УДК 004.8

ADAPTIVE AND EFFICIENT LANGUAGE MODELS: ARCHITECTURES FOR EDGE DEVICES

Isambetov Alexandr Vyacheslavovich
Group: ИИб-251 student, Institute of information technology, mechanical engineering and motor transport
Kuzzbass State Technical University, Kemerovo, Russia, alexshedmont@gmail.com
Shalina Violetta Vitalievna, Scientific supervisor: assistant of the department of foreign languages, Kuzzbass State Technical University

Introduction

Modern natural language processing has been revolutionized by Transformer-based LLMs* with hundreds of millions or billions of parameters. These models achieve state-of-the-art results, but their size and computational demands make deployment on resource-constrained devices impractical. For instance, GPT-3 model requires nearly 350GB of memory. Mobile and embedded devices typically have only a few GB of memory and limited GPU power, so naive execution of full-precision LLMs is infeasible. [1]

Yet on-device inference is highly desirable - it preserves user privacy, delivers instantaneous responses. For example, voice assistants need real-time text processing without cloud round-trips.

Running LLMs on-device faces challenges - limited RAM, VRAM*, memory, and most mobile processors lack high-throughput* floating-point units. To overcome these, researchers have developed compression techniques that shrink models while preserving accuracy. The most important approaches are quantization, knowledge distillation, and pruning. These can be used singly or in combination.

LLM - Large Language Model. A type of artificial intelligence program that is trained to work with text: understand, generate, retell, supplement, translate, and analyze it
VRAM — Video RAM, memory on GPU used for storing model weights/activations during inference.
throughput — the amount of work (e.g., inferences) a system can perform per unit time.

Quantization

Quantization compresses a model by using lower-bit representations. A simple form is converting 32-bit floats to 8-bit integers, reducing memory footprint by ~4x. In practice, quantization may use 8-bit, 4-bit, or mixed precision. [7]

Two main approaches exist:

1. Post-Training Quantization

Performed without additional training. Calibration data determines scaling factors for weights and activations.

2. Quantization-Aware Training

Quantization operations are simulated during fine-tuning* so the model adapts to lower precision.

Many mobile chips (ARM CPUs, DSPs, NPUs) include optimized integer units for 8-bit operations, enabling significant real-world speedups. Converting a BERT model to 8-bit can reduce file size to one-quarter of the original. This saves storage and reduces memory bandwidth during inference.

Very low bit-depth risks accuracy loss, but techniques like mixed precision or bias correction mitigate this. Most deployed models use 8-bit PTQ or QAT.

Key point - Quantization trades precision for large memory and speed gains. A BERT-base model (~440 MB) typically becomes ~110 MB in INT8, and inference can run several times faster on supported hardware. [7]

**fine-tuning* — continued training of a pre-trained or pruned model on task-specific data to recover or improve performance.

Knowledge Distillation

Knowledge distillation trains a small “student” network to imitate a large “teacher”. The student has fewer layers or smaller hidden dimensions. [1]

During distillation, the student learns to match the teacher's output probabilities or intermediate representations. The intuition is that the teacher's output distribution contains useful latent information.

Famous examples:

DistilBERT [1]

Has 6 layers instead of 12. ~66M parameters. ~40% smaller and faster than BERT-base. Almost identical accuracy on many tasks

TinyBERT [3]

Uses layerwise distillation; ~14M parameters while maintaining ~96% of BERT's accuracy.

MobileBERT [2]

Uses distillation from BERT-LARGE.
- ~25M parameters
- 4.3× smaller and 5.5× faster than BERT-base
- Small accuracy drop

MiniLM

Distills attention and value matrices. Retains >99% of teacher's performance with far fewer parameters.

Empirically, a distilled student model almost always outperforms a small model trained from scratch on the same data.

Key point: Distillation produces compact models with strong performance. For example, DistilBERT is 40% smaller, and MobileBERT is over 4× smaller with only minor accuracy loss. [1] [2]

Pruning

Pruning removes redundant parameters. [12]

1. **Unstructured pruning [11]**

Zeros out individual weights with little impact.
- Produces sparse matrices*
- Reduces storage
- Speedups require special sparse kernels or hardware

2. **Structured pruning [11]**

Removes entire neurons, attention heads, or layers.
- Produces a smaller dense model
- Works on all standard hardware
- Provides real inference speedups

Studies demonstrate substantial redundancy in Transformers. Pruning attention heads or intermediate dimensions can reduce size significantly with minor accuracy loss.

