Introduction to Deep Learning: Concepts and Terminologies

CSE 5194.01

Autumn ‘20

Arpan Jain
The Ohio State University
E-mail: jain.575@osu.edu
Outline

- Introduction
- DNN Training
- Essential Concepts
- Parallel and Distributed DNN Training
Deep Learning

• According to Yoshua Bengio
  “Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features”

• Deep Learning
  • Uses Deep Neural Networks and its variants
  • Based on learning data representation
  • It can be supervised or unsupervised
  • Examples Convolutional Neural Network (CNN), Recurrent Neural Network, Hybrid Networks

Source: https://thenewstack.io/demystifying-deep-learning-and-artificial-intelligence/
One Line (Unofficial) Definitions

• Machine Learning - Ability of machines to learn without being programmed

• Supervised Learning - We provide the machine with the “right answers” (labels)
  – Classification – Discrete value output (e.g. email is spam or not-spam)
  – Regression – Continuous output values (e.g. house prices)

• Unsupervised Learning - No “right answers” given. Learn yourself; no labels for you!
  – Clustering – Group the data points that are ”close” to each other (e.g. cocktail party problem)
    • finding structure in data is the key here!

• Features – Input attributes (e.g. tumor size, age, etc. in cancer detection problem)
  – A very important concept in learning so please remember this!

• Deep Learning – learning that uses Deep Neural Networks
Spot Quiz: Supervised vs. Unsupervised?

- Left Picture: Supervised/Unsupervised?
- Right Picture: Supervised/Unsupervised?

- What is X1 and X2?
- What do colors/shapes represent?
- What is the green line?
• To actually train a network, please visit: http://playground.tensorflow.org
Handwritten Numbers (Quick Demo)

- To try handwritten numbers, please visit: https://microsoft.github.io/onnxjs-demo/#/mnist
Outline

• Introduction

• DNN Training
  • Essential Concepts
  • Parallel and Distributed DNN Training
DNN Training: Forward Pass

Input Layer  Hidden Layer  Hidden Layer  Output Layer
DNN Training: Forward Pass

Forward Pass

Input Layer
Hidden Layer
Hidden Layer
Output Layer

X

$w_1$

$w_2$
DNN Training: Forward Pass

Forward Pass

Input Layer

Hidden Layer

Hidden Layer

Output Layer

$X$

$w_1$

$w_2$

$w_3$

$w_4$

$w_5$

$w_6$
DNN Training: Forward Pass

Input Layer

Hidden Layer

Hidden Layer

Output Layer
DNN Training: Forward Pass

Forward Pass

\[
\begin{align*}
\text{Input Layer} & : X \\
\text{Hidden Layer 1} & : W_1, W_2 \\
\text{Hidden Layer 2} & : W_3, W_4 \\
\text{Hidden Layer 3} & : W_5, W_6 \\
\text{Output Layer} & : W_7, W_8 \\
\text{Prediction} (\text{Pred}) & \\
\text{Error} & = \text{Loss}(\text{Pred}, \text{Output})
\end{align*}
\]
DNN Training: Backward Pass

Forward Pass

Input Layer → Hidden Layer → Hidden Layer → Output Layer

Error = Loss(Pred,Output)

Backward Pass

$E_f$ $E_o$
DNN Training: Backward Pass

Forward Pass

Input Layer

Hidden Layer

Hidden Layer

Output Layer

Backward Pass

Error = \text{Loss}(\text{Pred}, \text{Output})
DNN Training: Backward Pass

Forward Pass

Input Layer

Hidden Layer

Hidden Layer

Output Layer

Backward Pass

Error = Loss(Pred, Output)
DNN Training

Input Layer

Hidden Layer

Hidden Layer

Output Layer
Outline

- Introduction
- DNN Training

- Essential Concepts
- Parallel and Distributed DNN Training
Essential Concepts: Activation function and Back-propagation

- Back-propagation involves complicated mathematics.
  - Luckily, most DL Frameworks give you a one line implementation -- `model.backward()`

\[
a^{(2)} = g(\theta_1 x) \\
a^{(3)} = g(\theta_2 a^{(2)})
\]

What are Activation functions?
- RELU (a Max fn.) is the most common activation fn.
- Sigmoid, tanh, etc. are also used

I encourage everyone to take CSE 5526!
Essential Concepts: Stochastic Gradient Descent (SGD)

- **Goal of SGD:**
  - Minimize a cost fn.
  - $J(\theta)$ as a function of $\theta$
- **SGD is iterative**
- **Only two equations to remember:**
  
  \[
  \theta_i := \theta_i + \Delta \theta_i \\
  \Delta \theta_i = -\alpha \cdot \left( \frac{\partial J(\theta)}{\partial \theta_i} \right)
  \]

- $\alpha = \text{learning rate}$

**Diagram**

- Graph showing the function $J(\theta)$ with a point labeled $\theta_{\text{optimal}}$.

