



DEEP
LEARNING
INSTITUTE

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

Creating a model that is effective with *new 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) (<https://www.kaggle.com/datasets>) and [UCI](https://archive.ics.uci.edu/ml/datasets.html) (<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.

Until this point we have been working with a *very small* dataset of 16 images. We really need 10s of thousands of labeled images and don't (yet) have that for Louie. Luckily, Kaggle has a dataset that we can start with consisting of 18750 labeled images of dogs and cats. Instead of teaching our network who Louie is, let's start by teaching it what a dog is.

By the end of this task, you will have a trained neural network that can correctly classify dogs and cats that were not part of the training dataset.

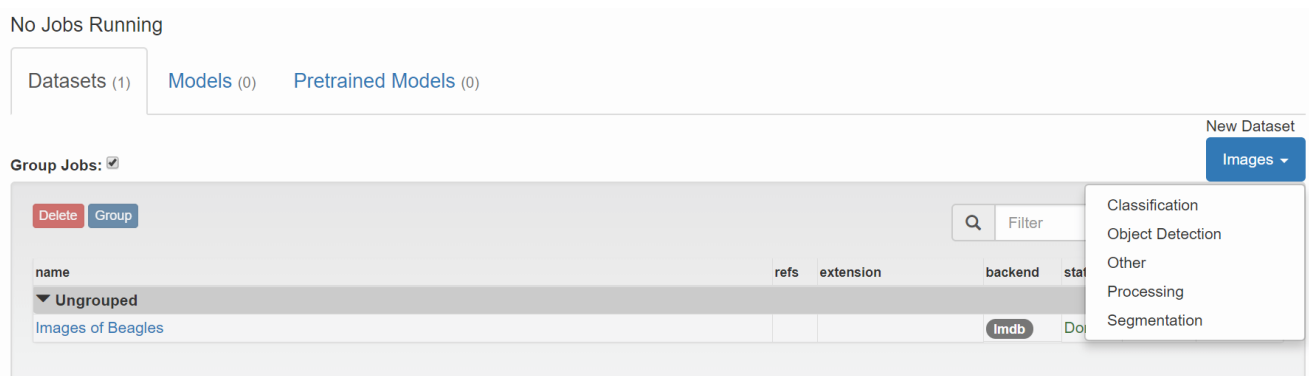
First, learn a bit more about DIGITS and the deep learning workflow by *creating* a training dataset from labeled data.

[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.



Ignore the "Images of Beagles" dataset from the last task (for now). Since we want our network to tell us which "class" each image belongs to, we ask DIGITS to prepare a **new** "classification" image dataset by selecting "Classification" from the blue "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

Point DIGITS to the folder where our data is stored by entering the following directory in the field called "Training Images":

```
/dli/data/dogscats/train
```

Use Image Folder

Use Text Files

Training Images ?

/dli/data/dogscats/train

Minimum samples per class ?

2

Maximum samples per class ?

% for validation ?

25

% for testing ?

10

☐ **Separate validation images folder**

☐ **Separate test images folder**

Note that this points to a folder in the training environment, not whatever computer you are working on. There are many ways to pass data back and forth between the two (highlighted in the ever improving [Resources: Next Steps for Independent Deep Learning](https://docs.google.com/document/d/1A8r1Shh0ssiRzrxNcraK7PJ_NUFay--EX1aBovpVMKU/edit) (https://docs.google.com/document/d/1A8r1Shh0ssiRzrxNcraK7PJ_NUFay--EX1aBovpVMKU/edit)).

To learn about how the data was structured prior to loading the dataset, hover your mouse over the ? next to "Training Images".

There are many ways to align input data with output labels. We created a folder with images with dogs called "dogs," and a folder of images with cats called "cats." We've put both of those folders into another folder called "dogscats". DIGITS recognized that there were two folders, so concluded that there were 2 classes. That's it. You will get to practice this with your own data shortly.

Note that we've allocated 25% in the field **% for validation**. You will read about this while the dataset loads.

To reduce the time required to prepare the dataset, change "Encoding" from "PNG(lossless)" to "None" and give the dataset a name.

When you select "Create", DIGITS will take about 3 minutes to prepare your data for training. While it does, read the section below to learn about what is happening.

DB backend

LMDB

Image Encoding ?

