Large Neural Network and Its Parallelization

Speaker: Jiayi Xu
11/01/2019
Outline

• Two papers

1. **Outrageously Large Neural Networks**: The Sparsely-Gated Mixture-of-Experts Layer

2. **Beyond Data and Model Parallelism for Deep Neural Networks**
   • Flex Flow
Large Neural Networks

• Why do we need large (e.g., deep) neural networks? Benefits?
• Challenge for large-scale data: underfitting
• Larger neural networks => Learn more knowledge from data
  • Generate more types of features
  • Approximate more complex function
Large Neural Networks

![Graph showing performance vs. amount of data for different types of neural networks.](image)

More data + Bigger model.

Courtesy of video “How Scale is Enabling Deep Learning”
A Challenge to Large Neural Networks: High computational cost

- Each input sample => entire network is evaluated
A Challenge to Large Neural Networks: High computational cost

• One solution: **conditional computation**
  • Utilize parts of network per sample wise
  • Each input sample goes different path of neural network for evaluation

• Benefits
  • Computational cost is reduced
  • Entire neural network can have large capacity

• Given a sample => which part of network should be evaluated/trained?
  • Gating mechanism
Paper 8.b: Outrageously Large Neural Networks

• Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer

• Authors: Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, Jeff Dean
  • Google Brain

• General idea
  • Gating Network
Overview of Sparsely Gated Mixture of Experts (MoE) Layer
Structure of Mixture of Experts (MoE)

- $n$ expert networks: $E_1(x), \ldots, E_n(x)$
- Gating network outputs sparse $n$ dimensional vector
  - Most of dimensions are zeros
  - If $G(x)_i = 0$, $E_i(x)$ is not evaluated
- Output of the MoE layer is passed through a sigmoid function + dropout

$$y = \sum_{i=1}^{n} G(x)_i E_i(x)$$
Hierarchical Mixture of Experts (MoE)

- If the number of experts are high
- Hierarchical MoE
- A primary gating mechanism chooses a sparse combination of experts
- Each of which is itself a secondary mixture of MoE with its own gating mechanism
Mixture of Experts: Basics

• Expert: a simple feed-forward network
• Key idea: Specialization
  • Each expert specializes in a subset of the training cases
  • They do not learn on cases for which they are not picked. So they can ignore stuff they are not good at modeling.
  • Make each expert focus on predicting the right answer for the cases where it is already doing better than the other experts
Mixture of Experts: A Spectrum of Models

• Very local models
  • E.g., nearest neighbors
    • Prediction
  • Benefit: very fast to fit
    • Just store training cases
  • Drawback: Local smoothing would usually improve prediction

• Fully global models
  • E.g., a polynomial
  • Benefit
    • Prediction is fast
  • Drawbacks
    • Maybe slow to fit, if many parameters
    • Unstable, parameters depend on all the data
Mixture of Experts: A Spectrum of Models

• Intermediate complexity
  • Very local models <-> Mixture of experts <-> Fully global models

• Good if the dataset contains several different regimes which have different relationships between input and output
  • E.g., financial data which depends on the state of the economy
  • Each expert specializes in a “regime”
Gating Mechanism

- **Softmax Gating**
  - An obvious, effective, and naive sparse gating mechanism
- Take the softmax of the Input multiplied by the weight matrix $W_g$
- However, not sparse

\[
G_\sigma(x) = \text{Softmax}(x \cdot W_g)
\]
Gating Mechanism

- **Top-K Gating**
  - Keep only the top $k$ values
  - Setting the rest to $-\infty$
    - Causes the corresponding gate values to equal 0 after softmax
- **Sparse**
  - If $G(x)_i = 0$, $E_i(x)$ is not evaluated

\[
G(x) = \text{Softmax}(\text{KeepTopK}(H(x), k))
\]

\[
\text{KeepTopK}(v, k)_i = \begin{cases} 
  v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\
  -\infty & \text{otherwise.}
\end{cases}
\]
Imbalanced Expert Utilization

