

Lab Guide to Train and Deploy a Convolutional Neural Network (CNN) Model

Overview

Convolutional Neural Networks (CNN) are designed to train machine learning models that can perform image classification tasks such as object detection, face detection, and emotion detection.

In this lab, we will use GPU-based virtual machines to train a CNN model that can recognize handwritten numbers. The model will be trained with popular MNIST dataset.

Objectives

- Using NVIDIA DIGITS to train a CNN model
- Testing and validating the trained model
- Running the trained model locally for inferencing

Prerequisites

- [Docker for Windows](#) / [Docker for Mac](#)
- [Git Client](#)
- A valid account with [Paperspace](#)

Setup

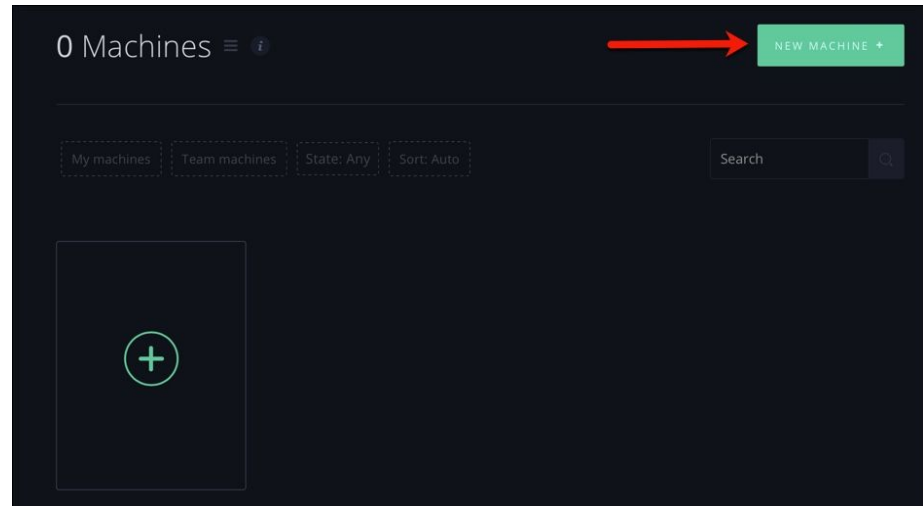
1. Clone the lab repo to access scripts and test data - `git clone https://github.com/janakiramm/cnn-lab`
2. Pull the Caffe CPU image for inference on local machine - `docker pull bvlc/caffe:cpu`

Step 1 - Launching Machine Learning in a Box VM in Paperspace

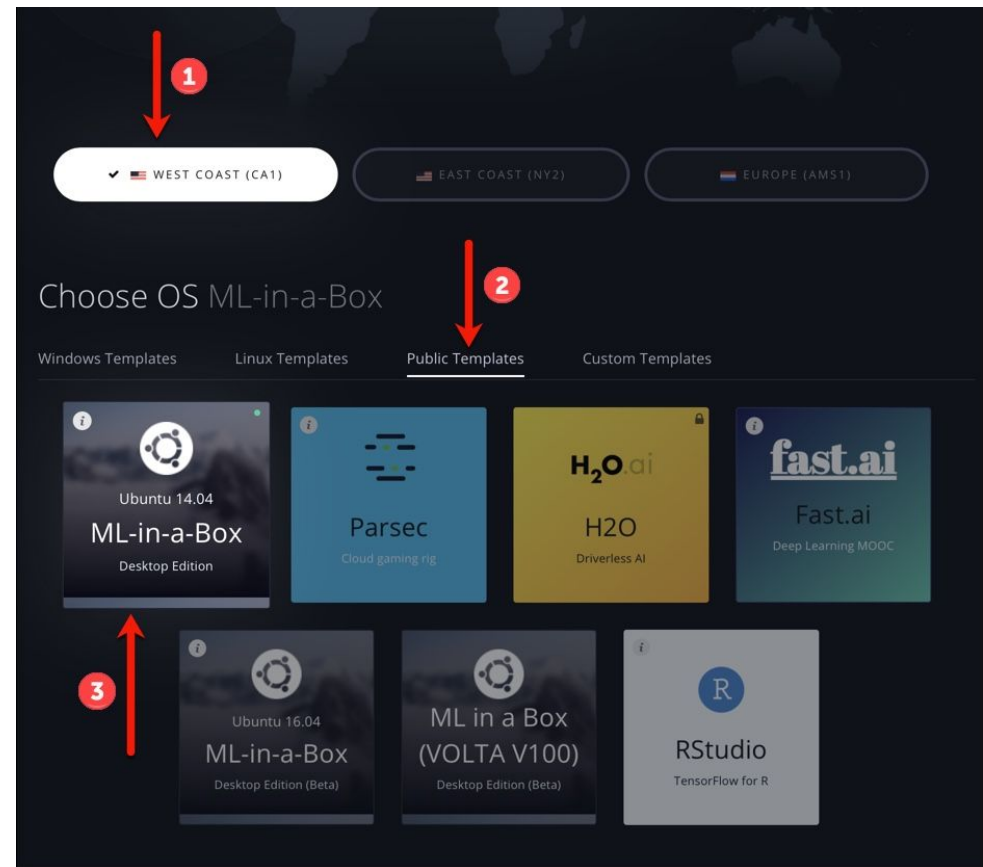
1. Sign-in to your Paperspace account
2. Click on **Machines** under Compute



1. Click on **New Machine** to create a new VM



1. Choose **West Coast** as the region
2. Select **Public Templates** to access MLiaB
3. Select **ML-in-a-Box** template
4. Scroll down to access the machine configuration settings.