*[Courtesy: https://www.jeremyjordan.me/gradient-descent/]
Essential Concepts: Learning Rate ($\alpha$)

- **Too low**: A small learning rate requires many updates before reaching the minimum point.

- **Just right**: The optimal learning rate swiftly reaches the minimum point.

- **Too high**: Too large of a learning rate causes drastic updates which lead to divergent behaviors.

*Courtesy: [https://www.jeremyjordan.me/nn-learning-rate/](https://www.jeremyjordan.me/nn-learning-rate/)*
Essential Concepts: Batch Size

- Batched Gradient Descent
  - Batch Size = N
- Stochastic Gradient Descent
  - Batch Size = 1
- Mini-batch Gradient Descent
  - Somewhere in the middle
  - Common:
    - **Batch Size** = 64, 128, 256, etc.
- Finding the optimal batch size will yield the fastest learning.

One full pass over N is called an **epoch** of training.

Courtesy: https://www.jeremyjordan.me/gradient-descent/
Mini-batch Gradient Descent (Example)
How to define the “size” of a model? (model is also called a DNN or a network)

Size means several things and context is important

- Model Size: # of parameters (weights on edges)
- Model Size: # of layers (model depth)
Essential Concepts: Accuracy and Throughput (Speed)

• What is the end goal of training a model with SGD and Back-propagation?
  – Of course, train the machine to predict something useful for you

• How do we measure success?
  – Well, accuracy of the trained model on “new” data is the metric of success

• How quickly we can reach there is:
  – ”good to have” for some models
  – “practically necessary” for most state-of-the-art models
  – In Computer Vision: images/second is the metric of throughput/speed

• Why?
  – Let’s hear some opinions from the class
Outline

• Introduction
• DNN Training
• Essential Concepts
• Parallel and Distributed DNN Training
Impact of Model Size and Dataset Size

- **Large models** $\rightarrow$ better accuracy
- **More data** $\rightarrow$ better accuracy

- Single-node Training; good for
  - Small model and small dataset

- Distributed Training; good for:
  - Large models and large datasets

**Model > Data**: Over-fitting? Multi-GPU or Distributed

**Data > Model**: Under-fitting? Multi-GPU or Distributed

---

Overfitting and Underfitting

- **Overfitting** – model > data → so model is not learning but memorizing your data
- **Underfitting** – data > model → so model is not learning because it cannot capture the complexity of your data

Parallelization Strategies

• What are the Parallelization Strategies
  – Model Parallelism
  – Data Parallelism (Received the most attention)
  – Hybrid Parallelism
  – Automatic Selection

**Need for Data Parallelism**

Let’s revisit Mini-Batch Gradient Descent

**Drawback:** If the dataset has 1 million images, then it will take forever to run the model on such a big dataset

**Solution:** Can we use multiple machines to speedup the training of Deep learning models? (i.e. Utilize Supercomputers to Parallelize)
Need for Communication in Data Parallelism

Problem: Train a single model on whole dataset, not 5 models on different sets of dataset
Data Parallelism

Machine 1

Machine 2

Machine 3

Machine 4

Machine 5

Gradients

Reduced Gradients

MPI AllReduce
Data Parallelism

- **Step 1: Data Propagation**
  - Distribute the data among GPUs

- **Step 2: Forward Backward Pass**
  - Perform forward pass and calculate the prediction
  - Calculate error by comparing prediction with actual output
  - Perform backward pass and calculate gradients

- **Step 3: Gradient Aggregation**
  - Call MPI_Allreduce to reduce the local gradients
  - Update parameters locally using global gradients
Impact of Large Batch Size

Large Batch Size is **bad** for Accuracy

But **good** for speed and scalability

Courtesy: [https://research.fb.com/publications/imagenet1kin1h/](https://research.fb.com/publications/imagenet1kin1h/)

A. A. Awan et al., S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters. PPoPP '17
Synchronous vs. Asynchronous Training

- Epochs per second (EPS)?
  - A variant of images/second
  - Basically, what is the speed of training the model
- Accuracy per Epoch (APE)?
  - E.g. 60% in one full pass over the dataset

- Async $\rightarrow$ Higher EPS but lower APE
- Sync $\rightarrow$ Higher APE but lower EPS

Review and Conclusion

• The concepts and terminologies discussed today will keep coming during the next lectures

• Please clarify any confusions early on

• Future papers and presentations will use these concepts in more complex ways!

• Questions/Comments?
Thank You!

Jain.575@osu.edu

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

The MVAPICH2 Project
http://mvapich.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/