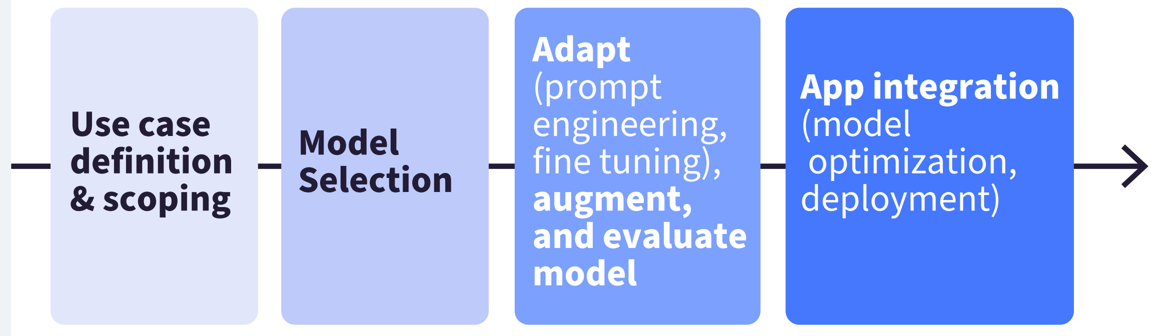
dataiku

LLM Compute Challenges and Scaling Laws

LARGE LANGUAGE MODEL CHOICE

Generative AI Project Lifecycle


Use case definition & scoping
Model Selection
Adapt (prompt engineering, fine tuning), augment, and evaluate model
App integration (model optimization, deployment)

Two options for model selection

- Use a pre-trained LLM.
- Train your own LLM from scratch.

But, in general...
...develop your application using a **pre-trained LLM,** except if you work with extremely specific data (i.e., medical, legal)

Hubs: Where you can browse existing models

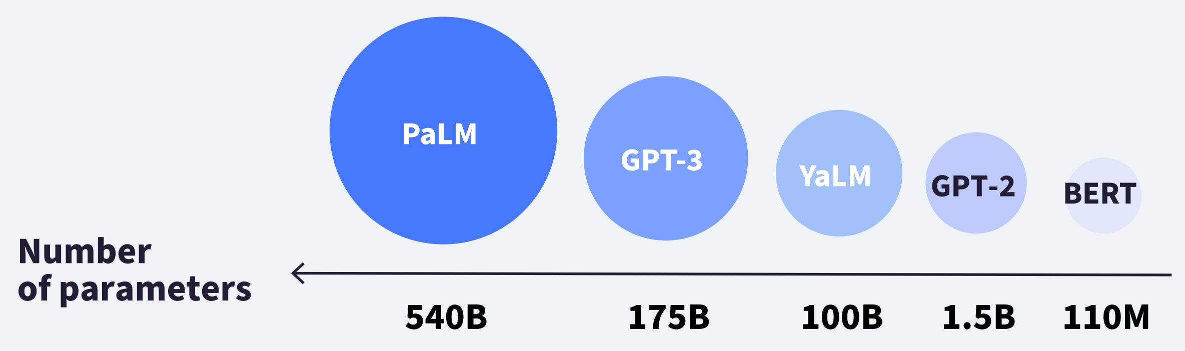
→ **Model Cards:** *List of best use cases, training details, limitations on models.*

The model **choice** will **depend on the details of the task** to carry out.

Model pre-training:

Model weights are adjusted in order to minimize the loss of the training objective.

It requires significant computational resources, (i.e., GPUs, due to high computational load).



COMPUTATIONAL CHALLENGES

Memory Challenge

```
RuntimeError : CUDA out of memory
```

→ LLMs are massive and require plenty of memory for training and inference.

To load the model into GPU RAM:

1 parameter (32-bit precision) = **4 bytes needed**
1B parameters = 4 x 10^9 bytes = 4GB of GPU

Pre-training requires storing additional components, beyond the model's parameters:

- Optimizer states (e.g., 2 for Adam)
- Gradients
- Forward activations
- Temporary variables

This could result in an additional **12-20 bytes of memory needed** per model parameter.

This would mean it requires **16 GB to 24 GB** of GPU memory to train a 1-billion parameter LLM, around **4-6x** the GPU RAM needed just for storing the model weights.

