L7: Neural Network 101
— DNN and GNN

Cong (Callie) Hao
callie.hao@ece.gatech.edu

Assistant Professor
ECE, Georgia Institute of Technology

Sharc-lab @ Georgia Tech https://sharclab.ece.gatech.edu/
Outline

• Introduction to Machine Learning
• Neural Networks
  o Multi-Layer Perceptron (MLP)
  o Convolution Neural Network (CNN)
  o Recurrent Neural Network (RNN)
  o Graph Neural Network (GNN)
• Optimization Opportunities
Why Machine Learning? Why Now?

• **Big Data**
  - Large unstructured data sets flood us everyday

• **Data Science**
  - Extract knowledge/insight from data

• **Machine Learning**
  - For specific tasks, resembles human intelligence
Overview Of The Machine Learning Process

• **Training**: Train the desired model, let machine learn intelligence
  - Computationally intensive
  - Huge amount of data, long training time
    - ImageNet: millions of training data. Weeks to train
  - GPUs, ASICs ...

• **Inference**: Infers things about new data based on its training
  - Computationally intensive
  - Real time – Mobile device, IoTs
  - ASICs, GPUs, FPGAs ...
  - Where most acceleration work focuses
    - But we shouldn’t ignore training!
Machine Learning Training is Heavy

Common carbon footprint benchmarks
in lbs of CO2 equivalent

- Roundtrip flight b/w NY and SF (1 passenger) | 1,984
- Human life (avg. 1 year) | 11,023
- American life (avg. 1 year) | 36,156
- US car including fuel (avg. 1 lifetime) | 126,000
- Transformer (213M parameters) w/ neural architecture search | 626,155

Chart: MIT Technology Review • Source: Strubell et al. • Created with Datawrapper
Outline

• Introduction to Machine Learning
• Neural Networks
  o Multi-Layer Perceptron (MLP)
  o Convolution Neural Network (CNN)
  o Recurrent Neural Network (RNN)
  o Graph Neural Network (GNN)
• Optimization Opportunities
Classification

• Identify to which of a set of categories an observation belongs
  o Or rather, which category is the most dominant

• Classification process
  o Provide examples of classes
    - Training data (label)
  o Adjust model weights for different input data
    - Training process
  o Assign all new input data to the model
    - Make prediction for classification
What about non-linear separable?

- Requires non-linearity
  - Called activation function
    - Rectified Linear Unit (ReLU)

- This is a Neural Network!
Basic Perceptron

- Basic Perceptron

\[
\sum_{i=1}^{m} (w_ix_i) + \text{bias} \\
\sum \\
\text{Inputs} \quad \text{Weights} \quad \text{Summation and Bias} \quad \text{Activation} \quad \text{Output}
\]

\[
f(x) = \begin{cases} 
1, & \text{if } \sum wx + b \geq 0 \\
0, & \text{if } \sum wx + b < 0
\end{cases}
\]

https://pub.towardsai.net/perceptron-a-basic-neural-network-model-for-deep-learning-21ae45e3216
Multi-Layer Perceptron (MLP)

• Terminology
  - Input layer, Hidden layer(s), Output layer
  - Deep Neural Network (DNN): more than one hidden layers
  - $X$: Input feature, $W$: Weights (Filter), $b$: bias, $Y$: Label
  - Activation function (at hidden layer)
Forward and Backward Propagation

• **Forward propagation (FP)**
  o Given the parameters and the input data, compute the label
  o Happens in *inference*

