COMS0018: PRACTICAL1 (Intro to Lab1)

Dima Damen
Dima.Damen@bristol.ac.uk

Bristol University, Department of Computer Science
Bristol BS8 1UB, UK

October 7, 2019
Several open-source software libraries are available for training DNNs:

- Caffe
- Theano
- Tensorflow
- Torch
Several open-source software libraries are available for training DNNs

- Caffe (Berkeley)
- Theano (University of Montreal)
- Tensorflow (Google Brain)
- Torch (adopted by Facebook AI)
Several open-source software libraries are available for training DNNs:

- Caffe (Berkeley)
- Theano (University of Montreal)
- Tensorflow (Google Brain)
- Torch (adopted by Facebook AI)

Due to the challenge in maintaining such software, ones created by universities are no longer maintained.
Several open-source software libraries are available for training DNNs:

- Caffe (Berkeley)
- Theano (University of Montreal)
- Tensorflow (Google Brain)
- Torch (adopted by Facebook AI)

Due to the challenge in maintaining such software, ones created by universities are no longer maintained.

This leaves us with Tensorflow and Torch as the current competitors.
Several open-source software libraries are available for training DNNs
- Caffe (Berkeley)
- Theano (University of Montreal)
- Tensorflow (Google Brain)
- Torch (adopted by Facebook AI)

Due to the challenge in maintaining such software, ones created by universities are no longer maintained.

This leaves us with Tensorflow and Torch as the current competitors.

In 2017 and 2018 we used Tensorflow to teach this unit.
PyTorch

- An unavoidable trend (Article on Sep 2018)

![Google Search: Past 6 Months to Prior 6 Months](https://towardsdatascience.com/which-deep-learning-framework-is-growing-fastest-3f77f14aa318)

Dima Damen
Dima.Damen@bristol.ac.uk
COMSM0018: FIRST CNN - 2019/2020
The main challenge in running the forward-backward algorithm is related to running time and memory size.

GPUs allow parallel processing for all matrix multiplications.

In DNN, all operations in both passes are in essence matrix multiplications.

---

1 https://developer.nvidia.com/cudnn
The main challenge in running the forward-backward algorithm is related to running time and memory size.

GPUs allow parallel processing for all matrix multiplications.

In DNN, all operations in both passes are in essence matrix multiplications.

The NVIDIA CUDA Deep Neural Network library (cuDNN) offers further optimised implementations of deep learning algorithms.

---

https://developer.nvidia.com/cudnn
Tensorflow - CPU vs GPU

https://github.com/jcjohnson/cnn-benchmarks

Dima Damen
Dima.Damen@bristol.ac.uk
COMSM0018: FIRST CNN - 2019/2020
Blue Crystal 4

BC4 uses Lenovo NeXtScale compute nodes, each comprising of two 14 core 2.4 GHz Intel Broadwell CPUs with 128 GiB of RAM. It also includes 32 nodes of two NVIDIA Pascal P100 GPUs plus one GPU login node, designed into the rack by Lenovo’s engineering team to meet the specific requirements of the University.\(^2\)

\(^2\)http://www.bristol.ac.uk/cabot/news/2017/blue-crystal-4.html
There are two ways to use the GPU logins in BC4
  - Interactive jobs - for lab sessions
  - Job queues - for off-lab and coursework work
There are two ways to use the GPU logins in BC4

- Interactive jobs - for lab sessions
- Job queues - for off-lab and coursework work

ACRC has reserved all 64 GPUs for this lab’s purposes :-)
Blue Crystal 4 - Interactive Jobs

1. First, you need to login to BC4
2. You can then reserve a GPU for interactive running
3. This GPU is hogged for your usage until it’s released
4. Please remember to release the GPU as soon as your job concludes
During training DNNs, you can observe the progress of the training using tensorboard.

Using a new terminal, you can open a port to observe the training process.

Make sure both terminals are properly closed to release the GPUs.
In this lab,

▶ You’ll get an introduction to PyTorch

Using the IRIS dataset - collected by biologist Ronald Fisher in his 1936 paper “The use of multiple measurements in taxonomic problems”.

In this lab,

▶ You’ll get an introduction to PyTorch
▶ You’ll build your first fully connected network

Using the IRIS dataset - collected by biologist Ronald Fisher in his 1936 paper "The use of multiple measurements in taxonomic problems".

In this lab,

- You’ll get an introduction to PyTorch
- You’ll build your first fully connected network
- Using the IRIS dataset - collected by biologist Ronald Fisher in his 1936 paper “The use of multiple measurements in taxonomic problems”.

Dima Damen
Dima.Damen@bristol.ac.uk
COMSM0018: FIRST CNN - 2019/2020
In this lab,

► You’ll get an introduction to PyTorch
► You’ll build your first fully connected network
► Using the IRIS dataset - collected by biologist Ronald Fisher in his 1936 paper “The use of multiple measurements in taxonomic problems”.

Data set [edit]

The dataset contains a set of 150 records under five attributes - petal length, petal width, sepal length, sepal width and species.

