Archives of Computational Methods in Engineering
https://doi.org/10.1007/s11831-026-10598-4

REVIEW

Building Expert Small Models: A Comprehensive Survey of Model Compression, Knowledge Distillation, and Augmented Inference

A. Aruna Gladys[1] · R. Aishwarya[2] · V. Vetriselvi[3] · Samuel Daphne[4] · P. N. Baskar[3] · Arun Amaithi Rajan[3]

Received: 4 February 2026 / Revised: 5 April 2026 / Accepted: 14 April 2026
© The Author(s) under exclusive licence to International Center for Numerical Methods in Engineering (CIMNE) 2026

Abstract

Large language models have strong general-purpose capabilities, but their high computational and energy costs limit their practicality in many real-world and resource-constrained settings. In contrast, Small Language Models (SLMs) offer a more efficient alternative for domain-specific applications, where targeted expertise and performance are more important. This survey provides a structured overview of methods that enable compact language models to achieve competitive performance without requiring large parameter counts. We first outline the foundational concepts behind modern Transformer-based models, including common pretraining and fine-tuning paradigms. We then review efficiency-oriented techniques such as parameter-level optimization, prompt-based adaptation, and knowledge distillation, highlighting how knowledge and task behavior can be transferred from large models to smaller ones. The survey also discusses recent approaches that augment language models with external memory, retrieval mechanisms, and tool access to enhance reasoning and decision-making without increasing model size. Finally, we briefly outline future research directions, including the role of efficient and augmented models in emerging agent-based and edge-deployed systems. This survey aims to provide researchers and practitioners with a clear methodological roadmap for designing compact and efficient models that are suitable for practical deployment.

1 Introduction

Artificial Intelligence (AI) has experienced immense progression over the past few decades, evolving from pure theoretical research to real-time problem-solving technology. In its early stages, machine learning focused on simpler algorithms that attempted to draw linear decision boundaries for classification tasks. These models could not capture the complex relationship between data and often required manual feature engineering and domain expertise [1]. The arrival of deep learning drastically shifted this paradigm. Deep learning models, particularly convolutional and recurrent architectures, can learn meaningful features directly from raw inputs, such as text, images, speech, and time-series signals. These models mapped the input data into high-dimensional latent spaces, capturing complex patterns and correlations without the need for human intervention for feature engineering [1, 2].

While the deep learning algorithms were widely successful across many different types of problems, such as classification and regression, their decision-making processes were opaque. The models applied multiple non-linear transformations in high-dimensional spaces, making it difficult to interpret why a particular output was produced. These models were often viewed as black boxes by the research community despite their improved performance [3]. This limitation led to the field of study called Explainable AI (XAI). This field aims to make machine decisions transparent and interpretable [4].

The introduction of the Transformer architecture by Vaswani et al. in 2017 [5] represents a major turning point in the field of AI. Traditional neural networks operate by

✉ A. Aruna Gladys
gladys.vimalan@gmail.com

1 Department of CSE (AI&ML), Easwari Engineering College, Chennai 600089, India

2 Department of Computing Technologies, SRM Institute of Science and Technology, Kattankulathur, Chennai 603203, India

3 Department of Computer Science and Engineering, College of Engineering Guindy, Anna University, Chennai 600025, India

4 Department of CSE (AI&ML), SRM Institute of Science and Technology, Ramapuram, Chennai 600089, India

Published online: 27 April 2026

sequentially processing data, whereas Transformers utilizes self-attention mechanism to capture dependencies in input sequences in parallel. This way of processing data enables the transformer to be more context aware as compared to any other type of neural network and therefore demonstrates superior model performance when compared to prior models. The Google Transformer model improved performance and redefined the structure of AI systems. Transformers have become the foundation for a wide array of model architectures, including encoder-only models (such as BERT [6]), decoder-only models (like GPT [7]), and encoder-decoder hybrids (like T5 [8] and BART). These architectures have become the backbone of modern AI systems, excelling in tasks ranging from text summarization to image captioning, and even scientific problem-solving. The timeline in Fig. 1 provides a visual summary of the progression and evolution of AI systems.

The models built on the Transformer backbone are usually referred to as large models. What makes these models "large" is their parameter count, often in the billions or even trillions. These models also possess abstract, contextual knowledge, and the key process that is responsible for the possession of massive knowledge is the self-attention mechanism. It allows the model to attend to different parts of its input and apply the learnt knowledge in contextually appropriate ways.

The capabilities of these large models extend beyond simple tasks like image recognition or sentiment analysis. With autoregressive models like GPT, we are no longer trying to distinguish between cats and dogs; instead, we are using AI to solve math problems, generate code, compose music, and even assist with breakthroughs in fields such as quantum mechanics, chemistry, and molecular biology [9]. In many cognitive tasks, machine intelligence has performance in par with humans. These developments have initiated a vision about the transition from narrow AI to Artificial General Intelligence (AGI), the systems that generalize knowledge across tasks and domains, and reason similarly to humans.

AGI is an open challenge to date. However, it is widely accepted that the large language models (LLMs) are early stepping stones towards AGI [10]. AGI aims to achieve a more generalized and universal problem-solving ability. The large models are still in the nascent stage, but the capabilities of current models, when integrated with tools, memory, and agentic workflows, show that the AI field is progressing toward general-purpose reasoning machines.

While the models developed by companies like OpenAI (ChatGPT), Google (Gemini), and xAI (Grok) represent the forefront of technology, they are all proprietary solutions requiring significant amounts of computing power that are also inflexible for customization. As a result, there exists a significant disparity between the capabilities of the technology and the ability to effectively use such technology by smaller organizations, research institutions, or startups with limited computing capabilities.

