(https://www.nvidia.com/dli/)

Image Classification with DIGITS

An introduction to Deep Learning

In this lab, you'll learn to **train** a **neural network** using clean **labeled data**. We'll introduce deep learning through the task of **supervised image classification,** where, given many images and their labels, you'll build a tool that can *predict* labels of *new* images.

The intent is to build the skills to start experimenting with deep learning. You'll examine:

- What it means to *train* vs. to *program*
- The role of data in artificial intelligence
- How to load data for training a neural network
- The role of a *network* in deep learning
- How to train a model with data

At the end of this lab, you'll have a trained neural network that can successfully classify images to solve a classic deep learning challenge:

How can we digitize handwriting?

Training vs. programming

The fundamental difference between artificial intelligence (AI) and traditional programing is that AI *learns* while traditional algorithms are *programmed.* Let's examine the difference through an example:

Imagine you were asked to give a robot instructions to make a sandwich using traditional computer programming, instruction by instruction. How might you start?

Maybe something like:
`Add two pieces of bread to an empty plate.`

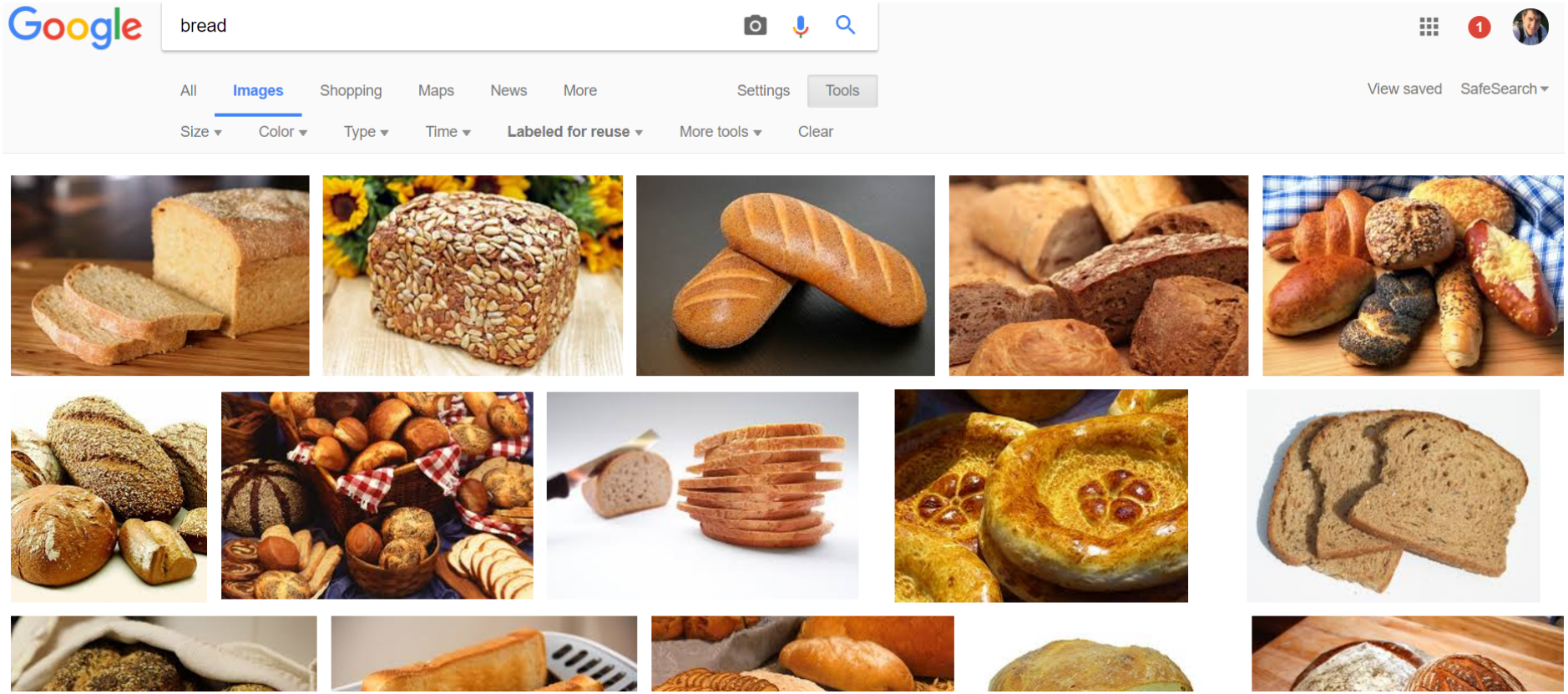
Run the code below to see what the robot might do with your first instruction. To run code, click on **In [ ]** and press **Shift + Enter**.

In [ ]:
```
readyPlate = 2*bread + emptyPlate
```

You likely got a "NameError." How would you then work to define "bread" to your robot? Not to mention other ingredients and how to combine them?

Computers only "see" images as the amount of red, blue, and green at each pixel. Everything else we want them to know, we would have had to describe in terms of pixels.

Artificial intelligence takes a different approach. Instead of providing instructions, we provide examples. Above, we could *show* our robot thousands of labeled images of bread and thousands of labeled images of other objects and ask our robot to learn the difference. Our robot could then build its own program to identify new groups of pixels (images) as bread.



Instead of instructions, we now need data and computers designed to learn.