1. Change the name of the machine with **<firstname-cnn>**
2. Uncheck **Auto Snapshot**. We don't need it for the lab.
3. Check **Public IP** to assign an IP address to the VM.

The screenshot displays the Interop ITX cloud console interface for configuring a new machine. At the top, there are tabs for 'Monthly' (selected) and 'Hourly'. Below this, four machine configurations are shown: GPU+, P4000 (selected with a green checkmark), P5000, and P6000. Each configuration lists its price per hour, monthly storage cost, and hardware specs (8 CPU, 30 GB RAM, 50 GB SSD). A note states: 'All hourly plans require a flat access fee which covers storage and maintenance costs. Learn more here'. The 'Choose Storage' section shows a dropdown set to '50 GB (\$5/month)'. The 'Machine Details' section shows a list with '1 machine' and a red arrow labeled '1' pointing to the 'jani-cnn' machine name field, with an 'ASSIGN MACHINE' button next to it. The 'Choose Network' section has a dropdown set to 'Default Network' and a link 'View/Configure networks'. The 'Options' section has two checkboxes: 'Auto Snapshot' (unchecked, with a red arrow labeled '2') and 'Public IP' (checked, with a red arrow labeled '3'). Descriptions for each option are provided below the checkboxes. At the bottom right, there is a yellow button with a 'J' logo and a '+' icon, and a link 'Learn more about snapshots and public IPs here'.

Monthly ☒ Hourly

GPU+	P4000	P5000	P6000
\$0.40 / hr ¹	\$0.40 / hr ¹	\$0.65 / hr ¹	\$0.90 / hr ¹
+ \$5 monthly storage	+ \$5 monthly storage	+ \$5 monthly storage	+ \$5 monthly storage
8 CPU 30 GB RAM Quadro M4000 50 GB SSD	8 CPU 30 GB RAM Quadro P4000 50 GB SSD	8 CPU 30 GB RAM Quadro P5000 50 GB SSD	8 CPU 30 GB RAM Quadro P6000 50 GB SSD

¹ All hourly plans require a flat access fee which covers storage and maintenance costs. [Learn more here](#)

Choose Storage 50 GB

50 GB (\$5/month)

Machine Details

1 machine + jani-cnn ASSIGN MACHINE

Choose Network

Default Network

[View/Configure networks](#)

Options

☐ Auto Snapshot ⓘ
A snapshot will be automatically created during the time interval you specify.

☒ Public IP ⓘ
A single static address for your machine. Perfect for servers, web apps, and more.


[Learn more about snapshots and public IPs here](#)

J +

1. Click on **Create Your Paperspace** to launch the VM

Payment

Note: All accounts must have an active credit card on file even if your total charge is \$0. We accept most cards (Visa, MasterCard, American Express, JCB, Discover, etc)



Promo Code

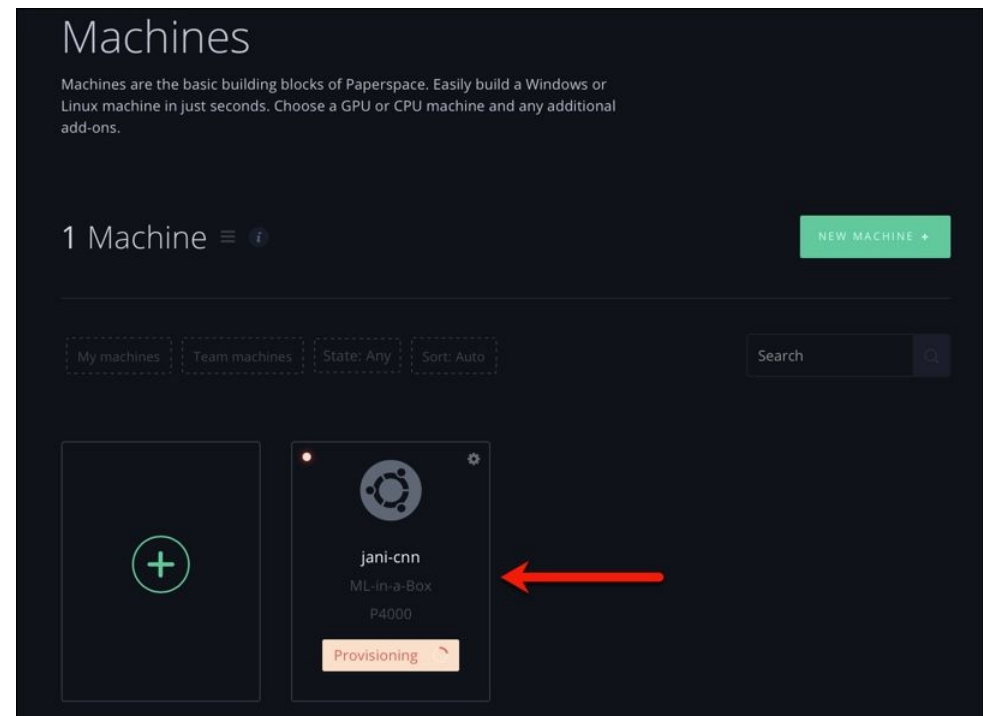
APPLY

1 x Paperspace P4000
ML-in-a-Box (West)