• Challenge: Possible that few experts are trained more rapidly as compared to the other experts

• Strong expert
  • => Higher probability for selection
  • => More samples for training
  • => Stronger

• Weak expert
  • => Lower probability for selection
  • => Less samples for training
  • => Weaker
Imbalanced Expert Utilization

- Noisy Top-K Gating
- Noise term
  - The Gaussian noise term helps with load balancing
    - Breaks the imbalance
  - The amount of noise per component is controlled by a second trainable weight matrix $W_{noise}$

\[
G(x) = \text{Softmax}(\text{KeepTopK}(H(x), k))
\]
\[
H(x)_i = (x \cdot W_g)_i + \text{StandardNormal()} \cdot \text{Softplus}((x \cdot W_{noise})_i)
\]
Imbalanced Expert Utilization

• Importance Loss

• Intuition for the loss:
  • For a batch of training examples, each expert has similar sum of gate values (i.e., selection probabilities)

• Importance of an expert
  • batchwise sum of the gate values

\[
\text{Importance}(X) = \sum_{x \in X} G(x)
\]
Imbalanced Expert Utilization

- Importance loss
  - The square of the Coefficient of Variation (CV) of the set of importance values, multiplied by a hand-tuned scaling factor $w_{importance}$
  - Encourage all experts to have equal importance

\[
\text{Importance}(X) = \sum_{x \in X} G(x)
\]

\[
L_{importance}(X) = w_{importance} \cdot CV(\text{Importance}(X))^2
\]

\[
\text{CV(a set of values)} = \frac{\sigma}{\mu}
\]
Application: Language Modeling

• Language modeling task
  • Predicting the next word or character in a document

• Dataset: 1 billion word language modeling benchmark
  • Shuffled unique sentences from news articles
  • Approximately 829 million words, with a vocabulary of 793,471 words
Application: Language Modeling

- Neural Network Architecture:
  - A series of LSTM layers
  - Adding a MoE layer in between each pair of LSTMs
    - A set of small feed-forward networks (the experts)
    - An even simpler gating network that chooses which expert to use
Application: Language Modeling

- Experiment 1:
- Perplexity <-> Model parameters
Application: Language Modeling

- Experiment 2:
- Perplexity $\leftrightarrow$ Computational budget
Application: Language Modeling

- Experiment 3:
  - Computational efficiency
    - Teraflops/GPU
      - Number of floating point operations required to process one training batch divided by the observed time and the number of GPUs in the cluster
      - NVIDIA theoretical maximum of TFLOPs/GPU: 4.29

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Perplexity 10 epochs</th>
<th>Test Perplexity 100 epochs</th>
<th>#Parameters excluding embedding and softmax layers</th>
<th>ops/timestep</th>
<th>Training Time 10 epochs</th>
<th>TFLOPS/GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Published Results</td>
<td>34.7</td>
<td>30.6</td>
<td>151 million</td>
<td>151 million</td>
<td>59 hours, 32 k40s</td>
<td>1.09</td>
</tr>
<tr>
<td>Low-Budget MoE Model</td>
<td>34.1</td>
<td>30.6</td>
<td>4303 million</td>
<td>8.9 million</td>
<td>15 hours, 16 k40s</td>
<td>0.74</td>
</tr>
<tr>
<td>Medium-Budget MoE Model</td>
<td>31.3</td>
<td></td>
<td>4313 million</td>
<td>33.8 million</td>
<td>17 hours, 32 k40s</td>
<td>1.22</td>
</tr>
<tr>
<td>High-Budget MoE Model</td>
<td><strong>28.0</strong></td>
<td></td>
<td>4371 million</td>
<td>142.7 million</td>
<td>47 hours, 32 k40s</td>
<td><strong>1.56</strong></td>
</tr>
</tbody>
</table>
Application: Language Modeling

