Image Classification with DIGITS

Hyungon Ryu
Certified Instructor, NVIDIA Deep Learning Institute
NVIDIA Corporation
DEEP LEARNING INSTITUTE

DLI Mission

Helping people solve challenging problems using AI and deep learning.

• Developers, data scientists and engineers
• Self-driving cars, healthcare and robotics
• Training, optimizing, and deploying deep neural networks
WHAT IS DEEP LEARNING?
Machine Learning

Neural Networks

Deep Learning
DEEP LEARNING EVERYWHERE

INTERNET & CLOUD
- Image Classification
- Speech Recognition
- Language Translation
- Language Processing
- Sentiment Analysis
- Recommendation

MEDICINE & BIOLOGY
- Cancer Cell Detection
- Diabetic Grading
- Drug Discovery

MEDIA & ENTERTAINMENT
- Video Captioning
- Video Search
- Real Time Translation

SECURITY & DEFENSE
- Face Detection
- Video Surveillance
- Satellite Imagery

AUTONOMOUS MACHINES
- Pedestrian Detection
- Lane Tracking
- Recognize Traffic Sign
ARTIFICIAL NEURONS

Biological neuron

Artificial neuron

Weights \( (W_n) \) = parameters

\[ y = F(w_1x_1 + w_2x_2 + w_3x_3) \]
MLP
ANN for MNIST

28x28=784

Input Layer

Hidden Layer

Output Layer

W11_1

W21_1

W14_3

W23_2

1

2

84

784

1

9
Pre-processing + ANN for MNIST

Features

768 x N

Input Layer

Hidden Layer

Output Layer

1

W11_1

2

W21_1

84

W23_2

9

W14_3

768 x N
Feature with Convolution Filter

edge

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>-2</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

blur

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.1</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>

sharpen

<table>
<thead>
<tr>
<th>-1</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>9</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Motion blur

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
CONVOLUTION

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
ARTIFICIAL NEURAL NETWORK
A collection of simple, trainable mathematical units that collectively
learn complex functions

Given sufficient training data an artificial neural network can approximate very complex
functions mapping raw data to output decisions
DEEP NEURAL NETWORK (DNN)

Application components:

Task objective
e.g. Identify face

Training data
10-100M images

Network architecture
~10s-100s of layers
1B parameters

Learning algorithm
~30 Exaflops
1-30 GPU days
DEEP LEARNING APPROACH

Train:

- Dog
- Cat
- Honey badger

Deploy:

- Dog

Errors

- Dog
- Cat
- Raccoon

DNN
DEEP LEARNING APPROACH - TRAINING

Process
- Forward propagation yields an inferred label for each training image
- Loss function used to calculate difference between known label and predicted label for each image
- Weights are adjusted during backward propagation
- Repeat the process

Input

Forward propagation

Backward propagation
ADDITIONAL TERMINOLOGY

• Hyperparameters - parameters specified before training begins
  • Can influence the speed in which learning takes place
  • Can impact the accuracy of the model
  • Examples: Learning rate, decay rate, batch size

• Epoch - complete pass through the training dataset

• Activation functions - identifies active neurons
  • Examples: Sigmoid, Tanh, ReLU

• Pooling - Down-sampling technique
  • No parameters (weights) in pooling layer
HANDWRITTEN DIGIT RECOGNITION
HANDWRITTEN DIGIT RECOGNITION
HELLO WORLD of machine learning?

- MNIST data set of handwritten digits from Yann Lecun’s website
- All images are 28x28 grayscale
  - Pixel values from 0 to 255
- 60K training examples / 10K test examples
- Input vector of size 784
  - 28 * 28 = 784
- Output value is integer from 0-9
CAFFE
NVIDIA Powers Deep Learning

Every major DL framework leverages NVIDIA SDKs
WHAT IS CAFFE?
An open framework for deep learning developed by the Berkeley Vision and Learning Center (BVLC)

- Pure C++/CUDA architecture
- Command line, Python, MATLAB interfaces
- Fast, well-tested code
- Pre-processing and deployment tools, reference models and examples
- Image data management
- Seamless GPU acceleration
- Large community of contributors to the open-source project

caffe.berkeleyvision.org
http://github.com/BVLC/caffe
CAFFE FEATURES
Deep Learning model definition

Protobuf model format
• Strongly typed format
• Human readable
• Auto-generates and checks Caffe code
• Developed by Google
• Used to define network architecture and training parameters
• No coding required!

name: "conv1"
type: "Convolution"
bottom: "data"
top: "conv1"

convolution_param {
  num_output: 20
  kernel_size: 5
  stride: 1
  weight_filler {
    type: "xavier"
  }
}
NVIDIA’S DIGITS
NVIDIA DIGITS
Interactive Deep Learning GPU Training System

Process Data
Configure DNN
Monitor Progress
Visualization
NVIDIA’S DIGITS
Interactive Deep Learning GPU Training System

• Simplifies common deep learning tasks such as:
  • Managing data
  • Designing and training neural networks on multi-GPU systems
  • Monitoring performance in real time with advanced visualizations
• Completely interactive so data scientists can focus on designing and training networks rather than programming and debugging
• Open source
Clicking DIGITS will bring you to this Home screen.