The Big Bang in Machine Learning

- **Data:** In the era of big data, we create ~2.5 quintillion bytes of data per day. Free datasets are available from places like Kaggle.com (https://www.kaggle.com/datasets) and UCI (https://archive.ics.uci.edu/ml/datasets.html). Crowdsourced datasets are built through creative approaches - e.g. Facebook asking users to "tag" friends in their photos to create labeled facial recognition datasets. More complex datasets are generated manually by experts - e.g. asking radiologists to label specific parts of the heart.
- **Computers designed to learn - software:** Artificial neural networks are inspired by the human brain. Like their biological counterparts, they are structured to understand (and represent) complex concepts, but their biggest strength is their ability to learn from data and feedback. The "deep" in deep learning refers to many layers of artificial neurons, each of which contribute to the network's performance.
- **Computers designed to learn - hardware:** The computing that powers deep learning is intensive, but not complex. Processing huge datasets through deep networks is made possible by parallel processing, a task tailor made for the GPU.

So how do we expose artificial neural networks to data?

By the end of this lab, you'll know how to load data into a deep neural network to create a trained model that is capable of solving problems with what it learned, not what a programmer told it to do.

Training a network to recognize handwritten digits

Since a computer "sees" images as collections of pixel values, it can't do anything with visual data unless it learns what those pixels represent.

What if we could easily convert handwritten digits to the digital numbers they represent?

1. We could help the post office sort piles of mail by post code. This is the problem that motivated Yann LeCun (http://yann.lecun.com/). He and his team put together the dataset and neural network that we'll use today and painstakingly pioneered much of what we know now about deep learning.
2. We could help teachers by automatically grading math homework. This the problem that motivated the team at answer.ky (http://answer.ky/), who used Yann's work to easily solve a real world problem using a workflow like what we'll work through now.
3. We could solve countless other challenges. What will you build?

We're going to train a deep neural network to recognize handwritten digits 0-9. This challenge is called "image classification," where our network will be able to decide which image belongs to which class, or group.

For example:
The following image should belong to the class '4':

whereas this next image should belong to the class '2':

It's important to note that this workflow is common to most image classification tasks, and is a great entry point to learning how to solve problems with Deep Learning.

Let's start.

Load and organize data

Let's start by bringing our data, in this case, thousands of images, into our learning environment. We're going to use a tool called DIGITS, where we can visualize and manage our data.

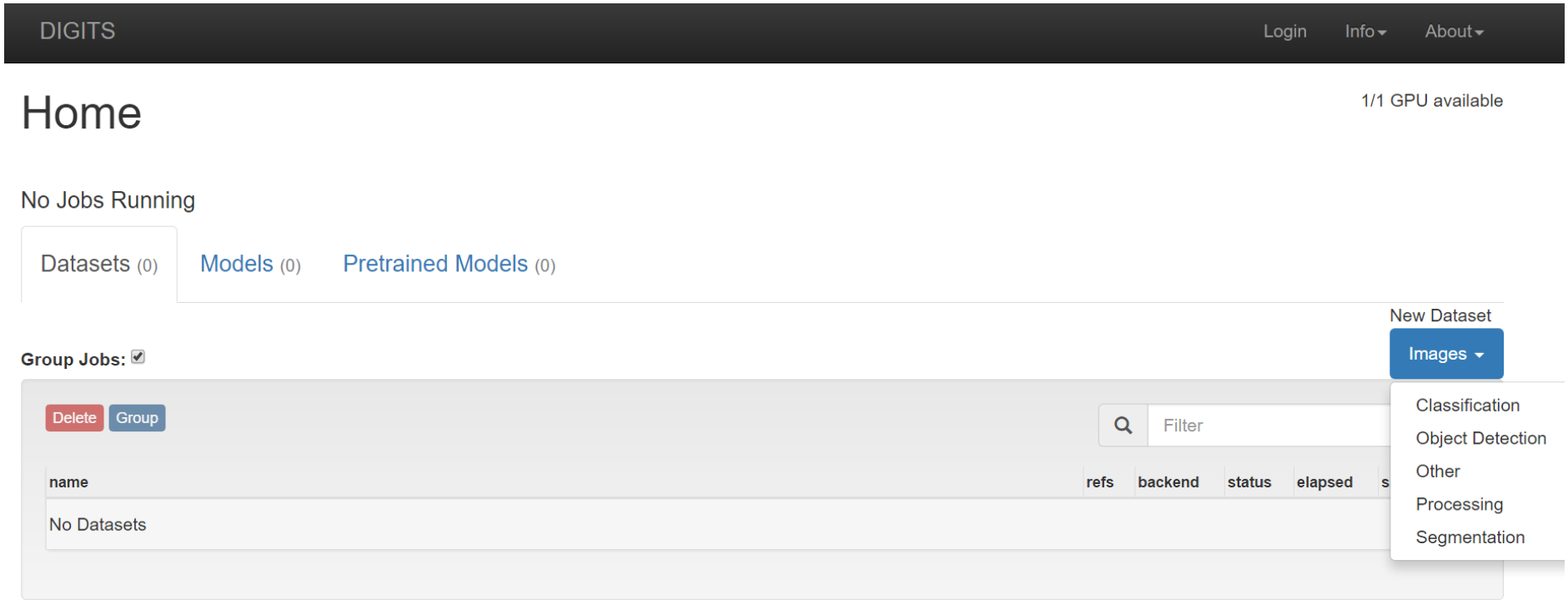
First, open DIGITS in a new tab using the following link.

Open DIGITS (/digits/).

