2 Background

1. Pre-training, adaptation tuning, utilization, and capacity evaluation.
2. **1**: LLMs display some surprising emergent abilities that may not be observed in previous smaller PLMs, which makes AI algorithms unprecedently powerful and effective; **2**: LLMs would revolutionize the way that humans develop and use AI algorithms; **3**: the development of LLMs no longer draws a clear distinction between research and engineering.
   In NLP, LLMs can serve as a general-purpose language task solver; in IR, AI chatbots offer new information seeking way; in CV, ChatGPT-like vision-language models are being developed to better serve multimodal dialogues
3. A mixture of diverse public textual including: webpages, conversation data, books&news, scientific data and code.
4. NLP task datasets, daily chat datasets and synthetic datasets.

   Alpaca is a dataset of 52,000 instructions and demonstrations generated by OpenAI's `text-davinci-003` engine.

   Training sample: the `text-davinci-003` engine generate the instruction data, and a new prompt is written for requirement of instruction generation. Multiple instructions are generated at once, and only a single instance will be generated for each instruction.
5. Data mixtures including: CommonCrawl, C4, Github, Wikipedia, Books, ArXiv, and StackExchange.

   1.4T tokens overall.
6. Quality Filtering, De-duplication, Privacy Reduction, Tokenization, and Discussion on Effect of Data Quality.
7. Tokenization aims to segment raw text into sequences of individual tokens, which are subsequently used as the inputs of LLMs.

   Algorithms examples: Byte-Pair Encoding (BPE) tokenization, WordPiece tokenization, Unigram tokenization.

   LLaMA uses BPE tokenization algorithm.
8. Encoder-decoder: Flan-T5.

   Causal decoder: OPT, BLOOM, and Gopher.

   Prefix decoder: GLM130B and U-PaLM.
9. Most language tasks can be cast as the prediction problem based on the input, so decoder-only LLMs can better learn how to accomplish tasks in a unified LM way. Some studies have also revealed that decoder-only LLMs can be naturally transferred to certain tasks by autoregressively predicting the next tokens, without fine-tuning.
10. Position embeddings (PE) are employed to inject absolute or relative position information for modeling sequences.

    PE examples: Absolute PE, Relative PE, Rotary PE, and ALiBi.
11. The language modeling task (LM) is the most commonly used objective to pre-train decoder-only LLMs.

Causal Language Modeling is an autoregressive method where the model is trained to predict the next token in a sequence given the previous tokens.

Masked Language Modeling is a training method used in models like BERT, where some tokens in the input sequence are masked, and the model learns to predict the masked tokens based on the surrounding context.

12. Outputs are generated based on a specific decoding strategy.

    Greedy Search: selects the token with the highest probability at the current step.

    Random Sampling: sample the word with higher probability while also retaining the possibilities of the rest words.

13. Instruction tuning is the approach to fine-tuning pre-trained LLMs on a collection of formatted instances in the form of natural language.

    It has significant effects in three aspects: Performance Improvement, Task Generalization, and Domain Specialization.

14. Alignment tuning enables LLMs to follow the expected instructions, which utilizes the technique of reinforcement learning with human feedback.

    It guarantees the LLMs to produce high-quality, harmless responses.

15. Parameter-efficient fine-tuning aims to reduce the number of trainable parameters while retaining a good performance as possible. It helps to developing a more lightweight adaptation approach in downstream tasks.

    Prefix tuning prepends a sequence of prefixes to each Transformer layer, while prompt tuning mainly focuses on incorporating trainable prompt vectors at the input layer.

16. Prompt engineering is the practice of designing inputs for AI tools that will produce optimal outputs. It's aimed to generate better results for LLMs with high-qualified inputs.

    Zero-shot learning is a technique whereby we prompt an LLM without any examples, attempting to take advantage of the reasoning patterns it has gleaned. Few-shot learning is a technique whereby we prompt an LLM with several concrete examples of task performance.

17. In-context learning focuses on few-shot learning and prompt engineering, while chain-of-thought prompting emphasizes the chain of thought, prompting complex reasoning.

18. Language generation, knowledge utilization, and complex reasoning.

    Perplexity is defined as the exponentiated average negative log-likelihood of a sequence.

    Hallucination occurs when a LLM generates a response that is either factually incorrect, nonsensical, or disconnected from the input prompt.

19. Human alignment is proposed to make LLMs act in line with human expectations. It's important for LLMs to generate high-quality, harmless responses.

