TensorFlow

# 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	<a href="#">Original paper [1998]</a>	28x28 (gray)
AlexNet	<a href="#">Original paper [2012]</a>	256x256
GoogLeNet	<a href="#">Original paper [2014]</a>	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, 07:59:48 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	<a href="#">Original paper [1998]</a>	28x28 (gray) <a href="#">Customize</a>
AlexNet	<a href="#">Original paper [2012]</a>	256x256
GoogLeNet	<a href="#">Original paper [2014]</a>	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	<a href="#">Original paper [1998]</a>	28x28 (gray)
Ⓞ AlexNet	<a href="#">Original paper [2012]</a>	256x256
Ⓞ GoogLeNet	<a href="#">Original paper [2014]</a>	256x256

Customize



Standard Networks Previous Networks Custom Network

Caffe Torch (experimental)

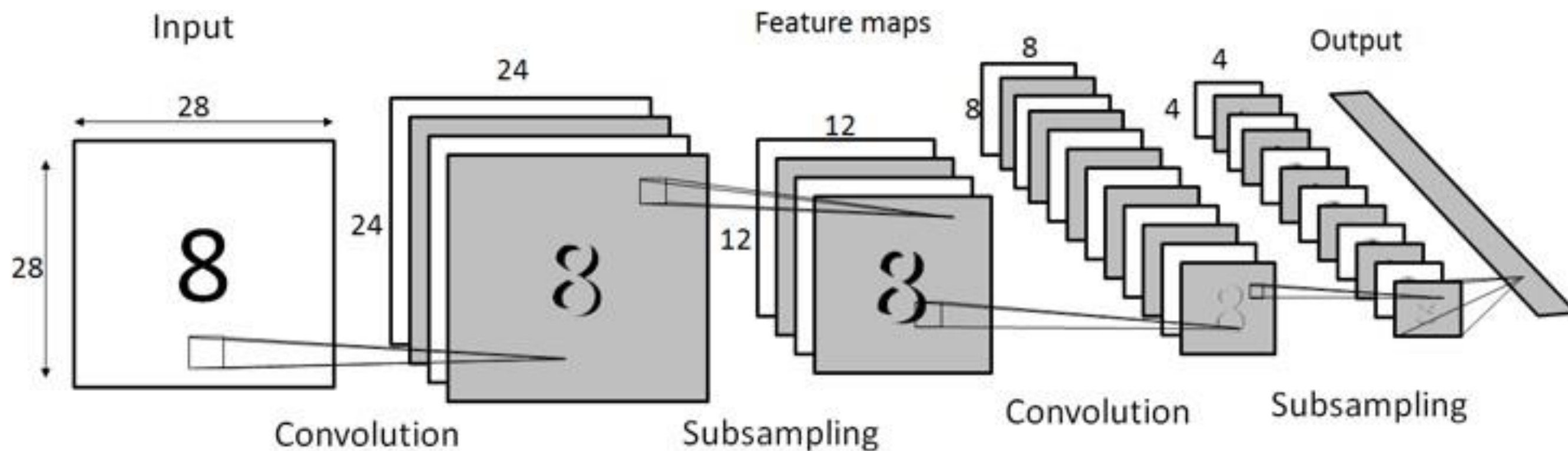
Custom Network ? Visualize

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

Pretrained model(s) ?

