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

Berkeley NLP

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?

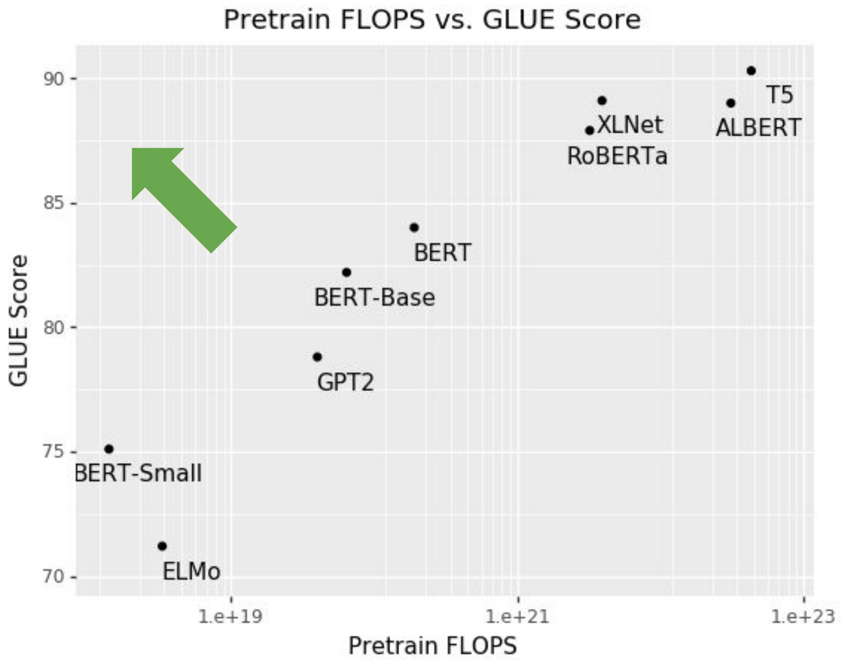
- One can **trade compute for accuracy**
  - Larger model size
  - Larger datasets
  - More training iterations
- **Computational constraints** are increasingly the bottleneck

Maximizing Computational Efficiency

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

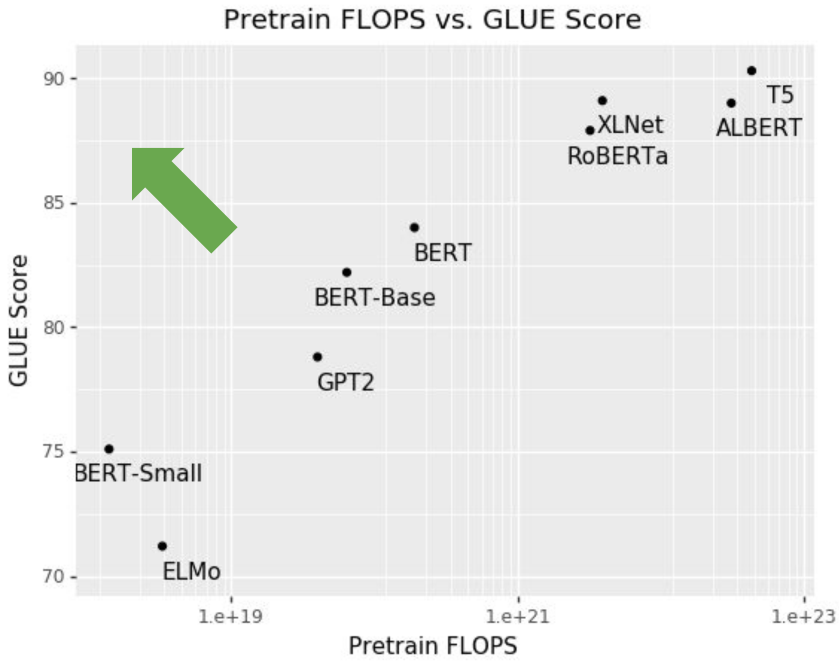
Maximizing Computational Efficiency

- 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

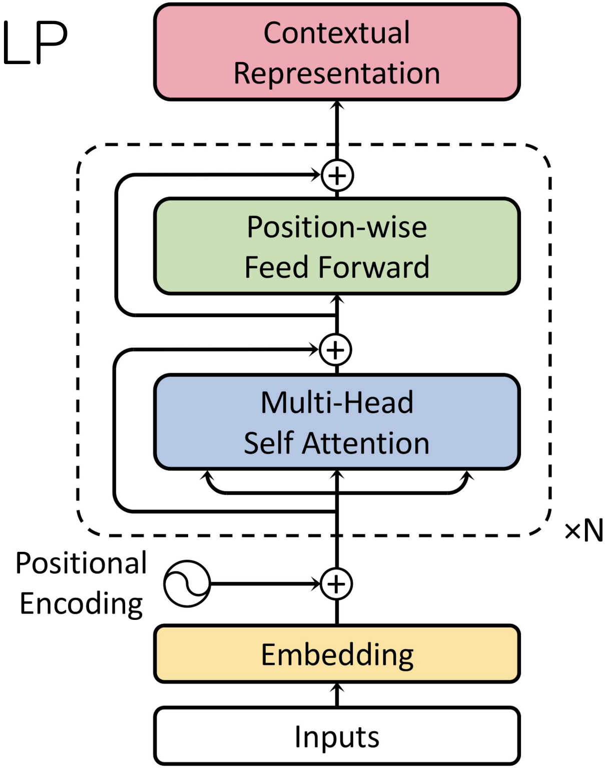
- Results:
  - **Large** models **train faster**
  - **Large** models are **efficient** at **inference** time

- Key idea: stop training early & compress heavily

Training Efficiency

Experimental Setup

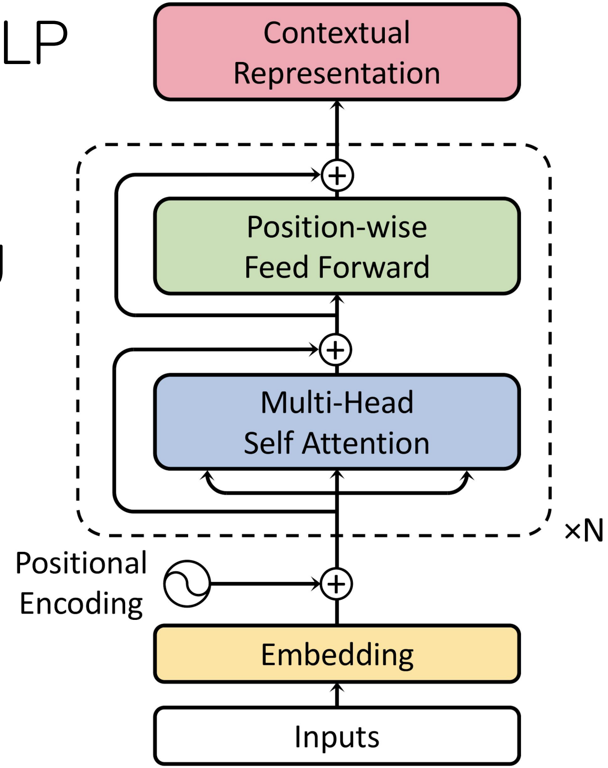
- Transformer models
  - Feedforward architecture, SoTA for NLP