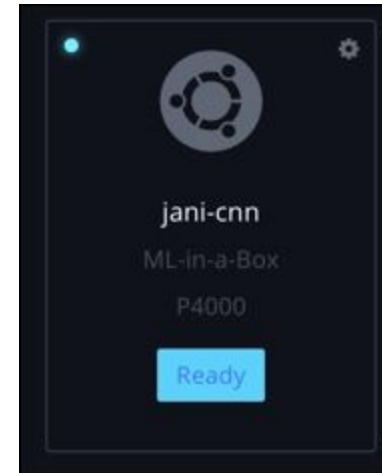
Hourly Cost	\$0.40
Storage Cost	\$5
<div></div>	
Total Billed Now:	\$0.00

CREATE YOUR PAPERSPACE

1. Wait for the machine to get provisioned.



1. You should see **Ready** when the machine is provisioned



Step 2 - Running NVIDIA DIGITS Container

1. Check your Inbox for details on accessing the VM.

Paperspace

Hi there,

Congratulations! Your new Paperspace Linux machine has been created, and it's waiting for you to start it up.

[Go to your machine](#)

[Read the release notes](#)

Your temporary sign-in password for machine **jani-cnn** is:

You can ssh into your new machine with the following command: `ssh paperspace@`

Happy computing!
- The Paperspace Team

1. Open terminal window and SSH into the VM with the credentials and IP address provided in the email

```
1.paperspace@psymuy4lc: ~ [ssh]
[~]$ ssh paperspace@
The authenticity of host ' ' can't be established.
ECDSA key fingerprint is SHA256:0hdUe1ptcIBWjJxRj8FZG/rDSr0b61ZPIRt1vkv1SHY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added ' ' (ECDSA) to the list of known hosts
.
paperspace@ 's password:
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-31-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

-----
WARNING! Your environment specifies an invalid locale.
This can affect your user experience significantly, including the
ability to manage packages. You may install the locales by running:

    sudo apt-get install language-pack-UTF-8
    or
    sudo locale-gen UTF-8

To see all available language packs, run:
    apt-cache search "^language-pack-[a-z][a-z]$"
To disable this message for all users, run:
    sudo touch /var/lib/cloud/instance/locale-check.skip
-----

paperspace@psymuy4lc:~$
```

1. Create a directory called **data** in your home directory
2. Launch NVIDIA DIGITS container with the following command

```
sudo nvidia-docker run --name digits -d -p 5000:5000
-v $PWD/data:/data nvidia/digits
```

```
paperspace@psymuy41c: ~ (ssh)
paperspace@psymuy41c:~$ sudo nvidia-docker run --name digits -d -p 5000:5000 -v
$PWD/data:/data nvidia/digits
Using default tag: latest
latest: Pulling from nvidia/digits
99ad4e3ced4d: Pull complete
ec5a723f4e2a: Pull complete
2a175e11567c: Pull complete
8d26426e95e0: Pull complete
46e451596b7c: Pull complete
79f02f6ab059: Pull complete
59d2eaa1372d: Downloading 83.38MB/453.4MB
4421cd6e3c30: Download complete
9a47758649e5: Download complete
ce8dfefeea4e7: Download complete
0a4202988172: Download complete
505bc2c93c81: Download complete
18dccdda6b6e: Download complete
a0dbf08682d2: Download complete
2795e82bc224: Downloading 205.4MB/307.6MB
a5f44492569e: Download complete
e10245179830: |
```

1. Verify that the DIGITS container is running with the following command

```
sudo docker ps
```

```
paperspace@psymuy41c:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATE
D                  STATUS             PORTS
NAMES
6add4eea38c3        nvidia/digits      "python -m digits"  About
a minute ago      Up About a minute  0.0.0.0:5000->5000/tcp, 6006/tcp
digits
paperspace@psymuy41c:~$
```

Step 3 - Training the Model

1. Create the MNIST dataset within the container

```
sudo docker exec -it digits sh -c "python -m  
digits.download_data mnist /data"
```