For example, multi-granular pruning methods have achieved >5× speedup with small accuracy reductions.

Pruning is usually followed by fine-tuning to recover accuracy. Combining pruning with distillation often yields better results than using either alone.

Key point: Pruning shrinks the architecture itself. Structured pruning gives tangible speedups while preserving most accuracy. [11]

**sparse matrices* — matrices with many zero entries; enable storage reduction but often need special kernels for runtime speedup.

Combined Techniques

In practice, optimal compression pipelines combine multiple methods. A common strategy:

1. Distill to a smaller model (BERT → DistilBERT).
2. Quantize to INT8*.
3. Prune unnecessary heads or layers. [1]

Research consistently shows combined pipelines outperform any individual technique. Alternating pruning and distillation can produce highly compact models with strong accuracy. [8]

Key point: State-of-the-art SLMs generally combine all three methods for maximum efficiency.

**INT8* — 8-bit integer representation commonly used in quantized models to reduce memory and accelerate inference on specialized hardware.

Conclusion

Bringing large language models to edge devices requires multiple complementary strategies. Quantization reduces memory by up to 4x, distillation cuts model size roughly in half while maintaining high accuracy, and pruning removes architectural redundancy. Combined, these techniques produce compact, fast, and accurate Small Language Models suitable for real-time on-device inference. [12]

Modern edge-optimized Transformers, like MobileBERT and SqueezeBERT - demonstrate multi-fold speedups and small memory footprints while preserving strong NLP performance. Compression pipelines routinely yield models under 100 MB that run smoothly on mobile processors. [2]

As hardware and methods continue to evolve, on-device LMs will approach cloud-level quality while operating within strict resource budgets.

References

1. Sanh V., Debut L., Chaumond J., Wolf T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter [Electronic resource]. – 2019. – https://arxiv.org/abs/1910.01108 [1]
2. Sun Z., Yu H., Song X., Liu R., Yang Y., Zhou D. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices [Electronic resource]. – 2020. – https://arxiv.org/abs/2004.02984 [2]
3. Jiao X., Yin Y., Shang L., Jiang X., Chen X., Li L., Wang F., Liu Q. TinyBERT: Distilling BERT for Natural Language Understanding [Electronic resource]. – 2020. – https://arxiv.org/abs/2004.08950 [3]
4. Iandola F.N., Kazemzadeh S., Ashraf K., Dahlen A., Mostajabi M., Nelson A., Keutzer K. SqueezeBERT: What can computer vision teach NLP about efficient

neural networks? [Electronic resource]. – 2020. – https://arxiv.org/abs/2006.11316
5. Wang S., Cao S., Wu Y., Lin Z., Liu Y., Yang M. Sparsifying Pretrained Transformers via Iterative Hard Thresholding [Electronic resource]. – 2020. – https://arxiv.org/abs/2004.07842
6. Hugging Face. Optimum Library Documentation [Electronic resource]. – https://huggingface.co/docs/optimum/index
7. Dettmers T., Zettlemoyer L., Lewis M., Yih W. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale [Electronic resource]. – 2022. – https://arxiv.org/abs/2208.07339
8. Kurtic E., Saveliev E., Oglic D., Osindero S. Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning [Electronic resource]. – 2022. – https://arxiv.org/abs/2201.00299
9. Zhang S., Sun A., Tay Y., Yu P.S. Deep learning based recommender system: A survey and new perspectives // ACM Computing Surveys. – 2019. – Vol. 52, №1. – P. 1–38.
10. Turc I., Chang M.W., Lee K., Toutanova K. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models [Electronic resource]. – 2019. – https://arxiv.org/abs/1908.08962
11. Xiong R., Yang Y., He D., Zheng K., Zhang H., Lan Y., Wang L., Liu T.Y. LayerDrop: Structured pruning with a dropout-based regularization [Electronic resource]. – 2020. – https://arxiv.org/abs/1909.11556 [11]
12. Frantar E., Alistarh D. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-shot [Electronic resource]. – 2023. – https://arxiv.org/abs/2301.00774