- Experiment 4:
- Perplexity $\leftrightarrow$ Larger Data
- Dataset
  - 100 billion word Google News Corpus
Application: Machine Translation

- **Experiment 1:**
  - En -> De
  - More parameters, similar training time, and better perplexity and BLEU

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Perplexity</th>
<th>Test BLEU</th>
<th>ops/timestep</th>
<th>Total #Parameters</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoE with 2048 Experts</td>
<td><strong>4.64</strong></td>
<td><strong>26.03</strong></td>
<td>85M</td>
<td>8.7B</td>
<td>1 day/64 k40s</td>
</tr>
<tr>
<td>GNMT (Wu et al., 2016)</td>
<td>5.25</td>
<td>24.91</td>
<td>214M</td>
<td>278M</td>
<td>1 day/96 k80s</td>
</tr>
<tr>
<td>GNMT +RL (Wu et al., 2016)</td>
<td>8.08</td>
<td>24.66</td>
<td>214M</td>
<td>278M</td>
<td>1 day/96 k80s</td>
</tr>
<tr>
<td>PBMT (Durrani et al., 2014)</td>
<td></td>
<td>20.7</td>
<td>20.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeepAtt (Zhou et al., 2016)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Results on WMT’14 En → De newstest2014 (bold values represent best results).
Application: Machine Translation

- **Experiment 2:**
  - En -> Fr

- More parameters, similar training time, and better perplexity and BLEU

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Perplexity</th>
<th>Test BLEU</th>
<th>ops/timenstep</th>
<th>Total #Parameters</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoE with 2048 Experts</td>
<td>2.69</td>
<td>40.35</td>
<td>85M</td>
<td>8.7B</td>
<td>3 days/64 k40s</td>
</tr>
<tr>
<td>MoE with 2048 Experts (longer training)</td>
<td><strong>2.63</strong></td>
<td><strong>40.56</strong></td>
<td>85M</td>
<td>8.7B</td>
<td>6 days/64 k40s</td>
</tr>
<tr>
<td>GNMT (Wu et al., 2016)</td>
<td>2.79</td>
<td>39.92</td>
<td>214M</td>
<td>278M</td>
<td>6 days/96 k80s</td>
</tr>
<tr>
<td>GNMT+RL (Wu et al., 2016)</td>
<td>2.96</td>
<td>39.22</td>
<td>214M</td>
<td>278M</td>
<td>6 days/96 k80s</td>
</tr>
<tr>
<td>PBMT (Durrani et al., 2014)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSTM (6-layer) (Luong et al., 2015b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSTM (6-layer+PosUnk) (Luong et al., 2015b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeepAtt (Zhou et al., 2016)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeepAtt+PosUnk (Zhou et al., 2016)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Application: Machine Translation

- Observation: different experts tend to become highly specialized based on syntax and semantics

- A few experts in the encoder of En→Fr translation
  - For each expert $i$, sort the inputs in a training batch in decreasing order of $G(x)_i$
  - Show the words surrounding the corresponding positions in the input sentences

<table>
<thead>
<tr>
<th>Expert 381</th>
<th>Expert 752</th>
<th>Expert 2004</th>
</tr>
</thead>
<tbody>
<tr>
<td>... with researchers, ...</td>
<td>... plays a core ...</td>
<td>... with rapidly growing ...</td>
</tr>
<tr>
<td>... to innovation</td>
<td>... plays a critical ...</td>
<td>... under static conditions ...</td>
</tr>
<tr>
<td>... tics researchers</td>
<td>... provides a legislative ...</td>
<td>... to swiftly ...</td>
</tr>
<tr>
<td>... the generation of ...</td>
<td>... play a leading ...</td>
<td>... to drastically ...</td>
</tr>
<tr>
<td>... technology innovations is ...</td>
<td>... assume a leadership ...</td>
<td>... the rapid and ...</td>
</tr>
<tr>
<td>... technological innovations, ...</td>
<td>... plays a central ...</td>
<td>... the fast est ...</td>
</tr>
<tr>
<td>... support innovation throughout ...</td>
<td>... taken a leading ...</td>
<td>... the Quick Method ...</td>
</tr>
<tr>
<td>... role innovation will ...</td>
<td>... established a reconciliation</td>
<td>... rec current ) ...</td>
</tr>
<tr>
<td>... research scientists</td>
<td>... played a vital ...</td>
<td>... provides quick access ...</td>
</tr>
<tr>
<td>... promoting innovation where ...</td>
<td>... have a central ...</td>
<td>... of volatile organic ...</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Conclusion