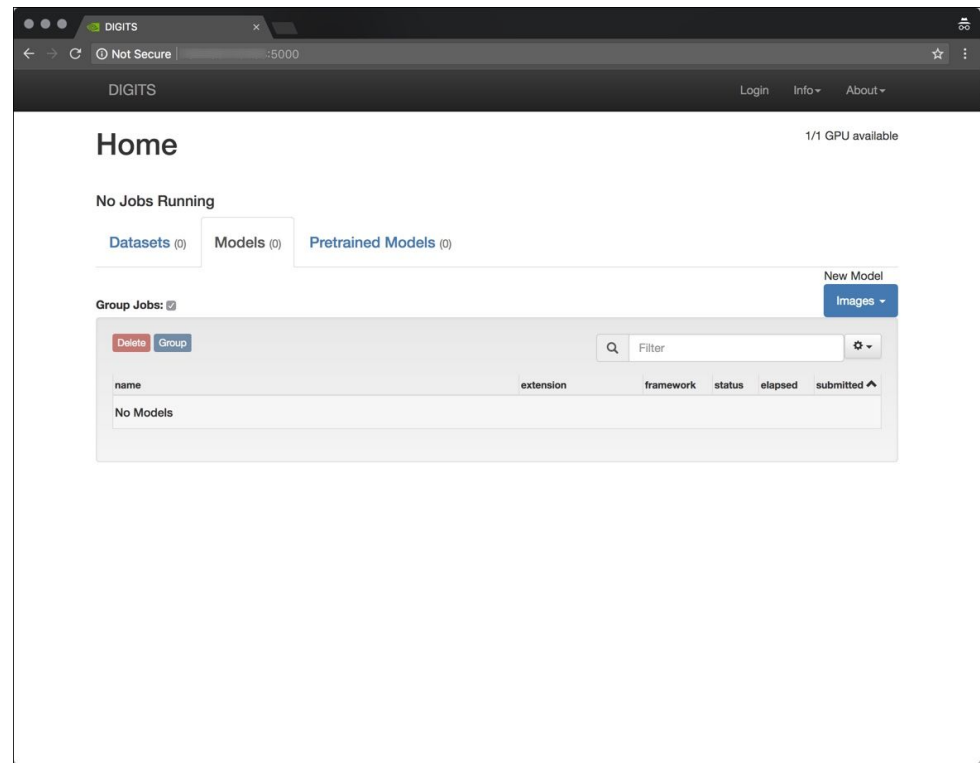
```
paperspace@psymuy4lc:~$ sudo docker exec -it digits sh -c  
"python -m digits.download_data mnist /data"  
Reading labels from /data/train-labels.bin ...  
Reading images from /data/train-images.bin ...  
Reading labels from /data/test-labels.bin ...  
Reading images from /data/test-images.bin ...  
Dataset directory is created successfully at '/data'  
Done after 15.3298709393 seconds.  
paperspace@psymuy4lc:~$
```

1. Verify the creation of the dataset by checking the local directory on the host

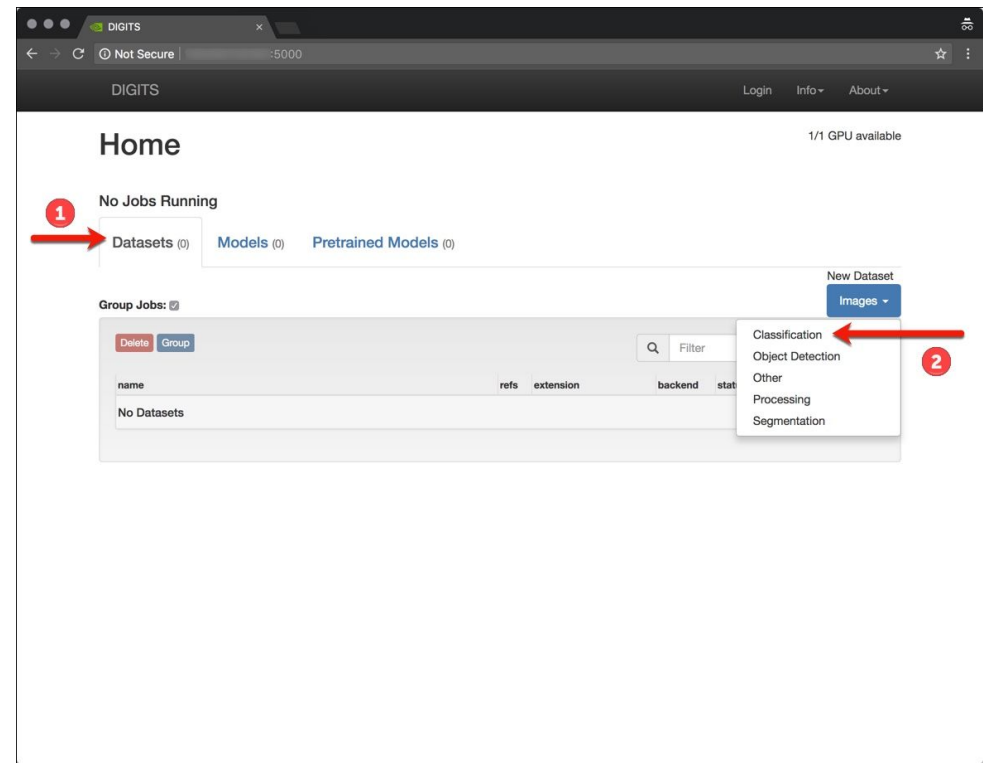
```
ls data
```

```
paperspace@psymuy4lc:~$ ls data  
t10k-images-idx3-ubyte.gz  train  
t10k-labels-idx1-ubyte.gz  train-images-idx3-ubyte.gz  
test                        train-images.bin  
test-images.bin            train-labels-idx1-ubyte.gz  
test-labels.bin            train-labels.bin  
paperspace@psymuy4lc:~$
```

1. Access DIGITS web application from the browser. The IP address is mentioned in the mail that you got during the creation of the VM. Don't forget to add port 5000 to the URL.