Clicking here will present different options for model and dataset creation.

Click here to see a list of existing datasets or models.
DIGITS - DATASET

Different options will be presented based upon the task.
DIGITS - MODEL

Differences may exist between model tasks

Define custom layers with Python

Can anneal the learning rate
DIGITS - TRAINING

Loss function and accuracy during training

Annealed learning rate
Once training is complete DIGITS provides an easy way to visualize what happened.
DIGITS PLUGINS

DIGITS Plugins
Image: Sunnybrook LV Segmentation
plugins/data/sunnybrook

DIGITS Plugins
Image: Regression
plugins/data/imageGradients

DIGITS Plugins
Text
plugins/data/textClassification
LAUNCHING THE LAB ENVIRONMENT
NAVGATING TO QWIKLABS

1. Navigate to: https://nvlabs.qwiklab.com

1. Login or create a new account
ACCESSING LAB ENVIRONMENT

3. Select the event specific In-Class Session Class in the upper left

3. Click the “Image Classification with DIGITS” Class from the list
LAUNCHING THE LAB ENVIRONMENT

5. Click on the Select button to launch the lab environment
   - After a short wait, lab Connection information will be shown
   - Please ask Lab Assistants for help!
LAUNCHING THE LAB ENVIRONMENT

6. Click on the Start Lab button

You should see that the lab environment is “launching” towards the upper-right corner.
7. Click on “here” to access your lab environment / Jupyter notebook
CONNECTING TO THE LAB ENVIRONMENT

You should see your “Getting Started With Deep Learning” Jupyter notebook.
JUPYTER NOTEBOOK

1. Place your cursor in the code

2. Click the "run cell" button

2. Confirm you receive the same result
STARTING DIGITS

Instruction in Jupyter notebook will link you to DIGITS
ACCESSING DIGITS

• Will be prompted to enter a username to access DIGITS
  • Can enter any username
  • Use lower case letters
LAB DISCUSSION / OVERVIEW
CREATE DATASET IN DIGITS

• Dataset settings
  • Image Type: Grayscale
  • Image Size: 28 x 28
  • Training Images: /home/ubuntu/data/train_small
  • Select “Separate test images folder” checkbox
  • Test Images: /home/ubuntu/data/test_small
  • Dataset Name: MNIST Small
CREATE MODEL

- Select the “MNIST small” dataset
- Set the number of “Training Epochs” to 10
- Set the framework to “Caffe”
- Set the model to “LeNet”
- Set the name of the model to “MNIST small”
- When training done, Classify One:

  /home/ubuntu/data/test_small/2/img_4415.png
EVALUATE THE MODEL

Accuracy obtained from validation dataset

Loss function (Validation)

Loss function (Training)
ADDITIONAL TECHNIQUES TO IMPROVE MODEL

• More training data
• Data augmentation
• Modify the network
FIRST RESULTS
Small dataset ( 10 epochs )

• 96% of accuracy achieved
• Training is done within one minute

<table>
<thead>
<tr>
<th>SMALL DATASET</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 : 99.00 %</td>
</tr>
<tr>
<td>1 : 99.90 %</td>
</tr>
<tr>
<td>2 : 69.03 %</td>
</tr>
<tr>
<td>8 : 71.37 %</td>
</tr>
<tr>
<td>8 : 85.07 %</td>
</tr>
<tr>
<td>8 : 99.69 %</td>
</tr>
<tr>
<td>8 : 54.75 %</td>
</tr>
</tbody>
</table>
FULL DATASET

6x larger dataset

- Dataset
  - Training Images: /home/ubuntu/data/train_full
  - Test Image: /home/ubuntu/data/test_full
  - Dataset Name: MNIST full

- Model
  - Clone “MNIST small”.
  - Give a new name “MNIST full” to push the create button
SECOND RESULTS
Full dataset (10 epochs)

- 99% of accuracy achieved
- No improvements in recognizing real-world images

<table>
<thead>
<tr>
<th></th>
<th>SMALL DATASET</th>
<th>FULL DATASET</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1: 99.90%</td>
<td>0: 93.11%</td>
</tr>
<tr>
<td>2</td>
<td>2: 69.03%</td>
<td>2: 87.23%</td>
</tr>
<tr>
<td>3</td>
<td>8: 71.37%</td>
<td>8: 71.60%</td>
</tr>
<tr>
<td>4</td>
<td>8: 85.07%</td>
<td>8: 79.72%</td>
</tr>
<tr>
<td>7</td>
<td>0: 99.00%</td>
<td>0: 95.82%</td>
</tr>
<tr>
<td>8</td>
<td>8: 99.69%</td>
<td>8: 100.0%</td>
</tr>
<tr>
<td>9</td>
<td>8: 54.75%</td>
<td>2: 70.57%</td>
</tr>
</tbody>
</table>
DATA AUGMENTATION
Adding Inverted Images