In contrast, the open-source LLM movement led by models like Meta's LLaMA series, Mistral, and Falcon offers an alternative to proprietary models. These model weights are freely available, and they can be modified and adapted for specific tasks, while also achieving similar performance in comparison with the proprietary models but with less compute and carbon footprint [11]. One of the major drawbacks of proprietary models like GPT-4 is that they are closed-source: their training data is not disclosed, and their model weights are inaccessible. Their architectures are controlled

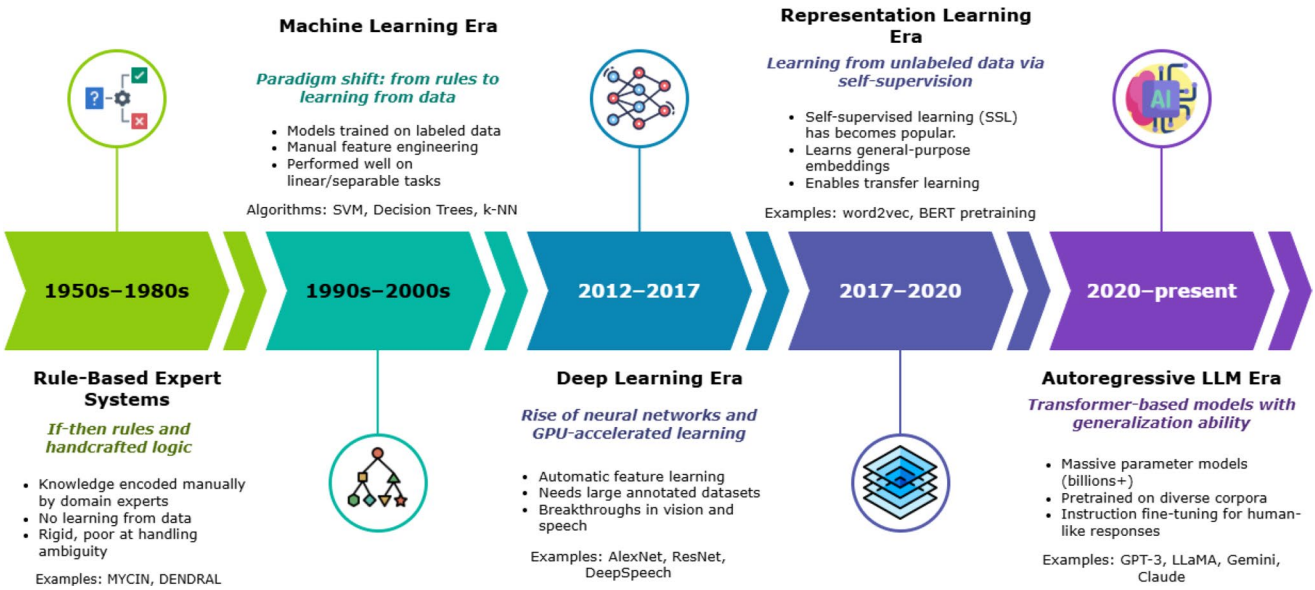


Fig. 1 Timeline illustrating the evolution from rule-based expert systems to modern generative models

by the companies that design and build them. Hence, these models are closed to customization and raise concerns about bias and auditability. Proprietary models are also costly to operate or fine-tune, often requiring APIs with usage-based pricing, and don't allow deep customization for niche tasks. Their behavior is governed by general-purpose instruction tuning, making them less suitable for highly domain-specific or regulated applications (e.g., legal compliance or scientific reasoning), where control, interpretability, and security are crucial.

In contrast, open-source models can be fine-tuned using parameter-efficient fine-tuning methods, such as LoRA [12] and QLoRA, which update only a small percentage (0.1 to 10%) of the total model parameters, making them computationally efficient even for users with limited resources. Full fine-tuning is also possible, as most of the model's weights and architecture are publicly available. This flexibility allows organizations to "bend and mold" the models to particular use cases: for example, building a legal expert assistant trained on national laws, or a coding assistant specialized in legacy systems. Such domain-specific customization is infeasible with most proprietary models. As a result, open-source LLMs are increasingly viewed as the ideal choice for targeted, high-accuracy tasks, particularly in enterprise, academic, or national contexts where cost, control, and transparency are crucial.

In recent years, Small Language Models (SLMs), lightweight counterparts of massive large models, have emerged as a practical and sustainable alternative [13]. Fine-tuning large models for narrow tasks in resource-constrained environments is not only challenging but increasingly unnecessary. SLMs offer clear advantages such as lower computational and energy requirements and cost-effective deployment. They are particularly well-suited for domain-specific industrial applications, where the goal is depth rather than breadth. For instance, a legal assistant specialized in national laws or a medical chatbot focused solely on disease-related knowledge. Although their scope is narrower than that of large models, with carefully designed training and optimization objectives, these smaller models can achieve expert-level performance. This survey consolidates and reviews the different categorizations of techniques that enable small models to become smarter and environmentally sustainable alternatives to the large parameter models.

This survey is written for an audience that may be new to Generative AI but eager to understand how to adapt large models for targeted use cases in academia, start-ups, or industrial R&D. It also helps researchers improve model intelligence while working within limited budgets and infrastructure, without compromising performance or interpretability.

This survey addresses the following central question: **How can these models be more innovative and efficient for solving narrow, real-world problems under constrained resources?**

In this article, we begin by clarifying the terminologies that define the landscape of foundation models, including distinctions between open source and proprietary models, pretraining and fine-tuning paradigms, and different architectural variants like encoder-only and decoder-only Transformers. We then present the core technical components of the survey, such as model distillation, knowledge distillation, and efficient fine-tuning techniques that utilize minimal computational resources.

