Train Large, Then Compress:
Rethinking Model Size for Efficient Training and Inference of Transformers

Berkeley
N L P

riselab
UC Berkeley

Zhuohan Li★

Eric Wallace★

Kevin Lin★

Sheng Shen★

Kurt Keutzer

Dan Klein

Joseph E. Gonzalez

State-of-the-art NLP models require millions of dollars to train

Devlin et al. (2019). We pretrain our model using 1024 V100 GPUs for approximately one day.

$4,600,000: The full cost of training GPT-3

Consumption	CO_2e (lbs)
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

Artificial intelligence / Machine learning

Training a single AI model can emit as much carbon as five cars in their lifetimes

Why is training so expensive?

- One can **trade compute for accuracy**

Why is training so expensive?

- One can **trade compute for accuracy**
  - Larger model size
  - Larger datasets
  - More training iterations

Why is training so expensive?

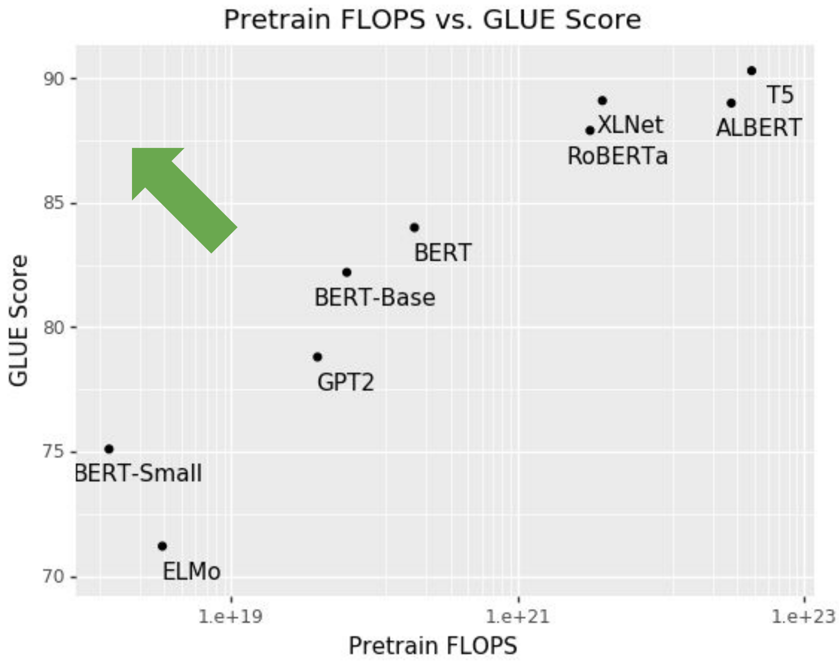
- One can **trade compute for accuracy**
  - ⬆ Larger model size
  - ⬆ Larger datasets
  - ⬆ More training iterations
- **Computational constraints** are increasingly the bottleneck

Maximizing Computational Efficiency

- The goal → maximize **computational efficiency**
  - highest possible accuracy given fixed hardware and training time

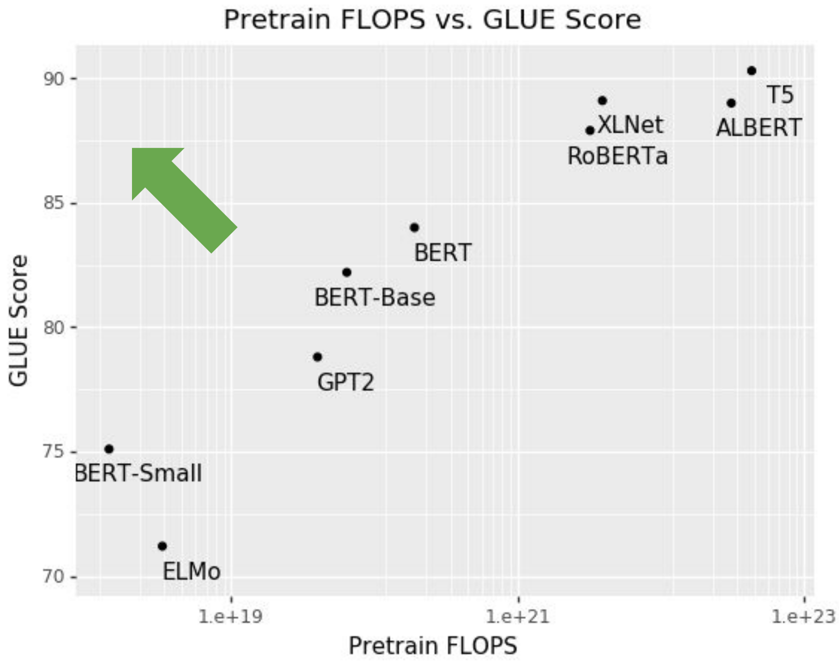
Maximizing Computational Efficiency

- The goal → maximize **computational efficiency**
  - highest possible accuracy given fixed hardware and training time



Maximizing Computational Efficiency

- The goal → maximize **computational efficiency**
  - highest possible accuracy given fixed hardware and training time



Rethinking Common Assumptions

- Conventional wisdom:
  - *Small* models **train** *faster*

Rethinking Common Assumptions

- Conventional wisdom:
  - *Small* models **train** *faster*
  - *Large* models are *infeasible* for **inference**

Rethinking Common Assumptions

- Conventional wisdom:
  - *Small* models **train** *faster*
  - *Large* models are *infeasible* for **inference**

small models = **fast** ↔ **large** models = **slow**

Rethinking Common Assumptions

- Conventional wisdom:
  - *Small* models **train** *faster*
  - *Large* models are *infeasible* for **inference**

small models = **fast** ↔ **large** models = **slow**

- Goal: study how **model size** affects training and inference efficiency

Rethinking Common Assumptions

- Conventional wisdom:
  - *Small* models **train** *faster*
  - *Large* models are *infeasible* for **inference**

small models = **fast** ↔ **large** models = **slow**

- Goal: study how **model size** affects training and inference efficiency
- Results:
  - **Large** models **train faster**

Rethinking Common Assumptions

- Conventional wisdom:
  - *Small* models **train** *faster*
  - *Large* models are *infeasible* for **inference**

  small models = **fast** ↔ **large** models = **slow**

- Goal: study how **model size** affects training and inference efficiency

- Results:
  - **Large** models **train faster**
  - **Large** models are **efficient** at **inference** time

Rethinking Common Assumptions

- Conventional wisdom:
  - *Small* models **train** *faster*
  - *Large* models are *infeasible* for **inference**

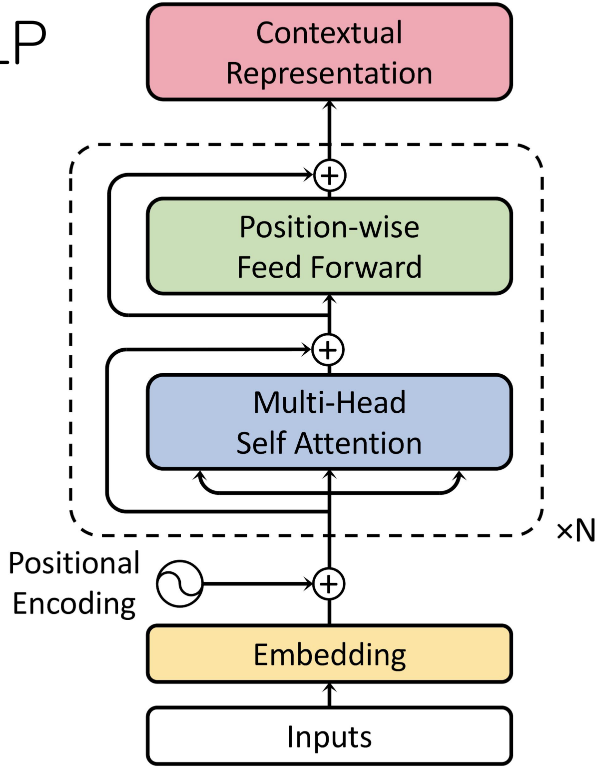
  small models = **fast** ↔ **large** models = **slow**

- Goal: study how **model size** affects training and inference efficiency
- Results:
  - **Large** models **train faster**
  - **Large** models are **efficient** at **inference** time
- Key idea: stop training early & compress heavily

Training Efficiency

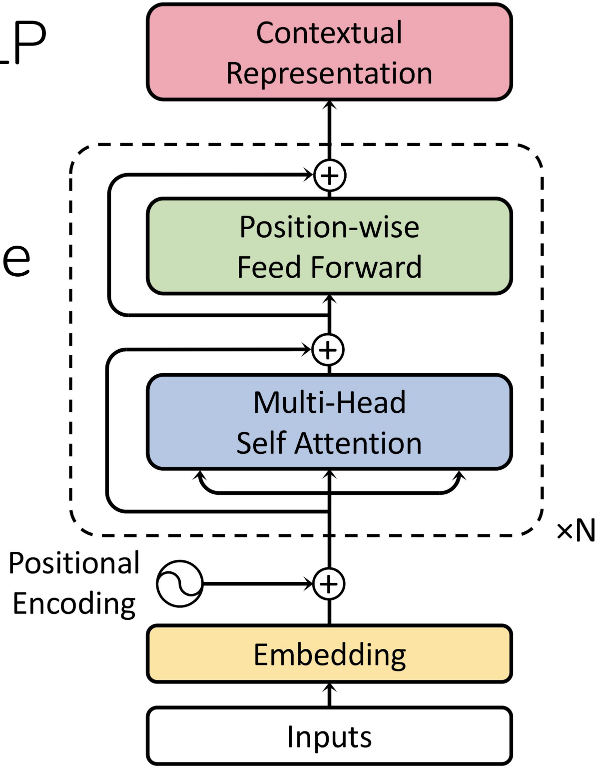
Experimental Setup

- Transformer models
  - feedforward architecture, SoTA for NLP



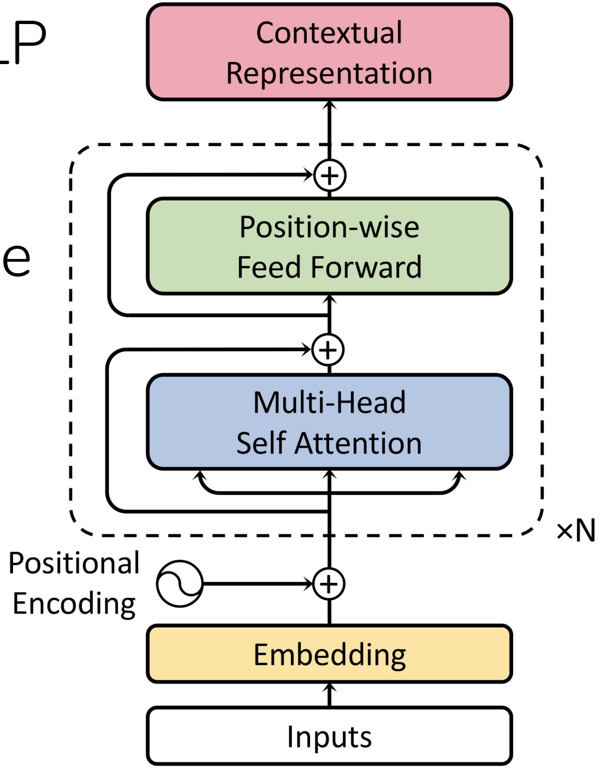
Experimental Setup

- Transformer models
  - feedforward architecture, SoTA for NLP
- We vary hidden size + model depth
  - measure loss at a given wall clock time
  - increase batch size to fill the GPU



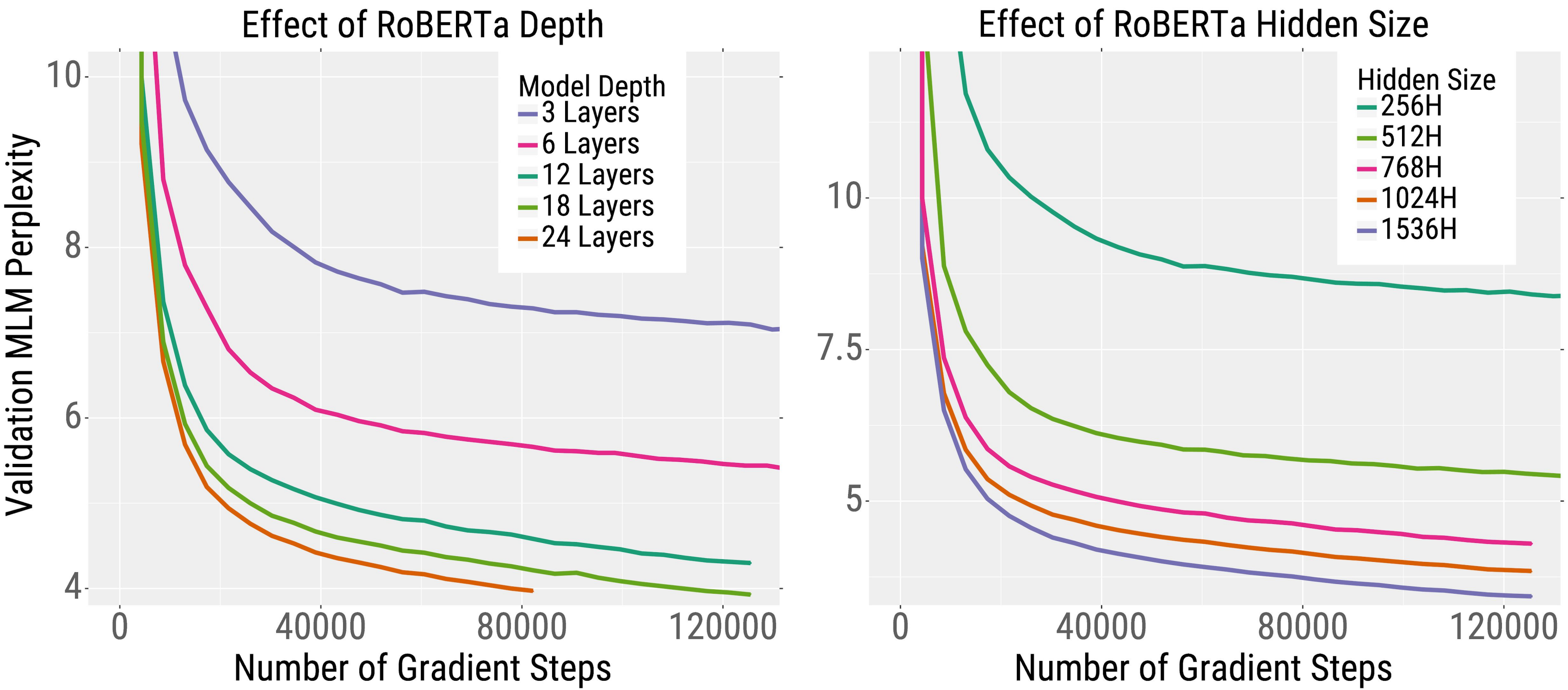
Experimental Setup

- Transformer models
  - feedforward architecture, SoTA for NLP
- We vary hidden size + model depth
  - measure loss at a given wall clock time
  - increase batch size to fill the GPU
- Task 1: **MLM pretraining + finetuning (RoBERTa)**
- Task 2: **machine translation**