Loading our first dataset

When you start DIGITS, you will be taken to the home screen where you can create new datasets or new models.

Begin by selecting the **Datasets** tab on the left.



Since we want our network to tell us which "class" each image belongs to, we ask DIGITS to prepare a "classification" image dataset by selecting "Classification" from the "Images" menu on the right.

At this point you may need to enter a username. If requested, just enter any name in lower-case.

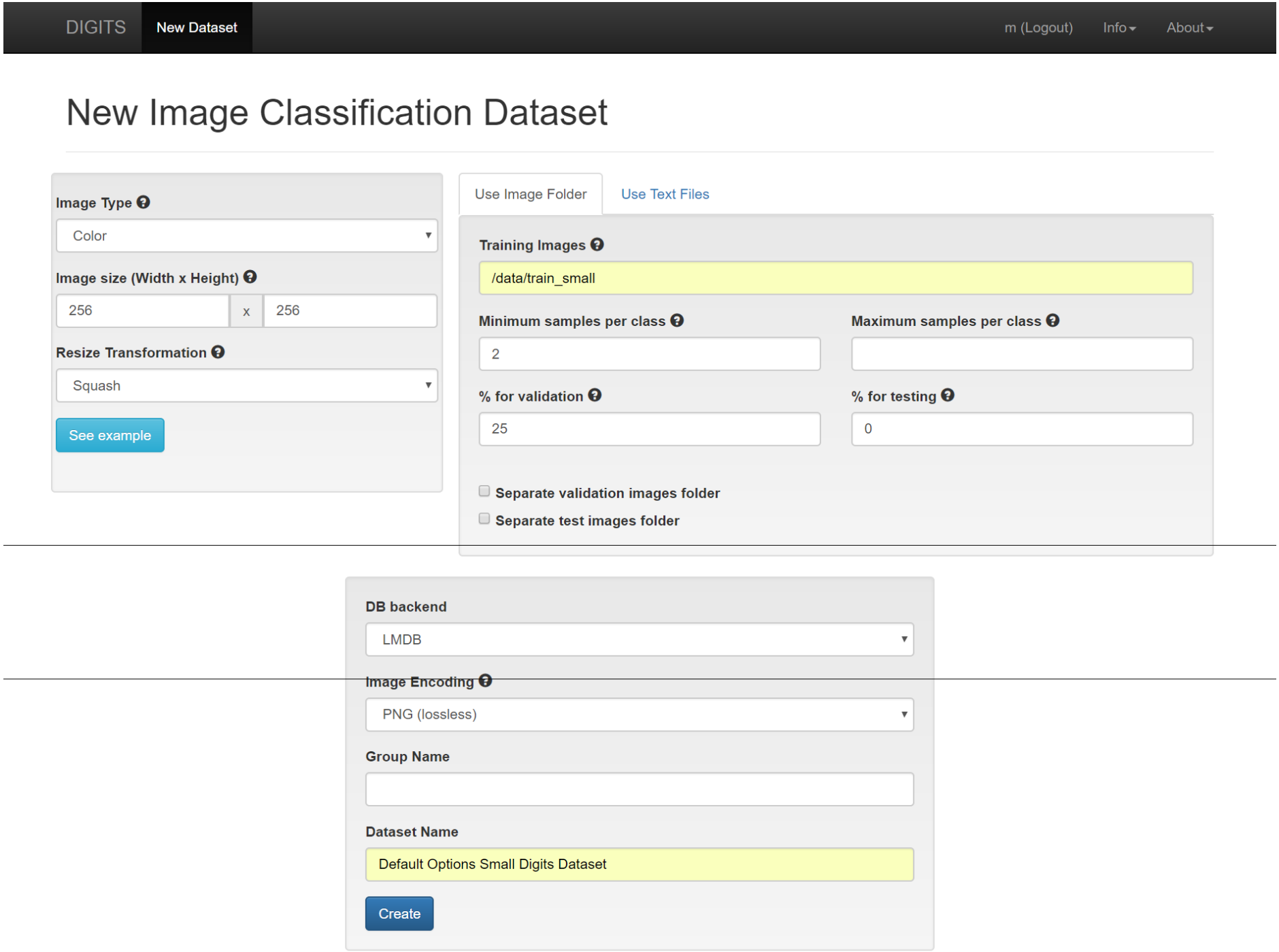
Loading and organizing our data

You'll see that you've got a lot of options around *how* to load a dataset. For this first runthrough, we're going to simplify and only fill out two fields.

1. Copy and paste the following filepath into the field "Training Images": `/data/train_small`
2. Name the dataset so that you can find it. We've chosen: `Default Options Small Digits Dataset`

Don't see "Training Images?" Click "DIGITS" on the top left and select "Datasets" before selecting "Images" and "Classification."

Note that we've already downloaded the dataset to the computer where DIGITS is running. You'll have a chance to explore it shortly and will learn methods for accessing data as you work through our labs.


DIGITS
New Dataset
m (Logout)
Info
About
New Image Classification Dataset
Image Type
Color
Image size (Width x Height)
256
x
256
Resize Transformation
Squash
See example
Use Image Folder
Use Text Files
Training Images
/data/train_small
Minimum samples per class
2
Maximum samples per class
% for validation
25
% for testing
0
Separate validation images folder
Separate test images folder
DB backend
LMDB
Image Encoding
PNG (lossless)
Group Name
Dataset Name
Default Options Small Digits Dataset
Create

Then press "Create."