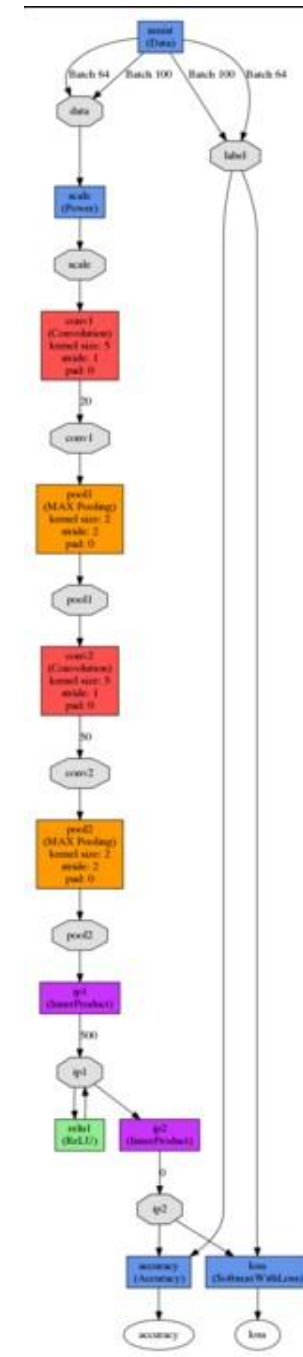
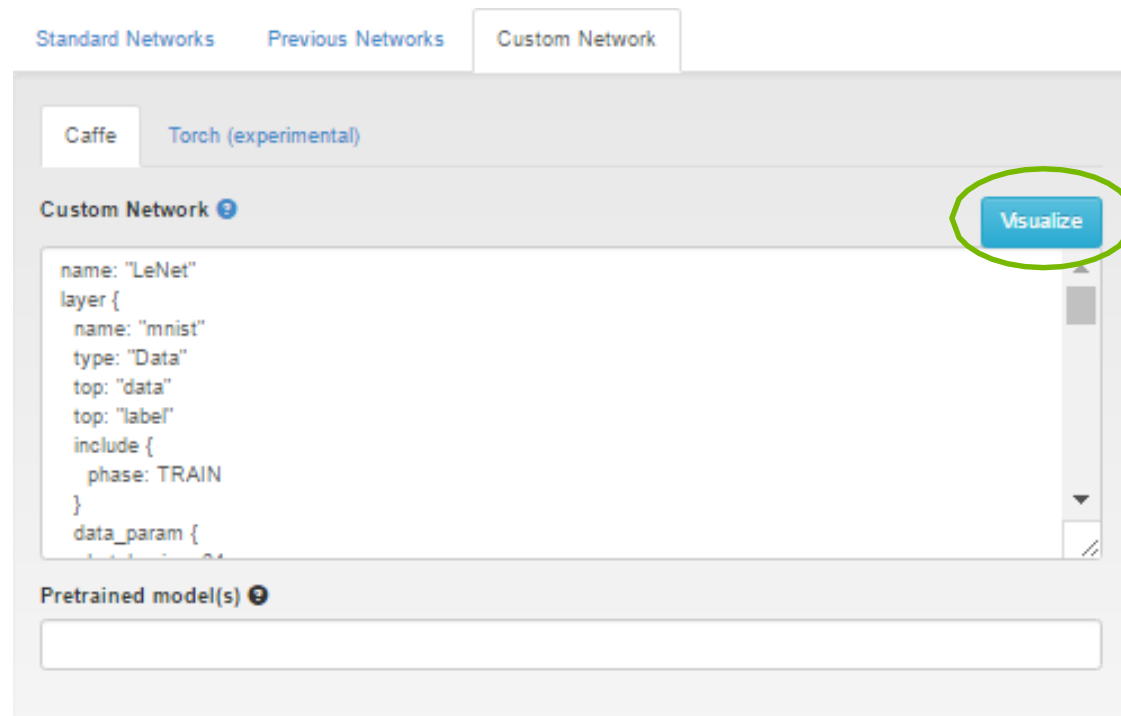
# LENET