1. Click on the **Datasets** tab
2. Click **Images** and select **Classification**
3. Provide a username if asked to login



1. Under **Training Images**, type `/data/train` to point DIGITS to the MNIST dataset
2. Type `mnist` as the **Dataset Name**
3. Click **Create** button to create the dataset

The screenshot shows the 'New Image Classification Dataset' form in the DIGITS application. The form is divided into several sections:

- Image Type:** A dropdown menu set to 'Color'.
- Image size (Width x Height):** Two input fields, both set to '256'.
- Resize Transformation:** A dropdown menu set to 'Squash'.
- See example:** A blue button.
- Training Images:** A section with a text input field containing '/data/train' (annotated with a red arrow and '1'). Below it are fields for 'Minimum samples per class' (set to '2') and 'Maximum samples per class' (empty). Further down are fields for '% for validation' (set to '25') and '% for testing' (set to '0'). There are also two checkboxes: 'Separate validation images folder' and 'Separate test images folder'.
- DB backend:** A dropdown menu set to 'LMDB'.
- Image Encoding:** A dropdown menu set to 'PNG (lossless)'.
- Group Name:** An empty text input field.
- Dataset Name:** A text input field containing 'mnist' (annotated with a red arrow and '2').
- Create:** A blue button at the bottom (annotated with a red arrow and '3').

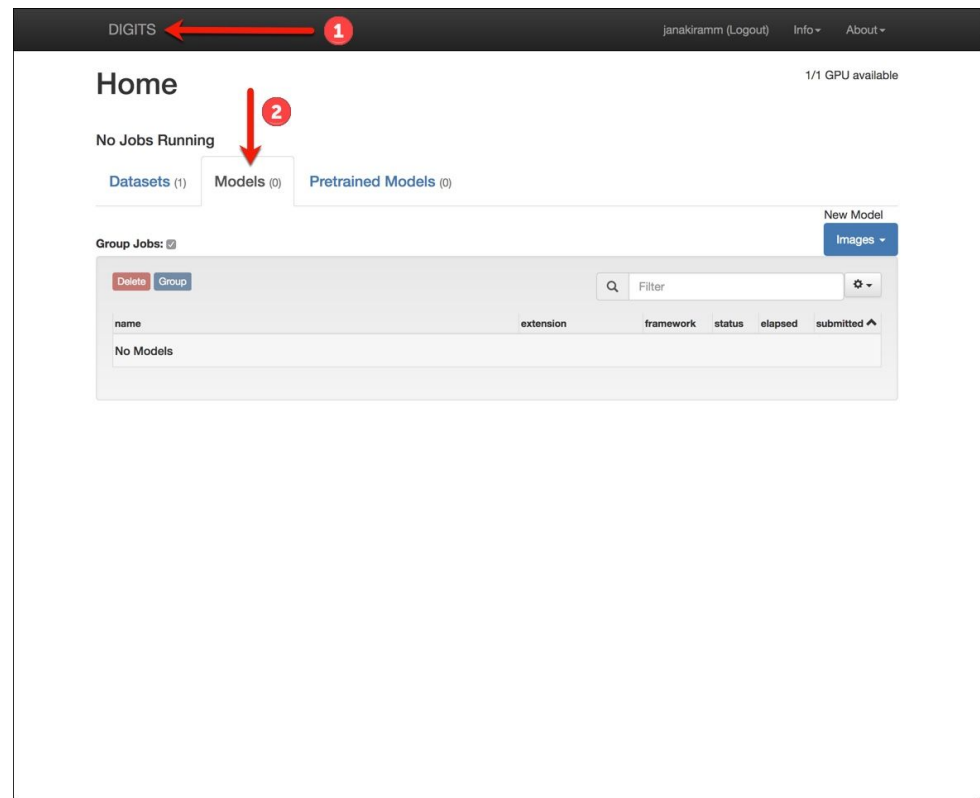
1. Wait for the train and val databases to get initialized. Both the jobs should be done before proceeding to the next step.

The screenshot displays the DIGITS web interface for an 'Image Classification Dataset' named 'mnist'. The interface is divided into several sections:

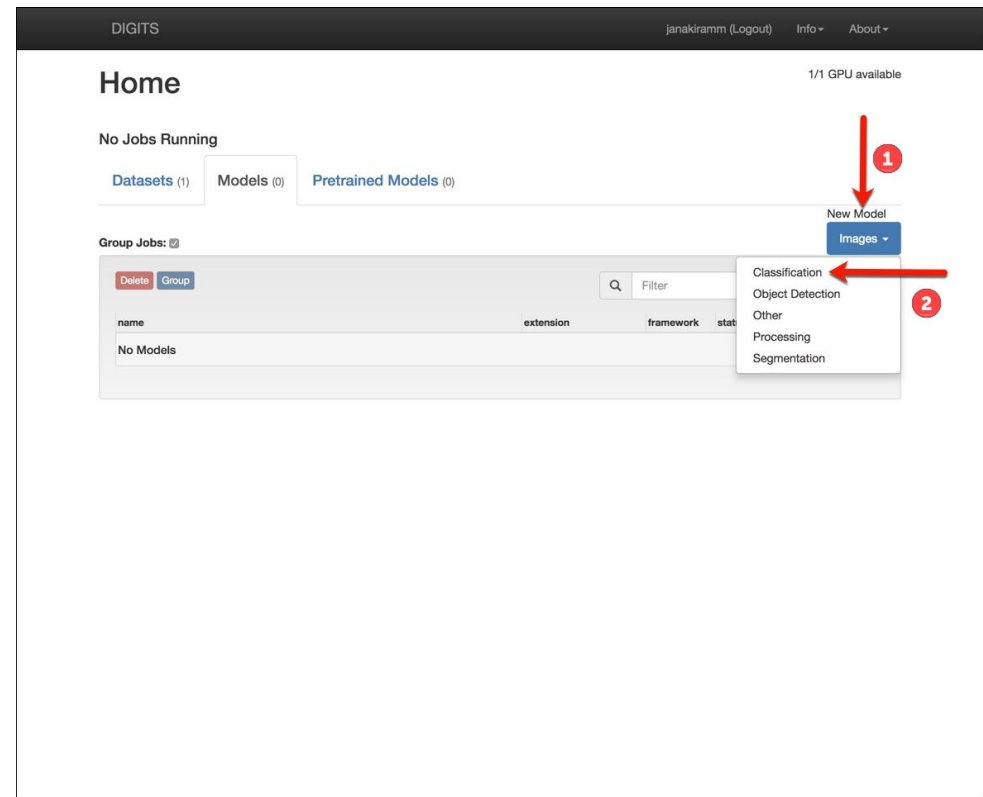
- Job Information:** Lists job details such as Job Directory (/jobs/20180415-081837-eeeb), Image Dimensions (256x256), Image Type (Color), Resize Transformation (Squash), DB Backend (Imdb), Image Encoding (png), DB Compression (none), and Dataset size (0 B).
- Parse Folder (train/val):** Shows the Folder path as /data/train.
- Create DB (train):** A section with a progress bar indicating 71% completion. It includes input files (train.txt, create_train_db.log) and a small bar chart showing a distribution of values between 5000 and 5500.
- Job Status Running:** A summary of job progress with timestamps:
 - Parse Folder (train/val) Running: Initialized at 08:18:37 AM (1 second), Running at 08:18:39 AM.
 - Create DB (train) Running: Estimated time remaining: 30 seconds. Initialized at 08:18:37 AM (3 seconds), Running at 08:18:41 AM.
 - Create DB (val) Running: Initialized at 08:18:37 AM (3 seconds), Running at 08:18:41 AM (43 seconds), Done at 08:19:24 AM (Total - 46 seconds).
- Notes:** A section at the bottom for additional information.

Red arrows point from the 'Create DB (train)' section to the 'Job Status Running' section, highlighting the progress of the database creation process.

1. Click the **DIGITS** logo and then click on the **Models** tab.



1. Click on **Images** and choose **Classification**



1. Choose mnist dataset created in the previous step.
2. Change the Training Epoche to 5
3. Scroll down to access the remaining options

DIGITS New Model janakiram (Logout) Info About

New Image Classification Model

Select Dataset ?

mnist

mnist

Done 08:20:18 AM

Image Size
256x256

Image Type
COLOR

DB backend
Imdb

Create DB (train)
45002 images

Create DB (val)
14998 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 ?
SGD (Stochastic Gradient Descent)

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

1. Under the **Standard Networks** tab, choose **AlexNet**
2. Type **mnist-model** as the **Model Name**
3. Click on **Create** button
4. Wait for the training job to start

The screenshot shows the AWS SageMaker console interface. At the top, there are tabs for 'Standard Networks', 'Previous Networks', 'Pretrained Networks', and 'Custom Network'. Under the 'Standard Networks' tab, there are sub-tabs for 'Caffe', 'Torch', and 'Tensorflow'. A table lists three networks: 'LeNet', 'AlexNet', and 'GoogLeNet'. The 'AlexNet' row is selected, indicated by a red arrow labeled '1'. Below the table, there is a form with a 'Group Name' field and a 'Model Name' field. The 'Model Name' field contains the text 'mnist-model', with a red arrow labeled '2' pointing to it. At the bottom of the form is a blue 'Create' button, with a red arrow labeled '3' pointing to it.