Experimental Setup

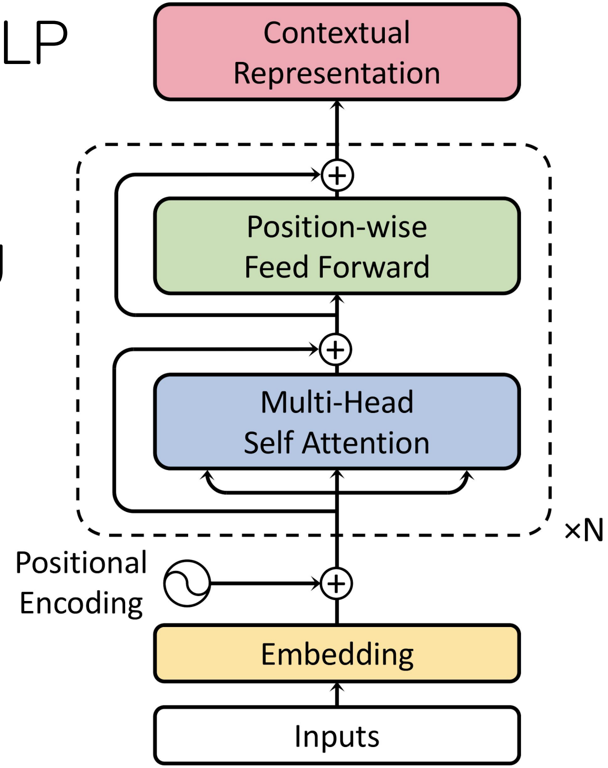
- Transformer models
  - Feedforward architecture, SoTA for NLP

- We vary hidden size + model depth
  increase batch size to fill the GPU

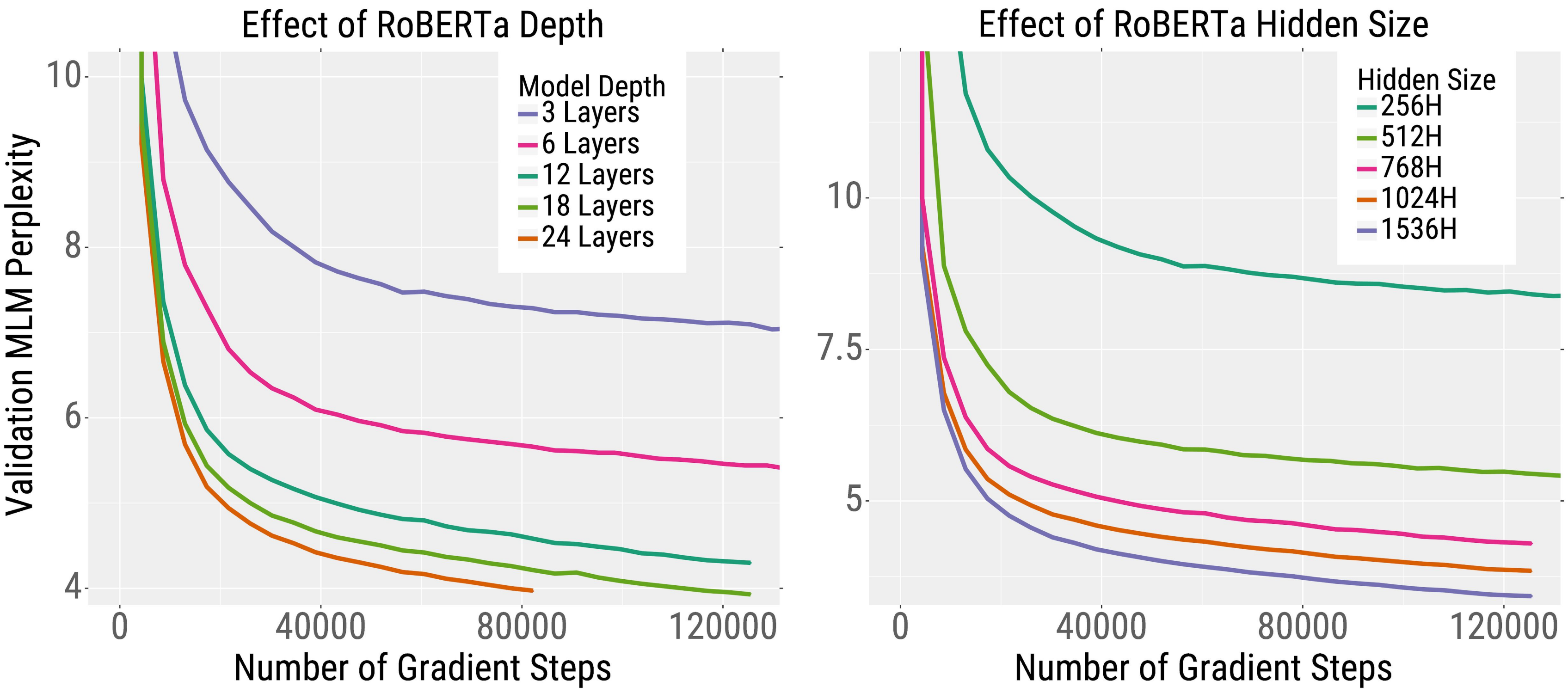


Experimental Setup

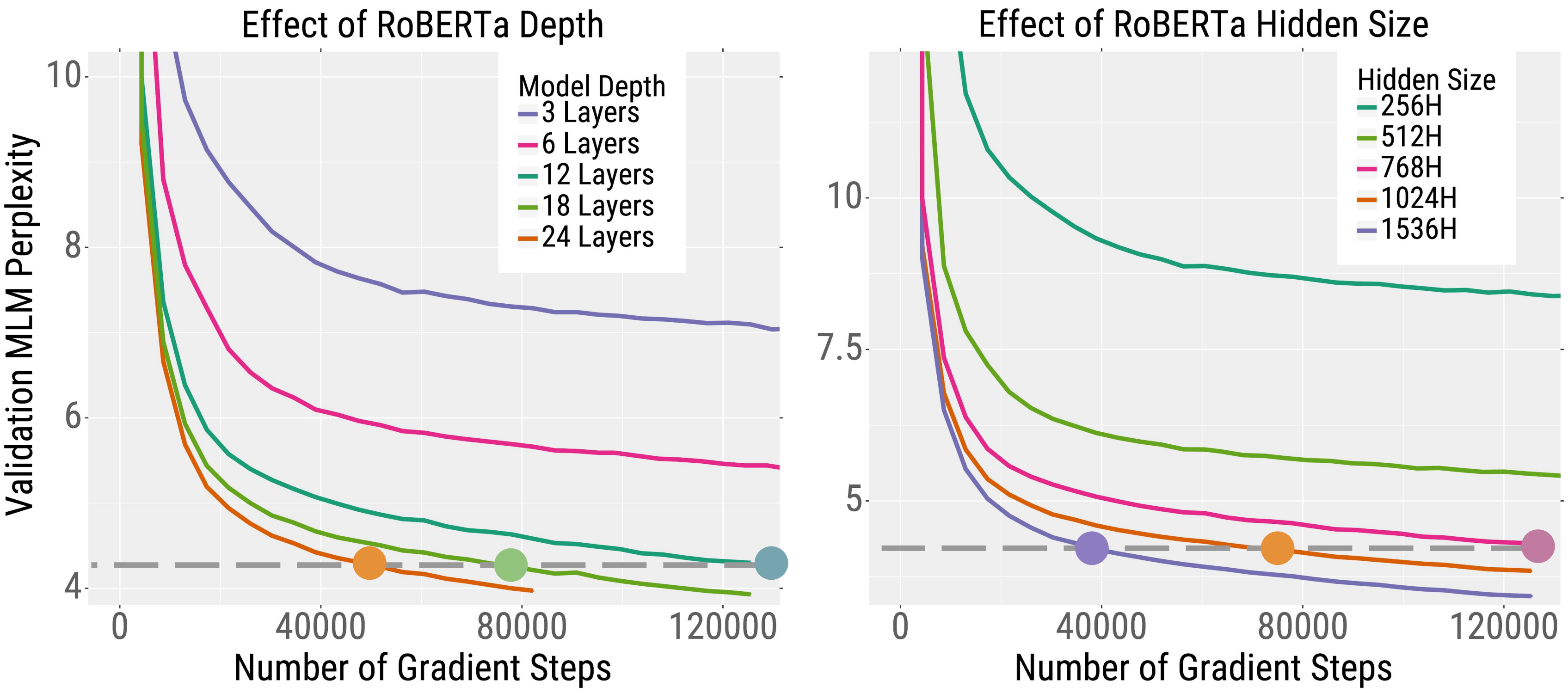
- Transformer models
  - Feedforward architecture, SoTA for NLP
- We vary hidden size + model depth
  increase batch size to fill the GPU
- Task 1: **MLM pretraining + finetuning (RoBERTa)**
- Task 2: **machine translation**