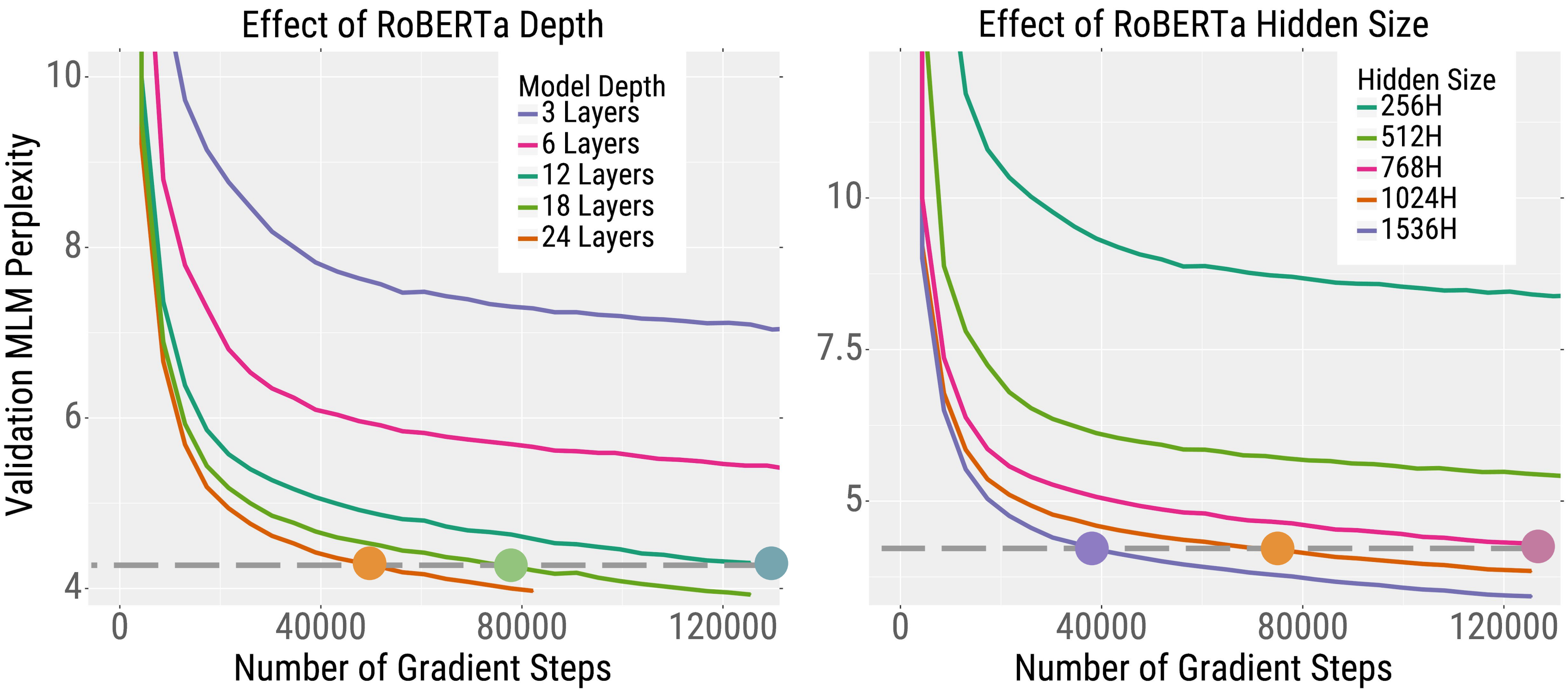


Deeper and Wider Models Converge in Fewer Steps

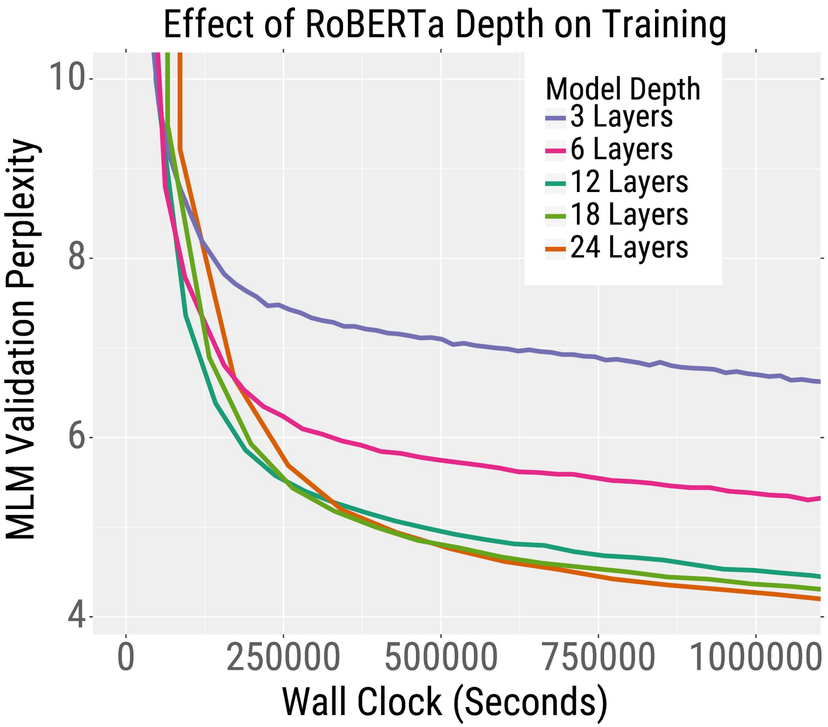
Deeper and Wider Models Converge in Fewer Steps