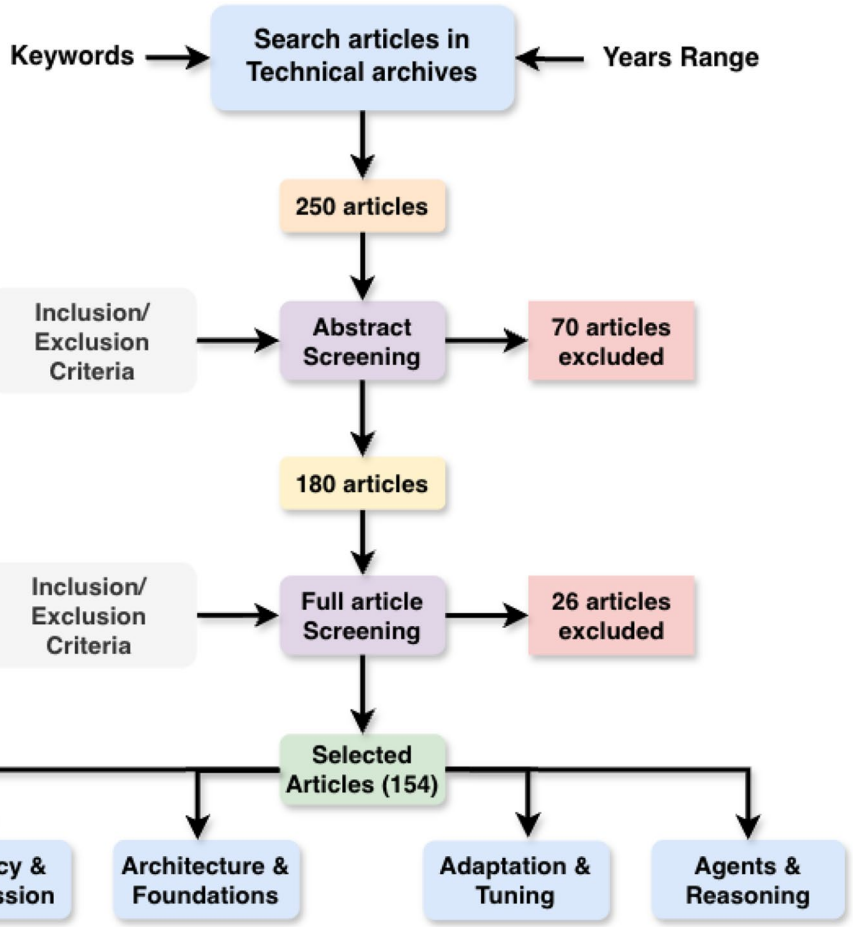
We also examine how models can be equipped with additional memory and decision-making abilities, thereby enabling agentic behaviour in these foundational models. These augmented models can support multi-step reasoning, use relevant tools, and advance towards goal-oriented AI. Finally, we discuss the evaluation metrics used for modern generative models and discuss how different metric suites are suited to specific domains, including healthcare assistants and real-time applications such as financial trading. This survey aims to provide readers with a structured roadmap for transforming general-purpose models into task-specific expert agents with stronger reasoning capabilities.

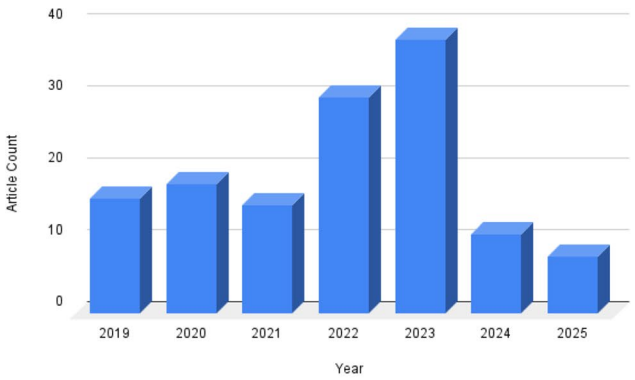
2 Search Methodology

Following Snyder's [14] guidelines in the article's study component, we've created a comprehensive literature search for our criteria on LLMs. This section focuses on how we collected articles for this review. We used standard databases (ACM, Springer, IEEE, arXiv, Elsevier) and standard keywords ("large language models", "agentic AI", "knowledge distillation", and "LLM evaluations") to search for articles from 2015 to 2025. We retrieved approximately 250 articles.

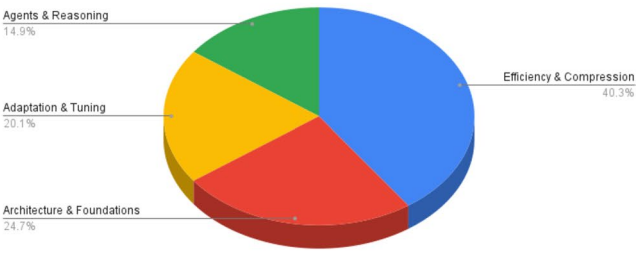
The two-part selection procedure consists of reviewing the abstract of each manuscript and the full manuscript. The focus for each manuscript was on breadth and depth of research and originality; however, following completion of the abstract review, by using our inclusion criteria, most manuscripts were rejected based on preliminary criteria. After nearly completing the manuscript review, there were very few manuscripts rejected. Approximately 70 manuscripts were withdrawn from consideration on the basis of their relevance to the selected topic. In the second review stage, the manuscripts were grouped to allow rapid analysis and review of all manuscripts and approximately 26 manuscripts were classified as out of scope because of a detailed examination of the subject; subsequently, we were left with

Fig. 2 Articles screening process





(a)



(b)

Fig. 3 Analysis on collected research articles

154 manuscripts to begin our survey. The review process is displayed in Fig. 2.