## Network Configuration

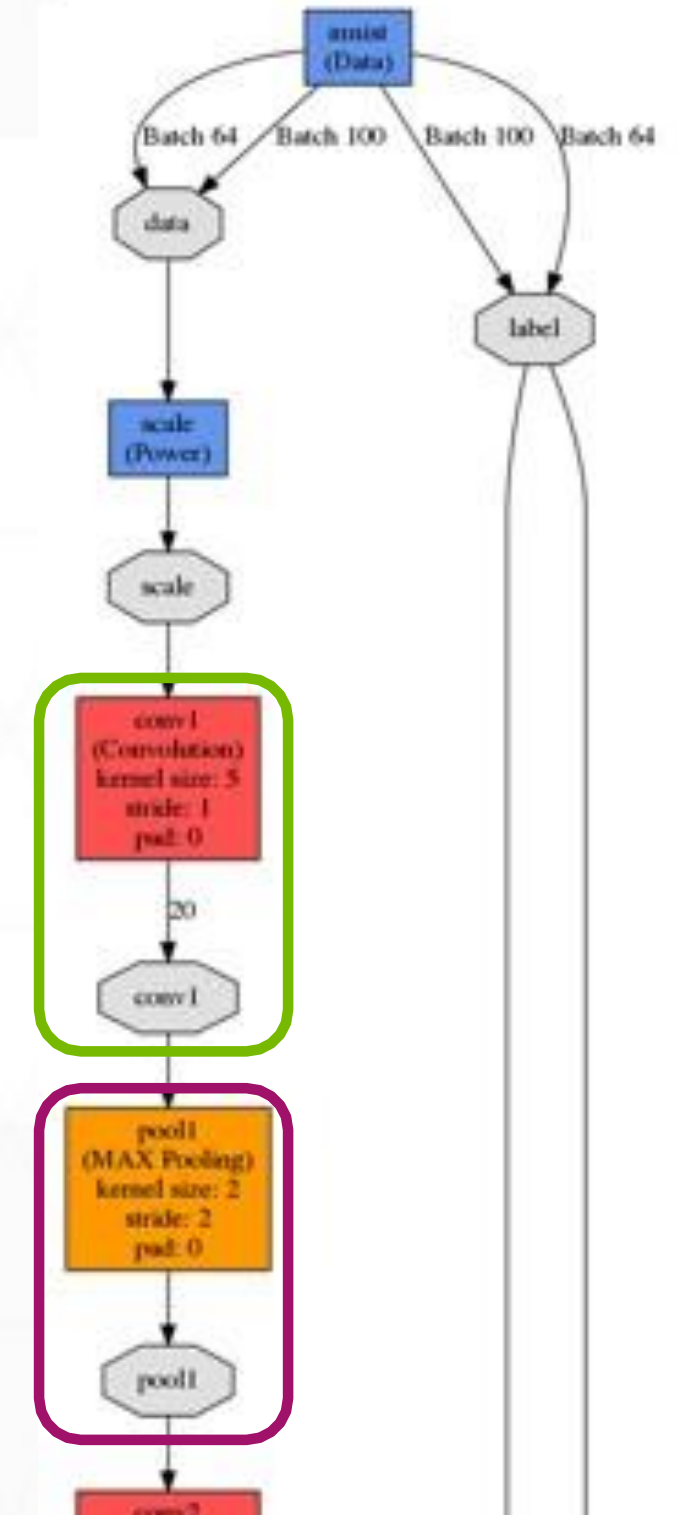