DIGITS is now creating your dataset from the folder. Inside the folder `train_small` there were 10 subfolders, one for each class (0, 1, 2, 3, ..., 9). All of the handwritten training images of '0's are in the '0' folder, '1's are in the '1' folder, etc.

Explore what our data looks like by selecting "Explore the db".

Your data

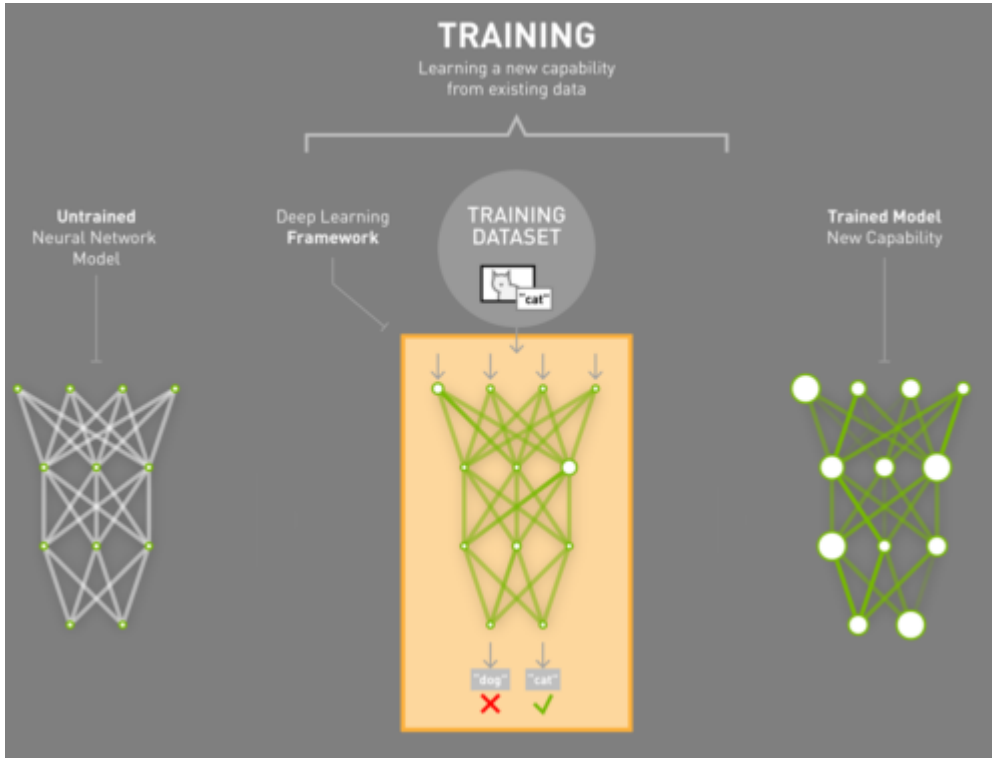
While there is an endless amount of analysis that we could do on the data, make sure you at least note the following:

1. This data is *labeled.* Each image in the dataset is paired with a **label** that informs the computer what number the image represents, 0-9. We're basically providing a question with its answer, or, as our network will see it, a desired output with each input. These are the "examples" that our network will learn from.
2. Each image is simply a digit on a plain background. Image classification is the task of identifying the *predominant* object in an image. For a first attempt, we're using images that only contain *one* object. We'll build skills to deal with messier data in subsequent labs.

This data comes from the MNIST (http://yann.lecun.com/exdb/mnist/) dataset which was created by Yann LeCun. It's largely considered the "Hello World," or introduction, to deep learning.

Learning from our data - Training a neural network

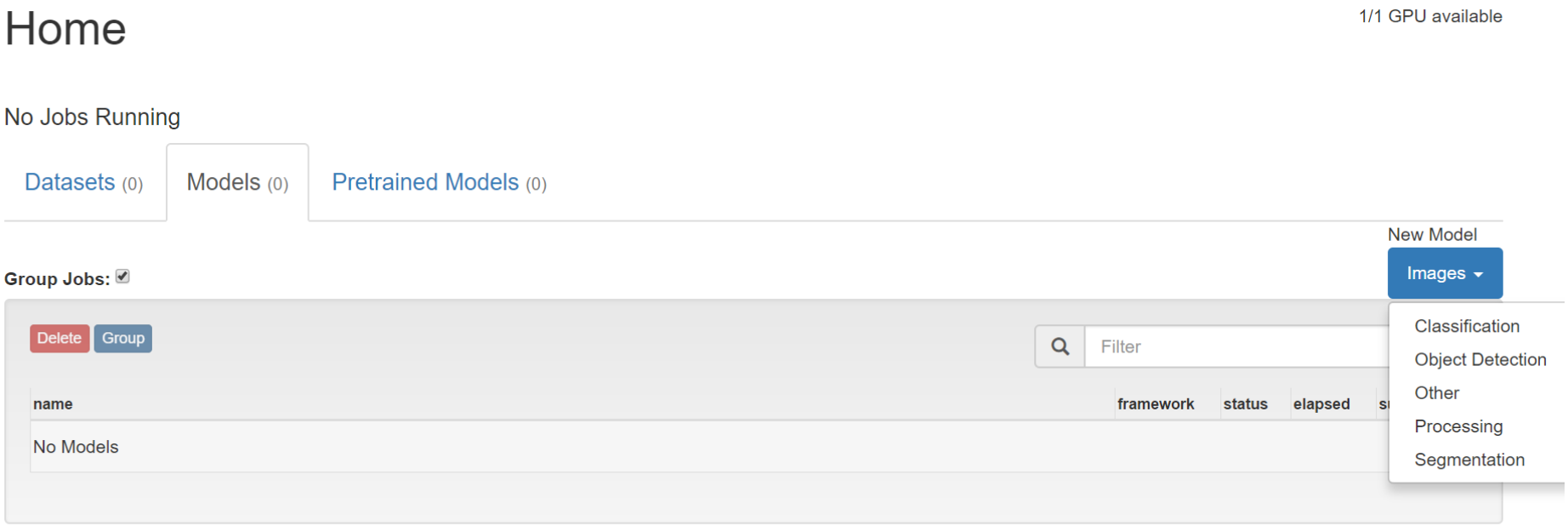
Next, we're going to use our data to *train* an artificial neural network. Like its biological inspiration, the human brain, artificial neural networks are learning machines. Also like the brain, these "networks" only become capable of solving problems with experience, in this case, interacting with data. Throughout this lab, we'll refer to "networks" as untrained artificial neural networks and "models" as what networks become once they are trained (through exposure to data).