The publication counts broken down by year (2019 through 2025) can be seen in Fig. 3(a) and the number of article selections broken down into the main categories we utilized for our document selections from the existing standard databases is shown in Fig. 3(b). As part of our analysis of this research area, there were no survey articles found that have comprehensively covered the combined perspectives of LLM's, knowledge distillation and agentic AI. This observation has led us to perform this survey.

3 Foundation Concepts of Large Models

One of the key limitations in deep learning is its reliance on large amounts of labeled data [15]. Digital data is plentiful and contains all sorts of content, including e-books, audio files, and social media posts, but annotating that data to complete a specific task, such as classification, generally requires input from subject matter experts. Additionally, the majority of the annotations available for AI data are simply class labels without any additional reasoning or explanations, thereby limiting the depth and utility of most AI datasets [16].

To address these limitations, modern models are built to learn directly from an unlabelled data source. In this section, we will discuss the foundational principles behind today's large models, with a focus on an important learning paradigm, that is Self-supervised learning. This technique enables models to learn from an enormous amount of unlabeled data. Then we shift to a discussion of the transformer model and the core components of its architecture, which enable it to learn from unlabelled data.

3.1 Learning without Labels: The Self-Supervised Revolution

To address the challenge mentioned above, self-supervised learning (SSL) emerged as a powerful alternative. Rather than depending on expensive human-generated labels, SSL learns useful representations directly from raw data by generating pseudo labels directly from raw data [17]. For example, in early Natural Language Processing (NLP) models, such as Word2Vec [18], a simple neural network was trained to predict a target word based on its surrounding context (using skip-gram and CBOW models). Initially, each word was represented using a one-hot encoding; however, after training, the model learned meaningful embeddings, where semantically similar words were closer in the high-dimensional space.

Following Word2Vec, models such as FastText and GloVe (word embeddings) gained popularity. However, a notable limitation of these approaches was their lack of contextual awareness. For example, GloVe [19] and Word2Vec models produce the same embeddings for polysemous words like "plant"(industrial facility vs. botanical plant), because it fails to consider the context in which the word appears.

Self-supervised learning has shown significant potential for learning useful features from images without ground truth annotation. One example of a self-supervised approach is to use image transformations, which produce supervisory signals that can be used to train a model to predict the transformation applied to an image (e.g., rotating an image by 90°, 180°, or 270°) [20]. Another strategy splits an image into patches, shuffles them, and requires the model to reconstruct the correct spatial arrangement, similar to solving a jigsaw puzzle [21]. Such models may not initially be able to classify objects, but they develop a sense of spatial awareness.

The development of self-supervised learning technique(SSL) coincided with the introduction of the Transformer model [5], which was originally developed for the "supervised neural machine translation" task. Although the Transformer was developed as a supervised model, the nature of its architecture has allowed it to be readily adapted into self-supervised approaches. In this paper, we will examine the key structural and functional features of the Transformer that made this adaptation successful.

3.2 Transformer Architecture: Structure and Efficient Parameter Usage

The development of artificial neural networks was inspired by the way the human brain's structures operate [22]. The gyral and sulcal folds of the cortex provide for more neural connections and thus enable a greater number of cognitive functions. Similarly, the depth of the hidden layers of an artificial neural network creates additional neural connections within an artificial neural network. Although adding more hidden layers and additional neurons to an artificial neural network can improve its learning abilities, when a model is too deep, it can overfit to the training data, which refers to when the network learns the details of the training data without being able to generalize it.

The training instability issue and learning from large-scale data have been addressed in transformers using Layer Normalization and Dropout. Layer Normalization normalizes the outputs of each of the sub-layers, so that they have consistent activation scales and stable gradient flow, which allows all components to contribute properly during training [23]. Layer Normalization offers a fundamentally different approach than conventional Batch Normalization because Batch Normalization normalizes activations across the batch dimension and therefore has problems with small batches and variable input sequence lengths. Layer Normalization solves these issues because each sample is normalized independently of other samples. Dropout is used as a regularization technique to randomly deactivate neurons to avoid overfitting. As a result, the architectural design of Transformers, particularly with these normalization and regularization strategies, renders them structurally well-suited for robust learning from large-scale data.

Each Transformer layer comprises two main sub-layers: an attention layer (discussed in Sect.3.5) and a feed-forward layer. Crucially, after both sub-layers, a Layer Normalization

(LN) operation is applied, immediately followed by a residual connection.

For a token's output vector $Y_i \in \mathbb{R}^D$, Layer Normalization is applied as:

$$\text{LayerNorm}(Y_i) = \frac{Y_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \cdot \gamma + \beta \tag{1}$$

Here, μ_i and σ_i^2 are the mean and variance computed across the D dimensions of the vector Y_i. γ and β are learnable scale and shift parameters. The small constant ϵ ensures numerical stability. This normalization is applied independently to each token.

3.3 Pretraining: Gaining Understanding from Large-Scale Data

Secondly, in terms of function, the Transformer model also mirrors human learning processes. Humans accumulate knowledge from their environment and apply it when needed. Similarly, Transformer-based models undergo pretraining on large corpora to acquire general knowledge, which is then fine-tuned for specific tasks. Pretraining strategies include:

3.3.1 Masked Language Modeling (MLM)

MLM [6] is a pretraining objective used in models like BERT, where 15% of the tokens in an input sentence are randomly selected and masked, and the model is trained to predict the original tokens at those masked positions. This is similar to fill-in-the-blank tasks, where the model must understand the context to infer the missing word.

In Masked Language Modeling (MLM), given a sequence of tokens $X = [x_1, x_2, \ldots, x_n]$, a subset of tokens is randomly masked, and the model is trained to predict the original tokens at those masked positions.