• The first paper to demonstrate major improvements from conditional computation in deep networks
• Achieving better than state-of-the-art accuracy
• Much more parameters yet similar training time
Paper 8.A: Flex Flow

- Beyond Data and Model Parallelism for Deep Neural Networks
- Authors: Zhihao Jia, Matei Zaharia, Alex Aiken
  - Stanford University
- General idea
  - Given a neural network, generate an optimized parallelism scheme automatically from a search space
Why parallelizing deep neural networks?

- DNN models have gotten progressively larger and more computationally expensive to train
  - Distributed computing is inevitable
- Iterating quickly is important for research progress
Common Parallelization Strategies

- Data parallelism
- Model parallelism
Common Parallelization Strategies

- Data parallelism
- Model parallelism
SOAP search space

• Four parallelizable dimensions
  • Sample
  • Operator
  • Attribute
  • Parameter
SOAP search space
Sample

• Sample
  • Partition training samples
  • Assign different training samples to different devices
• E.g., Data Parallelism
SOAP search space

Attribute

• Attribute
  • Partition attributes (e.g., different pixels) in a sample to different devices
• E.g., Spatial Parallelism

![Diagram showing partitioning of tasks to different processes](image-url)
SOAP search space
Operator

- Operator
  - Partition DNN operators
  - Assign different convolution operators to different devices
- E.g., Model Parallelism
SOAP search space
Parameter

- Parameter
  - Partition parameters in an operator
- E.g., parallelization configuration for a matrix multiplication operator.
Parallelization Strategies

- Strategies
  - Data parallelism
  - Model parallelism
  - Hybrid parallelism

- How to find an optimized strategy in the SOAP search space?
Overview of Flex Flow

• Flex Flow
  • Find an optimized parallelism strategy in the SOAP search space automatically

• Input: two graphs
  • Operator graph
    • Describe computation in the neural network
  • Device topology
    • Describes the set of available devices for training
    • Interconnections between devices
      • Bandwidth
Flex Flow
Overview of Execution Optimizer
Flex Flow Cost Model

• Assume each operator has a similar execution time for different samples

• Measure the execution time of each distinct operator once by given an architecture
  • Use this to estimate execution time for candidate strategies
Flex Flow
Markov Chain Monte Carlo (MCMC) Search Algorithm

- A current parallelism strategy $S$
- Randomly propose a new strategy $S^*$ based on $S$
- $S^*$ is accepted and becomes the new current strategy with the following probability

$$\alpha(S^*|S) = \min\left(1, \exp\left(\beta \cdot (\text{cost}(S) - \text{cost}(S^*))\right)\right)$$

- Preferring to move towards lower cost whenever that is readily available, but can also escape local minima
• \( S \rightarrow S^* \): Delta simulation algorithm

• Two steps:
  • An operator in the current parallelization strategy is selected at random
  • Its parallelization configuration is replaced by a random configuration

• Search algorithm terminates when
  • (1) the search time budget for current initial strategy is exhausted;
  • or
  • (2) the search procedure cannot further improve the best discovered strategy for half of the search time.
Evaluation of Flex Flow: Setup