None

Group Name

Dataset Name

Dogs and Cats

Create

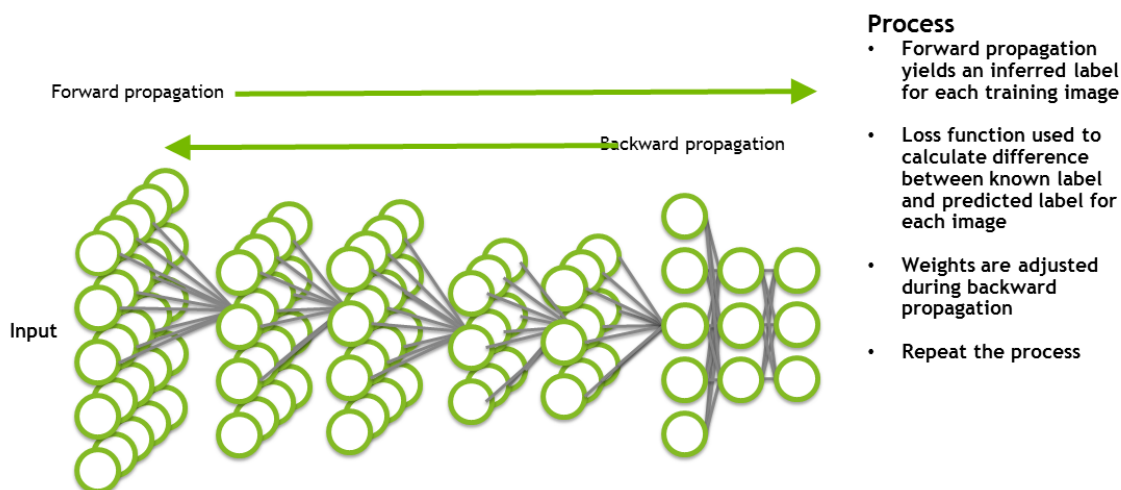
What is DIGITS doing?

You've asked DIGITS to take the images in the folder you specified and do two things:

- 1) Standardize them to the same size to match what the network you are training expects. We'll be training AlexNet again which was designed to take an input of 256X256 color images. Expect more on this in the next task.
- 2) Split them into two datasets, where 75% of each class is used for training and 25% is set aside for validation.

The training dataset will be used in the way we saw when we trained our Louie classifier; *forward propagate an image through the network*, generate an output, assess the *loss*, *backward propagate* the loss back through the network to update weights. (This is a core tenant of deep learning which will not be fully described here, as an understanding of the math *that the computer is doing* should not be a barrier to solving problems with deep learning, but for those who would like to learn more, save [this video from 3Blue1Brown](https://www.youtube.com/watch?v=llg3gGewQ5U) (<https://www.youtube.com/watch?v=llg3gGewQ5U>) to watch later.)