Let $\mathcal{M} \subset \{1, 2, \ldots, n\}$ be the set of masked indices. The MLM objective minimizes the cross-entropy loss over the masked token positions:

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in \mathcal{M}} \log P(x_i \mid X^{(\text{masked})}) \tag{2}$$

Here, $\mathcal{M}$ is the set of masked token indices, x_i is the original token at position i, and $X^{(\text{masked})}$ is the input sequence with 15% of its tokens masked or replaced according to the MLM strategy. To avoid overfitting to the [MASK] token, 80% of the selected masked tokens are replaced with [MASK], 10% with random tokens, and 10% are left unchanged, encouraging the model to rely on contextual understanding.

3.3.2 Causal Language Modeling (CLM)

CLM is primarily used in decoder-only transformer models such as GPT [24]. In this setting, the model learns to predict the next word in a sequence using only the previous context, without access to future tokens. This is enforced by applying a causal (triangular) attention mask during training. The learning objective is to maximize the probability of the next token given all previous ones. This autoregressive approach aligns with natural language generation tasks, where tokens are generated sequentially, one at a time.

In CLM, the probability of a sequence $w_1, w_2, \ldots, w_n$ is factorized as:

$$P(w_1, w_2, \ldots, w_n) = \prod_{t=1}^{n} P(w_t \mid w_1, w_2, \ldots, w_{t-1}) \tag{3}$$

where the model predicts each token w_t conditioned only on the preceding tokens, imposing causality during training.

3.3.3 Next Sentence Prediction (NSP)

NSP is a pre-training objective used in models like BERT, where the model is given two sentences (A, B) and trained to predict whether B logically follows A. It is formulated as a binary classification task using the loss:

$$\mathcal{L}_{\text{NSP}} = -\left[y \log P_{\text{NSP}} + (1 - y) \log(1 - P_{\text{NSP}})\right] \tag{4}$$

Here, $y \in \{0, 1\}$ is the ground truth label (1 if B follows A, otherwise 0), and P_{NSP} is the predicted probability that B is the next sentence after A. However, NSP has fallen out of favor in recent models, such as RoBERTa, because it was found to offer limited benefits and often encouraged the model to rely on surface-level clues.

Beyond the commonly discussed objectives, such as MLM, CLM, and NSP, several other pretraining strategies have proven effective in recent large language models. Permutation Language Modeling [25] (as in XLNet) enables bidirectional context learning by predicting tokens in randomly permuted orders. Denoising autoencoding objectives, used in models like BART [26] and T5 [8], train the model to reconstruct the original text from corrupted input, improving robustness. Span corruption, a variant used in T5, masks contiguous spans instead of individual tokens for richer contextual learning. Additionally, contrastive learning approaches in cross-modal models like CLIP [27]

enhance representation quality by distinguishing between semantically similar and dissimilar inputs. These objectives help models learn deeper structure and semantic relationships beyond word-level predictions.

Pretraining Transformer-based models begins with randomly initialized token embeddings. However, due to the self-attention mechanism, the embeddings evolve to become context-sensitive during training, allowing the model to capture nuanced distinctions, such as differentiating between a green plant and an industrial plant.

In computer vision, instead of words, image patches are used as inputs. Vision Transformers (ViTs) [28] take an image, divide it into "patches" of a fixed size, and then "flatten" the pixel data of each patch into a 1D array. These patches are then embedded in the same way word tokens are used in NLP. ViTs use a transformer encoder to process these patches. The self-attention mechanism provides ViTs with the ability to learn both local and global dependencies from the first layer onwards. In contrast to CNNs that only learn local, spatial patterns of a given image by applying convolutional filters, ViTs can also learn the global structure of an image, thus reducing misidentified features.

3.4 The Evolution of Attention Mechanisms

As we compare human and machine intelligence, humans build knowledge through observation, reading, and life experience. Pretraining in machine learning plays a similar role, gradually building a model's internal knowledge. The next step for both humans and machines is learning how to use that knowledge to solve specific problems. For instance, when asked about the Chola dynasty, a person naturally draws on their understanding of Indian history, rather than on unrelated knowledge such as chemical formulas. In much the same way, attention mechanisms in deep learning allow models to focus on the most relevant parts of the input. The idea of attention was first formalized in neural machine translation by Bahdanau et al. [29], where models learned to dynamically select which source words to emphasize while generating each target word. This attention mechanism overcomes the constraints of fixed-length context vectors in earlier encoder–decoder architectures, thus enabling more flexible models and has since become a foundational component of deep learning models.

For instance, translating the English sentence "I am going home" to Tamil as "naan veetukku poren" requires emphasizing the word "home" while generating "veetukku". When the model learns to focus on specific words, it leads to better alignment between source and target languages. Attention mechanism has found applications in computer vision [30], where models learn to focus on key pixels, objects, or channels, and this improves performance on tasks such as object detection and image captioning. However, this type of attention is different from self-attention, on which the modern models are built. The earlier attention mechanisms, such as the additive attention introduced by Bahdanau et al. [29], used an alignment-based approach to map the information being generated by an encoder with information being received by a decoder. In contrast, the self-attention mechanism proposed by Vaswani et al. [5] eliminates recurrence, enabling each token in the input sequence to attend to every other token simultaneously. This shift is one of the key components of the Transformer architecture, and it has made the model indispensable across modalities.

3.5 Self-Attention and Contextual Encoding

Self-attention allows each token in a sequence to pay attention to every other token, and thus captures global dependencies across the input. The mechanism is implemented by using three core components: Query (Q), Key (K), and Value (V) vectors.

Suppose there is a team reviewing multiple project proposals to identify which project matches the requirement. The Query represents the requirement itself, the Keys correspond to the available proposals, and the Values represent the execution plans behind those proposals. The more closely a project's proposal matches the requirement, the more significant an effect it will have on the overall decision. Instead of selecting one proposal, self-attention evaluates all available proposals, assigns a relevance rating, and combines them based on how well they correspond with the Query.