20. MMLU, BIG-bench, HELM, and a series of human exam benchmarks.

3 Optimization Methods

3.1 Gradient Accumulation

1. Suppose the linear model has the following expression:

$$\hat{y} = w \cdot x + b$$

For a batch of 8 images, divide the loss value by 4.

$$e_i = \frac{1}{4}(\hat{y}_i - y_i)$$

$$\nabla J(w,b) = \left(\frac{2}{n}\sum_{i=1}^{n}(e_i) \times x_i, \frac{2}{n}\sum_{i=1}^{n}(e_i) \right)$$

$$= \left(\frac{2}{n}\sum_{i=1}^{n}(\frac{\hat{y}_i - y_i}{4}) \times x_i, \frac{2}{n}\sum_{i=1}^{n}(\frac{\hat{y}_i - y_i}{4}) \right)$$

$$= \left(\frac{1}{16}\sum_{i=1}^{8}(\hat{y}_i - y_i) \times x_i, \frac{1}{16}\sum_{i=1}^{8}(\hat{y}_i - y_i) \right) \ (n=8)$$

Add 4 batches together:

$$\sum_{j=1}^{4}\nabla J_i(w,b) = \left(\frac{1}{16}\sum_{i=1}^{32}(\hat{y}_i - y_i) \times x_i, \frac{1}{16}\sum_{i=1}^{32}(\hat{y}_i - y_i) \right)$$

Which is identical to the non-accumulated gradient:

$$\nabla J(w,b) = \left(\frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \times x_i, \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \right)$$

$$= \left(\frac{1}{16}\sum_{i=1}^{32}(\hat{y}_i - y_i) \times x_i, \frac{1}{16}\sum_{i=1}^{32}(\hat{y}_i - y_i) \right) \ (n=32)$$

2. Each batch goes through the normalization process:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y_i = \gamma\hat{x}_i + \beta$$

   Where the mean μ_B and variance σ_B^2 are all derived from the current batch. Since these statistics are different in each batch, and also not identical to the whole non-accumulated data, the updates computed through gradient accumulation will most likely generate non-identical results to the non-accumulated result.

3.2 Gradient Checkpointing

1. Model inference doesn't require backward propagation, so inference does not require storing intermediate activations or gradients. Therefore memory consumption does not experience the "memory blow-up" problem.
2. For the first strategy, since all activations are discarded, the requirement will still be $O(n)$. For the second one, the requirement will be $O(\sqrt{n})$

3.3 Low-Rank Adaptation (LoRA)

1. The matrix rank is equal to the number of linearly independent rows (or columns) in that matrix.
2. $A = USV^T$ where: the columns of U and V are orthonormal and the matrix S is diagonal with positive real entries. U represents the left singular vectors of A and V^T represents the right singular vectors of A.
3. Because the transpose of orthogonal matrices is equal to their inverse:

$$U^T = U^{-1}\ ,\ V^T = V^{-1}$$
$$\because UU^{-1} = I\ ,\ VV^{-1} = I$$
$$\therefore UU^T = I\ ,\ VV^T = I$$

4. Its rank is n.
5. The reconstructed image will resemble the original image but with some loss of detail. Increasing the value of k will result in a reconstructed image that retains more detail and closely resembles the original image.
6. Its singular matrix will have only a few significant singular values, while the rest will be close to zero or negligible.
7. We only need to take in the top-k singular values. So $U \in R^{n\times k}$, $S \in R^{k\times k}$, $V \in R^{k\times n}$.

$$A = U \times \sqrt{S}, B = \sqrt{S} \times V^T$$

   where $\sqrt{S}$ means taking the square root of each of its diagonal elements.
8. Truncated SVD may fail to make a good approximation when the singular values decay slowly, meaning that even the smaller singular values contribute significantly to the matrix's structure. In such cases, truncating too many singular values can lead to a loss of important information, resulting in a poor approximation.

When they need to switch to another downstream task, they can recover W_0 by subtracting BA and then adding a different $B'A'$, a quick operation with very little memory overhead. This guarantees that they do not introduce any additional latency during inference and forward pass costs are saved.

3.4 Mixed Precision Training

FP32 master copy of weights is maintained in order to match the accuracy of the FP32 networks. On one hand, updates become too small to be represented in FP16. On the other hand, the ratio of the weight value to the weight update is very large. These two issues can be counteracted by updating in FP32.

Scaling up the gradients will shift the values to occupy more of the representable range and preserve values that are otherwise lost to zeros, so that they can match the accuracy achieved with FP32 training.

Yes, it still holds. This is primarily due to activations consuming the most memory during training, and storing them in FP16 format significantly reduces memory usage. Despite the 50% increase in memory requirements for maintaining an FP32 copy of weights alongside FP16, the reduction in memory from using FP16 for activations generally results in a net decrease in overall memory consumption. Moreover, FP16 usage enables the utilization of specialized GPU hardware like Tensor Cores for faster computations.