HPML Project

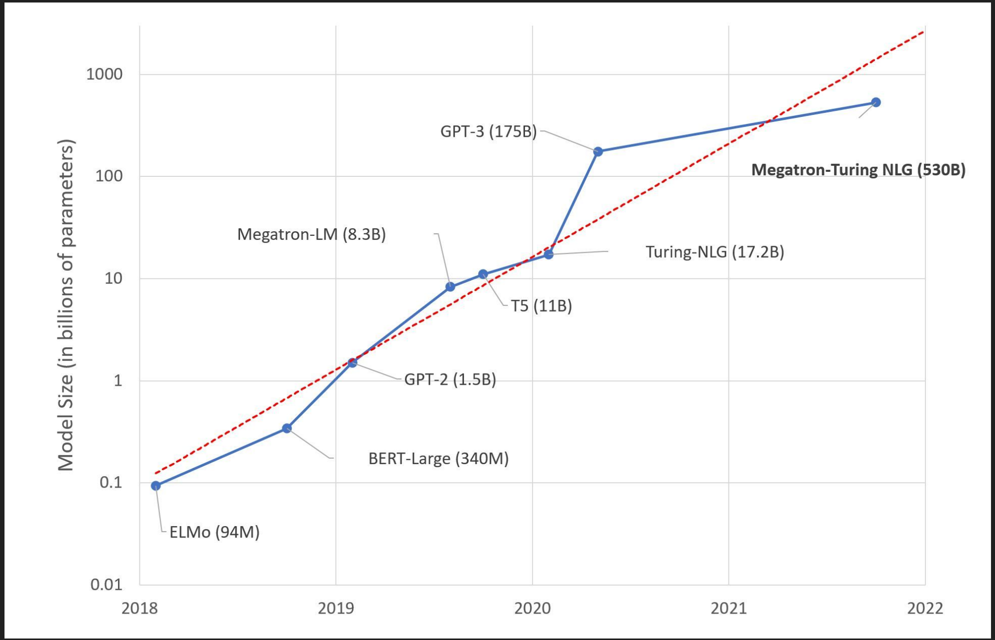
Austin Xiong (ax2156)
Capucine Leroux (cpl2146)

Executive Summary

- Goal: Optimize model performance on two different types of applications:
    - Computer Vision → VGGNet
    - NLP → LSTM model
- Approach: Use the methodology seen in class to optimize performance
    - Measure
    - Analyze
    - Optimize
- Benefits of our solution:
    1. VGGNet model (CV) → better training and inference time, small model size
    2. LSTM model (NLP) → better time and memory performance

Problem Motivation

- Reduce training time
  - Models parameters increasing
- Reduce inference time
  - Often exceeds training costs
- Reduce model size
  - Move to embedded



Background Work

VGGNet:

- VGGNet from "Very Deep Convolutional Networks for Large-Scale Image Recognition"
- VGGNet code adapted from https://github.com/kuangliu/pytorch-cifar

LSTM:

- LSTM for sentiment analysis: https://www.kaggle.com/code/affand20/imdb-with-pytorch
- IMDB dataset: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

Common tools:

- PyTorch tools (e.g. profiler, scheduler, quantization, pruning)

Technical Challenges

- Hardware - need a GPU to train deep learning models
  - GCP/AWS
- Some parts of PyTorch APIs do not support CUDA
  - Cannot run quantized models on GPU
- First NLP model used string type inputs not adapted to most techniques seen in class, changed model to use numerical inputs

Approach

VGGNet:

- GPU enabled VM instance using GCP
    - Nvidia P100
- Used PyTorch profiler to analyze CPU/GPU times and memory consumption
- Data Loader optimizations - varying number of workers
    - Default 0 - means main process must train and load
- Quantization
- TorchScript

Approach

LSTM:

- Google Colab platform
  - GPU V100, High-RAM
- Used PyTorch profiler to analyze CPU/GPU times and memory consumption
- Data Loader and Batch Size optimizations
  - Varying batch size and number of workers
- Pruning optimizations
  - Varying pruning ratio and pruning techniques (structured vs unstructured)

Implementation Details

- VGGNET
  - Vary Dataloaders
  - PyTorch quantization API
  - TorchScript tracing function
- LSTM
  - Profile initial model → identify bottlenecks using **pytorch profiler**
  - Vary Dataloaders' **num_workers** and **batch_size** → monitor time using **time.perf_counter()**
  - PyTorch pruning API → test **l2-structured** and **unstructured**, test different pruning ratios

