Enabling Large Dynamic Neural Network Training with Learning-based Memory Management and Tiered Memory

Jie Ren§, Dong Xu*, Shuangyan Yang*, Jiacheng Zhao†, Zhicheng Li† Christian Navasca‡, Chenxi Wang†, Harry Xu‡, Dong Li*
*University of California Merced, †University of Chinese Academy of Sciences
‡University of California Los Angeles §College of William and Mary

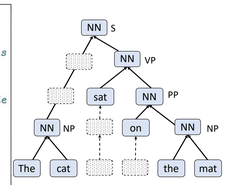
Introduction

- Dynamic structure of Dynamic Neural Networks (DyNNs) makes them very memory-hungry.
  - ❖ **Tensor offloading to CPU memory** is an effective way to train large AI models.
- Computation in dynamic neural network (DyNN) depends on the input. Different inputs will activate different model components.
  - ❖ Dynamism creates challenges to decide when to do tensor offloading

➢ We present DyNN-Offload, a memory management system to address the GPU memory capacity problem during DyNN training.

```
def Node():
  for i in range(N-1):
    p_i = tanh(W_1(c_i, c_{i+1}) + b)
    # use two candidate children's
    # representations c_i and c_{i+1}
    # to compute parent
  score_i = U^T p_i
    # the decision score of a node
  s(x, y) = Σ_{n∈nodes(y)} score_n
    # the score of a tree,
    # x is a sentence,
    # y is a parsing tree
  select max s(x_i, y)
    # select the nodes which
    # generates  the highest
    # score for the tree
```

(a)



Input Sentence: The cat sat on the mat

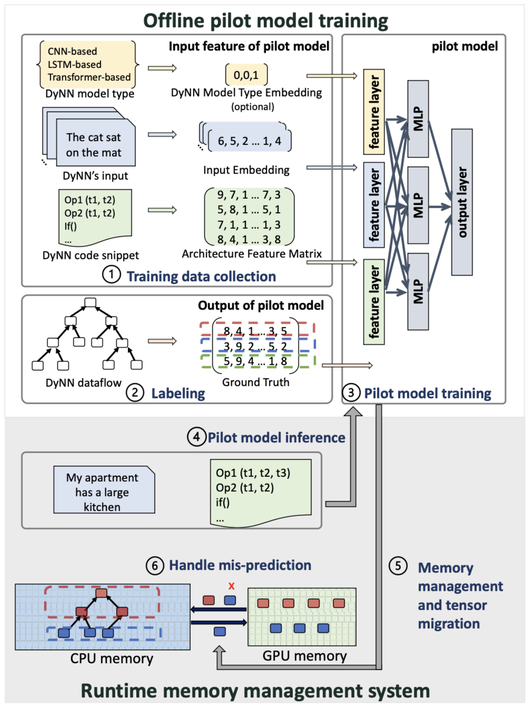
(b)

Design Philosophy

❖ The key to DyNN-Offload is ***learning for learning***, which involves using knowledge proactively gained from other input problems and DyNNs, instead of relying on profiling that lacks the flexibility to handle the dynamism of DyNN training.

Methodology

❖ Use a pilot model to predict **operator execution order** and **proactively prefetch tensors** from CPU to GPU memory to maximize overlap between computation and communication.



❖ DyNN-Offload Components

➢ **Pilot model.** A light neural network that takes (1) the input sample to the DyNN after embedding, (2) DyNN's static architecture, and (3) the basic NN type of DyNN as the input and output the execution plan.

➢ **Runtime system.** DyNN-Offload manages GPU memory for tensor migration and DyNN training based on double buffering.

➢ **Training system for the pilot model.** Collected execution traces will be fed to the pilot model.

❖ Challenge1: Tradeoff between pilot model complexity (model accuracy) and usefulness(model inference overhead)

✓ Solution 1: Use idiom based DyNN representation to capture salient properties of diverse DyNN architectures concisely.

❖ Challenge2: Effective use of the pilot model: determining *when* and *how* to query the pilot model

✓ Solution 2: Pilot model inference and data prefetch with execution blocks instead of tensor objects.

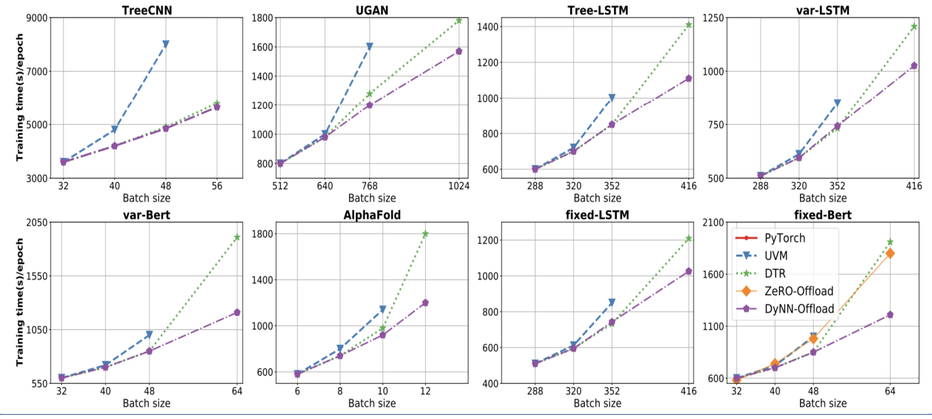
Evaluation Results

❖ **DyNN-Offload enables larger model and larger batch size trained with limited memory**

- DyNN-Offload allows for 8x and 6.3x larger deep and wide transformer models, respectively, compared to PyTorch-only solutions.
- With UVM, DTR[2] and DyNN-Offload, the largest batch size is **1.17×, 1.7×, and 3.6×** larger, compared with PyTorch-only solution.

❖ **DyNN-Offload effectively improves training throughput compared to other memory-saving techniques.**

- UVM perform worst because the on-demand transfer.
- DyNN-Offload outperforms DTR (a state-of-the-art to save GPU memory by recomputation) by 35% on average.
- For static NN, DyNN-Offload outperforms Microsoft ZeRO-Offload by 33% on aveage with three batch sizes, because of optimal partition decided by DyNN-Offload.



Conclusions

➢ DyNN-Offload is a memory management system enabling large DyNN training with limited GPU memory.

➢ Unlike the traditional profiling-based approach that lacks abilities to react to dynamism in DyNN, DyNN-Offload uses a learned approach to resolve dynamism and predict access order of tensors.

➢ DyNN-Offload shows that building a fast, accurate, and live ML model to guide performance optimization and analysis for DyNNs is feasible.

[1] M. Kirisame, S. Lyubomirsky, A. Haan, J. Brennan, M. He, J. Roesch, T. Chen, and Z. Tatlock, "Dynamic tensor rematerialization," 2021