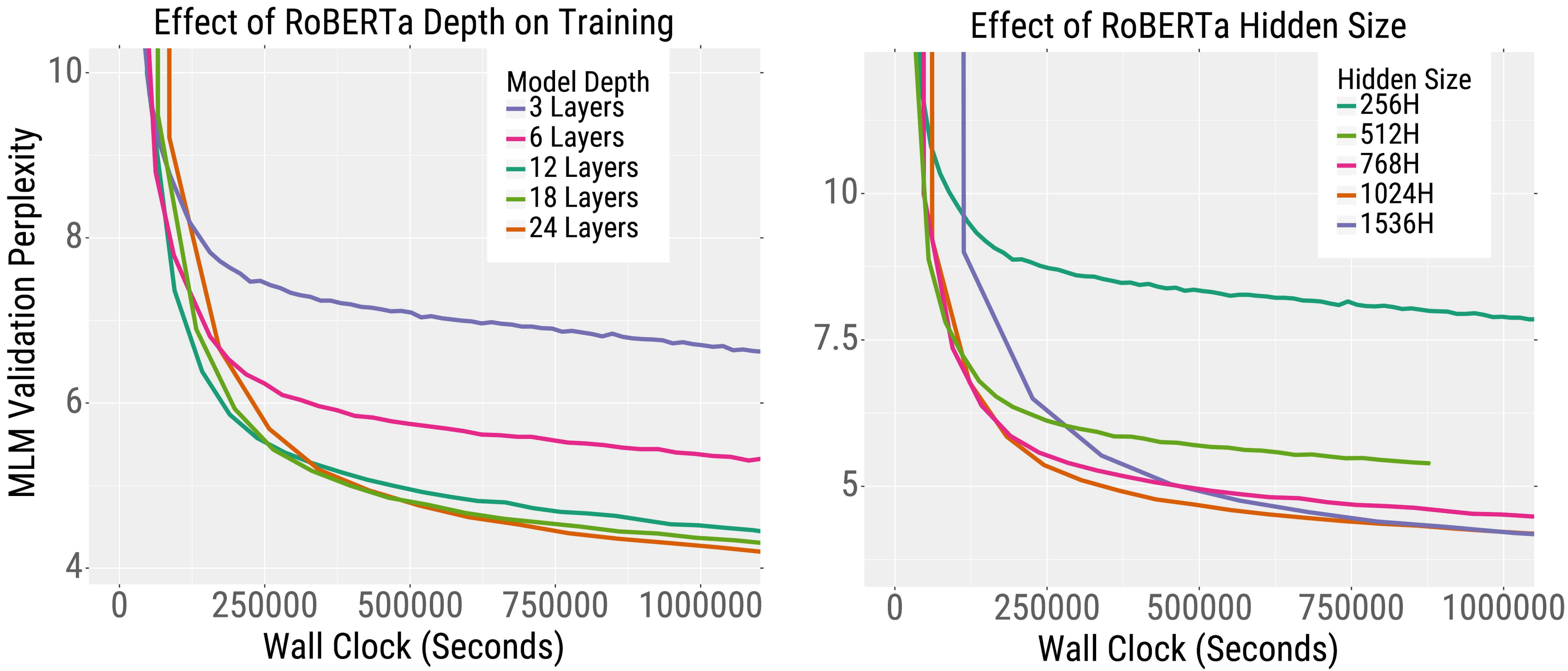
Deeper and Wider Models Converge in Fewer Steps



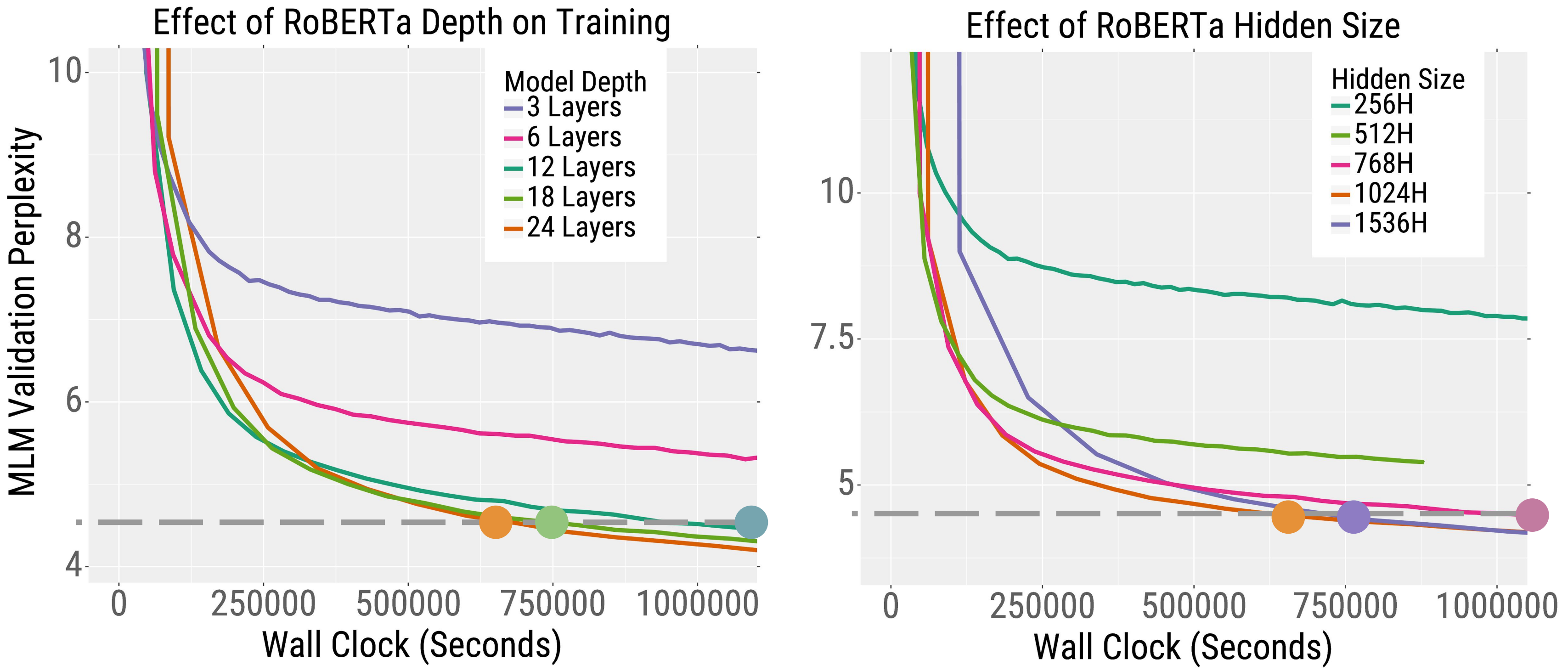
Deeper and Wider Models Converge in Fewer Steps