### Fisher’s Iris Data [hide]

<table>
<thead>
<tr>
<th>Dataset Order</th>
<th>Sepal length</th>
<th>Sepal width</th>
<th>Petal length</th>
<th>Petal width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>2</td>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>3</td>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>4</td>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>5</td>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.3</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>6</td>
<td>5.4</td>
<td>3.9</td>
<td>1.7</td>
<td>0.4</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>7</td>
<td>4.6</td>
<td>3.4</td>
<td>1.4</td>
<td>0.3</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>8</td>
<td>5.0</td>
<td>3.4</td>
<td>1.5</td>
<td>0.2</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>9</td>
<td>4.4</td>
<td>2.9</td>
<td>1.4</td>
<td>0.2</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>10</td>
<td>4.9</td>
<td>3.1</td>
<td>1.5</td>
<td>0.1</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>11</td>
<td>5.4</td>
<td>3.7</td>
<td>1.5</td>
<td>0.2</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>12</td>
<td>4.8</td>
<td>3.4</td>
<td>1.6</td>
<td>0.2</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>13</td>
<td>4.8</td>
<td>3.0</td>
<td>1.4</td>
<td>0.1</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>14</td>
<td>4.3</td>
<td>3.0</td>
<td>1.1</td>
<td>0.1</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>15</td>
<td>5.8</td>
<td>4.0</td>
<td>1.2</td>
<td>0.2</td>
<td><em>I. setosa</em></td>
</tr>
</tbody>
</table>
In this lab,

- Sepal Length
- Sepal Width
- Petal Length
- Petal Width
- Species
In this lab,
In this lab,
In this lab,
In this lab,

Sepal Length: 4.3
Sepal Width: 3.0
Petal Length: 1.1
Petal Width: 0.1

Labels:
- 1: setosa
- 0: versicolor
- 0: virginica
In this lab,
In this lab,
In this lab,
In this lab,
In this lab,
In this lab,
In this lab,
In this lab,

- Our focus is on the weight tensors... $W_1 [4, 100], W_2 [100, 3] \rightarrow \text{total: 700 weights to train}$
- To train... 150 samples!!!!
First Steps,

- Test your BC4 connection
First Steps,

- Test your BC4 connection
- Let us know once you’ve reserved your first GPU
First Steps,

- Test your BC4 connection
- Let us know once you’ve reserved your first GPU
- You will need this connection for all labs, and for your project
Introduction to PyTorch,

- We suggest that you use Google Colaboratory [optional but helpful]
Introduction to PyTorch,

- We suggest that you use Google Colaboratory [optional but helpful]
- Introduction to PyTorch basic operations
Introduction to PyTorch,

- We suggest that you use Google Colaboratory [optional but helpful]
- Introduction to PyTorch basic operations
- Important: Tensor and tensor dimensions - 1D, 2D, 3D, 4D!
Introduction to PyTorch,

- We suggest that you use Google Colaboratory [optional but helpful]
- Introduction to PyTorch basic operations
- Important: Tensor and tensor dimensions - 1D, 2D, 3D, 4D!
- Think about tensor reshaping and their effect
Tensor Reshaping,
From Theory to Practice,

```python
for t=0, 1, 2, ... do
    pick next training sample
    FORWARD PASS: compute all layer outputs
    compute derivative of cost function w.r.t. final layer
    BACKWARD PASS: compute all deltas
    update all weights based on deltas and activities

    for epoch in range(0, 100):
        logits = model.forward(features['train'])
        loss = criterion(logits, labels['train'])
        loss.backward()
        optimizer.step()

    logits = model.forward(features['test'])
```
From Theory to Practice,

epoch: 0 train accuracy: 48.00, loss: 1.22696
epoch: 1 train accuracy: 48.00, loss: 1.03830
epoch: 2 train accuracy: 72.00, loss: 0.90800
epoch: 3 train accuracy: 72.00, loss: 0.82028
epoch: 4 train accuracy: 74.00, loss: 0.75852
epoch: 5 train accuracy: 77.00, loss: 0.71211
epoch: 6 train accuracy: 78.00, loss: 0.67529
epoch: 7 train accuracy: 78.00, loss: 0.64492
epoch: 8 train accuracy: 79.00, loss: 0.61916
epoch: 9 train accuracy: 81.00, loss: 0.59687
epoch: 10 train accuracy: 82.00, loss: 0.57729
epoch: 11 train accuracy: 83.00, loss: 0.55990
epoch: 12 train accuracy: 83.00, loss: 0.54429
epoch: 13 train accuracy: 83.00, loss: 0.53019
epoch: 14 train accuracy: 83.00, loss: 0.51736
epoch: 15 train accuracy: 83.00, loss: 0.50563
epoch: 16 train accuracy: 84.00, loss: 0.49484
epoch: 17 train accuracy: 84.00, loss: 0.48488
epoch: 18 train accuracy: 85.00, loss: 0.47565
epoch: 19 train accuracy: 85.00, loss: 0.46706
epoch: 20 train accuracy: 86.00, loss: 0.45904
epoch: 21 train accuracy: 85.00, loss: 0.45152
epoch: 22 train accuracy: 85.00, loss: 0.44447
epoch: 23 train accuracy: 85.00, loss: 0.43782
epoch: 24 train accuracy: 85.00, loss: 0.43154
epoch: 25 train accuracy: 85.00, loss: 0.42559
epoch: 26 train accuracy: 86.00, loss: 0.41995
epoch: 27 train accuracy: 86.00, loss: 0.41459
epoch: 28 train accuracy: 86.00, loss: 0.40947
epoch: 29 train accuracy: 87.00, loss: 0.40459
epoch: 30 train accuracy: 87.00, loss: 0.39992
epoch: 31 train accuracy: 87.00, loss: 0.39544
epoch: 32 train accuracy: 88.00, loss: 0.39115
From Theory to Practice,

- We will also learn to plot these loss and accuracy curves
From Theory to Practice,

- We will also learn to plot these loss and accuracy curves
- Make sure you always distinguish train curves from test curves
By the end of the lab,

- We need 1 zip file
By the end of the lab,

- We need 1 zip file

**Preparing Lab_1 Portfolio**

You should by now have the following files, which you can zip under the name `Lab_1_<username>.zip`

```
Lab_1_<username>.zip
|-- logs
|-- train_fully_connected.py
```

Store this zip safely. You will be asked to upload all your labs' portfolio to **Blackboard at Week 7**
And now....

READY....

STEADY....

GO...