For image classification (and some other tasks), DIGITS comes pre-loaded with award-winning networks. As we take on different challenges in subsequent labs, we'll learn more about selecting networks and even building our own. However, to start, weighing the merits of different networks would be like arguing about the performance of different cars before driving for the first time. Building a network from scratch would be like building your own car. Let's drive first. We'll get there.

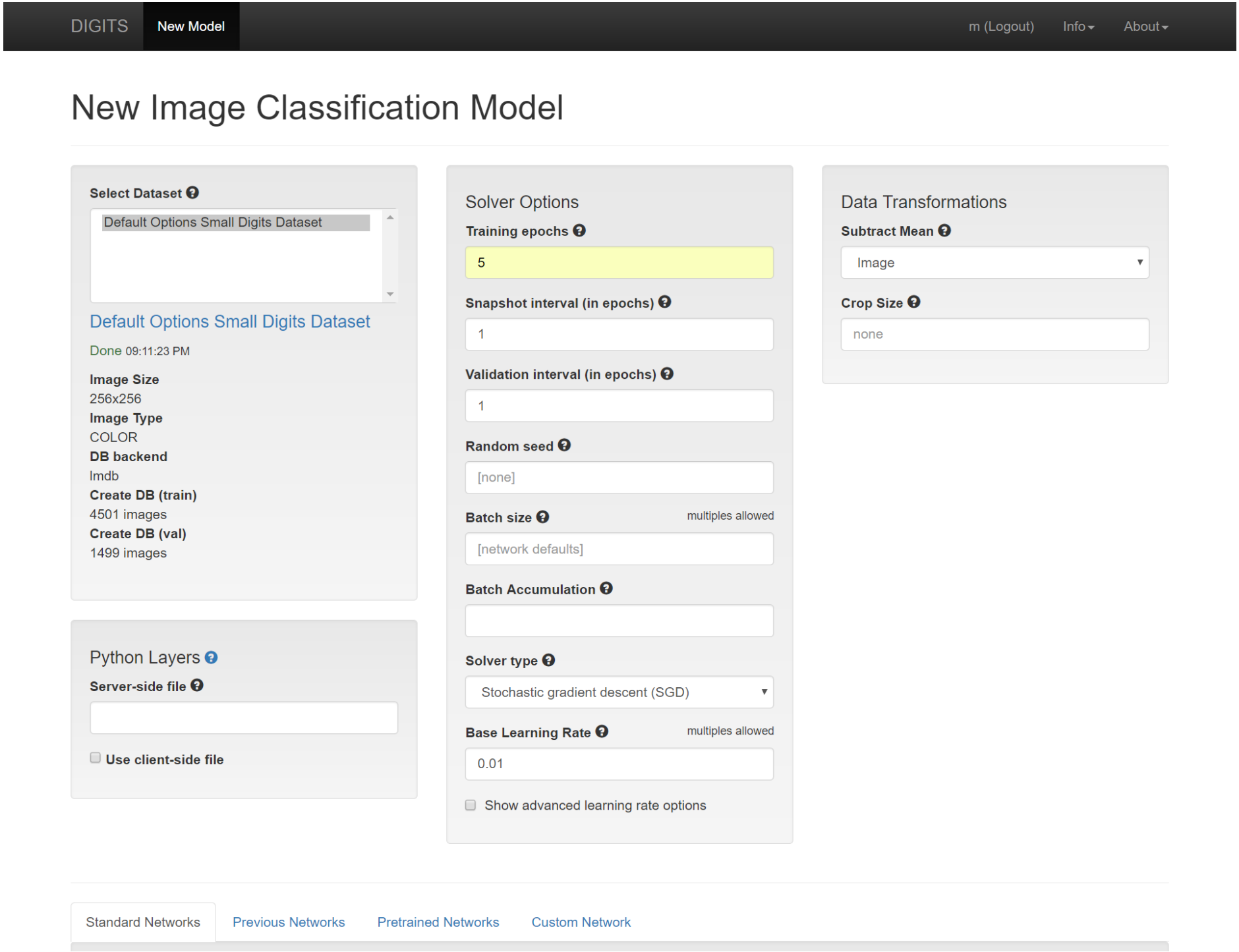
Go to the tab where DIGITS is still open and return to the main screen by clicking "DIGITS" on the top left of the screen.