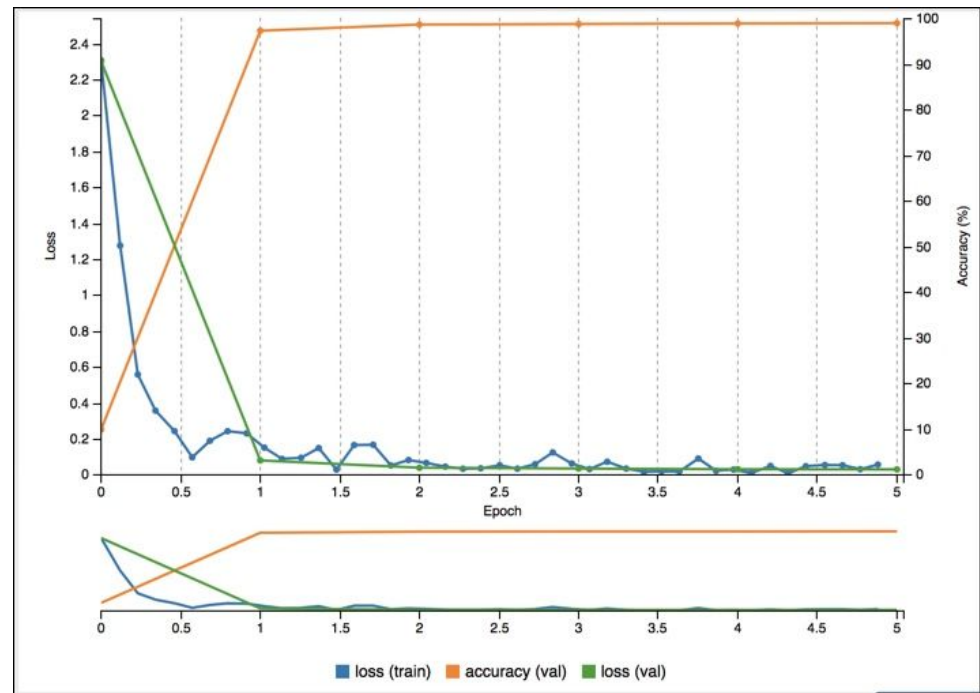
Network	Details	Intended image size
<input type="radio"/> LeNet	Original paper [1998]	28x28 (gray)
<input checked="" type="radio"/> AlexNet	Original paper [2012]	256x256
<input type="radio"/> GoogLeNet	Original paper [2014]	256x256

Group Name

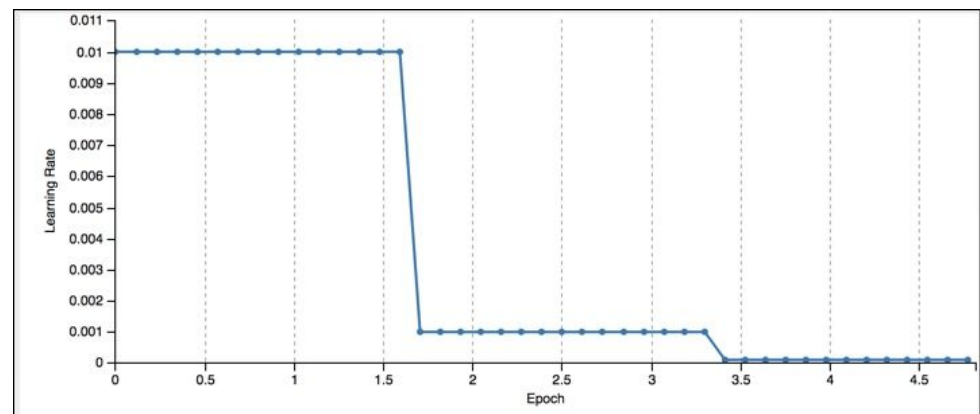
Model Name

Create

1. Notice how the accuracy improved after a few epochs



1. Notice that the learning rate is adjusted with each passing epoch



Step 4 - Testing and Validating the Model

1. In the **Image Path** text box, type the following `/data/test/3/00018.png`
2. Click on the **Classify One** button
3. The browser will open a new tab

The screenshot shows the 'Trained Models' web interface. On the left, under 'Test a single image', the 'Image Path' text box contains '/data/test/3/00018.png' and is marked with a red arrow and a circled '1'. Below it, the 'Classify One' button is marked with a red arrow and a circled '2'. On the right, under 'Test a list of images', there are fields for 'Upload Image List', 'Image folder (optional)', 'Number of images use from the file', and 'Number of images to show per category', along with buttons for 'Classify Many' and 'Top N Predictions per Category'.

Trained Models

Select Model
Epoch #10

[Download Model](#) [Make Pretrained Model](#)

Test a single image

Image Path ?
/data/test/3/00018.png

Upload image
[Browse...](#)

☐ Show visualizations and statistics ?

[Classify One](#)

Test a list of images

Upload Image List
[Browse...](#)
Accepts a list of filenames or urls (you can use your val.txt file)

Image folder (optional)

Relative paths in the text file will be prepended with this value before reading

Number of images use from the file
All
Leave blank to use all


[Classify Many](#) ?

Number of images to show per category
9

[Top N Predictions per Category](#) ?

1. Notice that the model predicted the input image as digit 3 with an probability of 50%
2. Try testing the model with other images that represent different digits

mnist-model Image Classification Model



Predictions	
3	49.49%
8	41.83%
0	5.01%
9	2.51%
5	0.71%

Job Status Done

- Initialized at 10:43:19 AM (1 second)
- Running at 10:43:20 AM (4 seconds)
- Done at 10:43:24 AM (Total - 5 seconds)

Infer Model Done ▾

Notes

None

Step 5 - Running the Model Locally for Inferencing

1. Click on the **Download Model** button under **Trained Models** section
2. Uncompress the downloaded file in your home directory

Trained Models

Select Model
Epoch #10

Download Model **Make Pretrained Model**

Test a single image
Image Path ?
/data/test/3/00018.png

Upload image
Browse...

☐ Show visualizations and statistics ?