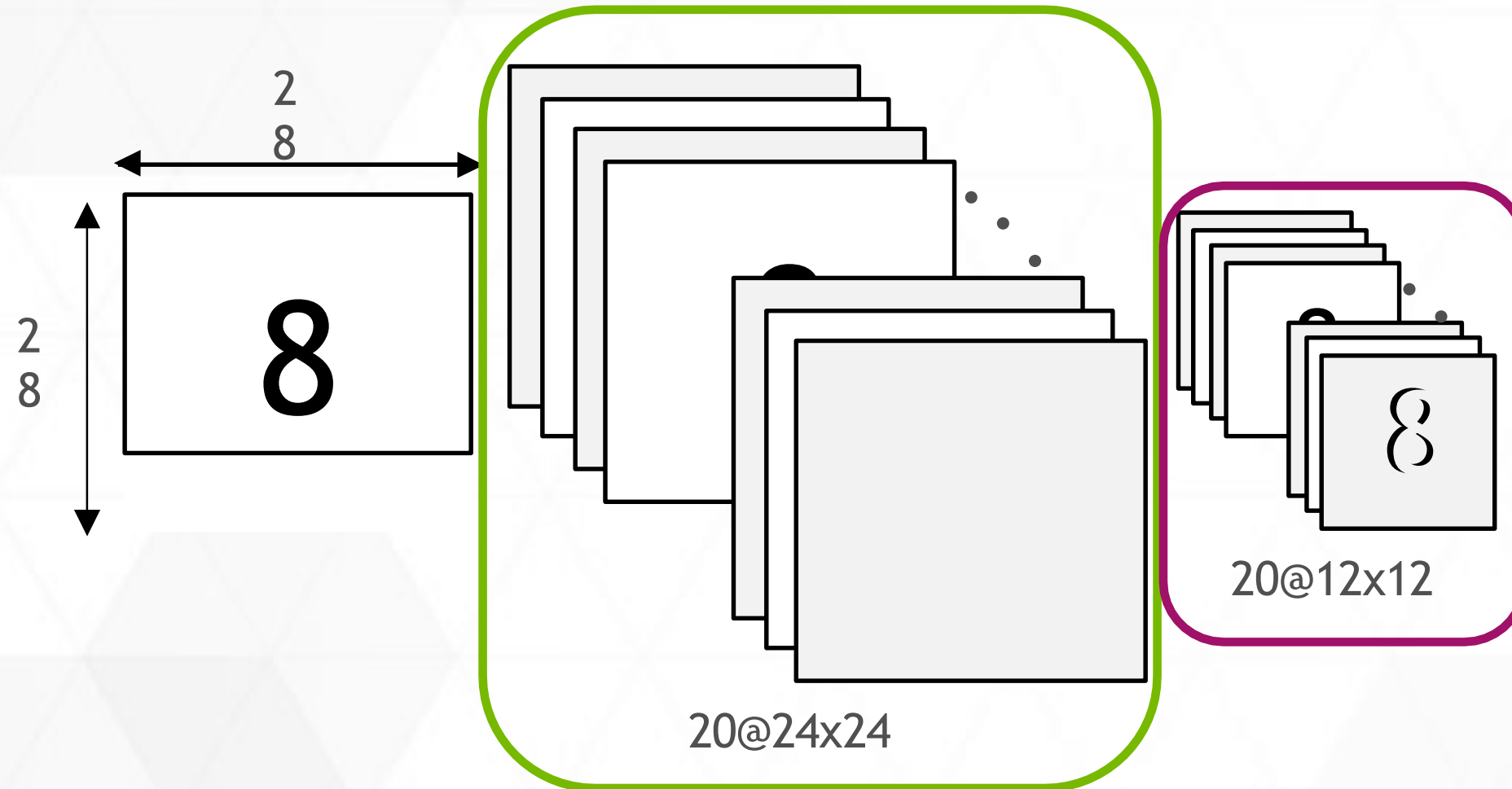