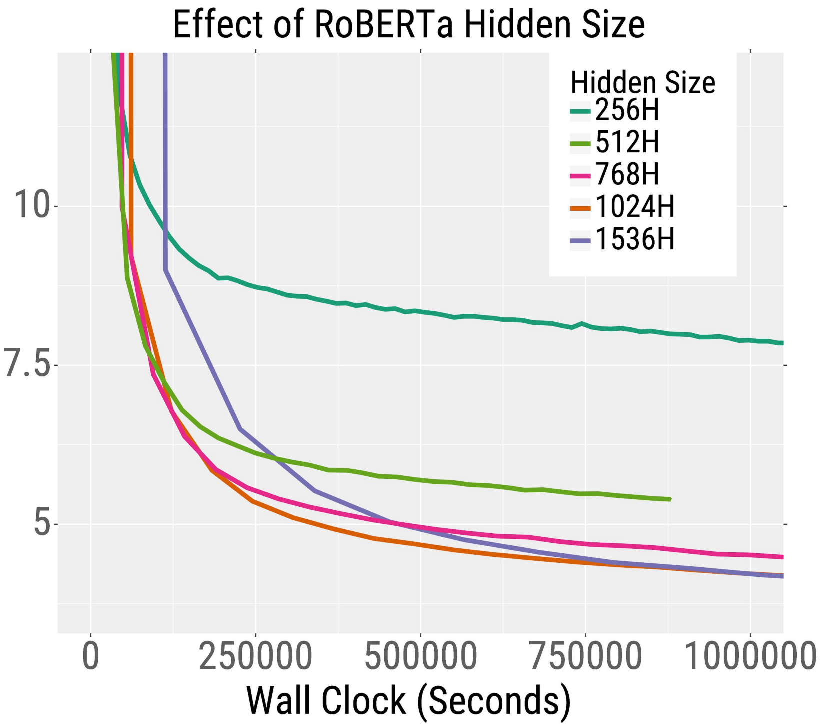


Deeper and Wider Models Converge in Fewer Steps

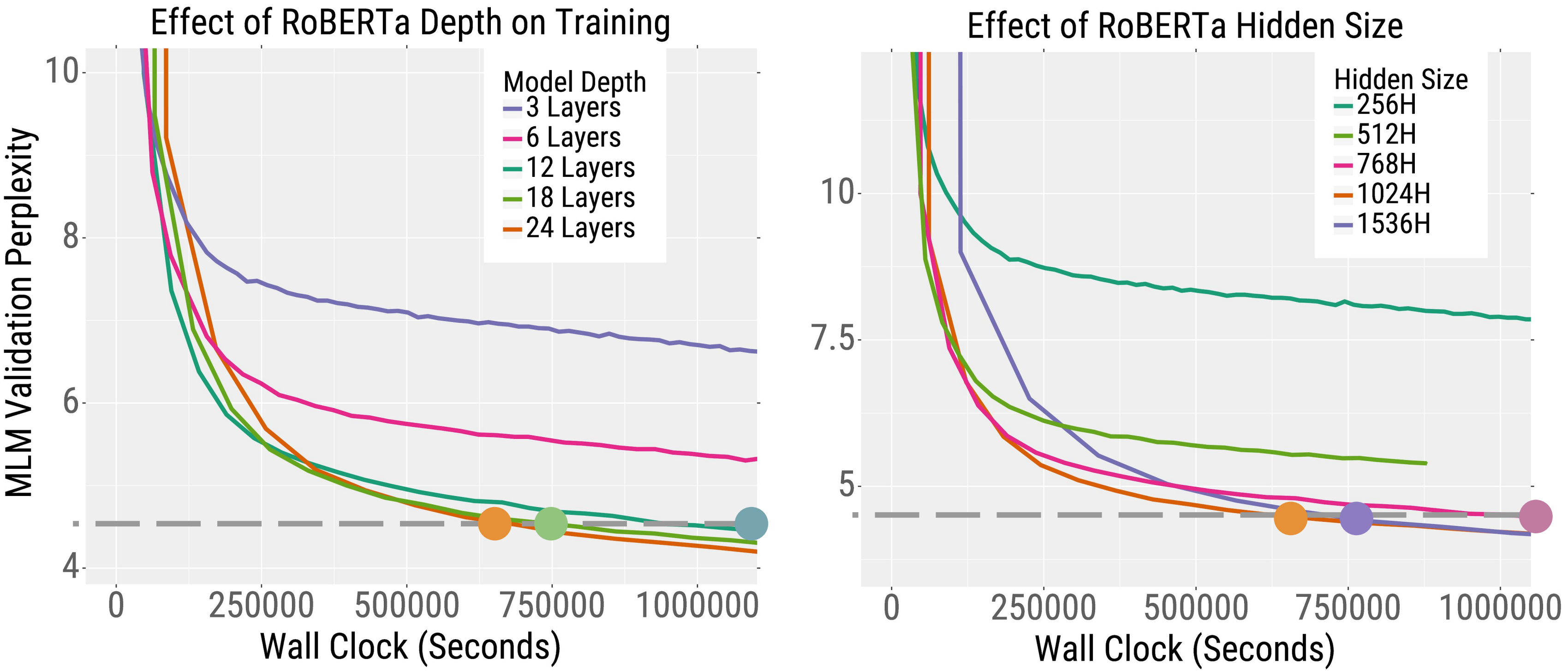


Deeper and Wider Models Converge in Less Wall Clock Time

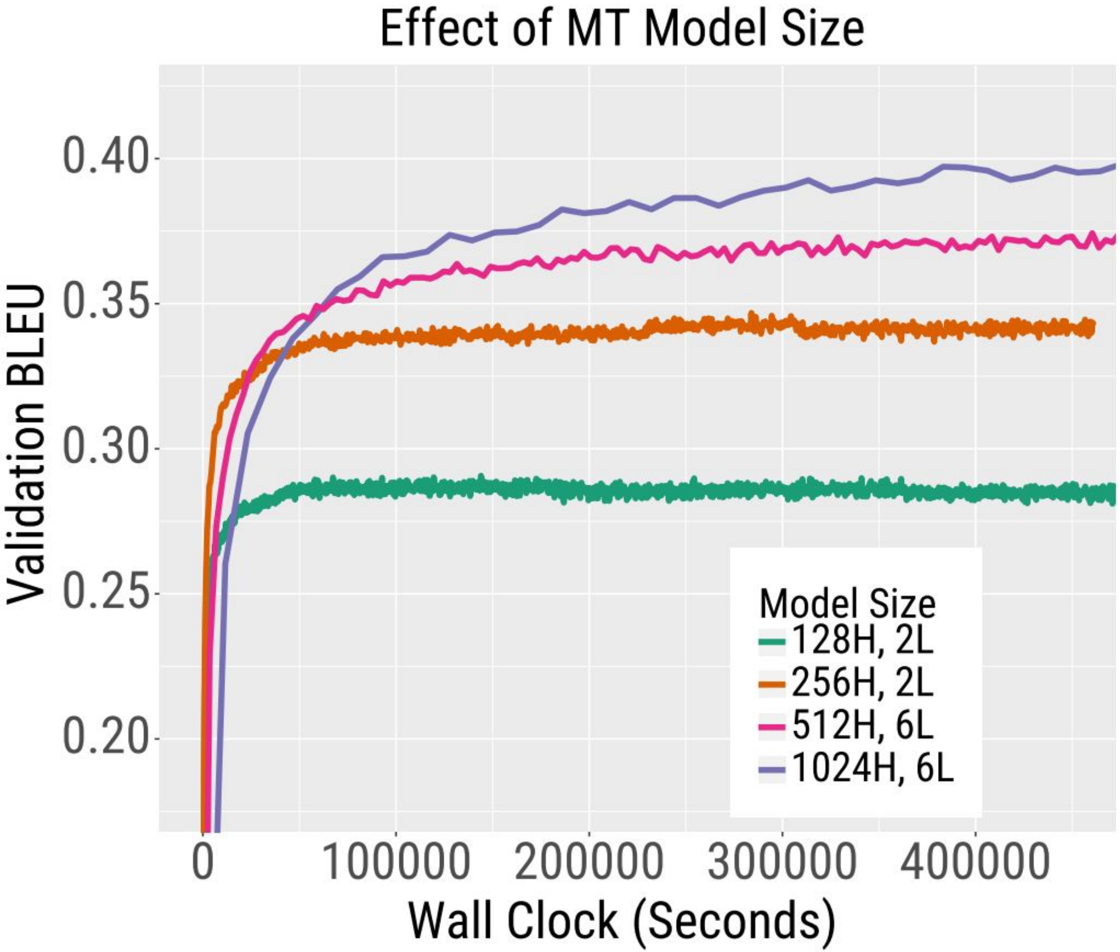




Deeper and Wider Models Converge in Less Wall Clock Time



Same Trends Hold for Machine Translation



Why Do Larger Models Train Faster?

- Larger models reduce **training** error faster

Why Do Larger Models Train Faster?

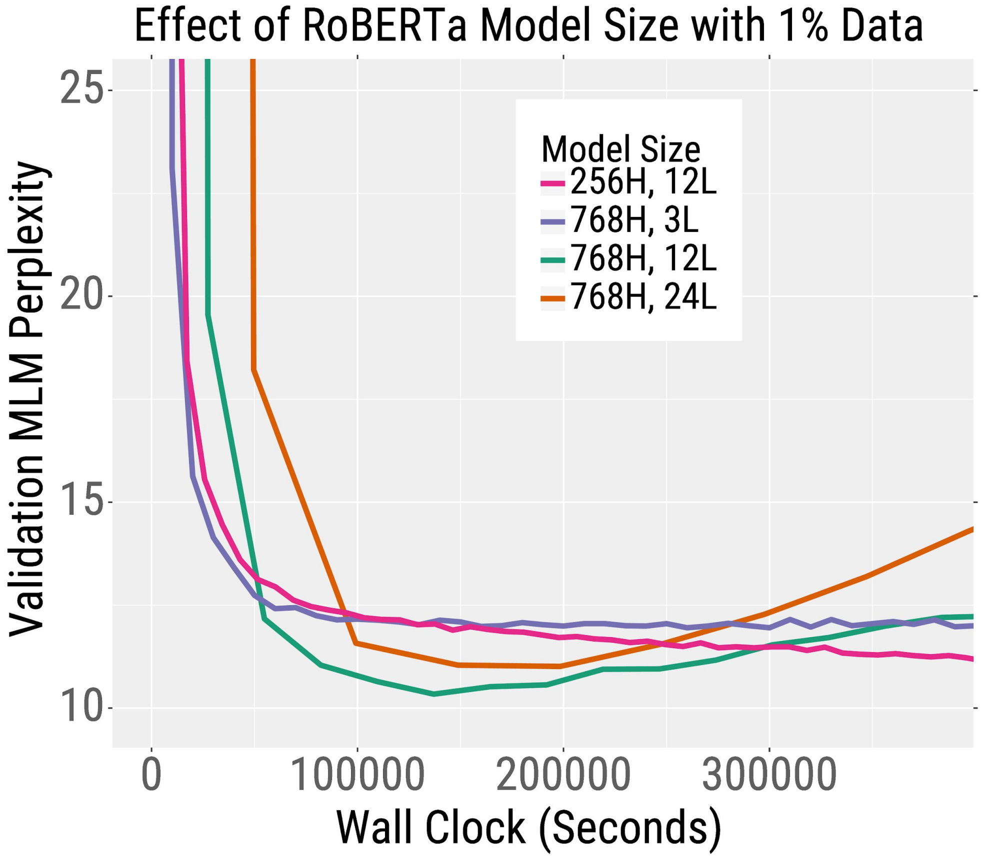
- Larger models reduce **training** error faster
- MLM training has "**unlimited**" data → overfitting <u>*not*</u> a concern
- Thus, larger models also minimize **validation** error faster

Why Do Larger Models Train Faster?

- When overfitting *is* a concern, be careful of how big you go

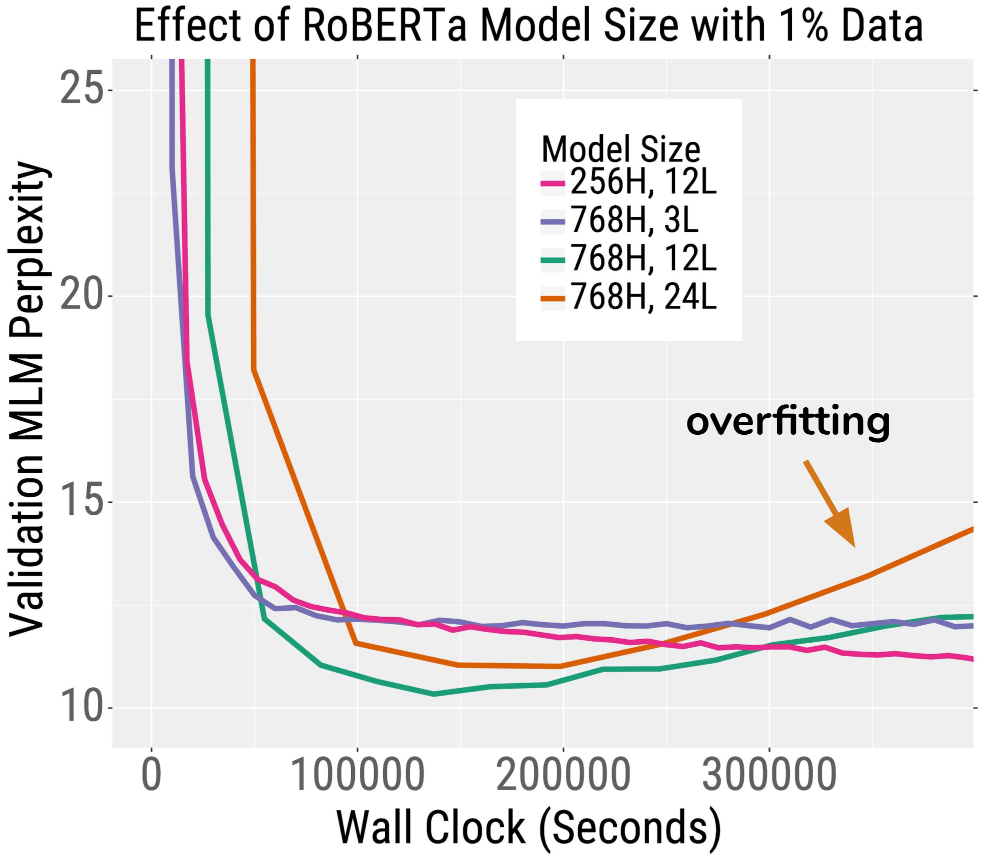
Why Do Larger Models Train Faster?

- When overfitting *is* a concern, be careful of how big you go



Why Do Larger Models Train Faster?

- When overfitting *is* a concern, be careful of how big you go



Inference Efficiency

Large models are **fast** at **training** time

Large models are **slow** at **inference** time

Trade-off between large and small models?

✓ Large models are **fast** at **training** time

✗ Large models are **slow** at **inference** time

Trade-off between large and small models? **No!**

✓ We show that larger models are **more** *compressible*

Experimental Setup

- Fix training time for models of different sizes
- Two compression techniques: pruning & quantization

Experimental Setup

- Fix training time for models of different sizes
- Two compression techniques: **pruning** & quantization
  - ➔ Set weights to 0
    - ◆ Reduces memory
    - ◆ Reduces FP operations

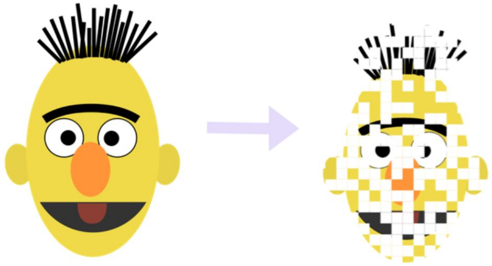


Image: Rasa

Experimental Setup

- Fix training time for models of different sizes
- Two compression techniques: **pruning** & quantization
  - ➔ Set weights to 0
    - ◆ Reduces memory
    - ◆ Reduces FP operations

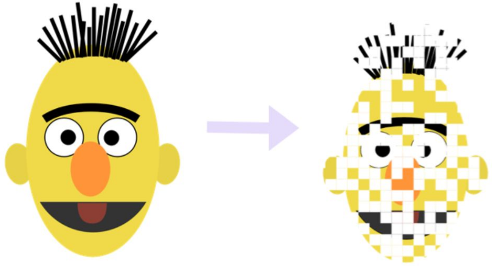


Image: Rasa

Experimental Setup

- Fix training time for models of different sizes
- Two compression techniques: pruning & **quantization**
  - ➔ Store weights in low precision
    - ◆ Reduces memory
    - ◆ Accelerates speed on certain hardware
    - ◆ Post-hoc quantize with no additional training time