- Pixel(Inverted) = 255 - Pixel(original)
- White letter with black background
  - Black letter with white background
- Training Images: /home/ubuntu/data/train_invert
- Test Image: /home/ubuntu/data/test_invert
- Dataset Name: MNIST invert
# DATA AUGMENTATION

Adding inverted images (10 epochs)

<table>
<thead>
<tr>
<th></th>
<th>SMALL DATASET</th>
<th>FULL DATASET</th>
<th>+INVERTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 : 99.90 %</td>
<td>0 : 93.11 %</td>
<td>1 : 90.84 %</td>
</tr>
<tr>
<td>2</td>
<td>2 : 69.03 %</td>
<td>2 : 87.23 %</td>
<td>2 : 89.44 %</td>
</tr>
<tr>
<td>3</td>
<td>8 : 71.37 %</td>
<td>8 : 71.60 %</td>
<td>3 : 100.0 %</td>
</tr>
<tr>
<td>4</td>
<td>8 : 85.07 %</td>
<td>8 : 79.72 %</td>
<td>4 : 100.0 %</td>
</tr>
<tr>
<td>7</td>
<td>0 : 99.00 %</td>
<td>0 : 95.82 %</td>
<td>7 : 82.84 %</td>
</tr>
<tr>
<td>8</td>
<td>8 : 99.69 %</td>
<td>8 : 100.0 %</td>
<td>8 : 100.0 %</td>
</tr>
<tr>
<td>8</td>
<td>8 : 54.75 %</td>
<td>2 : 70.57 %</td>
<td>2 : 96.27 %</td>
</tr>
</tbody>
</table>
MODIFY THE NETWORK
Adding filters and ReLU layer

layer {
    name: "pool1"
    type: "Pooling"
    ...
}

layer {
    name: "reluP1"
    type: "ReLU"
    bottom: "pool1"
    top: "pool1"
}

layer {
    name: "reluP1"
    ...
}

layer {
    name: "conv1"
    type: "Convolution"
    ...
    convolution_param {
        num_output: 75
    }
    ...
}

layer {
    name: "conv2"
    type: "Convolution"
    ...
    convolution_param {
        num_output: 100
    }
    ...
}
MODIFY THE NETWORK

Adding ReLU Layer

pool1
(MAX Pooling)
kernel size: 2
stride: 2
pad: 0

pool1

conv2
(Convolution)
kernel size: 5
stride: 1
pad: 0

relu1
(ReLU)

pool1

conv2
(Convolution)
kernel size: 5
stride: 1
pad: 0
## MODIFIED NETWORK

Adding filters and ReLU layer (10 epochs)

<table>
<thead>
<tr>
<th></th>
<th>SMALL DATASET</th>
<th>FULL DATASET</th>
<th>+INVERTED</th>
<th>ADDING LAYER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 : 99.90 %</td>
<td>0 : 93.11 %</td>
<td>1 : 90.84 %</td>
<td>1 : 59.18 %</td>
</tr>
<tr>
<td>2</td>
<td>2 : 69.03 %</td>
<td>2 : 87.23 %</td>
<td>2 : 89.44 %</td>
<td>2 : 93.39 %</td>
</tr>
<tr>
<td>3</td>
<td>8 : 71.37 %</td>
<td>8 : 71.60 %</td>
<td>3 : 100.0 %</td>
<td>3 : 100.0 %</td>
</tr>
<tr>
<td>4</td>
<td>8 : 85.07 %</td>
<td>8 : 79.72 %</td>
<td>4 : 100.0 %</td>
<td>4 : 100.0 %</td>
</tr>
<tr>
<td>5</td>
<td>0 : 99.00 %</td>
<td>0 : 95.82 %</td>
<td>7 : 82.84 %</td>
<td>2 : 62.52 %</td>
</tr>
<tr>
<td>6</td>
<td>8 : 99.69 %</td>
<td>8 : 100.0 %</td>
<td>8 : 100.0 %</td>
<td>8 : 100.0 %</td>
</tr>
<tr>
<td>7</td>
<td>8 : 54.75 %</td>
<td>2 : 70.57 %</td>
<td>2 : 96.27 %</td>
<td>8 : 70.83 %</td>
</tr>
</tbody>
</table>
WHAT’S NEXT

• Use / practice what you learned
• Discuss with peers practical applications of DNN
• Reach out to NVIDIA and the Deep Learning Institute