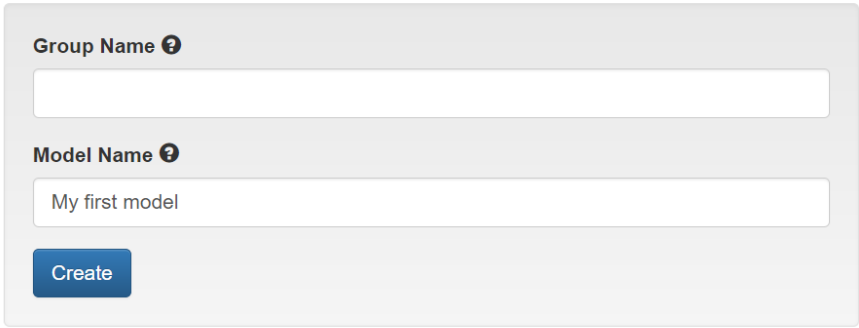
Creating a new model in DIGITS is a lot like creating a new dataset. From the home screen, the "Models" tab will be pre-selected. Click "Images" under "New Model" and select "Classification", as we're creating an image classification model to match our image classification dataset and image classification task.



Again, for this first round of training let's keep it simple. The following are the fewest settings you could possibly set to successfully train a network.

1. We need to choose the dataset we just created. Select our **Default Options Small Digits Dataset** dataset.
2. We need to tell the network how long we want it to train. An **epoch** is one trip through the entire training dataset. Set the number of **Training Epochs** to 5 to give our network enough time to learn something, but not take all day. This is a great setting to experiment with.
3. We need to define which network will *learn* from our data. Since we stuck with default settings in creating our dataset, our database is full of 256x256 color images. Select the network **AlexNet**, if only because it expects 256x256 color images.
4. We need to name the model, as hopefully we'll do a lot of these. We chose **My first model.**


DIGITS
New Model
m (Logout)
Info
About
New Image Classification Model
Select Dataset
Default Options Small Digits Dataset
Default Options Small Digits Dataset
Done 09:11:23 PM
Image Size
256x256
Image Type
COLOR
DB backend
lmdb
Create DB (train)
4501 images
Create DB (val)
1499 images
Python Layers
Server-side file
Use client-side file
Solver Options
Training epochs
5
Snapshot interval (in epochs)
1
Validation interval (in epochs)
1
Random seed
[none]
Batch size
multiples allowed
[network defaults]
Batch Accumulation
Solver type
Stochastic gradient descent (SGD)
Base Learning Rate
multiples allowed
0.01
Show advanced learning rate options
Data Transformations
Subtract Mean
Image
Crop Size
none
Standard Networks
Previous Networks
Pretrained Networks
Custom Network