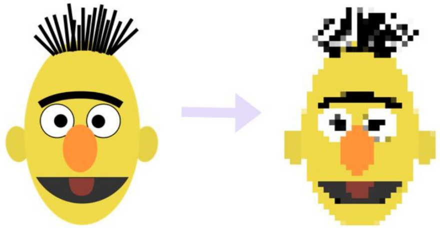
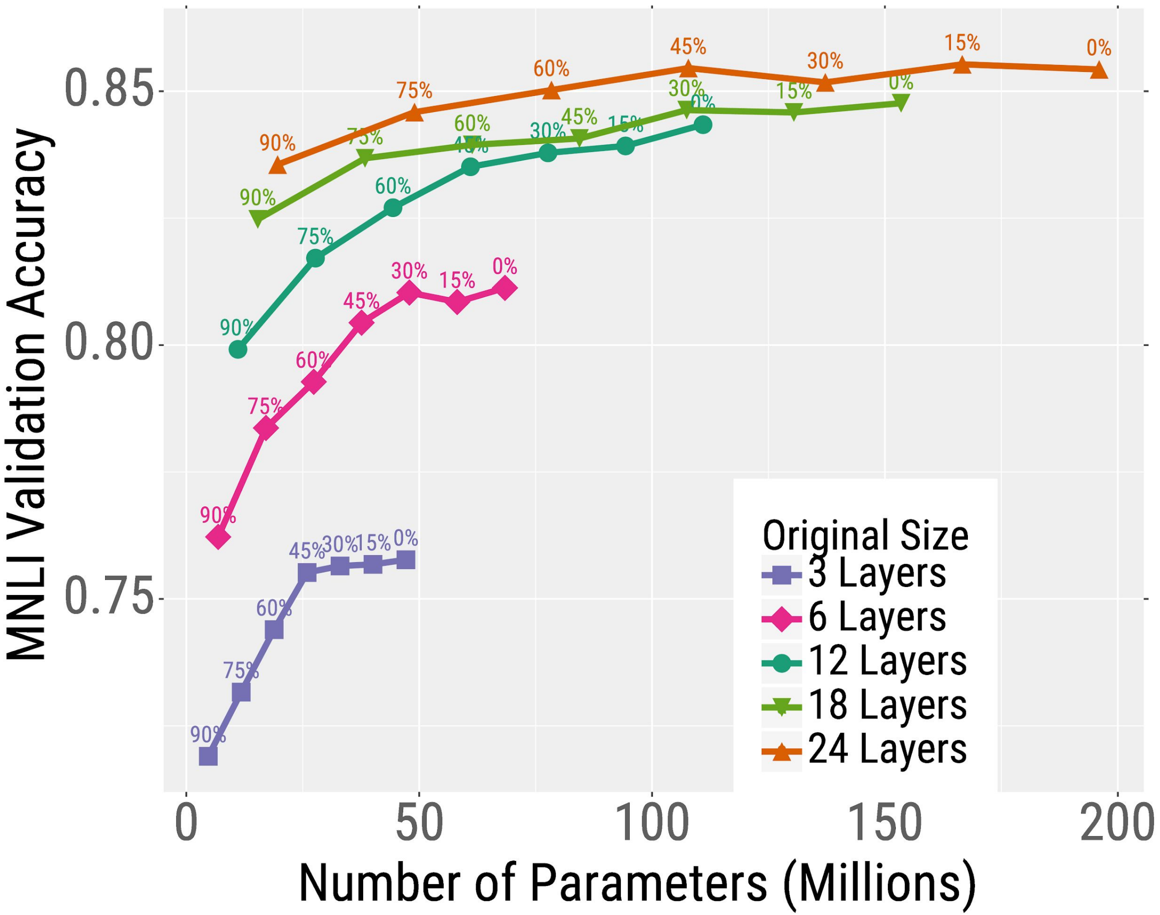
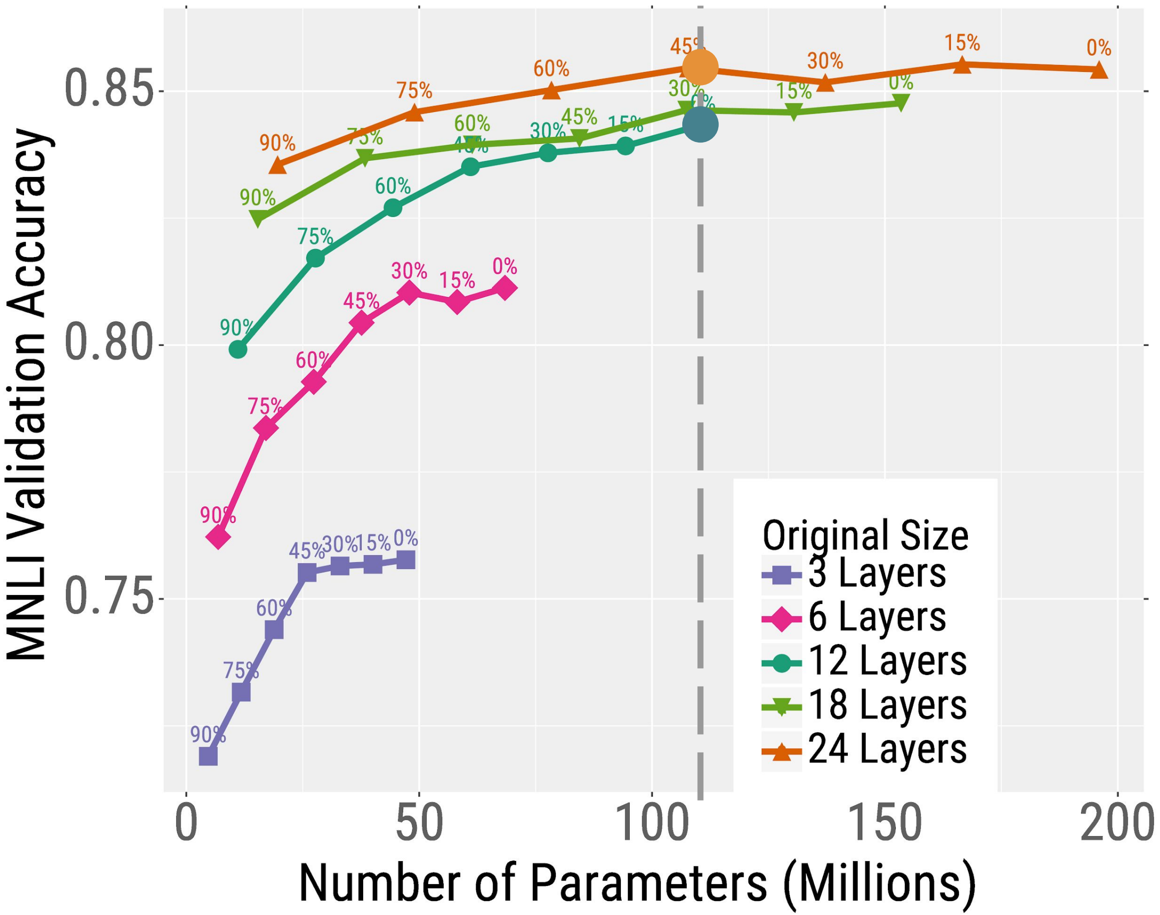


Image: Rasa

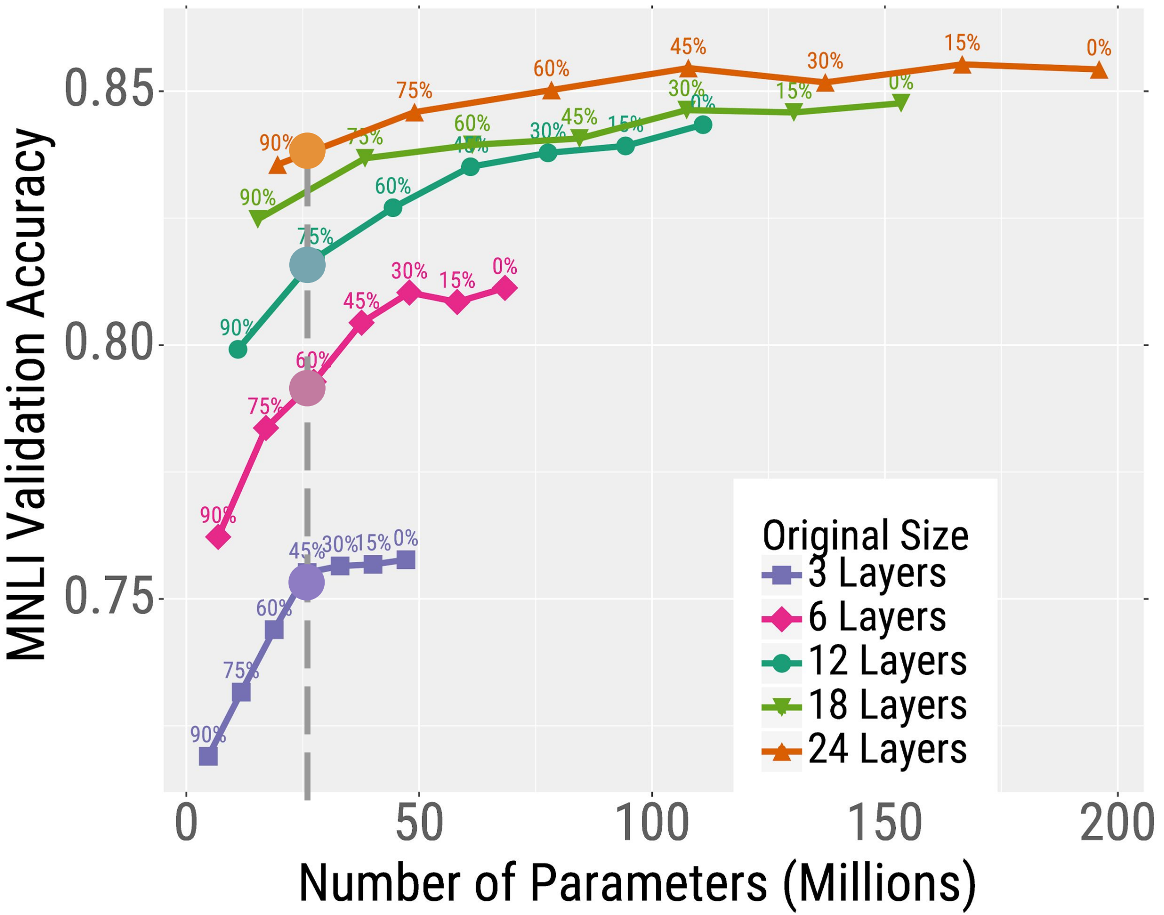
Deeper and Wider Models are More Robust to Pruning



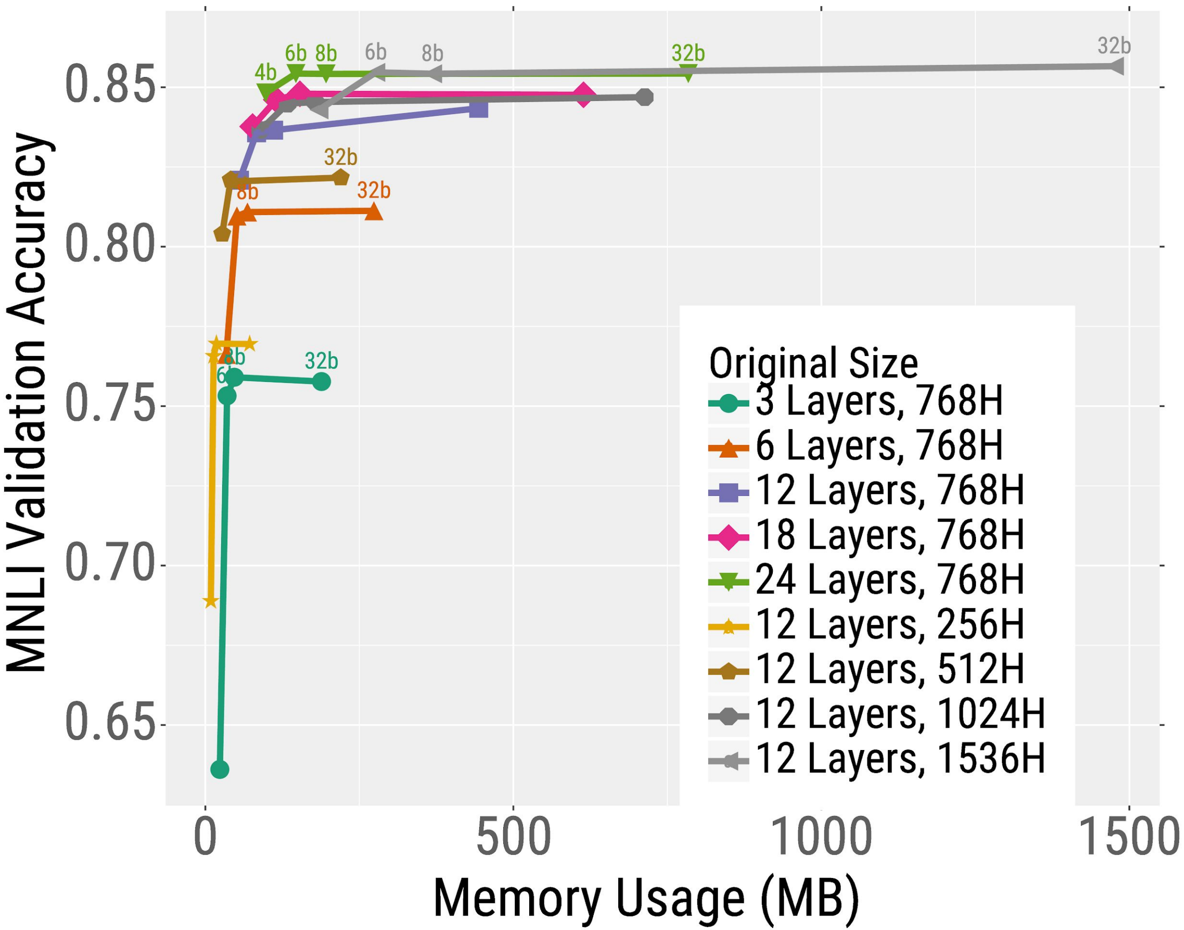
Deeper and Wider Models are More Robust to Pruning