Mathematically, for any given token, similarity scores are computed between the query vector and the key vectors of all tokens in the sequence. The scores are then used to weight the corresponding value vectors. This weighted sum is the contextualized representation of each token, where the information from the entire sequence is integrated. Unlike RNNs, this computation is parallelizable and thus highly efficient.

The input sequence $X \in \mathbb{R}^{n \times d}$ is projected into Query, Key, and Value matrices:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V \tag{5}$$

where $W^Q, W^K, W^V \in \mathbb{R}^{d \times d_k}$ are learned projection matrices.

The attention output is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{6}$$

This output is then layer normalized, along with a residual connection to the input X, before being passed to the next layer:

$$\text{Output} = \text{LayerNorm}(X + \text{Attention}(Q, K, V)) \tag{7}$$

The attention mechanism differs between the encoder and decoder in the Transformer architecture. In the encoder, self-attention operates over the entire input sequence, allowing each token to attend to all others. However, in the decoder, self-attention is masked, often using a triangular (causal) mask, to prevent access to future tokens, as the model generates output in an autoregressive manner. This ensures that each position can only attend to previous or current positions in the sequence. Additionally, because the original Transformer was designed for neural machine translation, the decoder also includes a cross-attention mechanism. This allows the decoder to attend to the encoder's output representations, functioning similarly to traditional attention mechanisms used in earlier sequence-to-sequence models.

To preserve word order, which self-attention does not inherently capture, the Transformer introduces positional encodings that are added to the input embeddings. This ensures the model can still differentiate between tokens based on their position in the sequence.

3.6 Multi-Head Self-Attention in Transformers

To capture different kinds of relationships at the same time, Transformers use a mechanism called multi-head attention. Each attention head learns a distinct set of Query-Key-Value transformations, enabling the model to focus on different aspects of the input in parallel.

Consider the following passage: Eric goes to the supermarket on a rainy day carrying an umbrella. He buys chips and later plays in a muddy puddle.

To answer "Where did Eric go?", attention should be placed on Eric and the supermarket. To answer "What did he buy?", attention should focus on Eric and the chips. Multi-head attention enables the model to simultaneously attend to diverse associations, capturing rich, layered semantics across the input. This concept is illustrated in Fig. 4.

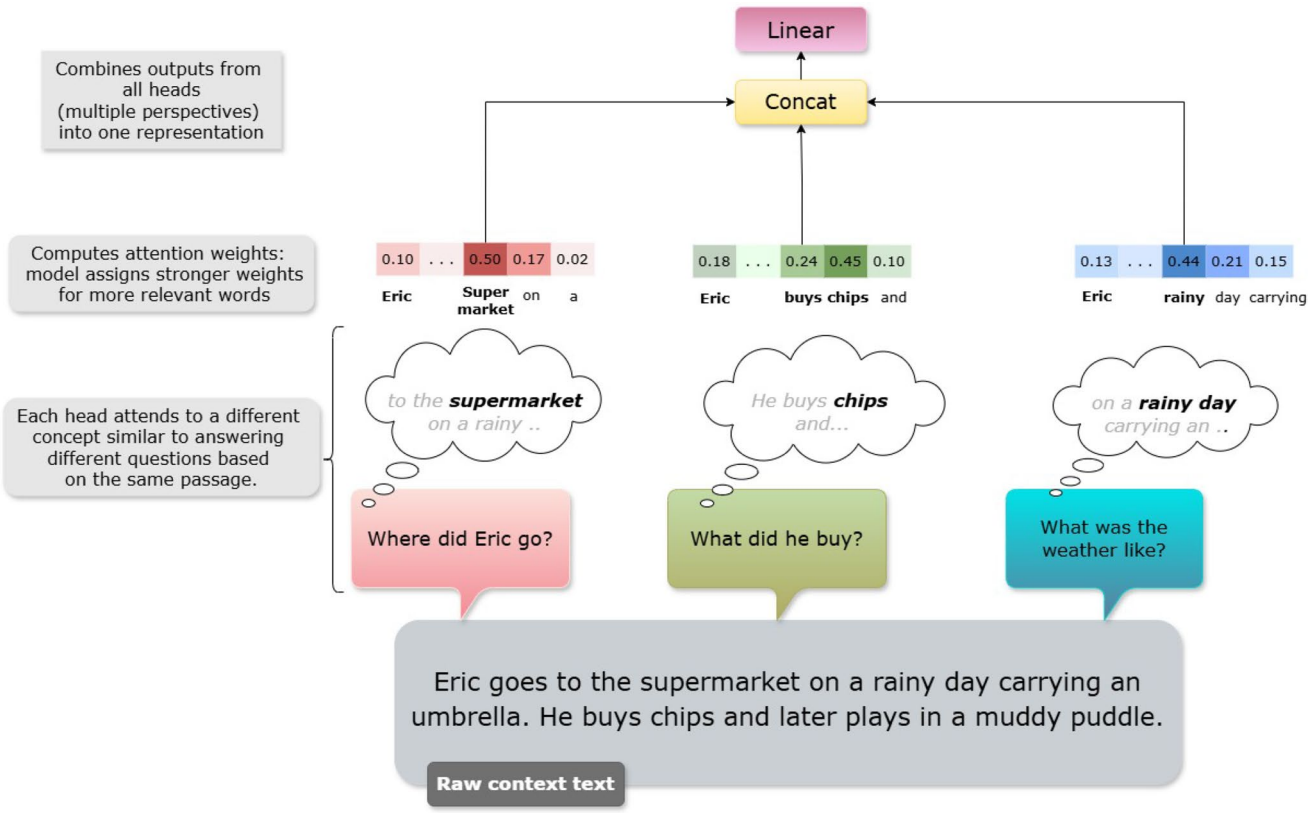


Fig. 4 An illustration of multi-head attention capturing different semantic relationships within a passage. Each attention head focuses on distinct aspects such as entities, actions, or locations enabling the model to answer varied questions like "where did Eric go?" or "What did he buy?" by attending to relevant tokens in parallel

3.7 Transformer Variants: Encoder, Decoder, and Encoder-Decoder Architectures

The original Transformer architecture comprises both an encoder and a decoder. Over time, different model variants specialized in one or the other:

Encoder-only models like BERT are well-suited for tasks that require a deep understanding of the input (e.g., classification, question answering). BERT introduced masked language modeling and next sentence prediction as training objectives, resulting in embeddings that capture both token-level and sentence-level semantics.