Hence, the memory needed for LLM training is:

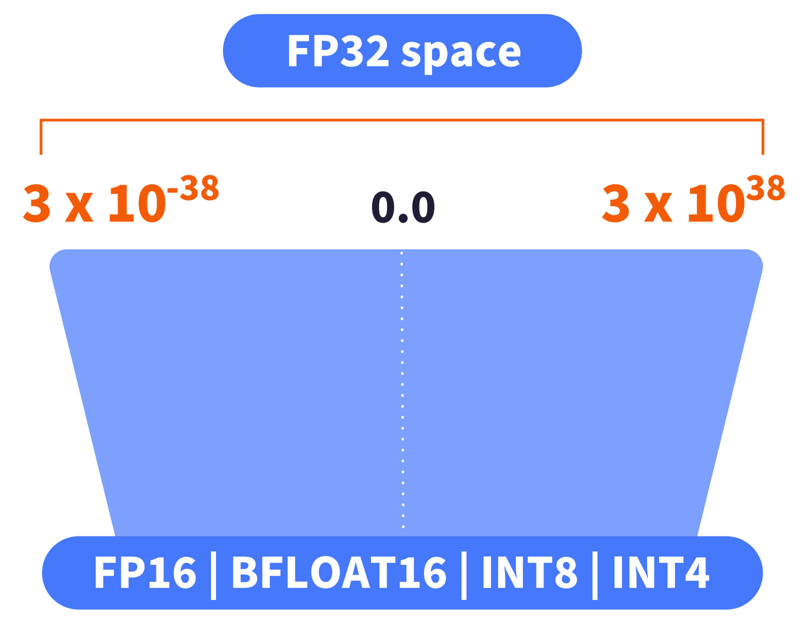
→ Excessive for consumer hardware

→ Even demanding for data center hardware (for single processor training).
For instance, NVIDIA A100 supports up to 80GB of RAM.

QUANTIZATION

How can you reduce memory for training?

Quantization: Decrease memory to store the weights of the model by converting the precision from 32bit to 16bit or 8bit integers.



Quantization maps the FP32 numbers to a lower precision space by employing scaling factors determined from the range of the FP32 numbers.

→ In most cases, quantization **strongly reduces memory requirements** with a **limited loss** in prediction.

BFLOAT16 is a popular alternative to FP16:

- Developed by Google Brain
- Balances memory efficiency and accuracy
- Wider dynamic range
- Optimized for storage and speed in ML tasks

e.g., FLAN T5 pre-trained using BFLOAT16

Benefits of quantization:

→ Less memory

→ Potentially better model performance

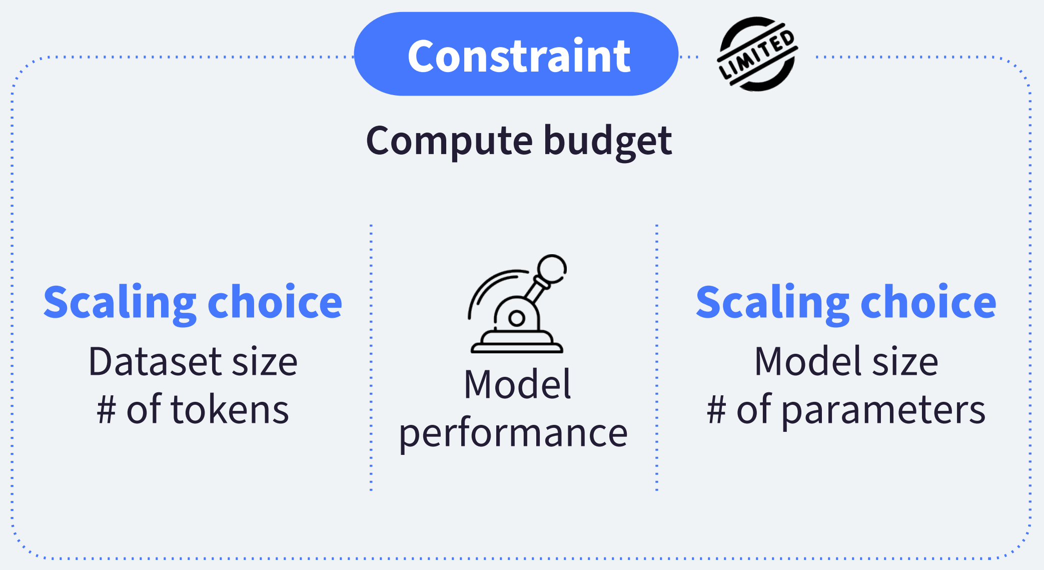
→ Higher calculation speed

SCALING LAWS

How big do the models need to be?
The goal is to maximize model performance.

Researchers explored trade-offs between the dataset size, the model size, and the compute budget:

Increasing compute may seem ideal for better performance, but practical constraints like hardware, time, and budget limit its feasibility.



It has been empirically shown that, as the compute budget remains fixed:

→ **Fixed model size:** Increasing training dataset size improves model performance.

→ **Fixed dataset size:** Larger models demonstrate lower test loss, indicating enhanced performance.

What's the optimal balance?

Once scaling laws have been estimated, we can use the **Chinchilla approach**, i.e., we can choose the dataset size and the model size to train a **compute-optimal model**, which maximizes performance for a given compute budget. The compute-optimal training dataset size is ~20x the number of parameters.

© 2024 Dataiku