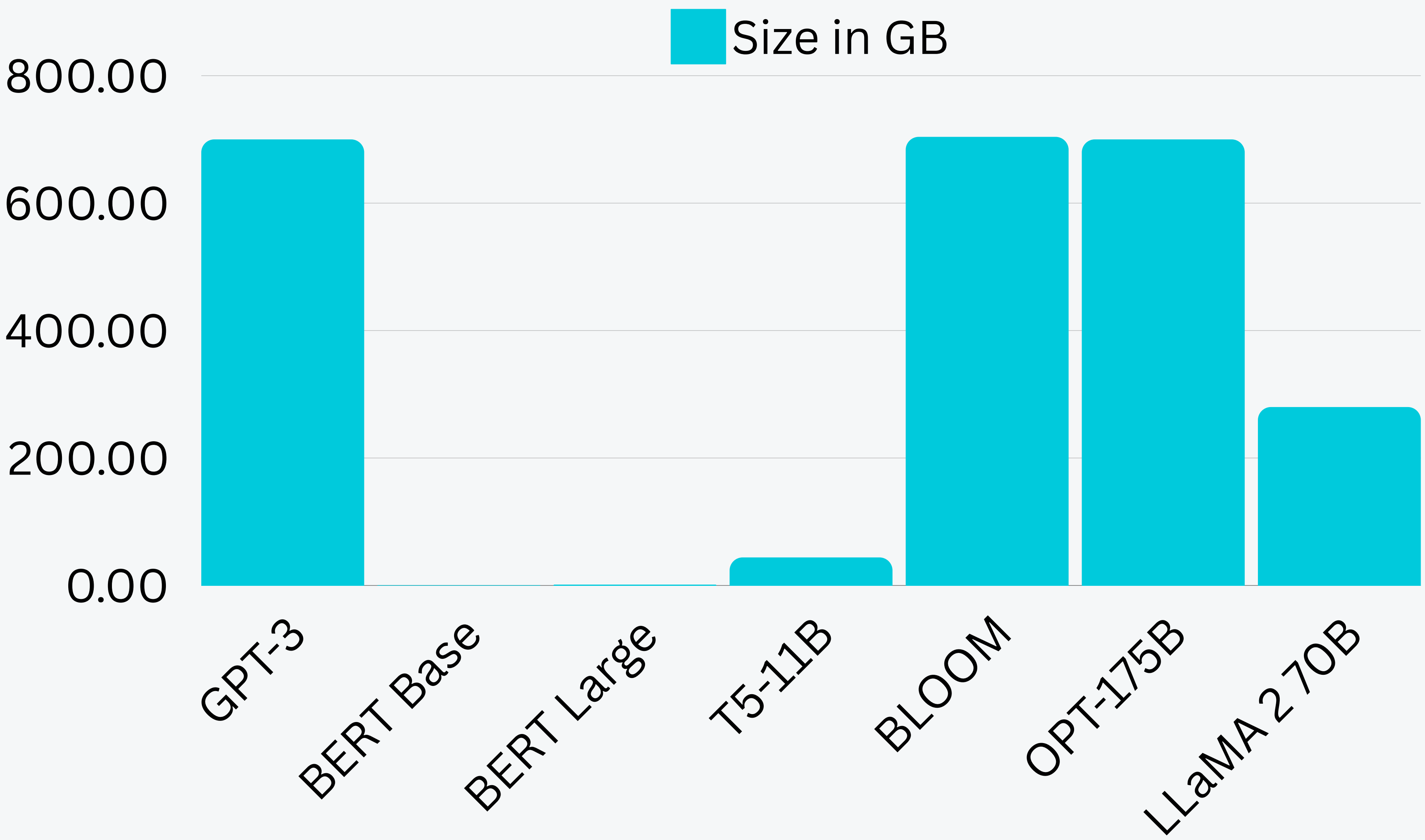
POST 36 - GEN AI

GENERATIVE AI
FOR ALL

Large Language Model(LLM) Size:

The Internal Mechanics of AI Model Storage



DINESHLAL

Introduction: What is Model Storage Size?

- **The storage size of a generative AI model refers to the amount of memory required to store the trained model on a disk or in memory.**
- It is influenced by several internal features, mainly the parameters (weights and biases), but other factors like parameter precision and additional components such as embeddings also play a significant role.