Classify One

Test a list of images
Upload Image List
Browse...
Accepts a list of filenames or uris (you can use your val.txt file)

Image folder (optional)
Relative paths in the text file will be prepended with this value before reading

Number of images use from the file
All
Leave blank to use all

Classify Many ?

Number of images to show per category
9

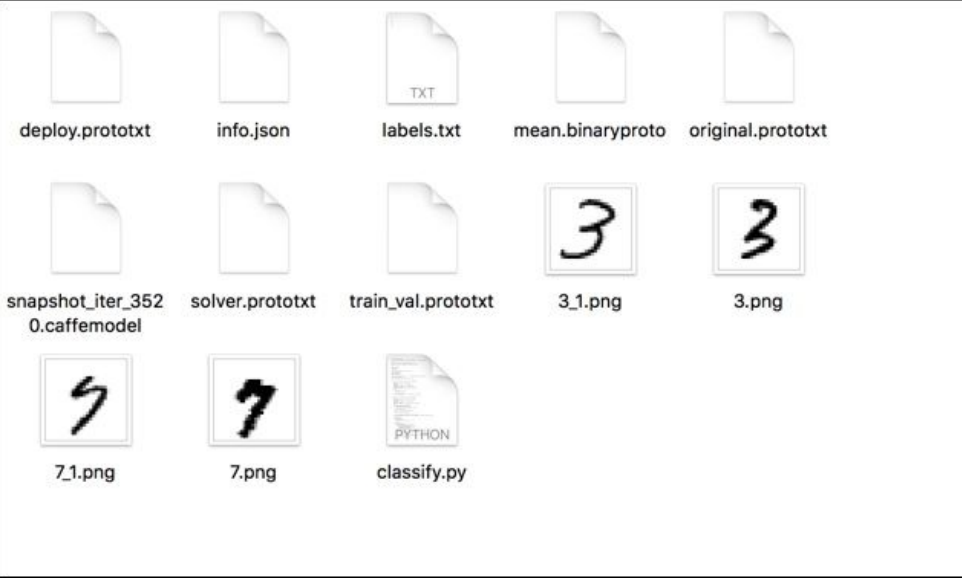

Top N Predictions per Category ?

1. Verify that the Caffe Intel image is available on your local machine

docker images

```
[~]$ docker images
REPOSITORY          TAG          IMAGE ID          CREATED           SIZE
bvlc/caffe           cpu         0b577b836386     3 days ago      1.64GB
[~]$
```

1. Clone the Github repo to access scripts and test images
git clone <https://github.com/janakiramm/cnn-lab>

<p>1. Copy all the files from the repo to the directory containing the downloaded model from DIGITS</p> <pre>cd cnn-lab cp * ~/<Model_Directory></pre>	
<p>1. Find the file with the extension caffemodel, and set its name as a variable</p> <pre>export MODEL_NAME="snapshot_iter_3520.caffemodel"</pre>	

1. Launch the Docker container and pass the parameters

```
docker run -it --rm --name caffe -v $PWD:/infer  
bvlc/caffe:cpu bash -c "cd /infer && python -W  
ignore classify.py -m mean.binaryproto -l labels.txt  
$MODEL_NAME deploy.prototxt 3_1.png --nogpu"
```

```
[20180415-101947-e7dc_epoch_10.0]$ docker run -it --rm --name caffe -v  
$PWD:/infer bvlc/caffe:cpu bash -c "cd /infer && python -W ignore clas  
sify.py -m mean.binaryproto -l labels.txt $MODEL_NAME deploy.prototxt 3  
_1.png --nogpu"  
Processed 1/1 images in 0.341568 seconds ...  
----- Prediction for 3_1.png -----  
-----  
99.9999% - "3"  
0.0001% - "8"  
0.0000% - "9"  
0.0000% - "5"  
0.0000% - "7"  
  
Script took 1.695304 seconds.  
[20180415-101947-e7dc_epoch_10.0]$
```

1. Try the inferencing by sending different images

```
docker run -it --rm --name caffe -v $PWD:/infer  
bvlc/caffe:cpu bash -c "cd /infer && python -W  
ignore classify.py -m mean.binaryproto -l labels.txt  
$MODEL_NAME deploy.prototxt 7.png --nogpu"
```

```
[20180415-101947-e7dc_epoch_10.0]$ docker run -it --rm --name caffe -v  
$PWD:/infer bvlc/caffe:cpu bash -c "cd /infer && python -W ignore clas  
sify.py -m mean.binaryproto -l labels.txt $MODEL_NAME deploy.prototxt 7  
.png --nogpu"  
Processed 1/1 images in 0.342311 seconds ...  
----- Prediction for 7.png -----  
-----  
99.9215% - "7"  
0.0354% - "9"  
0.0119% - "1"  
0.0095% - "2"  
0.0068% - "5"  
  
Script took 1.629630 seconds.  
[20180415-101947-e7dc_epoch_10.0]$
```

Step 6 - Cleaning Up Resources

1. Access Paperspace Console by visiting <https://www.paperspace.com/console/machines>
2. Switch to list mode by clicking on the icon next **Machine**
3. Select the checkbox against your machine
4. Click **Deactivate** to shutdown the machine