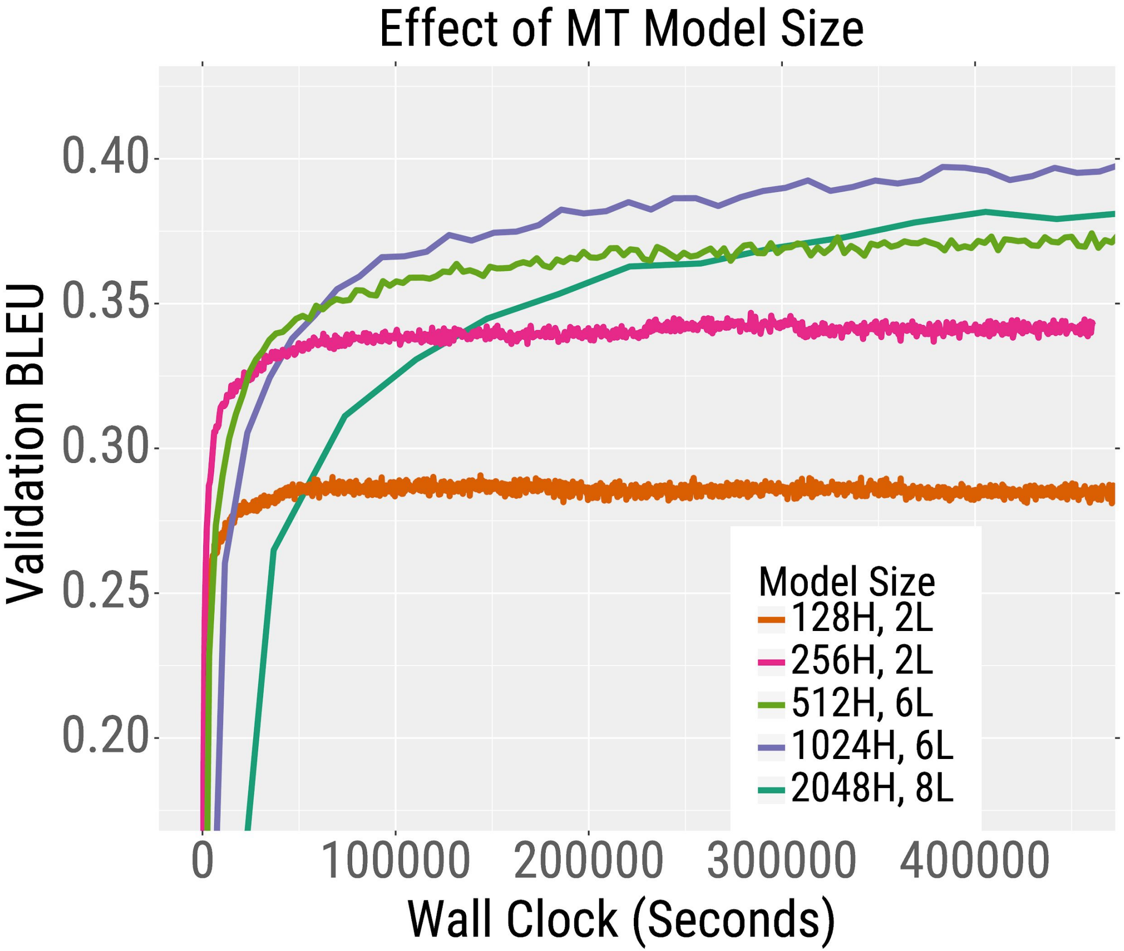
Deeper and Wider Models Converge in Less Wall Clock Time



Deeper and Wider Models Converge in Less Wall Clock Time



Same Trends Hold for Machine Translation



Why Do Larger Models Train Faster?

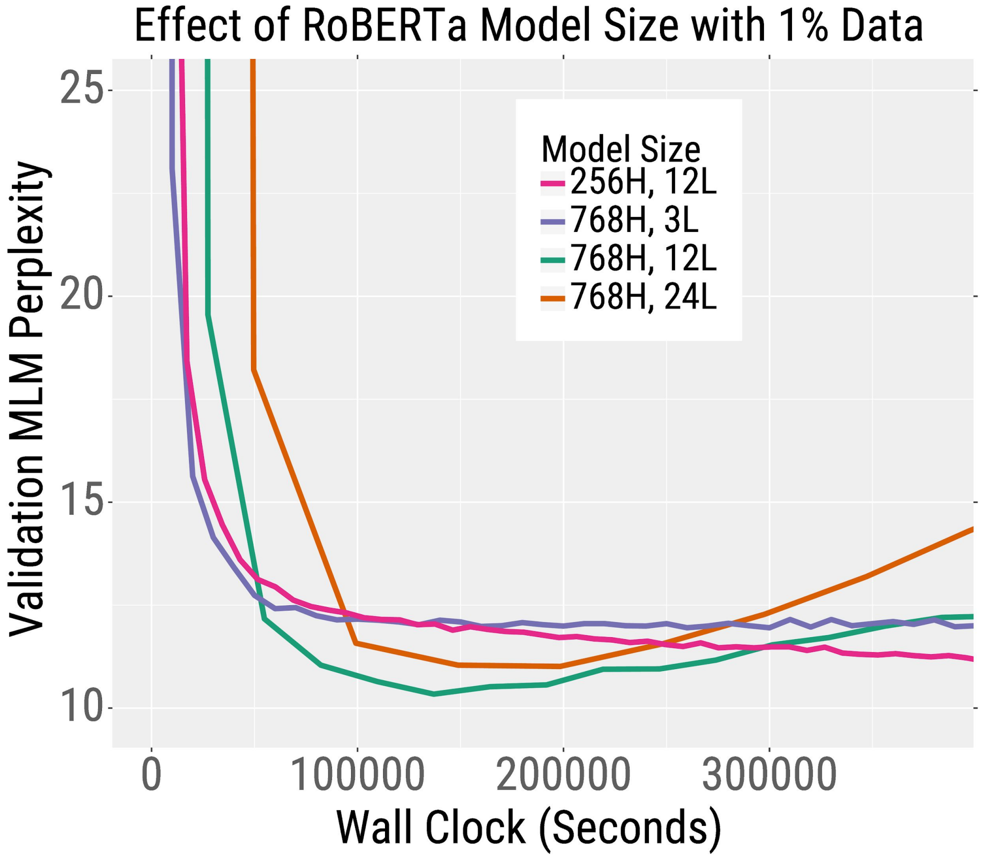
- Larger models reduce **training** error faster

Why Do Larger Models Train Faster?