Decoder-only models like GPT are autoregressive, generating text by predicting the next token conditioned on previous ones. These models are powerful for generation tasks such as story completion and code generation.

Encoder-decoder models such as T5, FLAN-T5 are designed for sequence-to-sequence tasks such as translation or summarization. The encoder processes the input, and the decoder generates the output sequence conditioned on the encoder's representations.

3.8 Cross-Modal Transformers and Contrastive Pretraining

Large cross-modal models, such as CLIP (Contrastive Language–Image Pre-training) and BLIP (Bootstrapped Language–Image Pre-training), extend Transformer architectures to model multiple modalities jointly.

CLIP [27] aligns image and text embeddings via contrastive learning. Given matched (image, caption) pairs, the model is trained to maximize their similarity in a shared embedding space, while pushing mismatched pairs apart. This enables zero-shot classification by measuring the similarity between an image's embedding and the embeddings of candidate labels described in natural language.

BLIP [31] introduces a unified framework combining vision-language pretraining with generative and contrastive objectives. It incorporates both vision-language modeling and image-text matching, enabling superior performance in tasks such as visual question answering and captioning.

These models are built on the principle of pairwise representation learning, where a joint representation space is learnt so that semantically related information from different modalities ends up close together. This strategy is closely connected to contrastive learning, which encourages models to distinguish similar and different samples based on their features, thereby learning robust representations of the input data.

Cross-modal systems such as CLIP and BLIP can then be fine-tuned for domain-specific applications, such as visual question answering, and medical image–text retrieval. Fine-tuning enables these models to adapt their general representations to specialized datasets by updating either the entire model or a subset of parameters using techniques such as linear probing or parameter-efficient fine-tuning. Since both CLIP (via OpenCLIP) and BLIP are open-source, they offer flexibility for modification in multimodal applications, particularly in high-precision multimodal applications, including those involving medical data.

3.9 Instruction Fine-Tuning and Training Objectives

All the foundational components discussed, such as self-attention, pretraining strategies, and layer normalization, contribute to the model's capability to generalize during the instruction fine-tuning stage. Different families of large models are trained with distinct objectives: encoder-decoder models are supervised with structured input-output pairs (e.g., summarization or translation), while cross-modal models, such as CLIP and BLIP, adopt contrastive learning strategies to align visual and textual modalities. Decoder-only models, typically autoregressive in nature (e.g., GPT), are often instruction-tuned using large corpora of request–response pairs. In this setup, the model learns to generate appropriate responses one token at a time, conditioned on the input request. Pretraining equips the model with general linguistic and world knowledge. At the same time, instruction tuning leverages this capability by utilizing human-annotated samples, thereby making the output more aligned and coherent in downstream applications.

3.10 Model Size Considerations for Fine-Tuning

Table 1 compares different model types on various parameters. It is to be noted that the larger models offer strong reasoning capabilities, while the smaller models are found to be effective for targeted applications with limited data.

Fig. 5 offers a conceptual analogy rather than a direct comparison. Just as humans develop through neural structures, knowledge intake, and domain specialization, large models progress through transformer architectures, large-scale pretraining, and fine-tuning with methods such as instruction tuning, contrastive learning, and RLHF. The analogy highlights the shared trajectory from broad learning to task-specific expertise, while remaining illustrative rather than biological.

While Sect.3 described the foundations of building and aligning large language models, including pre-training on massive corpora and the family of instruction and preference-based fine-tuning methods, these approaches assume access to substantial computational resources and vast training data. In many practical scenarios, however, deploying or adapting a full-scale LLM is neither affordable nor

Table 1 Comparison of model types, sizes, and use cases

Model Type	Parameter Size	Ideal Dataset Size	Specific Use Cases	Why It's Suitable
Small LLM	1B–3B	∼1–10 M examples	– Embedded assistants on edge devices – Educational bots for schools	Lightweight, fast inference, low compute cost, easy on-device deployment
Medium LLM	7B–13B	∼10–100 M examples	– HR/Legal/Finance chatbots – Customer service bots	Balanced capability and compute; captures domain-specific language patterns
Large LLM	30B+	100 M+ examples	– Legal document drafting – Scientific research assistants – Enterprise copilots	High reasoning, multi-level logic, and nuanced understanding of long contexts
Cross-Modal Model	150 M–1.5B	∼5–100 M image-text pairs	– Medical image reporting – Vision-language assistants	Trained with contrastive objectives; aligns vision and text for paired data tasks