Key factors affecting model storage size

- **Number of parameters:** More parameters (weights and biases) increase storage size.
- **Precision of parameters:** Lower precision (e.g., FP16 or INT8) decreases storage requirements.
- **Model architecture:** Complex architectures (e.g., multiple dense layers, large attention heads, extensive embeddings) lead to larger storage needs.
- **Compression techniques:** Methods like quantization, pruning, and knowledge distillation help reduce storage size without drastically compromising model performance.

Number of Parameters

Number of Parameters: Weights and Biases

- **Weights**: In neural networks, weights form the bulk of the parameters, particularly in dense layers and attention mechanisms in generative models like Transformers.
- **Biases**: While biases are fewer in number compared to weights, they still contribute to the total parameter count.

Number of Parameters

Storage Calculation:

- For a model with NNN parameters, the storage size is influenced by both the number of parameters and their precision: Storage **Size=N×Precision**

Example Calculation

- Suppose a model has 10 million parameters (weights + biases), and each parameter is stored as a 32-bit (4-byte) floating-point number. **Storage Size=10,000,000×4 bytes=40 MB**

Precision of parameters

Floating-Point Precision

- **32-bit (FP32) precision:** Commonly used during training for higher numerical accuracy.
- **16-bit (FP16) precision:** Often used during inference or in hardware-efficient models to reduce storage requirements while maintaining adequate accuracy.
- **8-bit (INT8) quantization:** Further compresses the model by representing parameters as 8-bit integers, significantly reducing storage size but with potential impacts on model performance.

Precision of parameters

Storage Impact of Precision

- The precision of each parameter dictates how much storage is needed:
- **FP32 (32 bits):** 4 bytes per parameter.
- **FP16 (16 bits):** 2 bytes per parameter, halving the storage size compared to FP32.
- **INT8 (8 bits):** 1 byte per parameter, reducing the storage size by a factor of 4 compared to FP32.

Model Architecture

Number and Type of Layers

- **Fully Connected (Dense) Layers**: These layers often have the most parameters, contributing significantly to storage size. The number of neurons and the way they are interconnected (weights) directly impact size.
- **Convolutional Layers:** In convolutional neural networks (CNNs), the number of filters, kernel sizes, and strides affect the total number of parameters, albeit typically fewer than dense layers.

Model Architecture

Number and Type of Layers

- **Attention Mechanisms:** Generative models like Transformers utilize attention layers, where storage size depends on the number of attention heads, hidden dimensions, and sequence lengths.

Model Architecture

Embedding Layers

- **Embeddings**: In models dealing with text (e.g., GPT), word or token embeddings are stored as matrices. The size of these matrices depends on:
- **Vocabulary size:** Total number of unique words or tokens.
- **Embedding dimension:** Length of each word's vector representation.

Model Architecture

Storage Calculation:

Embedding Storage = Vocabulary Size×Embedding Dimension×Precision

For example, an embedding layer with a vocabulary of 50,000 words, an embedding dimension of 300, and 32-bit precision:

Storage=50,000×300×4 bytes=60 MB

Compression Techniques

To optimize storage size, various techniques can be applied:

- **Quantization**
- Converts parameters to lower precision (e.g., FP16 or INT8) without significantly impacting model performance. This reduction in bit-width directly reduces the storage size.
- Example: Moving from FP32 to INT8 reduces storage requirements by 75%.

Compression Techniques

To optimize storage size, various techniques can be applied:

- **Pruning**
- Removes less important parameters (those close to zero), creating a sparse model that requires fewer storage resources.
- The storage size reduction depends on the amount of pruning applied and the sparsity pattern.

DINESHLAL

Compression Techniques

To optimize storage size, various techniques can be applied:

- **Knowledge Distillation**
- Trains a smaller "student" model using the outputs of a larger "teacher" model. The resulting student model has fewer parameters, thus requiring less storage.

Compression Techniques

To optimize storage size, various techniques can be applied:

- **Model Compression Formats**
- Checkpoint files: Typically contain model parameters and may include extra information for training. Depending on format and compression (e.g., .ckpt, .pt, .h5), the storage size varies.
- ONNX (Open Neural Network Exchange): A format that allows saving models in an optimized, portable format, sometimes reducing storage size further due to optimized graph representations.

Conclusion

- By understanding and manipulating these internal features, one can optimize model storage size, balancing the trade-offs between model performance and resource efficiency.
- This is particularly important when deploying models on devices with limited memory, such as mobile phones, IoT devices, or embedded systems.

DINESHLAL

THANK YOU

- Special thanks to Gemini and ChatGPT for all the help on content
- Follow along for more informative articles in Generative AI space