• Two clusters
  1. 4 nodes
     • Each node
       • two Intel 10-core E5-2600 CPUs
       • 256GB main memory
       • four NVIDIA Tesla P100 GPUs
     • GPUs on the same node are connected by NVLink
     • Nodes are connected over 100GB/s EDR Infiniband
  2. 16 nodes
     • Each node
       • two Intel 10-core E5-2680 CPUs
       • 256GB main memory
       • four NVIDIA Tesla K80 GPUs
     • GPUs are connected by a separate PCI-e switch, and all GPUs are connected to CPUs through a shared PCI-e switch
     • Nodes are connected over 56 GB/s EDR Infiniband
Evaluation of Flex Flow: Setup

- Six DNN benchmarks
  - Three CNNs for image classification
    - AlexNet
    - Inception_V3
    - ResNet-101
  - Three RNNs for NLP tasks
    - RNNTC
    - RNNLM
    - NMT
- Synchronous training
- Use data parallelism and a randomly generated strategy as the initial candidates for the search
Evaluation of Flex Flow: End-to-end Performance

• Accuracy

<table>
<thead>
<tr>
<th>DNN</th>
<th>Description</th>
<th>Dataset</th>
<th>Reported Acc.</th>
<th>Our Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>A 12-layer CNN</td>
<td>Synthetic data</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Inception-v3</td>
<td>A 102-layer CNN with Inception modules (Szegedy et al., 2014)</td>
<td>ImageNet</td>
<td>78.0%</td>
<td>78.0%</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>A 101-layer residual CNN with shortcut connections</td>
<td>ImageNet</td>
<td>76.4%</td>
<td>76.5%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DNN</th>
<th>Description</th>
<th>Dataset</th>
<th>Reported Acc.</th>
<th>Our Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNNTC</td>
<td>4 recurrent layers followed by a softmax layer</td>
<td>Movie Reviews (Movies)</td>
<td>79.8%</td>
<td>80.3%</td>
</tr>
<tr>
<td>RNNLM</td>
<td>2 recurrent layers followed by a softmax layer</td>
<td>Penn Treebank (Marcus et al.)</td>
<td>78.4%</td>
<td>76.1%</td>
</tr>
<tr>
<td>NMT</td>
<td>4 recurrent layers followed by an attention and a softmax layer</td>
<td>WMT English-German (WMT)</td>
<td>19.67%</td>
<td>19.85%</td>
</tr>
</tbody>
</table>

* top-1 accuracy for single crop on the validation dataset (higher is better).
* word-level test perplexities on the Penn Treebank dataset (lower is better).
* BLEU scores (Papineni et al., 2002) on the test dataset (higher is better).
Evaluation of Flex Flow: End-to-end Performance

• Training Loss
  • Training curves of Inception-v3 in TensorFlow and FlexFlow
  • The model is trained on 16 P100 GPUs (4 nodes)
Evaluation of Flex Flow: Per-iteration training performance

- Num Samples/Second/GPU
- Six DNN benchmarks
Evaluation of Flex Flow: Per-iteration training performance

- Samples/Second
- Compare with other automated frameworks
Evaluation of Flex Flow: Execution Simulator Accuracy

- Comparison between the simulated and actual execution time for different DNNs and device topologies.
Evaluation of Flex Flow: Search Algorithm

• Compare the best discovered strategies by Flex Flow with the global optimal strategies for small executions
• Test modes
  • CNNs: LeNet (6-layer CNN) and a variant of RNNLM
• Use Depth-first search + A* to find global optimal strategies
• Flex Flow to obtain the global optimal strategy
  • LeNet: 0.8 hours
  • RNNLM: 18 hours
Conclusion

• This paper presents FlexFlow, a deep learning system that automatically finds efficient parallelization strategies in the SOAP search space for DNN training.

• FlexFlow uses a guided randomized search procedure to explore the space and includes an execution simulator that is an efficient and accurate predictor of DNN performance.

• Evaluate FlexFlow with six real-world DNN benchmarks on two GPU clusters and show FlexFlow significantly outperforms state-of-the-art parallelization approaches
The End.

• Q&A