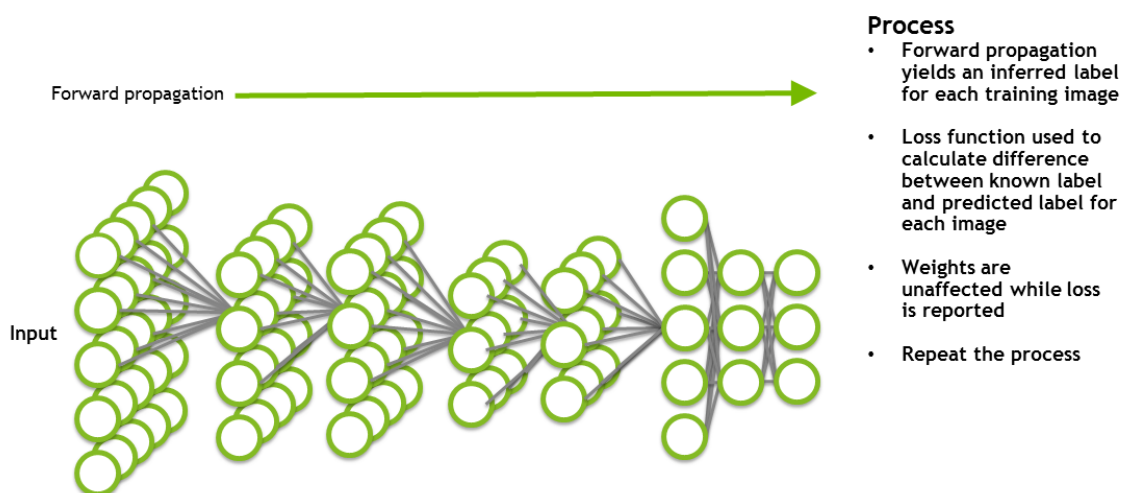
DEEP LEARNING APPROACH - TRAINING



Using validation data to assess performance

The validation dataset will be used to assess performance on *new data* using a technique that human learning can not. Validation data is fed through the network to generate an output, but the network *does not learn anything* from the data. Loss is reported but the model itself is unchanged! We can use the same validation dataset to assess our performance on new data over and over again while continuing to treat it like *new data*.

DEEP LEARNING APPROACH - VALIDATION



Once the datasets have loaded, feel free to learn a bit more about them by mousing over the histogram that is created and viewing the actual data by selecting "Explore the db."

You have now seen how to load data into DIGITS. These same steps are required no matter where you plan to train your network.

Training

Your next job is to train a neural network model the same way you did with the Louie dataset.

Start from DIGITS' home screen by selecting "DIGITS" in the top left corner. Next, create a new classification model by selecting **New Model** and then **Classification**. From here, the steps are identical to when you trained the smaller dataset given the following information:

Train the network "AlexNet" on the data you just loaded for 5 epochs.

If you can do this independently, know that you can:

Train an image classification network from pre-loaded data.

If you need a refresher, visit the notebook from the last task, [Train a Model](#).
([../task1/task/Train%20a%20Model.ipynb](#))

This time, examine the graph that is generated as the network trains. Work to interpret its results, which we'll discuss shortly. The key question: **how can you use the graph to assess the performance of your model as it trains?**

Note: If the training time is greater than 15 minutes, you likely asked for more than 5 epochs (iterations).

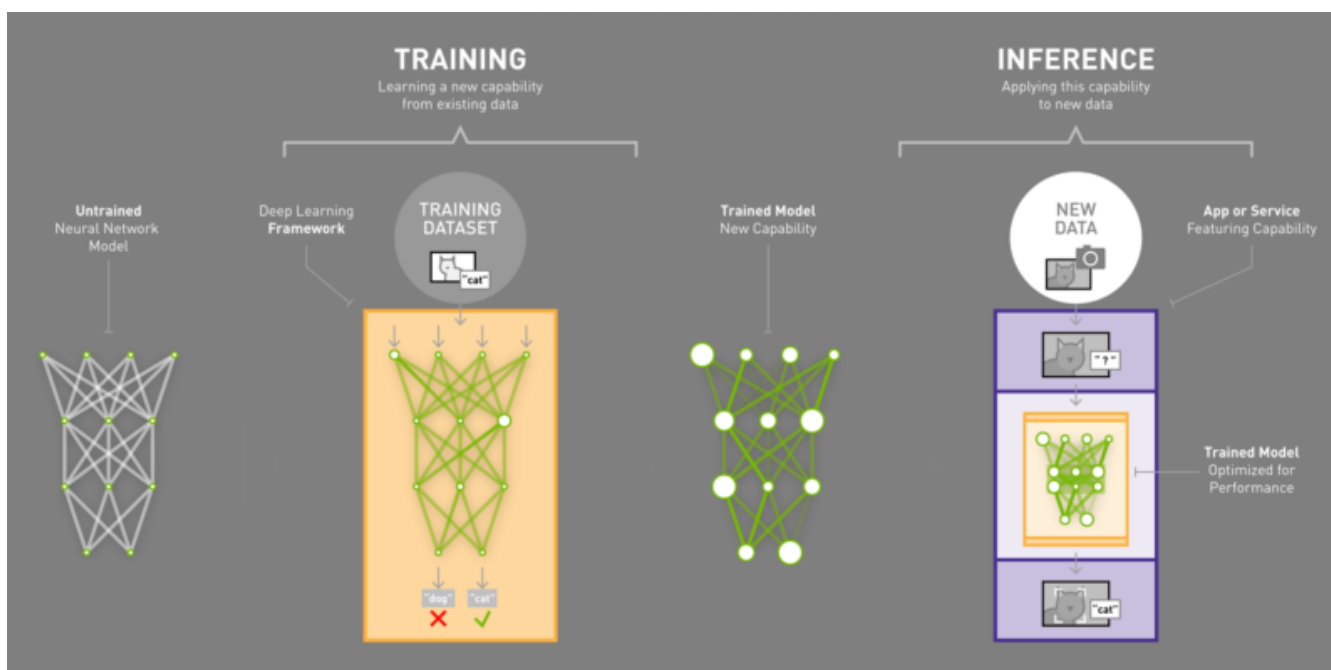
At the top of the screen, you can 'Abort Job' and then 'Clone Job' to check the instructions above.