- Larger models reduce **training** error faster
- MLM training has "**unlimited**" data ⟶ overfitting *not* a concern
- Thus, larger models also minimize **validation** error faster

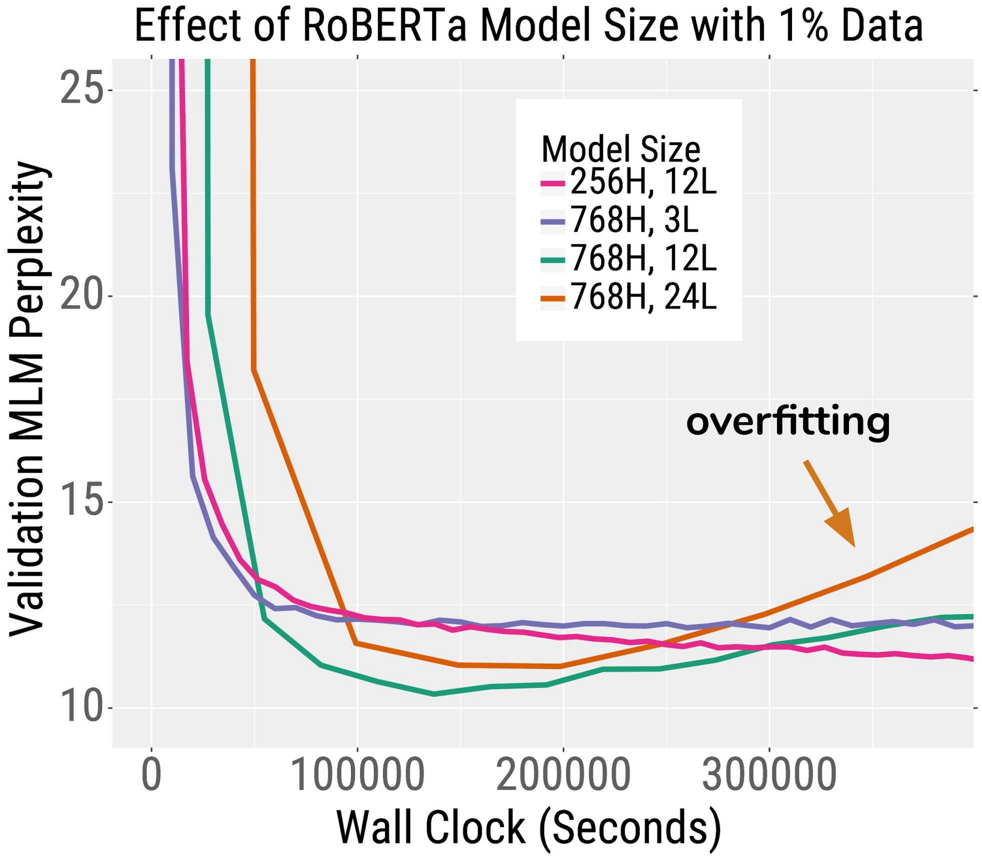
Why Do Larger Models Train Faster?

- 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



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

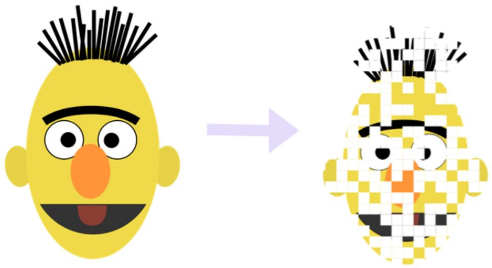
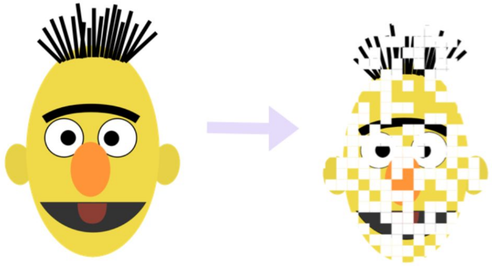