# NVIDIA DIGITS

## Train a Network-Advance Options



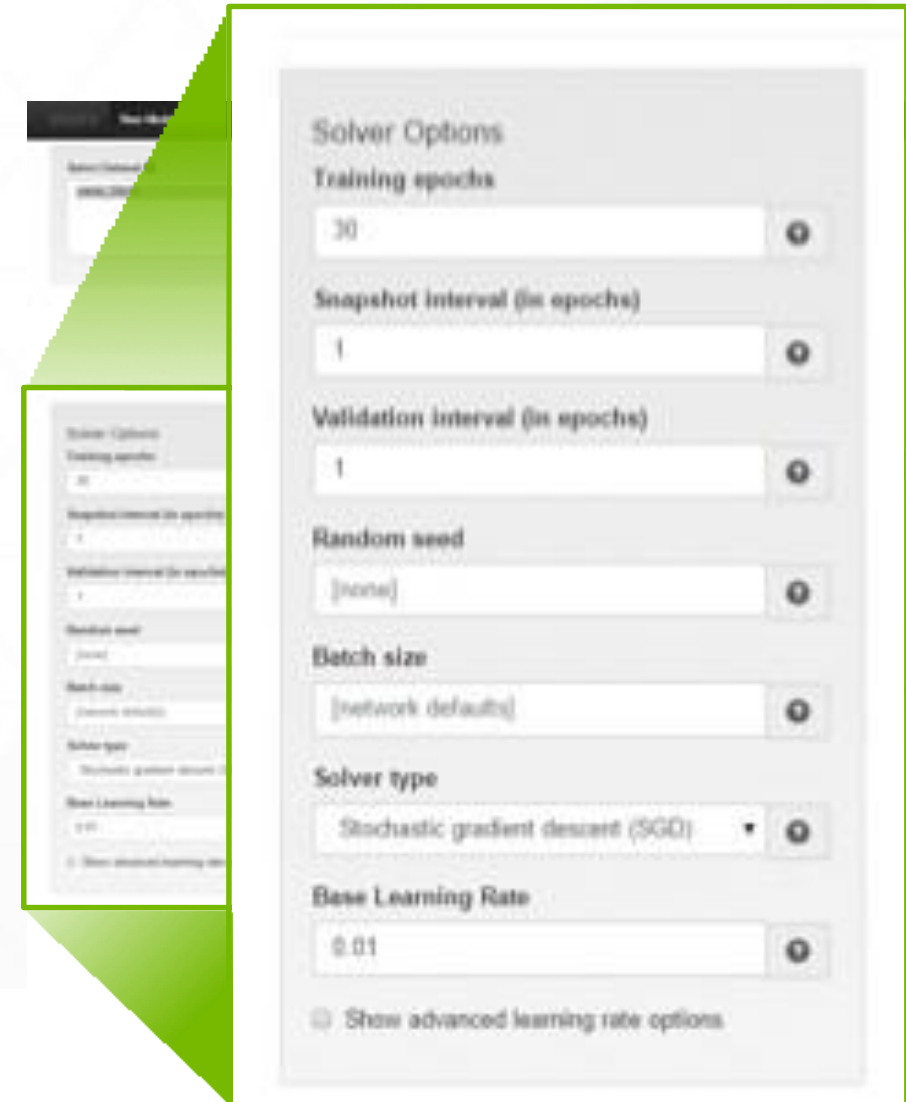
# NETWORK PARAMETERS



# 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 'Advance Options' panel. The panel is titled 'Solver Options' and 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, there are input fields for 'Training epochs', 'Validation interval (in epochs)', 'Batch size', 'Number of GPUs', and 'Data directory'. On the right, there are fields for 'Data Transformations' and 'Model Name'. In the center, there is a table showing the hardware configuration:

Device	Model	Model Image Size
GPU0	ImageNet-1000	224x224
GPU1	ImageNet-1000	224x224
GPU2	ImageNet-1000	224x224

At the bottom, there is a 'Model Name' field and a 'Create' button.

Multiple GPU system

The screenshot shows the 'New Image Classification Model' interface for a multiple GPU system. It includes the same input fields as the single GPU system, but with additional options for 'Data Transformations' and 'Model Name'. The hardware configuration table is also present. A green box highlights the 'Number of GPUs' field, which is set to 1. Below this, there is a section titled 'Select which GPUs you would like to use' with a list of available GPUs.

Select the number of GPUs you want to use.