As it trains, read the section below.

Inference

From your model job page. We will dig into how to interpret the content of this page when we return to the main course, but for now, let's test an image that our network has never seen before to see that we've been successful.

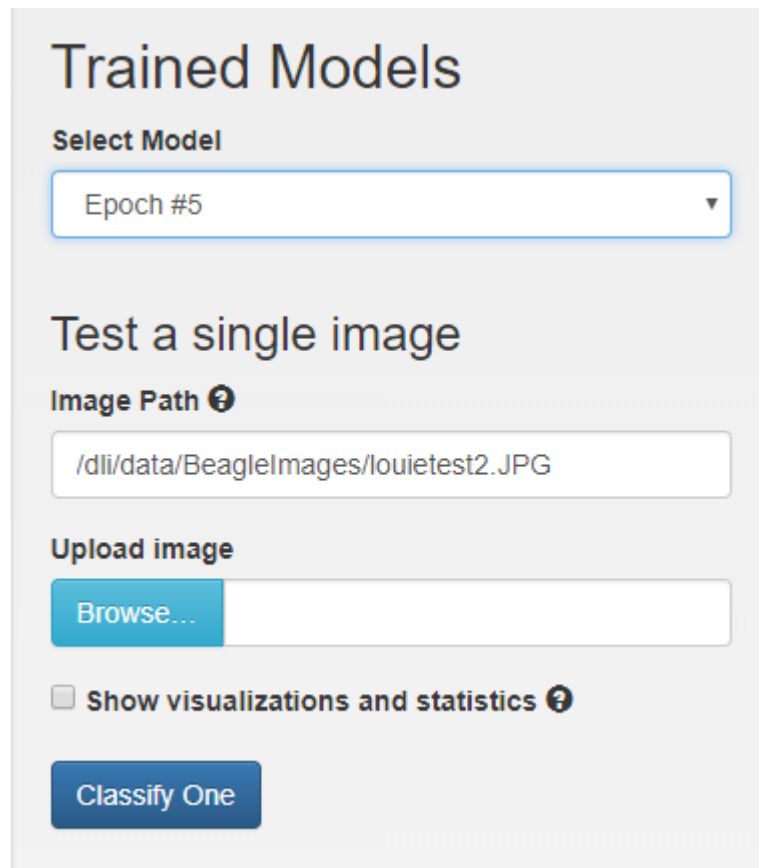
Inference is the process of making decisions based on what was learned. The power of our trained model is that it can now classify **unlabeled** images.



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

`/dli/data/BeagleImages/louietest2.JPG`

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.



You can interpret the results in two ways:

- 1) It worked! We took an untrained neural network, exposed it to thousands of *labeled* images, and it outputs a statistically significant confidence that a dog should be classified as a dog. Congratulations!
- 2) We're not there yet. Human learners would be 100% confident that that image contained a dog. Our model still has significant loss.

Both are correct, but for the moment, celebrate number 1 and we'll examine performance as soon as you close this notebook.

Testing other images

Feel free to check any images in our original dataset as none of them were in this training set.

You can see the images' filepaths by clicking the cell below and pressing **Shift and Enter** (a technique you will use often throughout the rest of the course). You can then test using the full filepath in the field "Image Path."

```
In [1]: !ls /dli/data/BeagleImages/Louie
```

```
louie1.JPG  louie3.JPG  louie5.JPG  louie7.JPG
louie2.JPG  louie4.JPG  louie6.JPG  louie8.JPG
```

Note that that same workflow would work with almost any image classification task. You could train AlexNet to classify images of images of you from images of me, images of handwritten digits from one another, images of healthy vs unhealthy patients, etc. We'll see this first-hand in our next task.

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

Let's return to the course to add some context to what we just did. When you close this window, you'll see an option to "Stop" this GPU instance. Select it after closing this window.



DEEP
LEARNING
INSTITUTE

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