Image: Rasa

Experimental Setup

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



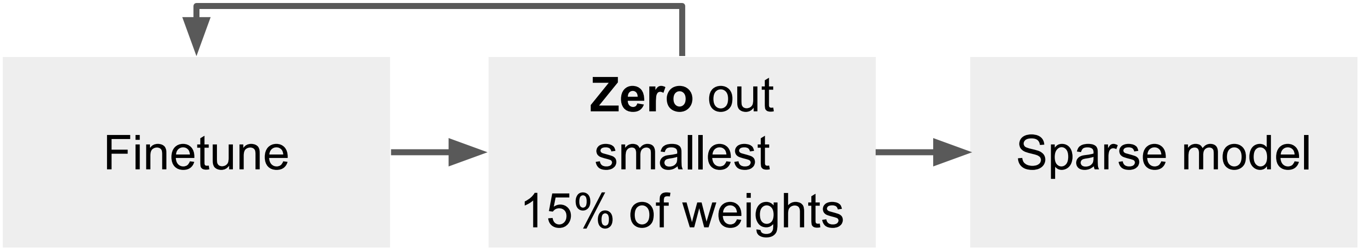


Image: Rasa

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

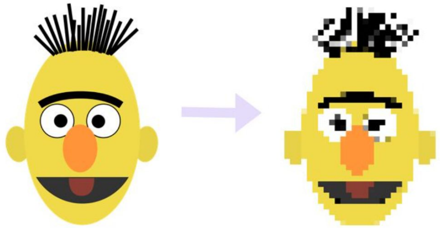
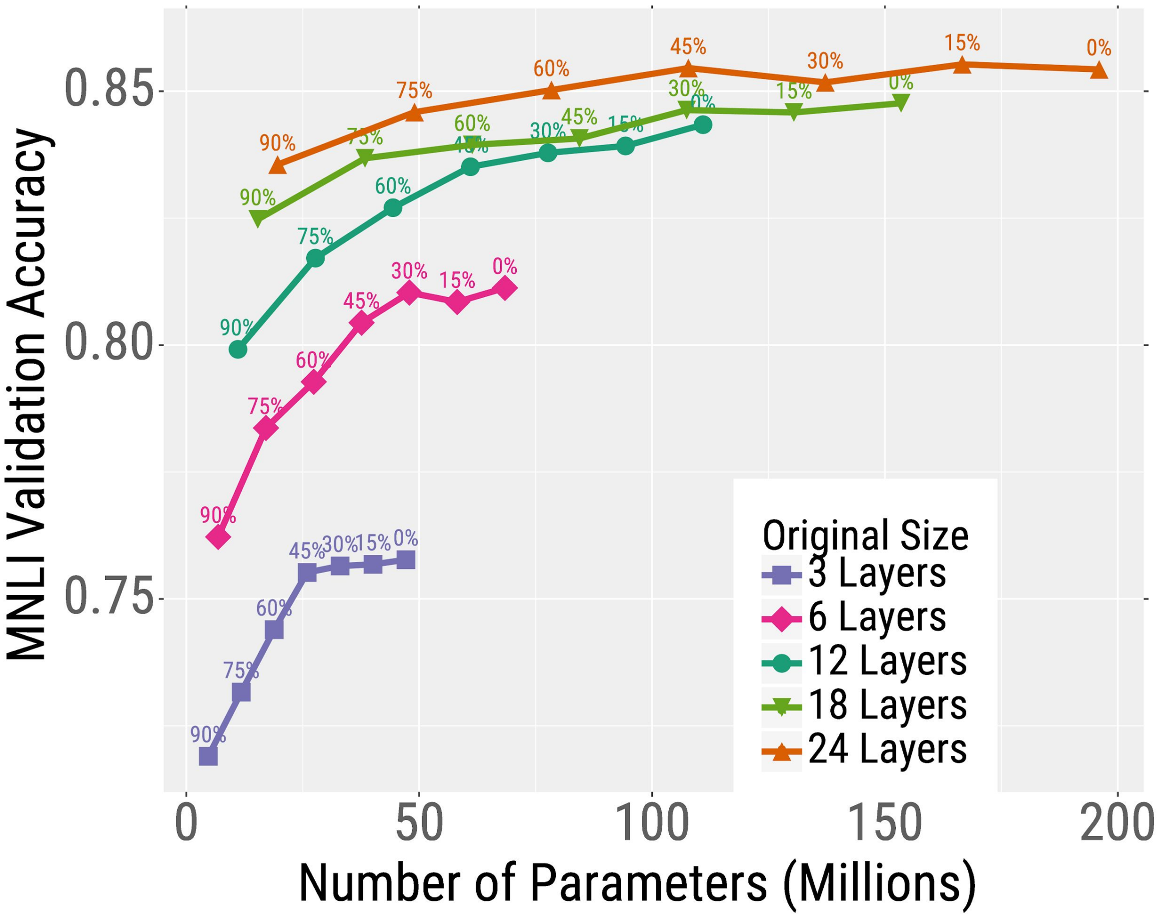
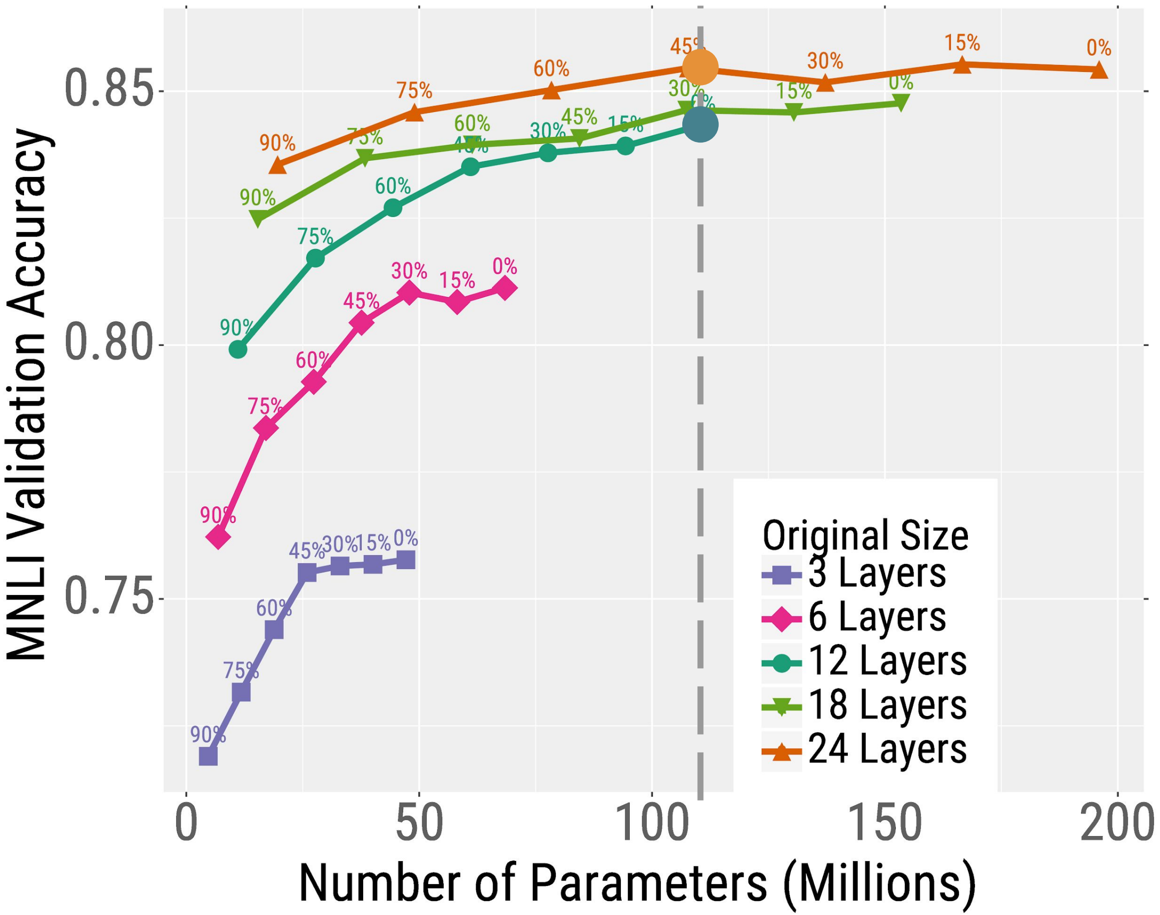


Image: Rasa

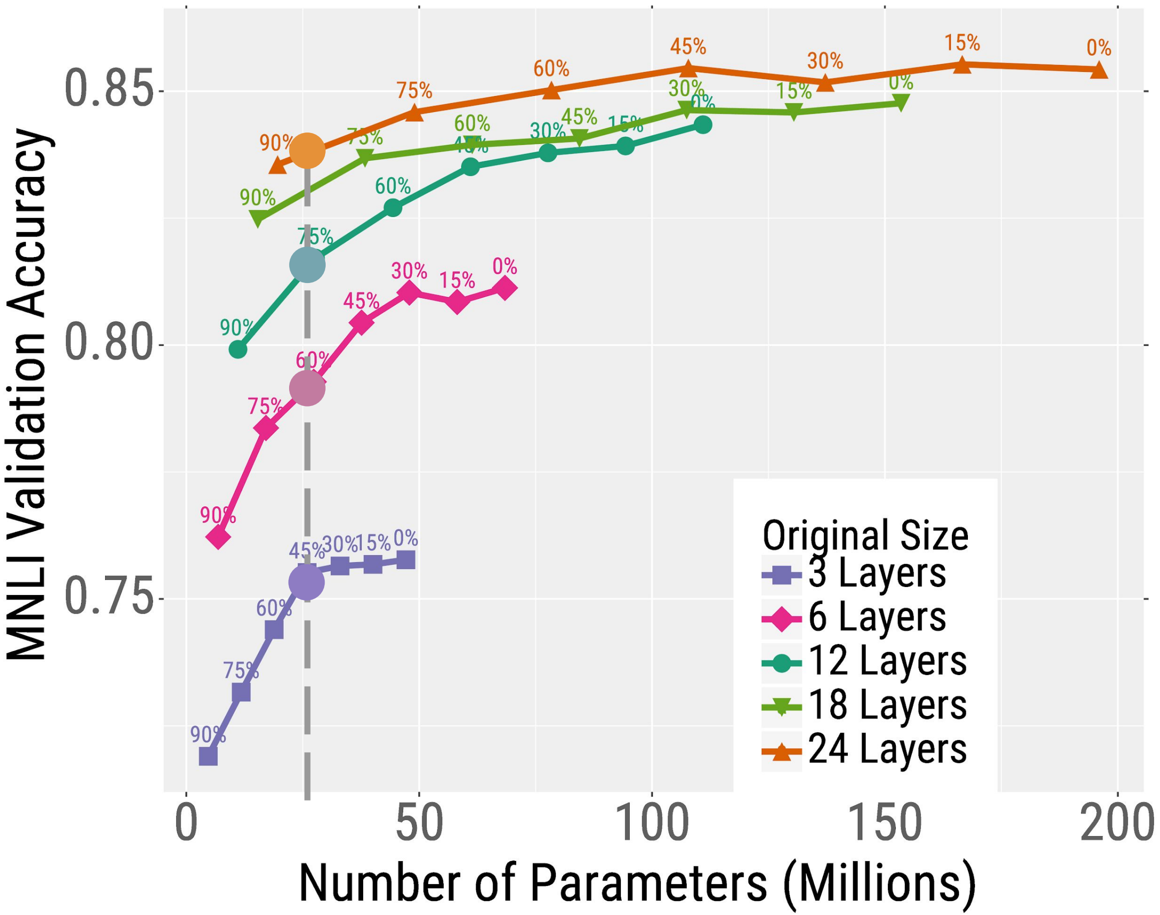
Deeper and Wider Models are More Robust to Pruning