Deeper and Wider Models are More Robust to Pruning

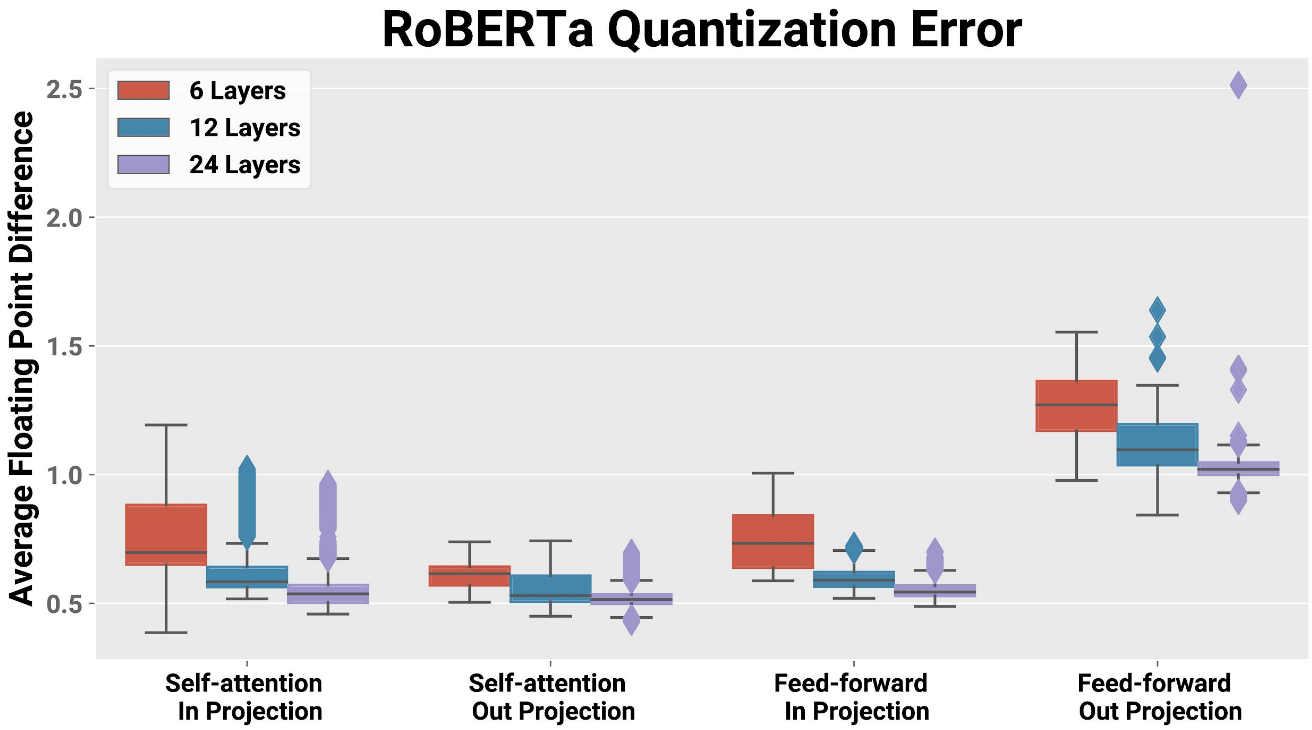


Deeper and Wider Models are More Robust to Quantization



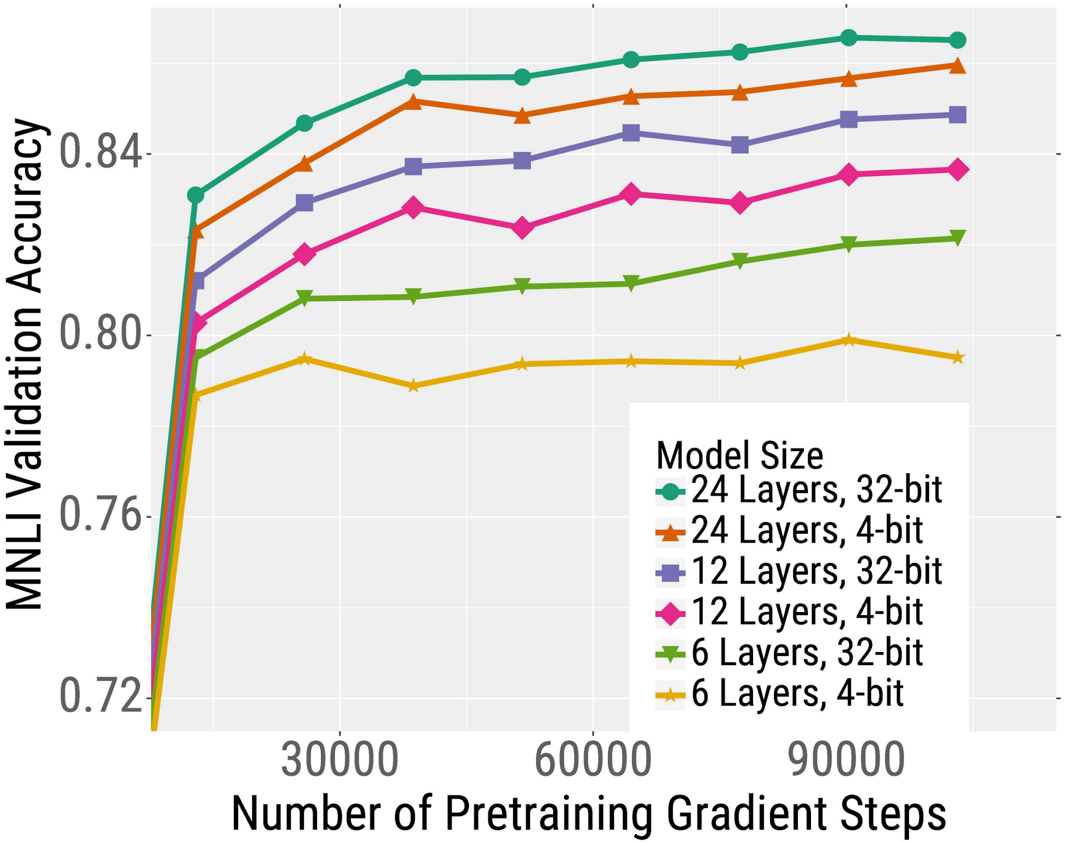
Why Do Larger Models Compress Better?

- Quantization/Pruning error is smaller for larger models



Why Do Larger Models Compress Better?

- Size, not convergence, determines compressibility



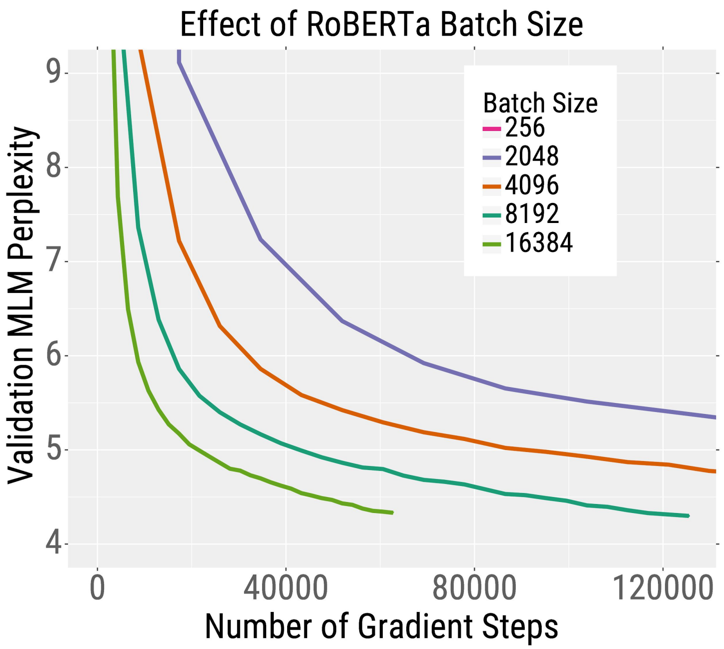
Practical Takeaways

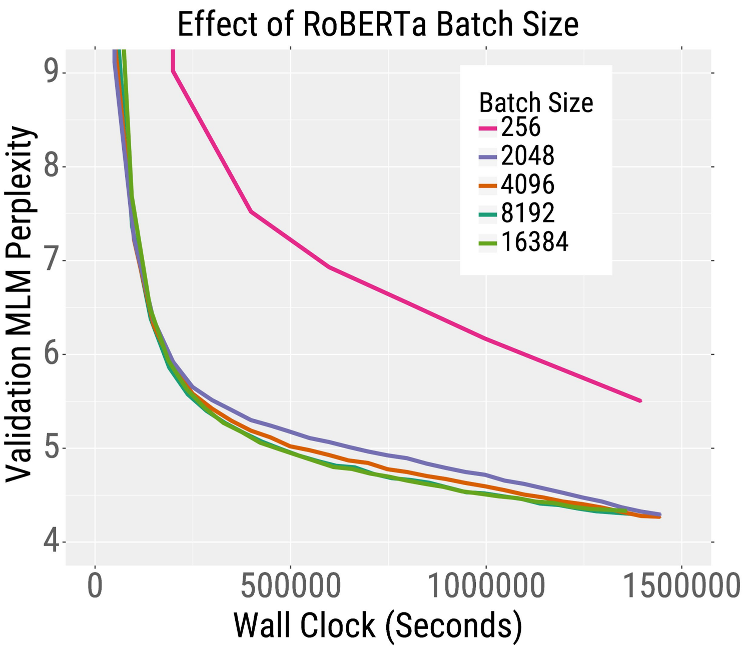
Practical Takeaways

- **Increase model width, sometimes depth**

Practical Takeaways

- Increase model width, sometimes depth
- **Increase model size not batch size**





Practical Takeaways

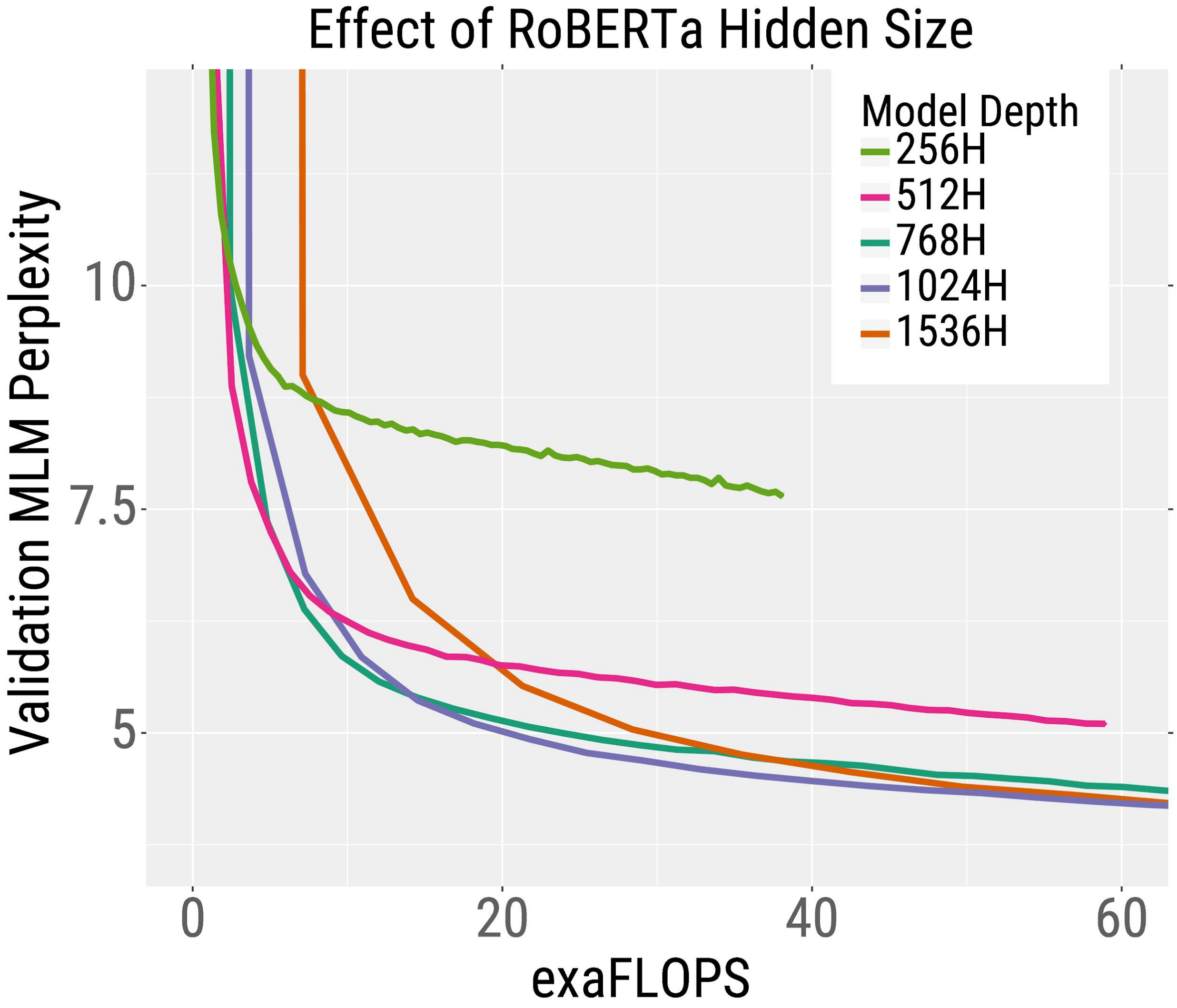
- Increase model width, sometimes depth
- Increase model size not batch size
- **Apply compression methods like pruning/quantization**
  - little to no training overhead
  - compress model up to 8x without hurting performance