Demo/Experiment Design Flow

In the experimental evaluation, we will show:

For the VGGNet model:

- Initial profiling
- Experimental findings of optimal number of dataloader workers
- Utilization of quantization and torchscripting

For the LSTM model:

- Initial profiling analysis and identification of the bottleneck
- Experimental findings of optimal batch size and number of workers
- Experimentations around pruning techniques and final performance

Experimental Evaluation

- Quantization float-32 to int-8, smaller model size, slightly greater accuracy

Profiler for memory consumption

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	CPU Mem	Self CPU Mem	CUDA Mem	Self CUDA Mem	# of Calls
aten::empty	0.48%	500.000us	5.63%	5.879ms	136.721us	24.02 Kb	24.02 Kb	98.04 Mb	98.04 Mb	43
aten::randn	0.06%	59.000us	0.16%	168.000us	168.000us	24.00 Kb	0 b	0 b	0 b	1
aten::normal_	0.07%	72.000us	0.07%	72.000us	72.000us	0 b	0 b	0 b	0 b	1
aten::zeros	0.01%	13.000us	0.04%	42.000us	42.000us	4 b	0 b	0 b	0 b	1
aten::zero_	0.00%	3.000us	0.00%	3.000us	3.000us	0 b	0 b	0 b	0 b	1
Scatter	1.67%	1.743ms	3.24%	3.384ms	3.384ms	0 b	0 b	24.00 Kb	0 b	1
cudaStreamCreateWithPriority	1.16%	1.213ms	1.16%	1.213ms	18.953us	0 b	0 b	0 b	0 b	64
aten::chunk	0.02%	16.000us	0.07%	69.000us	69.000us	0 b	0 b	0 b	0 b	1
aten::split	0.02%	20.000us	0.05%	53.000us	53.000us	0 b	0 b	0 b	0 b	1
aten::narrow	0.01%	11.000us	0.03%	33.000us	33.000us	0 b	0 b	0 b	0 b	1

Self CPU time total: 104.332ms

Evaluating trained model (not quantized)

[==>] Step: 7ms | To 100/100 | Loss: 0.996 | Acc: 65.110% (6511/10000)

Model size: 36.965367 MB

Model size: 9.301691 MB

Profiler for memory consumption

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	CPU Mem	Self CPU Mem	# of Calls
aten::empty	8.12%	448.000us	8.12%	448.000us	16.593us	1.20 Mb	1.20 Mb	27
aten::_empty_affine_quantized	10.89%	601.000us	10.89%	601.000us	25.042us	660.00 Kb	660.00 Kb	24
aten::quantize_per_tensor	1.01%	56.000us	1.01%	56.000us	56.000us	6.00 Kb	6.00 Kb	1
aten::resize_	0.22%	12.000us	0.22%	12.000us	12.000us	20 b	20 b	1
aten::randn	0.91%	50.000us	5.56%	307.000us	307.000us	24.00 Kb	0 b	1
aten::normal_	1.12%	62.000us	1.12%	62.000us	62.000us	0 b	0 b	1
aten::item	0.29%	16.000us	0.43%	24.000us	12.000us	0 b	0 b	2
aten::_local_scalar_dense	0.14%	8.000us	0.14%	8.000us	4.000us	0 b	0 b	2
aten::contiguous	0.16%	9.000us	2.17%	120.000us	120.000us	6.00 Kb	0 b	1
aten::clone	1.47%	81.000us	2.01%	111.000us	111.000us	6.00 Kb	0 b	1