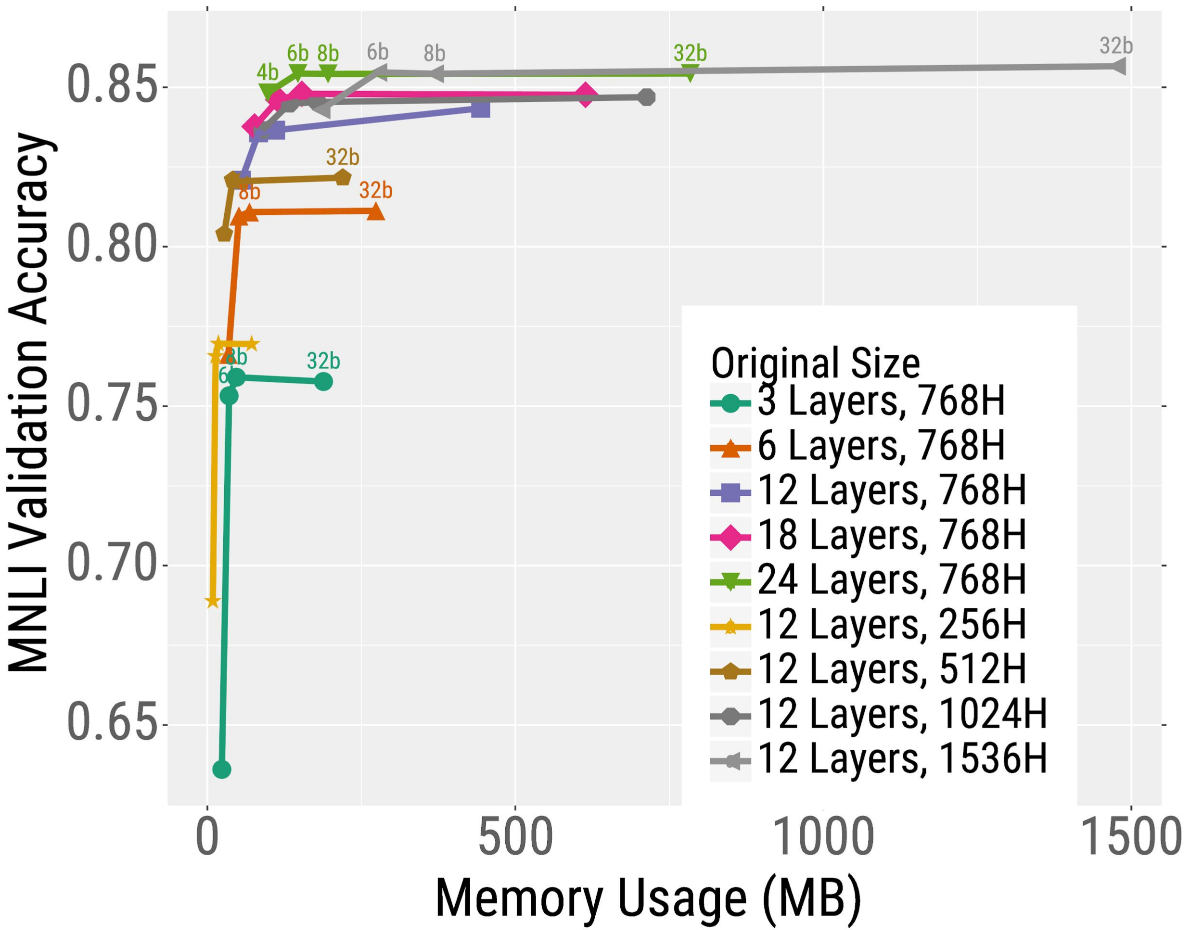
Deeper and Wider Models are More Robust to Pruning