• **Backward propagation (BP)**
  o To determine the desired model parameters (weights) during training
  o Given the input data (X) and the target label (T), how to compute the parameters (W and b)
  o Happens in *training*
    - Training needs both FP and BP
• Take a 2-layer MLP as an example

$$g(g(g(x \cdot W_1 + b_1) \cdot W_2 + b_2) + b_3) = y$$

Activation function

Vector-matrix multiplication
Different Type of Neural Networks

• **Real-world ML models are much more complicated than MLP**
  - Large number of layers → Deep Neural Networks (**DNN**)  
  - On image data with weight sharing → Convolution Neural Networks (**CNN**)  
  - With time dependence → Recurrent Neural Networks (**RNN**)  
  - On graph data → Graph Neural Networks (**GNN**)  
  - With complex valued weights and activations → Complex Valued Neural Network (**CVNN**)
• Introduction to Machine Learning
• Neural Networks
  o Multi-Layer Perceptron (MLP)
  o Convolution Neural Network (CNN)
  o Recurrent Neural Network (RNN)
  o Graph Neural Network (GNN)
• Optimization Opportunities
Convoluted Neural Network

• MLP for image?
  o Consider an image of size 250 x 250
  o Vectorize the 2D image to a 1D vector as input feature
  o For each hidden node, it requires 250x250 weights ~ 62,500
  o How about multiple hidden layer? Bigger image?
  o Too many weights, computational and memory expensive

• Can we better utilize the “image” features and also reduce the number of weights?
• Observation: even the input image shifts, the output (classification) stays unchanged.
  o A CAT is still a CAT!
  o If a feature is useful in some locations during training, detectors for that feature will be available in all locations during testing.

• Translation Invariant: if a detector (filter) learnt a useful feature to detect 'CAT', it will capture 'CAT' wherever its location is in an image at testing time.
Basic Structure of CNN

INPUT (28 x 28 x 1)

Conv_1
Convolution (5 x 5) kernel valid padding

Max-Pooling (2 x 2)

n1 channels (24 x 24 x n1)

Conv_2
Convolution (5 x 5) kernel valid padding

Max-Pooling (2 x 2)

n1 channels (12 x 12 x n1) → n2 channels (8 x 8 x n2)

fc_3
Fully-Connected Neural Network ReLU activation

fc_4
Fully-Connected Neural Network

(with dropout)

0
1
2
...
9

OUTPUT

n3 units

CNN Terminologies

Filter (Kernel) 3x3, 5x5, etc.
Receptive Field

Feature Maps
(Activations, Tensors)

Fully-Connected (FC) Layers

INPUT

CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING

FEATURE LEARNING

FLATTEN
FULLY CONNECTED
SOFTMAX

CLASSIFICATION
A Closer Look at CNN Computation

\[ \sum \left( a \times 1 + b \times 2 + c \times 3 
+ f \times 4 + g \times 5 + h \times 6 
+ k \times 7 + l \times 8 + m \times 9 \right) \]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>J</td>
</tr>
<tr>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>o</td>
</tr>
<tr>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
</tr>
<tr>
<td>u</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
</tr>
</tbody>
</table>

\[
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array}
\]
A Closer Look at CNN Computation

\[
\sum \ b \times 1 + c \times 2 + d \times 3 \\
+ g \times 4 + h \times 5 + i \times 6 \\
+ l \times 7 + m \times 8 + n \times 9
\]
A Closer Look at CNN Computation

\[
\sum m \times 1 + n \times 2 + o \times 3 + r \times 4 + s \times 5 + t \times 6 + w \times 7 + x \times 8 + y \times 9
\]

One input channel  One kernel channel  One output channel
CNN Computation with Multiple Channels

3 input channel

3 kernel channel

1 output channel

\[ \sum (\sum \sum \sum ) \]
CNN Computation with Multiple Channels

for(int h = 0; h < 5; h++) {
    for(int w = 0; w < 5; w++) {
        float sum = 0;
        for(int ci = 0; ci < 3; ci++) {
            for(int m = 0; m < 3; m++) {
                for(int n = 0; n < 3; n++) {
                    sum += A[ci][h+m][w+n] * W[ci][m][n];
                }
            }
            B[h][w] = sum;
        }
    }
}
CNN Computation with Multiple Channels