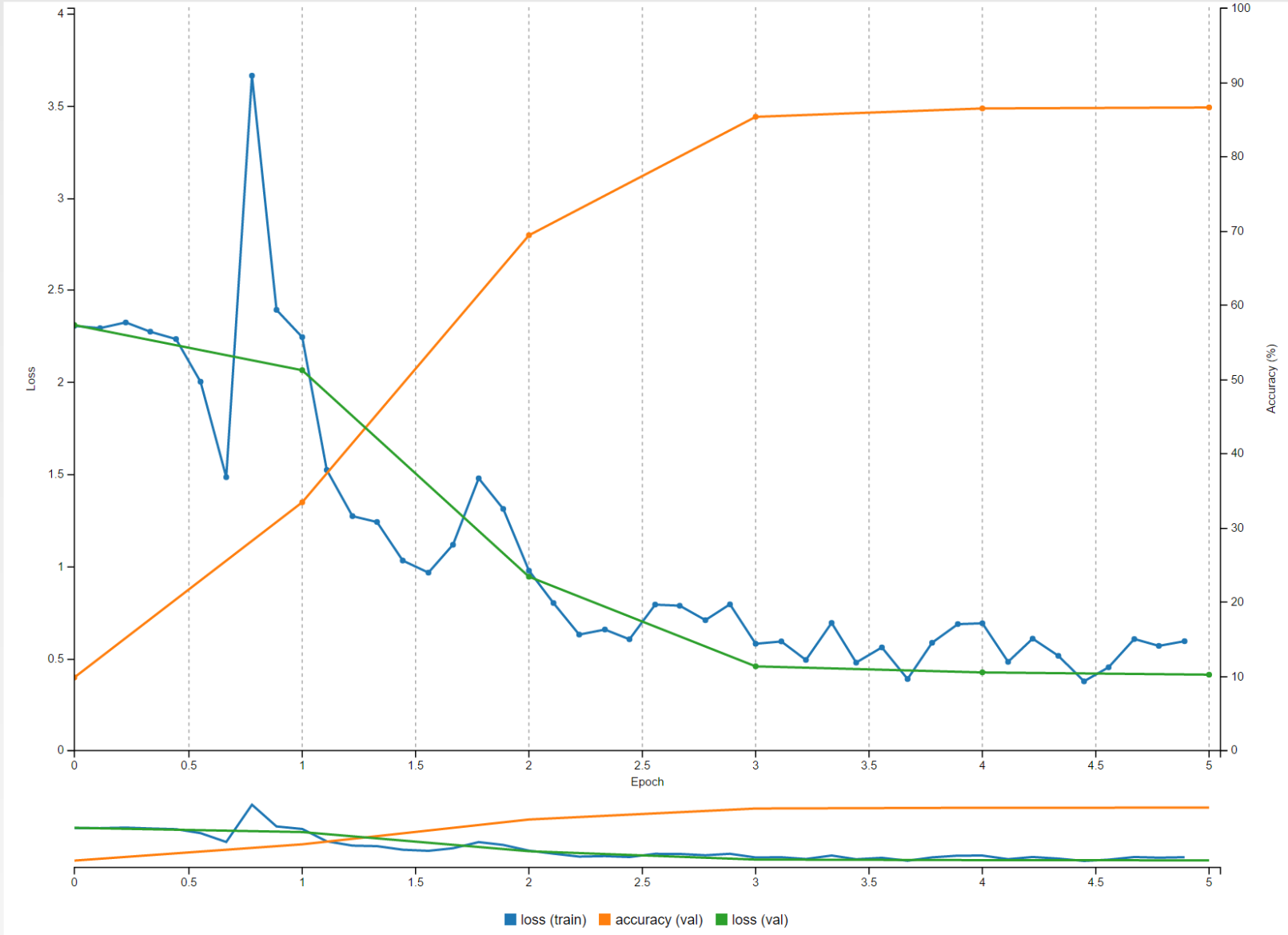


When you have set all of these options, press the Create button.

You are now training your model! For this configuration, the model training should complete in less than 5 minutes. You can either watch it train, continue reading, or grab a cup of coffee.

When done, the Job Status on the right will say "Done", and your training graph should look something like:

DIGITS Image Classification Model m (Logout) Info About

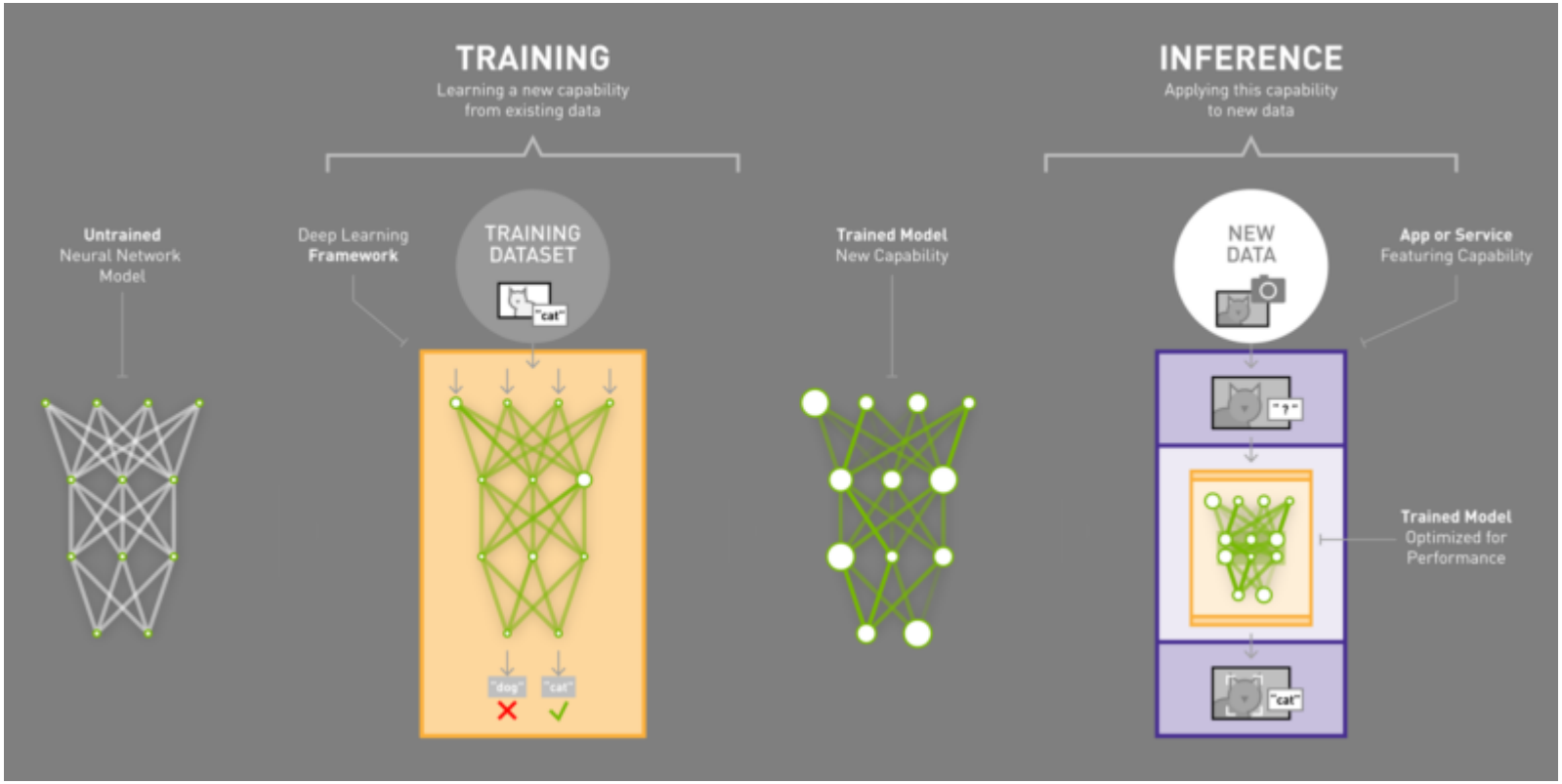


We'll dig into this graph as a tool for improvement, but the bottom line is that after 5 minutes of training, we have built a model that can map images of handwritten digits to the number they represent with an accuracy of about 87%!