Self CPU time total: 5.519ms

Experimental Evaluation

- Little to no change in CPU inference time, but >2x speed up in GPU time

```
Profiler for inference
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA %    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                        model_inference         0.01%     209.000us       100.00%        1.922s        1.922s       0.000us         0.00%       1.239ms       1.239ms             1
                                   DataParallel.forward         0.08%       1.491ms        99.98%        1.922s        1.922s       0.000us         0.00%       1.239ms       1.239ms             1
                                           aten::conv2d         0.00%      72.000us        99.76%        1.918s     239.730ms       0.000us         0.00%       1.037ms     129.625us             8
                                      aten::convolution         0.00%      88.000us        99.75%        1.918s     239.721ms       0.000us         0.00%       1.037ms     129.625us             8
                                     aten::_convolution         0.01%     241.000us        99.75%        1.918s     239.710ms       0.000us         0.00%       1.037ms     129.625us             8
                                aten::cudnn_convolution         0.05%     977.000us        99.71%        1.917s     239.627ms     988.000us        79.74%     988.000us     123.500us             8
maxwell_scudnn_winograd_128x128_ldg1_ldg4_relu_tile2...         0.00%       0.000us         0.00%       0.000us       0.000us     530.000us        42.78%     530.000us     132.500us             4
void cudnn::winograd::generateWinogradTilesKernel<0,...         0.00%       0.000us         0.00%       0.000us       0.000us     211.000us        17.03%     211.000us      52.750us             4
maxwell_scudnn_winograd_128x128_ldg1_ldg4_mobile_rel...         0.00%       0.000us         0.00%       0.000us       0.000us     130.000us        10.49%     130.000us      65.000us             2
                                        aten::batch_norm         0.00%      68.000us         0.05%       1.047ms     130.875us       0.000us         0.00%     127.000us      15.875us             8
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 1.923s
Self CUDA time total: 1.239ms

 [================================================================>]  Step: 7ms | To 100/100 | Loss: 1.011 | Acc: 66.110% (6611/10000)
Parallel model test batch inference time (GPU): 2.5399973579997095 sec

Converting model to torchscript...
 [================================================================>]  Step: 6ms | To 100/100 | Loss: 1.011 | Acc: 66.110% (6611/10000)
Traced parallel model test batch inference time (GPU): 1.1010754359999737 sec

Profiler for inference
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA %    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                        model_inference         0.01%     241.000us        99.99%        1.955s        1.955s       0.000us         0.00%       1.039ms       1.039ms             1
                                                forward         0.00%      12.000us        99.98%        1.955s        1.955s       0.000us         0.00%       1.039ms       1.039ms             1
                                   DataParallel.forward         0.01%     249.000us        99.98%        1.955s        1.955s       0.000us         0.00%       1.039ms       1.039ms             1
                                     aten::_convolution         0.01%     269.000us        99.79%        1.951s     243.922ms       0.000us         0.00%     958.000us     119.750us             8
                                aten::cudnn_convolution         0.04%     819.000us        99.76%        1.951s     243.843ms     913.000us        87.87%     913.000us     114.125us             8
maxwell_scudnn_winograd_128x128_ldg1_ldg4_relu_tile2...         0.00%       0.000us         0.00%       0.000us       0.000us     480.000us        46.20%     480.000us     120.000us             4
void cudnn::winograd::generateWinogradTilesKernel<0,...         0.00%       0.000us         0.00%       0.000us       0.000us     208.000us        20.02%     208.000us      52.000us             4
maxwell_scudnn_winograd_128x128_ldg1_ldg4_mobile_rel...         0.00%       0.000us         0.00%       0.000us       0.000us     117.000us        11.26%     117.000us      58.500us             2
void implicit_convolve_sgemm<float, float, 128, 5, 5...         0.00%       0.000us         0.00%       0.000us       0.000us      84.000us         8.08%      84.000us      42.000us             2
                                             aten::add_         0.01%     156.000us         0.01%     231.000us      28.875us      45.000us         4.33%      45.000us       5.625us             8
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 1.955s
Self CUDA time total: 1.039ms
```

Experimental Evaluation

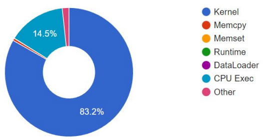
LSTM → 1. *Initial profiling analysis and identification of the bottleneck*