3 input channel

Kernel 1

Output Channel 1

Kernel 2

Output Channel 2
• Introduction to Machine Learning
• Neural Networks
  o Multi-Layer Perceptron (MLP)
  o Convolution Neural Network (CNN)
  o Recurrent Neural Network (RNN)
  o Graph Neural Network (GNN)
• Optimization Opportunities
Recurrent Neural Network

- So far, the neural network only considers the **current input**. What about **time**?
- We assume all inputs are independent, this may not be good assumption for certain tasks
  - Time sequence, for example, speech and handwriting
- **Input is a sequence, how to learn the sequential information?**
  - Make the neural network considers the previous output together with the current input

---

**This is a sentence**

 Dies ist ein Satz

**Callie is teaching her class**

 ???
Recurrent Neural Network

• A class of neural network with backward edges
  o Can learn sequential information
  o Indirectly factor in all previous inputs
  o Vanishing effect: The input long time ago has very little effect
  o Only short-term memory, like human (for example myself)

![Diagram of Recurrent Neural Network](https://wiki.tum.de/display/lfdv/Recurrent+Neural+Networks+-+Combination+of+RNN+and+CNN)
• **How about long-term memory?**
• **Can a neural network learn the pattern of both type of memories?**
  o Long + Short

• **Instead of only using the current input and previous output, also add a memory component**
  o If the memory brings information of long time ago, it will directly affect the current output
Long Short-Term Memory (LSTM)

- Long-term Memory
- Previous Output (short-term)

\[ h_{t+1} \rightarrow A \rightarrow h_t \rightarrow A \rightarrow h_{t+1} \]

\[ X_{t-1} \rightarrow A \rightarrow X_t \rightarrow A \rightarrow X_{t+1} \]
Outline

• Introduction to Machine Learning
• Neural Networks
  o Multi-Layer Perceptron (MLP)
  o Convolution Neural Network (CNN)
  o Recurrent Neural Network (RNN)
  o Graph Neural Network (GNN)
• Optimization Opportunities
Graph Neural Network (GNN)

- Traditional neural networks are designed for simple sequences & grids

[Slide credit: http://web.stanford.edu/class/cs224w]
• **Reality:** A lot of real-world data does not “live” on grids
  - Arbitrary size and complex topological structure
  - No fixed node ordering or reference point
  - Often dynamic and have multimodal features
Graph Neural Network (GNN)

Main Idea: Pass massages between pairs of nodes and agglomerate

[Slide credit: Structured deep models: Deep learning on graphs and beyond]
How is GNN Computed?

- **Key idea:** Generate node embeddings based on local network neighborhoods
  - **Node embedding:** a vector to represent the node features

[Slide credit: http://web.stanford.edu/class/cs224w]
Outline

• Introduction to Machine Learning

• Neural Networks
  o Multi-Layer Perceptron (MLP)
  o Convolution Neural Network (CNN)
  o Recurrent Neural Network (RNN)
  o Graph Neural Network (GNN)

• Optimization Opportunities
Opportunities for ML Acceleration

• **Four basic principles:**
  o Specialization
  o Parallelism
  o Memory localization and optimization
  o Reducing overhead

• **What else?**
  o Huge model size
  o And it’s getting crazier!
Huge ML Model Size

2018 (left) through 2019 (right)

94M  110M  340M  465M  665M  330M  1.5B  340M  355M  1.5B  8.3B  1.5B  11B  2.6B  17B  9.4B  175B

ELMo  GPT  BERT-Large  Transformer  XLM  MT-DNN  GPT-2  XLNet  RoBERTa  CTRL  MegatronLM  Grover-Mega  T5-11B  Meena  Turing-NLG  DistilBERT  BST 9.4B  GPT-3

https://moon-walker.medium.com/ai-service%EB%A5%BC-%EC%9C%84%ED%95%9C-%ED%98%84%EC%8B%A4%EC%A0%81%EC%9D%BB-%EC%A0%91%EA%B7%BC-%EB%B0%A9%EB%B2%95-3-massive-ai-inference-94f75b0fc64f
Model Compression

https://neuralmagic.com/blog/pruning-overview/
Data Quantization

- From 32-bit floating point (FP32) to 8-bit integer (INT8)