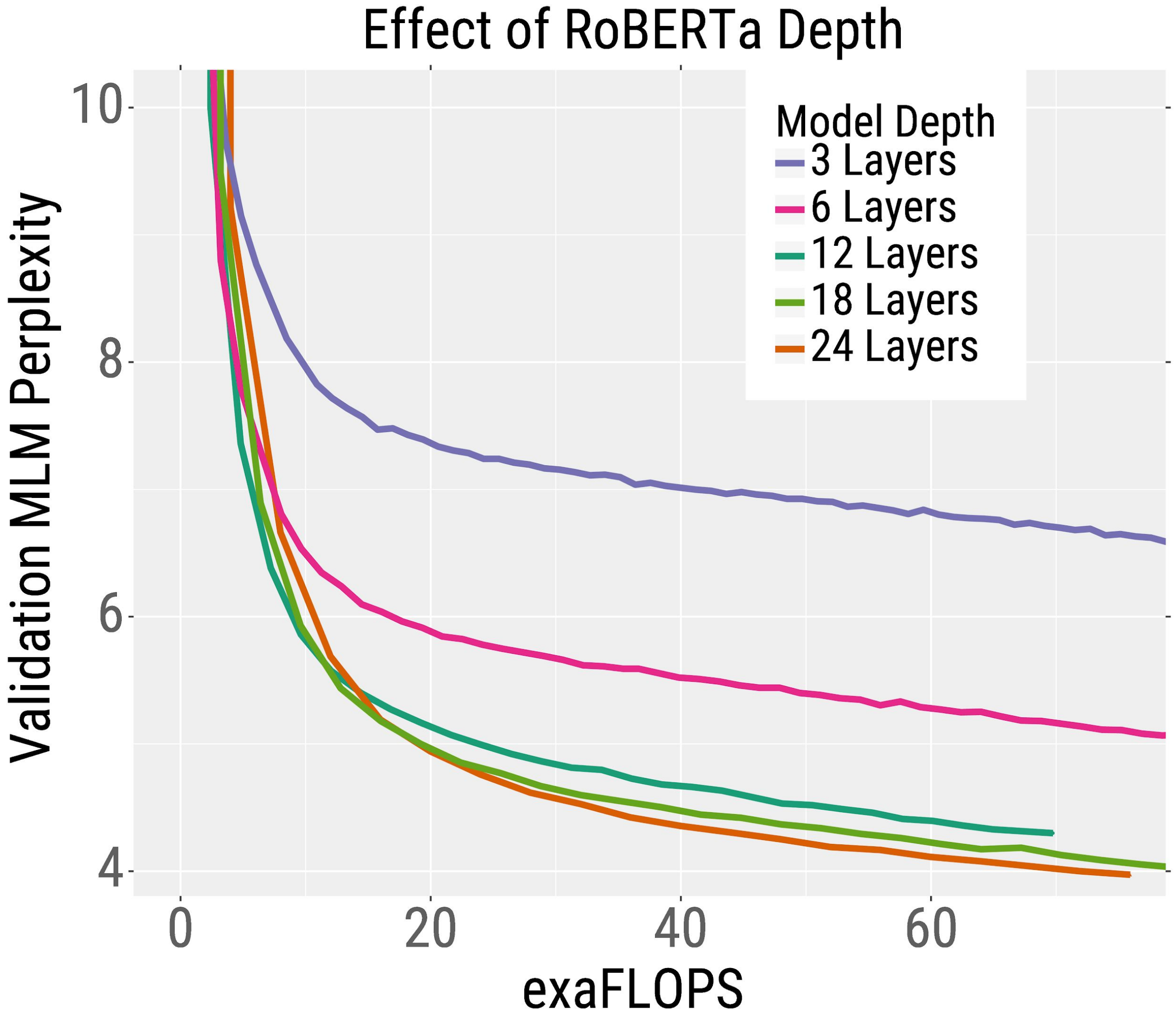
Conclusion

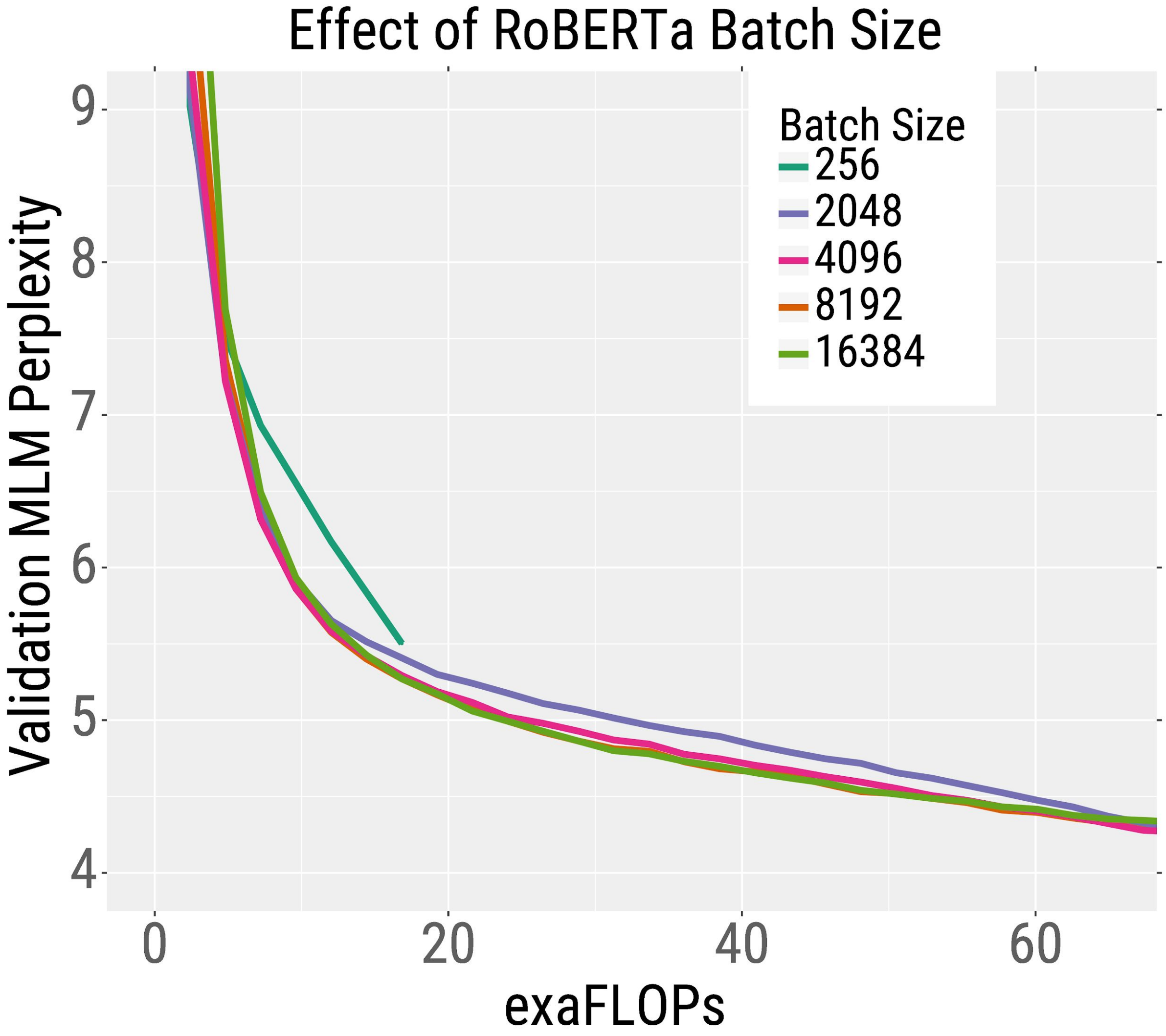
Common Practice	Train Small Model →	Stop Training When Converged →	Lightly Compress
Optimal	Train Large Model →	Stop Training Early →	Heavily Compress