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	CPU Mem	Self CPU Mem	CUDA Mem	Self CUDA Mem	# of Calls
aten::lstm	0.37%	5.510ms	36.02%	537.105ms	9.103ms	0.000us	0.00%	1.504s	25.493ms	0 b	0 b	1.86 Gb	0 b	59
aten::_cudnn_rnn	17.02%	253.780ms	35.53%	529.904ms	8.981ms	1.502s	98.60%	1.504s	25.493ms	0 b	0 b	1.86 Gb	-10.07 Gb	59
volta_sgemm_64x32_sliced1x4_tn	0.00%	0.000us	0.00%	0.000us	0.000us	1.015s	66.66%	1.015s	33.608us	0 b	0 b	0 b	0 b	30208
volta_sgemm_128x64_tn	0.00%	0.000us	0.00%	0.000us	0.000us	261.200ms	17.15%	261.200ms	279.060us	0 b	0 b	0 b	0 b	936

```
Overall Time: 49.64097316499999s

Train:
Total train Time: 18.132872709000026s
Train Time per epoch:
18.132872709000026s

Validation:
Total validation Time:
1.923322245999998s
Validation per epoch: 1.923322245999998s

Test:
Total test Time: 1.5564957660000118s
```

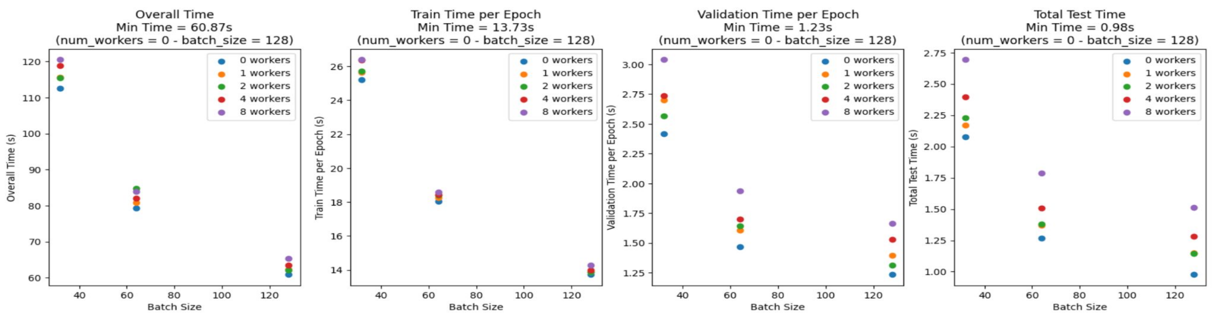
Execution Summary

Category	Time Duration (us)	Percentage (%)
Average Step Time	37,273,784	100
Kernel	31,014,603	83.21
Memcpy	217,258	0.58
Memset	18,847	0.05
Runtime	0	0
DataLoader	0	0
CPU Exec	5,396,088	14.48
Other	626,988	1.68



Conclusion: the computations of the model are expensive in memory which is the main bottleneck but also take up most of the time.

LSTM → 2. *Experimental findings of optimal batch size and number of workers*

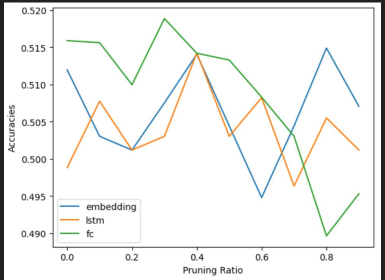


Conclusion: Optimal parameters are num_workers=0, batch_size=128.
Due to a memory overhead ?

Experimental Evaluation

LSTM → 3. *Experimentations around pruning techniques and final performance*

- *Compared structured vs unstructured → preferred structured*
- *Tested layers' sensitivity to pruning ratio*



Conclusion: Varies slightly between 0.49 and 0.52 → not sensitive
Chose a 0.9 pruning ratio for every layer.

- *Final performance*

```
Before pruning:
The model has 34,731,777
trainable parameters

After pruning:
The model has 3,528,662
non-zero trainable
parameters
```

```
Overall Time: 24.326757582999562s

Train:
Total train Time: 13.63664408800014s
Train Time per epoch: 13.63664408800014s

Validation:
Total validation Time:
1.869992953000292s
Validation per epoch: 1.869992953000292s

Test:
Total test Time: 1.0925553890001538s
```

Final model

```
Overall Time: 49.64097316499999s

Train:
Total train Time: 18.132872709000026s
Train Time per epoch:
18.132872709000026s

Validation:
Total validation Time:
1.923322245999998s
Validation per epoch: 1.923322245999998s

Test:
Total test Time: 1.5564957660000118s
```

First model

Conclusion:
Gain in time and memory

Conclusion

- The number of workers can help decreasing time but not always
- Quantization and pruning are two techniques that can help solve memory bottlenecks and speed-up the process
- Pytorch script also allows for faster inference and serialization for non-python environments