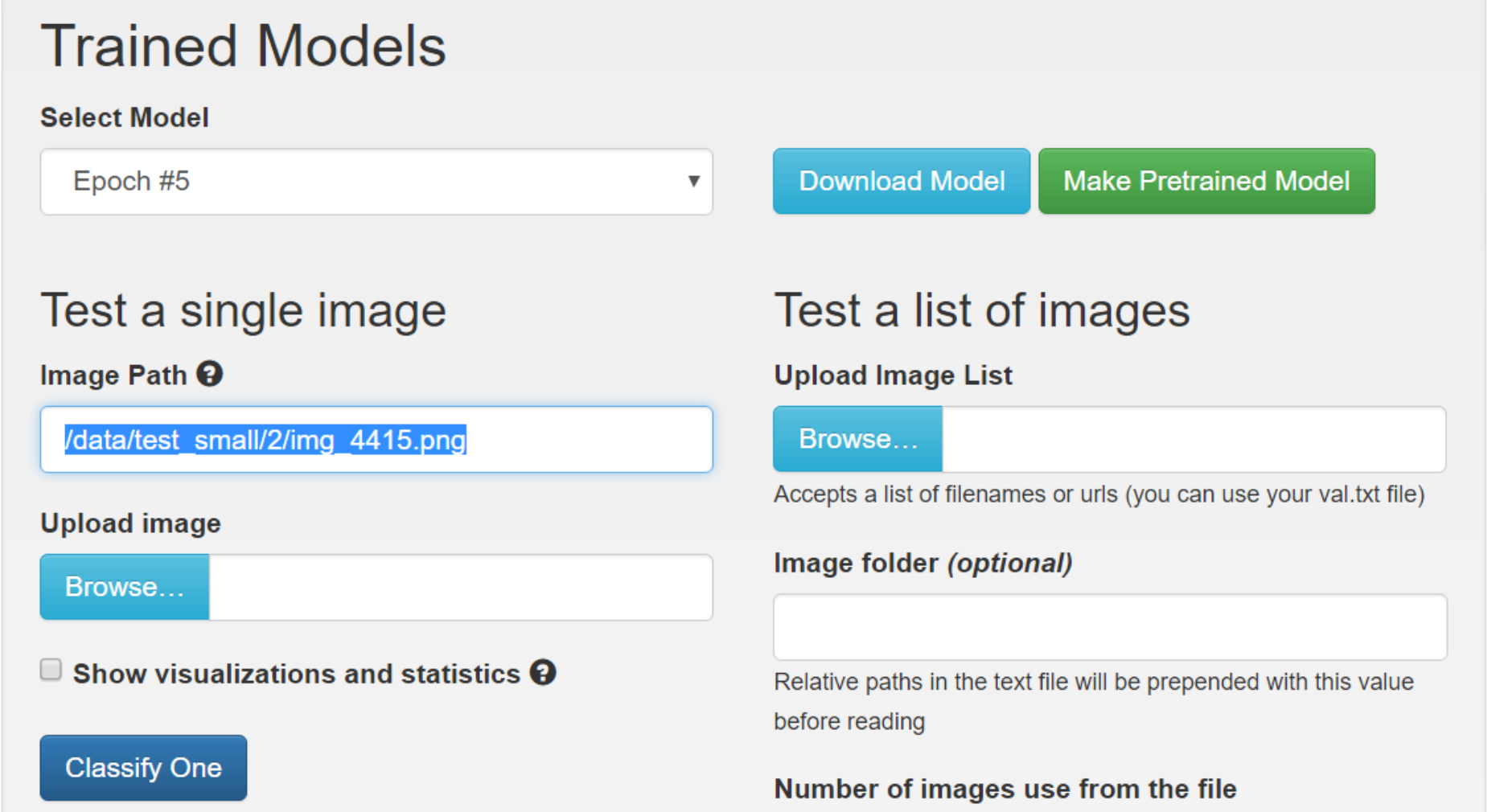
Let's test the ability of the model to identify **new** images.

Inference

Now that our neural network has learned something, *inference* is the process of making decisions based on what was learned. The power of our trained model is that it can now classify **unlabeled** images.



We'll use DIGITS to test our trained model. At the bottom of the model window, you can test a single image or a list of images. On the left, type in the path `/data/test_small/2/img_4415.png` in the Image Path text box. Select the **Classify One** button. After a few seconds, a new window is displayed with the image and information about its attempt to classify the image.



It worked! (Try again if it didn't). You took an untrained neural network, exposed it to thousands of *labeled* images, and it now has the ability to accurately predict the *class* of *unlabeled* images. Congratulations!

Note that that same workflow would work with almost any image classification task. You could train AlexNet to classify images of dogs from images of cats, images of you from images of me, etc. If you have extra time at the end of this lab, theres another dataset with 101 different classes of images where you can experiment.

While you have been successful with this introductory task, there is a lot more to learn.

In the next notebook, you will work to define and improve performance.

Continue to the next notebook (Image%20Classification%20with%20DIGITS%20-%20Improving%20Performance.ipynb)

(https://www.nvidia.com/en-us/deep-learning-ai/education/)