Intuition: 1.0001 and 0.9992 (in most cases) make no difference in ML
Data Quantization

Floating point $x_f$

Signed Int8 $x_q$

Rounding

Outlier Clipping

• Introduction to ML
• MLP, CNN, RNN, GNN
• ML Acceleration Opportunities

• Lab 1 Review
  o Will publish a few example solutions with well-written reports
Lab 1 Review

Paralyze Computation?  Overlap Communication?

Image Credit: Tejas S Shah
Lab 1 Review – Paralyze Computation

\[ x = \text{Gigantic adder tree for partial sum accumulation} \]
Lab 1 Review – Paralyze Computation

Alternative: Parallelism at another dimension
Loop Reorder to avoid accumulation

Image Credit: Tejas S Shah
Loop Reorder to avoid accumulation

```java
for (int i = 0; i < M; i++) {
    for (int j = 0; j < K; j++) {
        for (int p = 0; p < N; p++) {
            MatC[i][j] += MatA[i][p] * MatB[p][j];
        }
    }
}
```

Code Credit: Tejas S Shah
To completely avoid long partial sum accumulation + another dimension
Lab1 Review – Overlap Communication

Before

```java
for(int i = 0; i < M; i++) {
    for(int j = 0; j < N; j++) {
        MatA[i][j] = MatA_DRAM[i][j];
    }
}
for(int i = 0; i < N; i++) {
    for(int j = 0; j < K; j++) {
        MatB[i][j] = MatB_DRAM[i][j];
    }
}
for(int i = 0; i < M; i++) {
    for(int j = 0; j < K; j++) {
        MatC[i][j] = 0;
    }
}
```

After

```java
int maxMN = M < N? N : M;
int maxNK = N < K? K : N;
for(int i = 0; i < maxMN; i++) {
    for(int j = 0; j < maxNK; j++) {
        if (i < M && j < N)
            MatA[i][j] = MatA_DRAM[i][j];
        if (i < N && j < K)
            MatB[i][j] = MatB_DRAM[i][j];
        if (i < M && j < K)
            MatC[i][j] = 0;
    }
}
```
Lab1 Review – Overlap Communication

```c
void matrix_mul( FIX_TYPE MatA_DRAM[M][N], FIX_TYPE MatB_DRAM[N][K], FIX_TYPE MatC_DRAM[M][K])
{
#pragma HLS interface m_axi depth=100000 port=MatA_DRAM offset=slave bundle=memA
#pragma HLS interface m_axi depth=100000 port=MatB_DRAM offset=slave bundle=memB
#pragma HLS interface m_axi depth=100000 port=MatC_DRAM offset=slave bundle=memC
#pragma HLS interface s_axilite port=return
```

**Note** A single port (e.g., memA) cannot be partitioned or accessed simultaneously

- Coming from different DRAM buses
- So MatA, MatB, and MatC can read/write simultaneously
Overlapping Read, Write, and Compute

```c
colC: for (int j=0; j<K; j++){
    elemC: for (int p=0; p<N; p++) {
        MatB[p][j] = MatB_DRAM[p][j];
        if (j>0 && p<M)
            MatC_DRAM[p][j-1] = MatC[p][j-1];
        MatC[i][j] += MatA[i][p] * MatB[p][j];
    }
}
storeC: for (int i = 0; i < M; i++) {
        MatC_DRAM[i][K-1] = MatC[i][K-1];
}
```

- Load only when necessary
- Can even use a single register

Code Credit: Ashwin Bhat
• How to fully utilize the bus bandwidth? 32 bit → 512 bit?
  o Data packing
• How to reduce the memory? Do we really need to store all data on-chip?
  o Ping-pong buffer
• Is 32-bit really necessary?
  o Low precision

• Future lectures