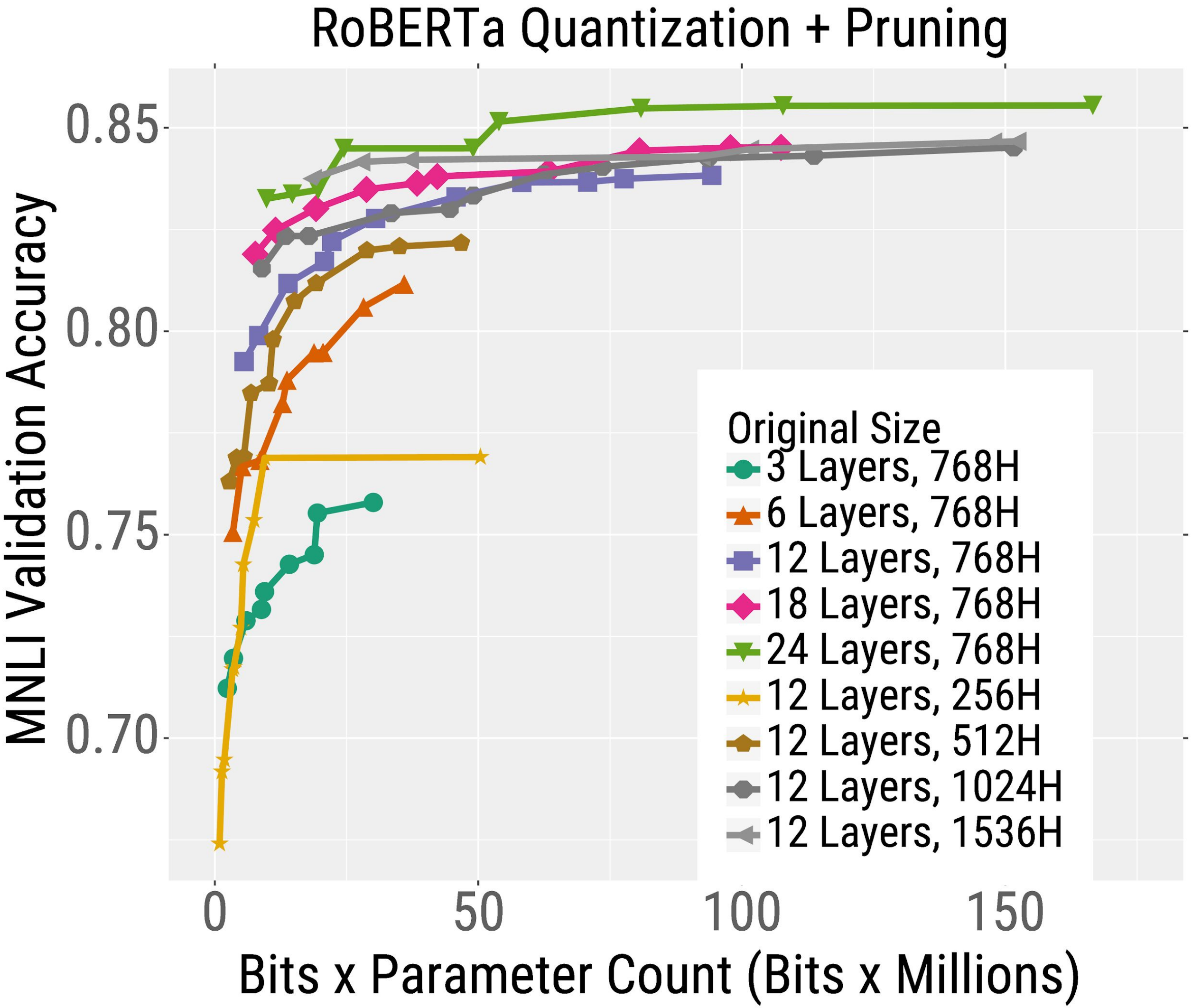
Blog and Paper available

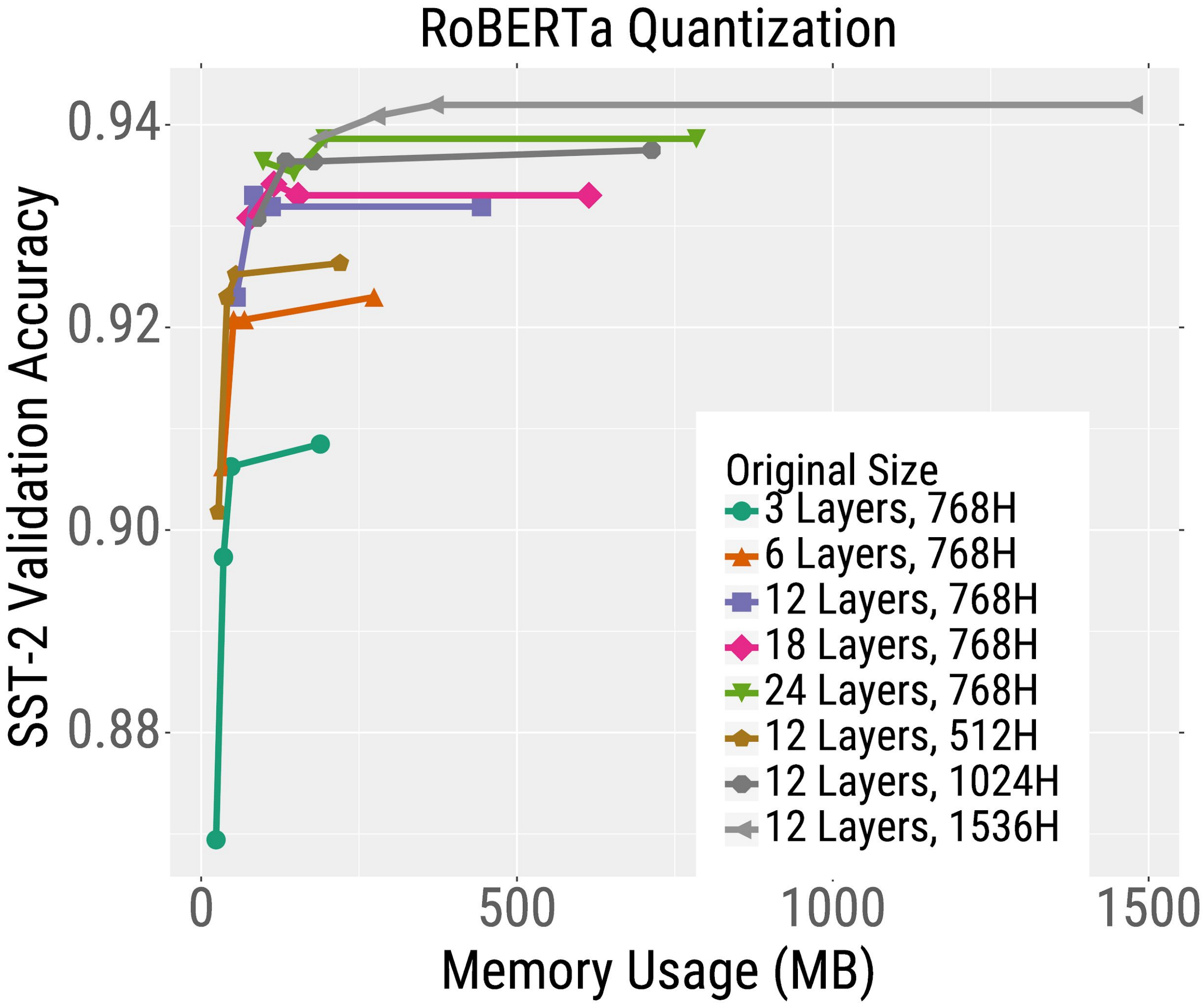
Bonus Slides

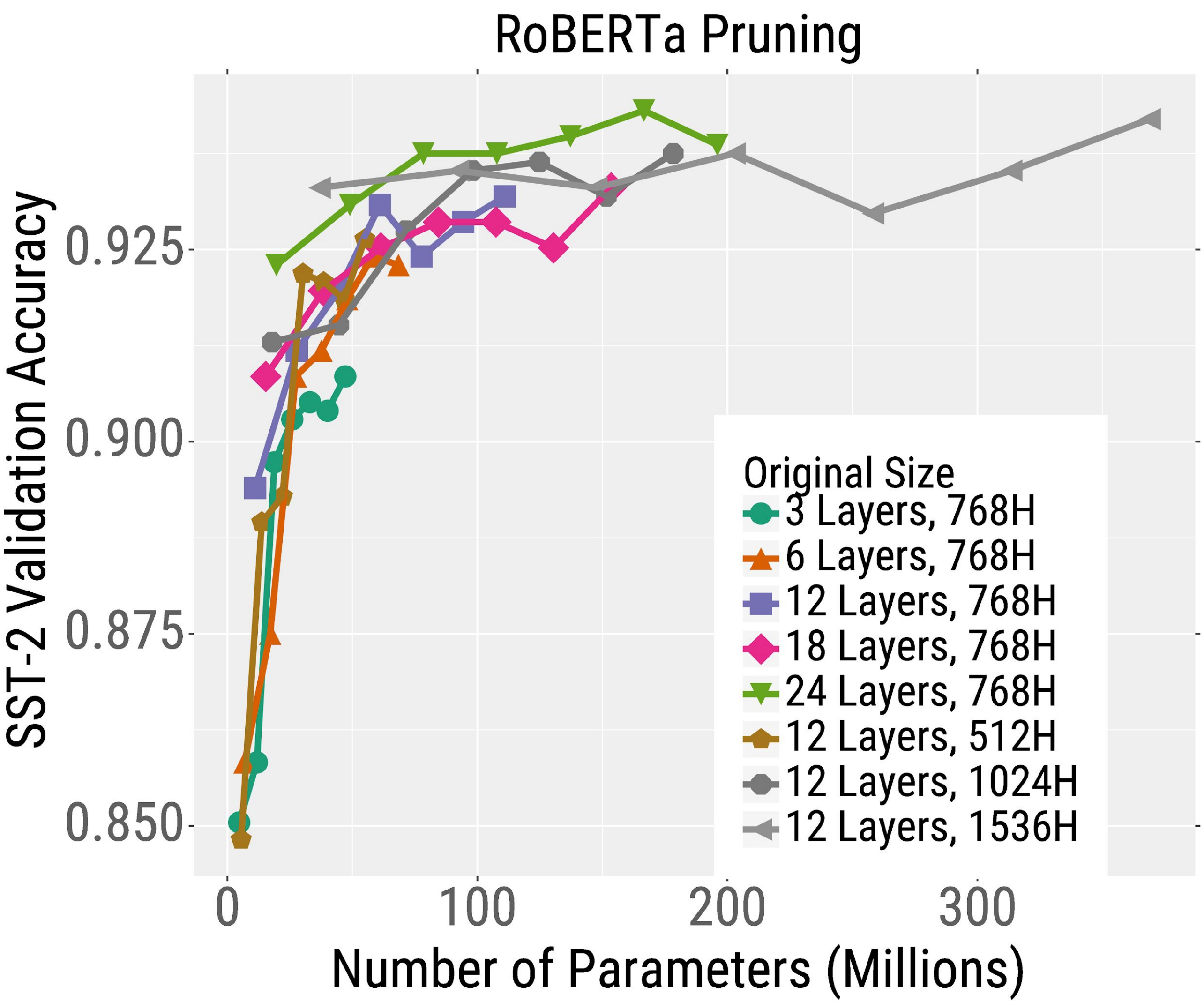

Effect of RoBERTa Hidden Size
Model Depth
256H
512H
768H
1024H
1536H
Validation MLM Perplexity
10
7.5
5
0
20
40
60
exaFLOPS

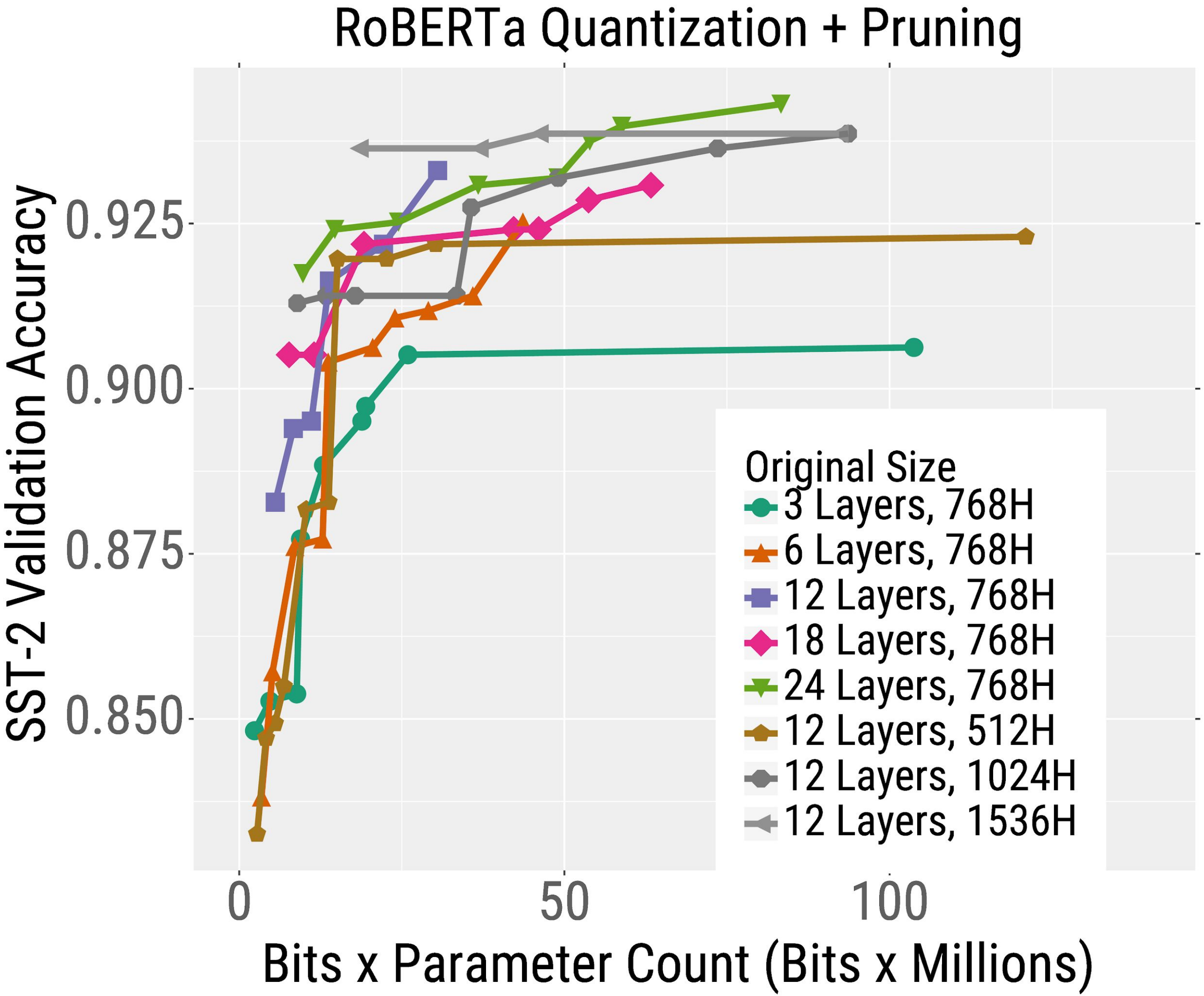

Effect of RoBERTa Depth
Model Depth
3 Layers
6 Layers
12 Layers
18 Layers
24 Layers
Validation MLM Perplexity
10
8
6
4
0
20
40
60
exaFLOPS

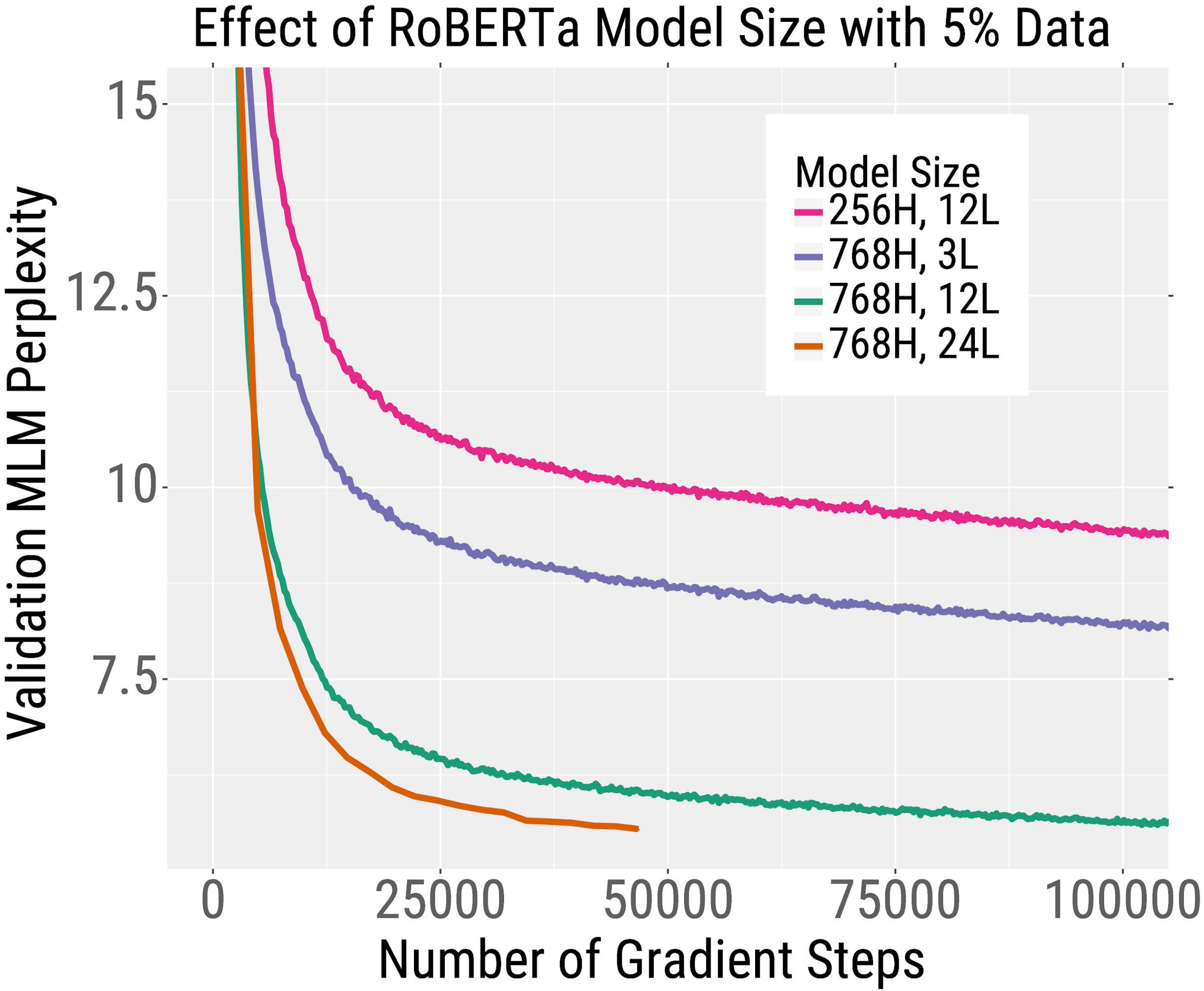

Effect of RoBERTa Batch Size
Validation MLM Perplexity
exaFLOPs
Batch Size
256
2048
4096
8192
16384
9
8
7
6
5
4
0
20
40
60


RoBERTa Quantization + Pruning
MNLI Validation Accuracy
0.85
0.80
0.75
0.70
0
50
100
150
Bits x Parameter Count (Bits x Millions)
Original Size
3 Layers, 768H
6 Layers, 768H
12 Layers, 768H
18 Layers, 768H
24 Layers, 768H
12 Layers, 256H
12 Layers, 512H
12 Layers, 1024H
12 Layers, 1536H

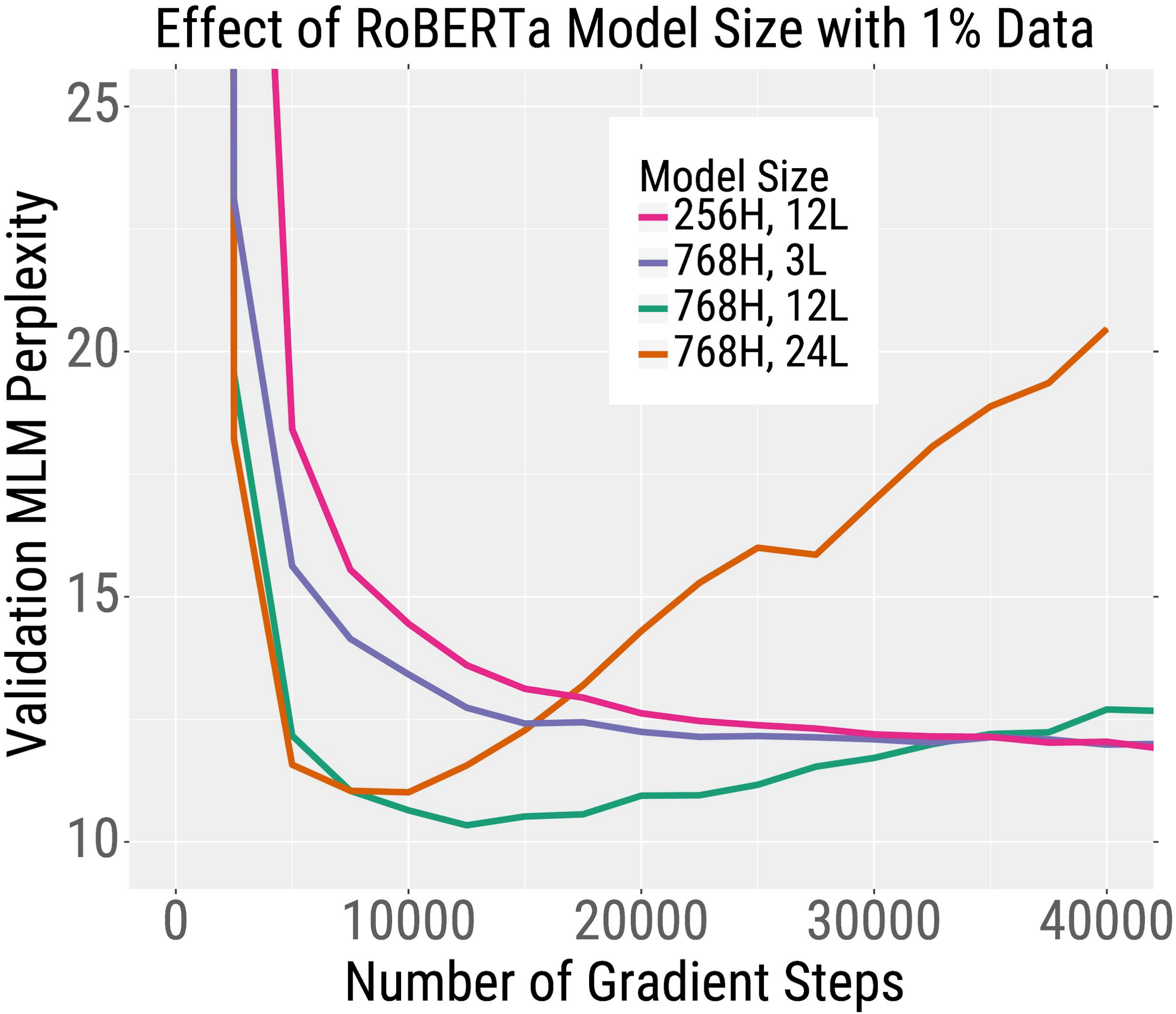

RoBERTa Quantization
SST-2 Validation Accuracy
0.94
0.92
0.90
0.88
0
500
1000
1500
Memory Usage (MB)
Original Size
3 Layers, 768H
6 Layers, 768H
12 Layers, 768H
18 Layers, 768H
24 Layers, 768H
12 Layers, 512H
12 Layers, 1024H
12 Layers, 1536H