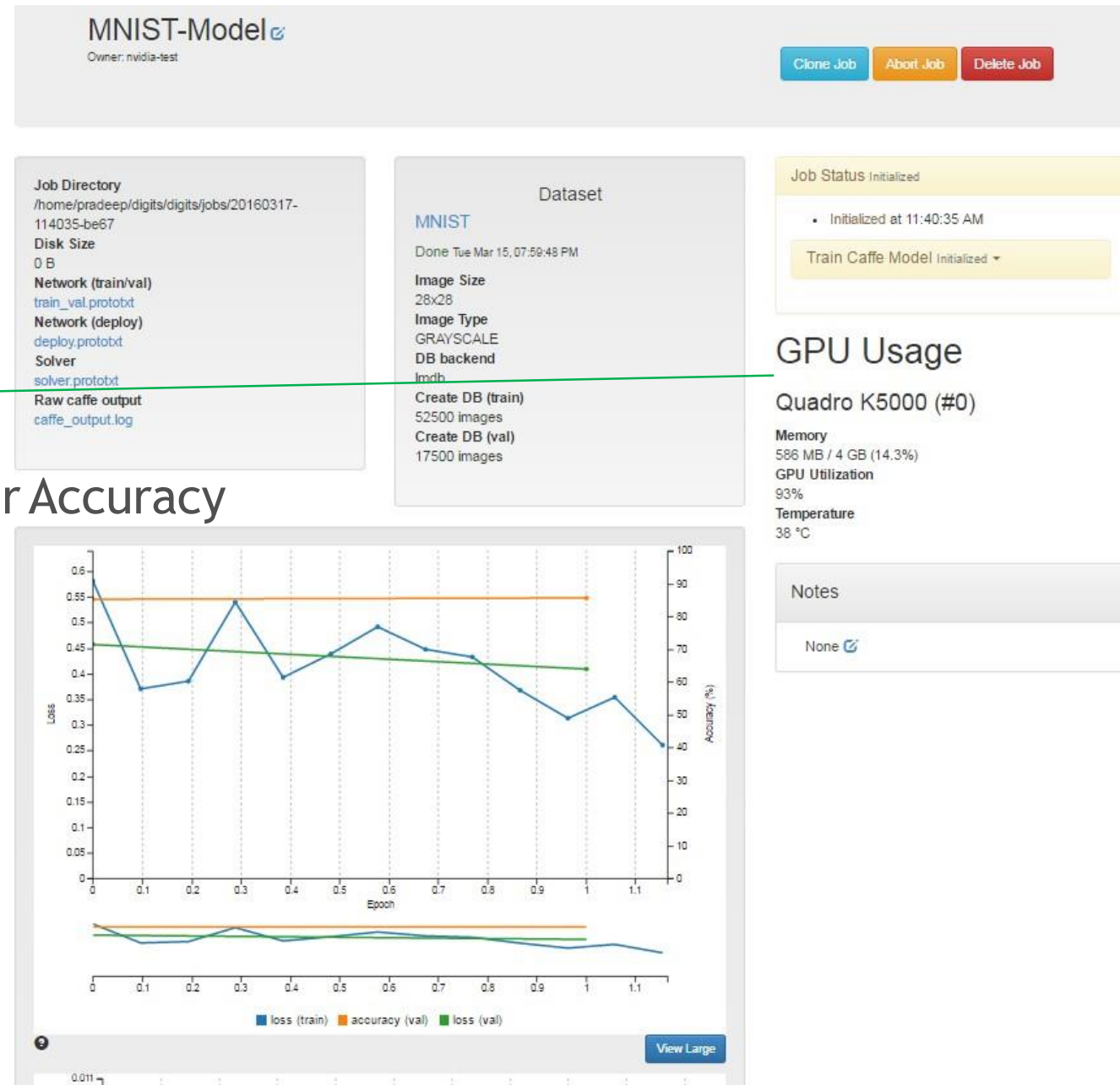
# NVIDIA DIGITS

## Training Results

Monitor GPU, Memory usage and Temperature

Monitor Accuracy

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



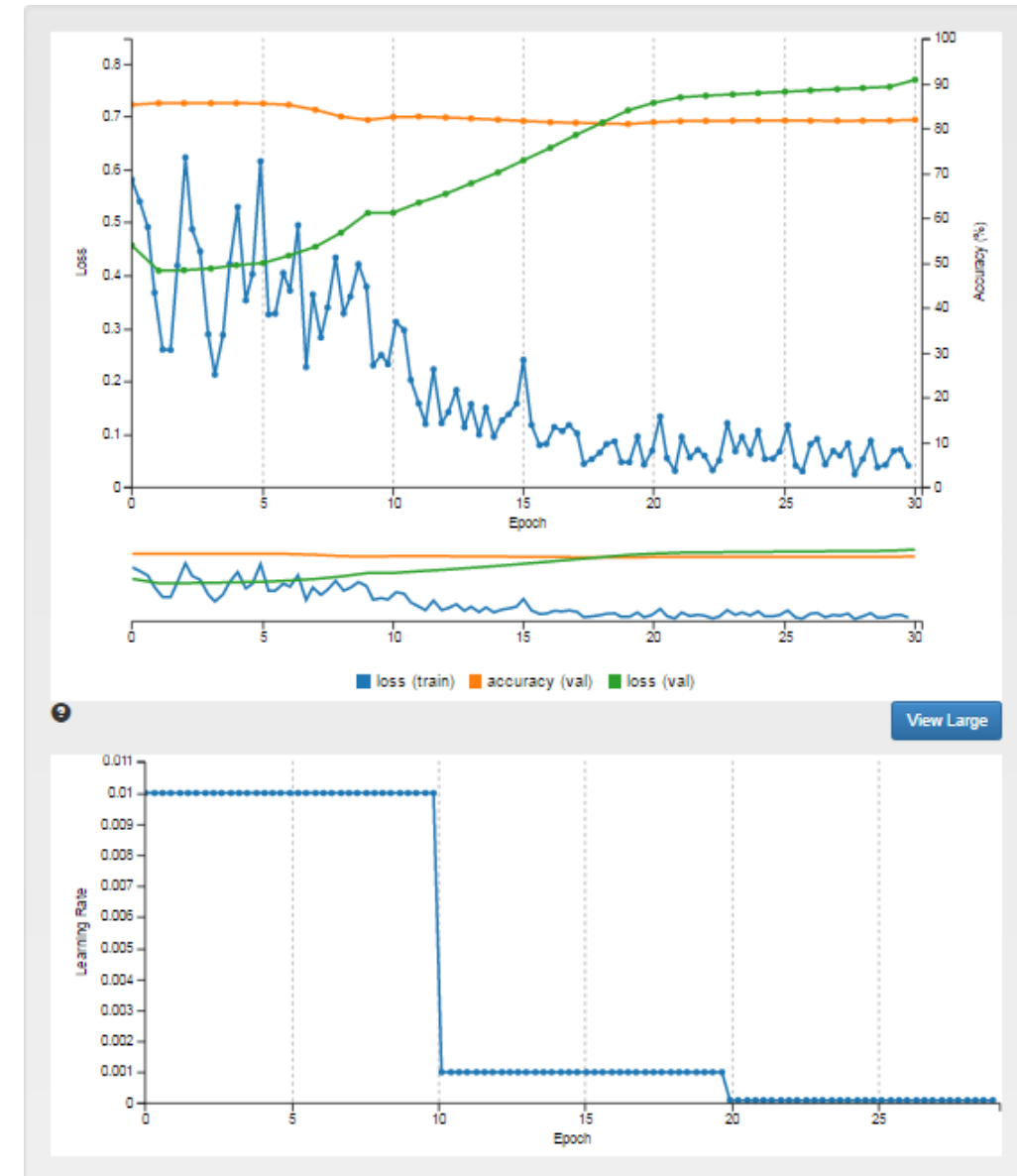


# NVIDIA DIGITS

## 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

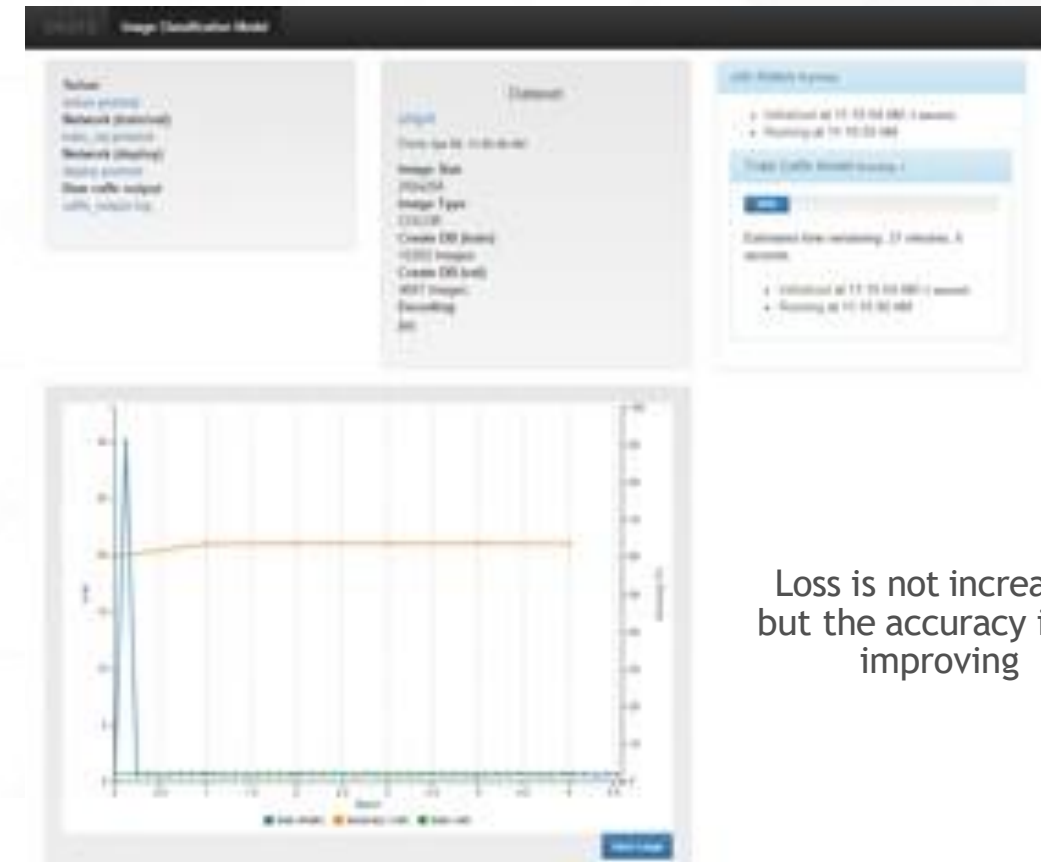
How can I use DIGITS to tell me this is happening?

Overfitting



Loss continues to decrease with training data but increases with validation

Underfitting



Loss is not increasing but the accuracy is not improving

Validation data helps you identify when/if this occurs!

# DIGITS Features at a Glance

# 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](https://nvidia.qwiklab.com)
  2. Go to “Getting Started with DIGITS” lab at [bit.ly/dlnvlab2](https://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](http://developer.nvidia.com/deep-learning-courses)