Deeper and Wider Models are More Robust to Pruning

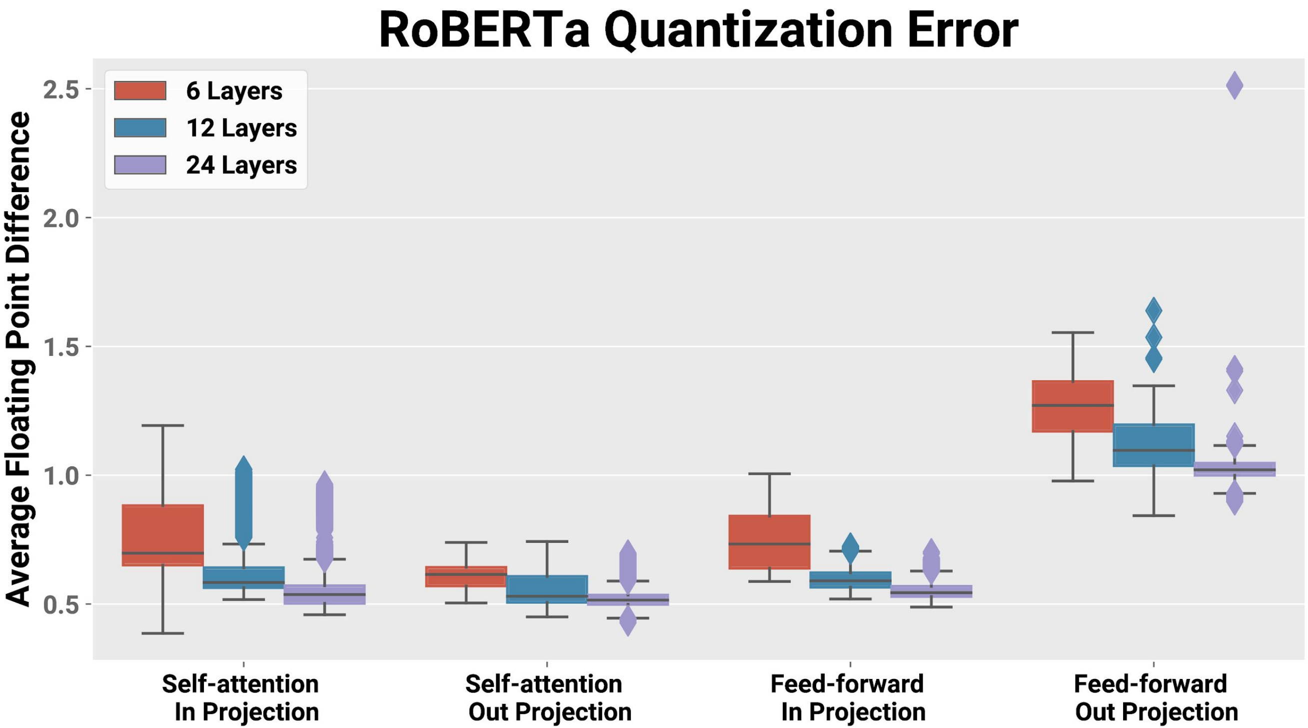


Deeper and Wider Models are More Robust to Quantization



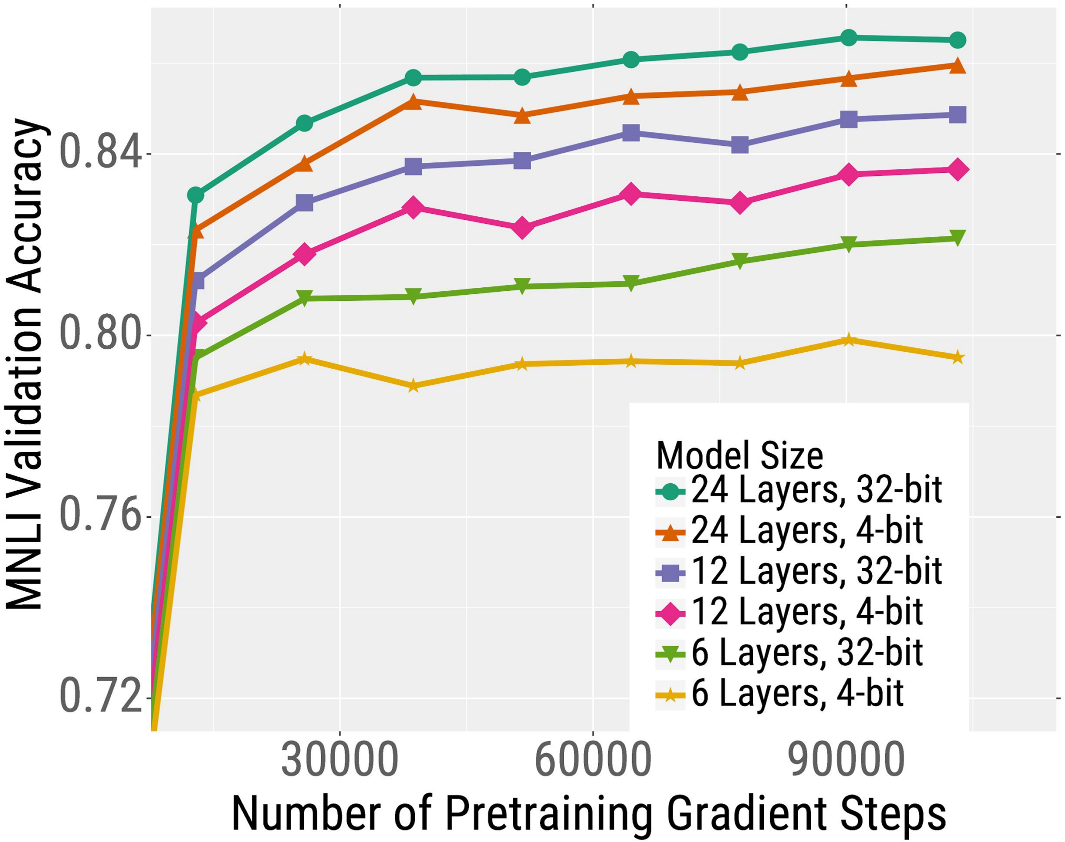
Why Do Larger Models Compress Better?

- Quantization/Pruning error is smaller for larger models



Why Do Larger Models Compress Better?

- Size, not convergence, determines compressibility



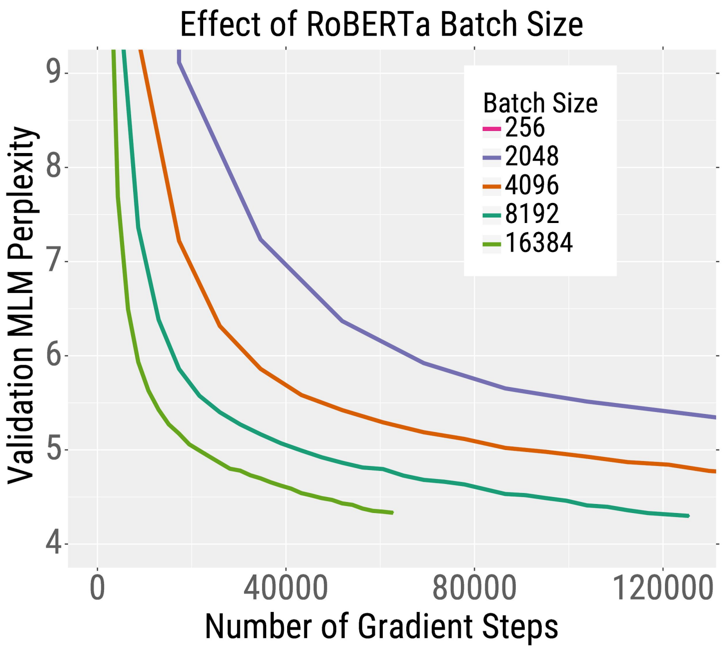
Practical Takeaways

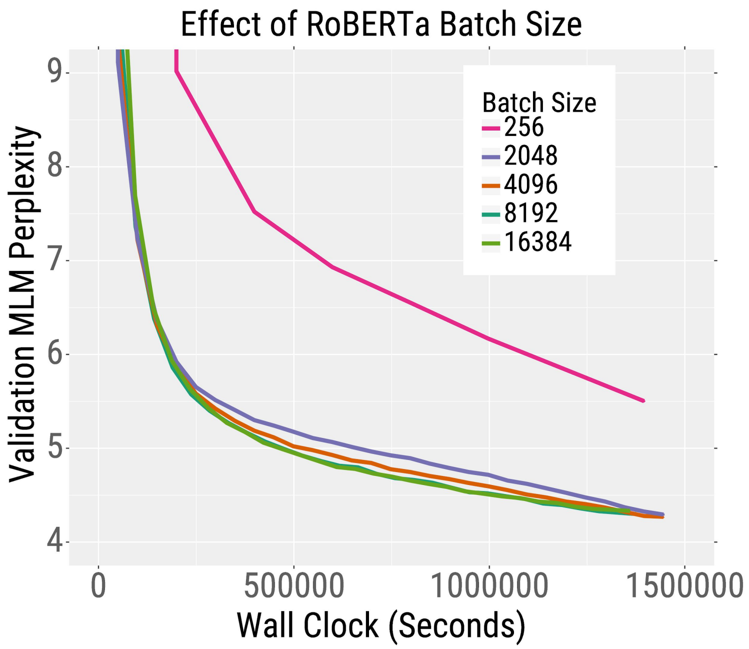
Practical Takeaways

- **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