RoBERTa Pruning
SST-2 Validation Accuracy
0.850
0.875
0.900
0.925
0
100
200
300
Number of Parameters (Millions)
Original Size
3 Layers, 768H
6 Layers, 768H
12 Layers, 768H
18 Layers, 768H
24 Layers, 768H
12 Layers, 512H
12 Layers, 1024H
12 Layers, 1536H

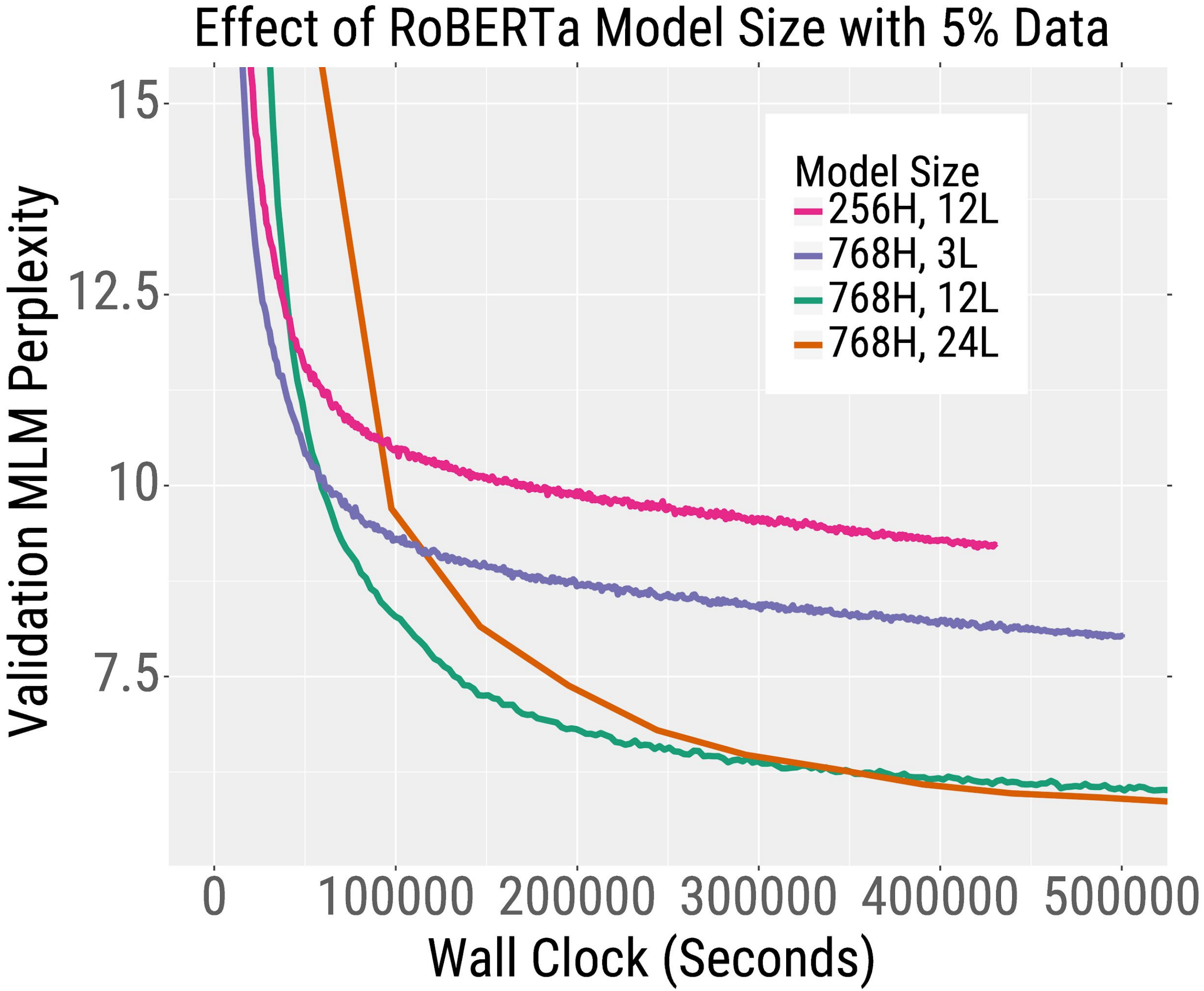

RoBERTa Quantization + Pruning
SST-2 Validation Accuracy
0.925
0.900
0.875
0.850
0
50
100
Bits x Parameter Count (Bits x Millions)
Original Size
3 Layers, 768H
6 Layers, 768H
12 Layers, 768H
18 Layers, 768H
24 Layers, 768H
12 Layers, 512H
12 Layers, 1024H
12 Layers, 1536H

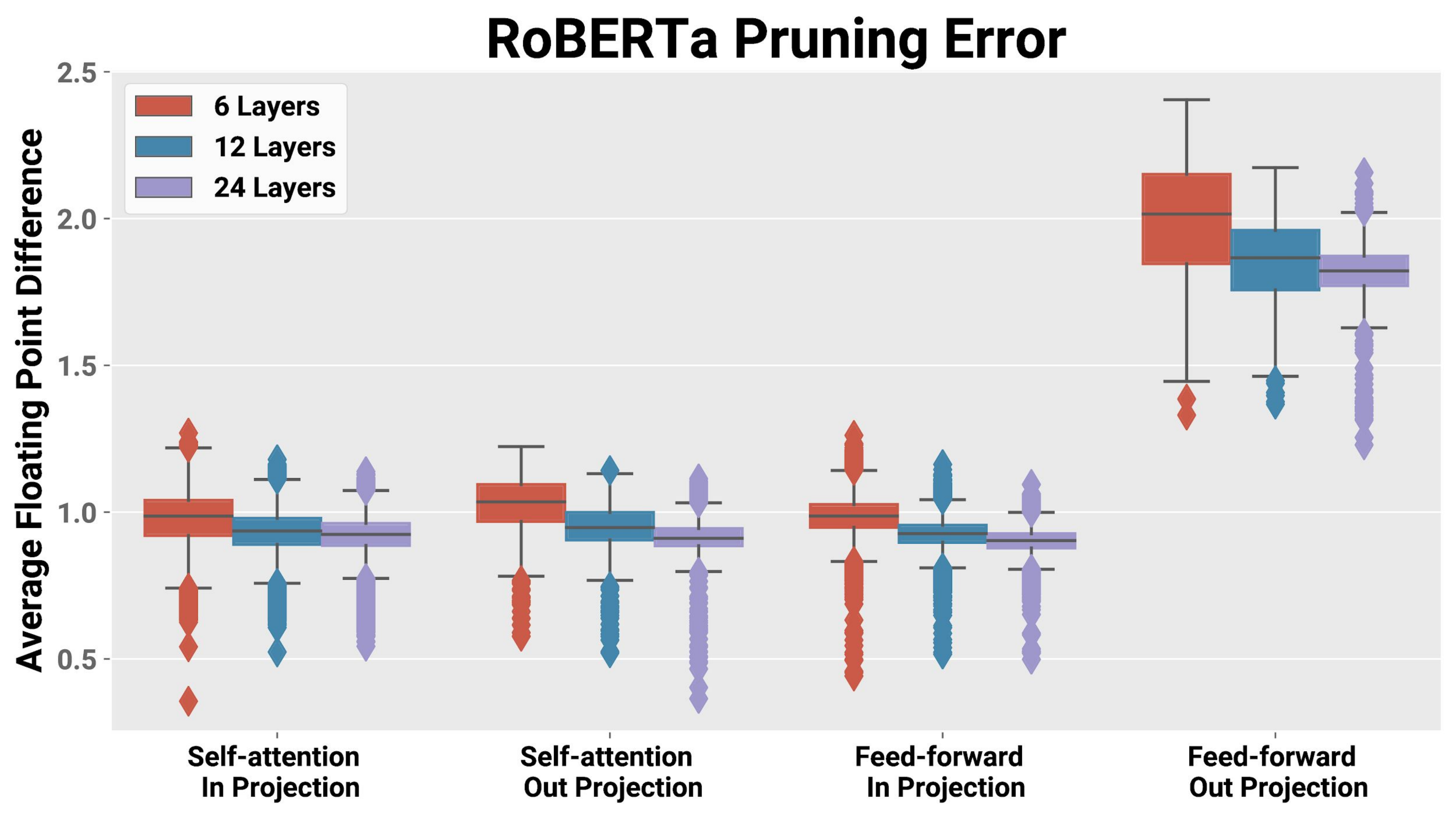


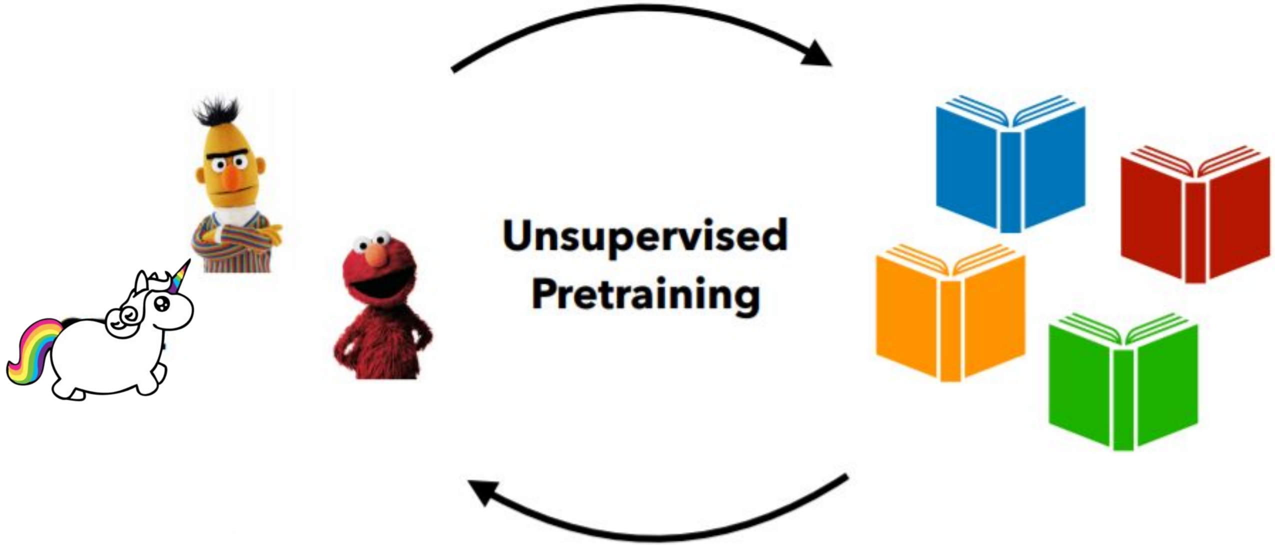
Effect of RoBERTa Model Size with 5% Data
Validation MLM Perplexity
15
12.5
10
7.5
0
25000
50000
75000
100000
Number of Gradient Steps
Model Size
256H, 12L
768H, 3L
768H, 12L
768H, 24L



Effect of RoBERTa Model Size with 1% Data
Model Size
256H, 12L
768H, 3L
768H, 12L
768H, 24L
Validation MLM Perplexity
25
20
15
10
0
10000
20000
30000
40000
Number of Gradient Steps


Effect of RoBERTa Model Size with 5% Data
Validation MLM Perplexity
15
12.5
10
7.5
0
100000
200000
300000
400000
500000
Wall Clock (Seconds)
Model Size
256H, 12L
768H, 3L
768H, 12L
768H, 24L


RoBERTa Pruning Error
6 Layers
12 Layers
24 Layers
Average Floating Point Difference
2.5
2.0
1.5
1.0
0.5
Self-attention In Projection
Self-attention Out Projection
Feed-forward In Projection
Feed-forward Out Projection


Unsupervised
Pretraining