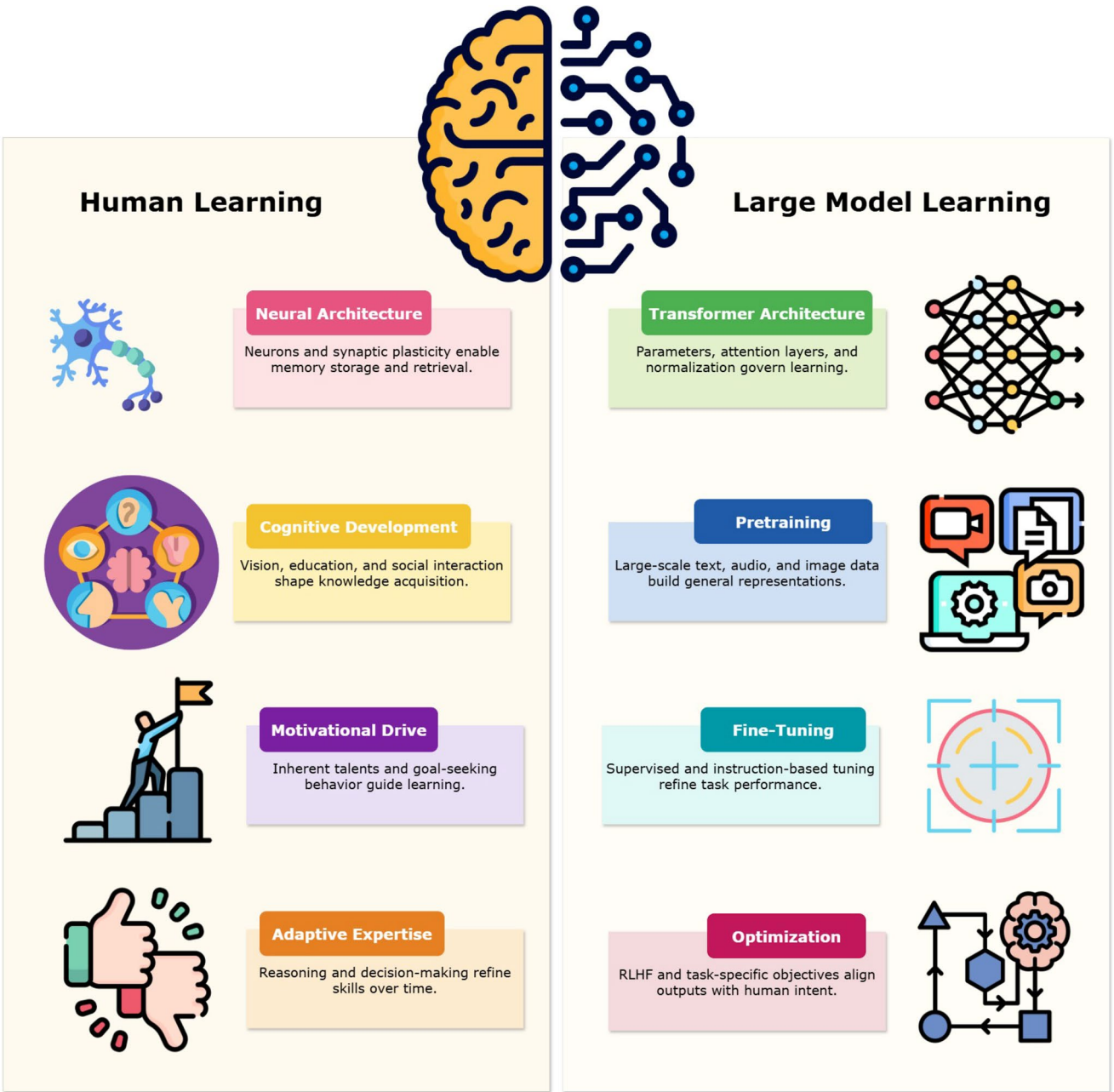


Fig. 5 Analogy between human learning and large model training (conceptual illustration)

necessary. Instead, practitioners increasingly rely on techniques that make smaller models more capable, without reconstructing the entire training pipeline. Fine-tuning the open-source models is also discouraged due to various challenges listed in Fig. 6. Section 4 shifts focus to such strategies, examining methods that operate directly on the model itself, either by modifying its internal parameters or by steering its behaviour through prompting in the embedding space before moving to approaches that leverage external teachers and distillation frameworks. The flowchart that gives an overview of the Sects. 4 and 5 is shown in Fig. 7.

4 Model-Intrinsic Approaches for Improving LLM Performance Without External Resources

Large Language Models (LLMs) have achieved impressive results across numerous natural language processing (NLP) tasks, but substantial computational and memory requirements are still a major issue. For example, GPT-3 [32], with 175 billion parameters, requires around 350 GB of memory in FP16 precision and at least five A100 GPUs (80 GB each) for efficient inference, making deployment in practical and resource-constrained settings highly challenging.

4.1 Parameter-Level Techniques

Model compression has become a widely adopted strategy to overcome these challenges. The main objective is to reduce model size and computational cost without compromising much on performance. This will lead to better inference efficiency on devices with limited resources. It also addresses environmental concerns by lowering energy consumption and carbon emissions [33–35]. A range of compression methods has been explored, including pruning [36–38], quantization [39–41], low-rank approximation [42, 43], parameter sharing [44, 45], dynamic networks [46–48], efficient neural architecture design [49, 50] and prompt tuning [51, 52]. These approaches try to balance model size and accuracy, making LLMs more practical for real-world use cases.

4.1.1 Pruning

Pruning is a very effective technique for model compression and is utilized for improving the efficiency of a neural network by getting rid of the unnecessary parameters. Pruning involves removing unnecessary weights, neurons, filters, or even whole subsections, resulting in a sparser model. Pruning is a great technique for achieving compression and is useful in scenarios where commodity hardware is being utilized [53].

Pruning techniques can be ranked based on the granularities of the weight parameters being removed (Fig. 8). Unstructured pruning involves the removal of individual weight parameters or connections. This form of pruning leads to very sparse matrices. Even though the compression ratios obtained through pruning are high [33], the random sparsity prevents a substantial speedup on traditional computing systems. When considering structured pruning techniques, larger entities like channels, filters, or even whole networks can be pruned, leading to those targeting hardware accelerators and showing viable inference benefit.

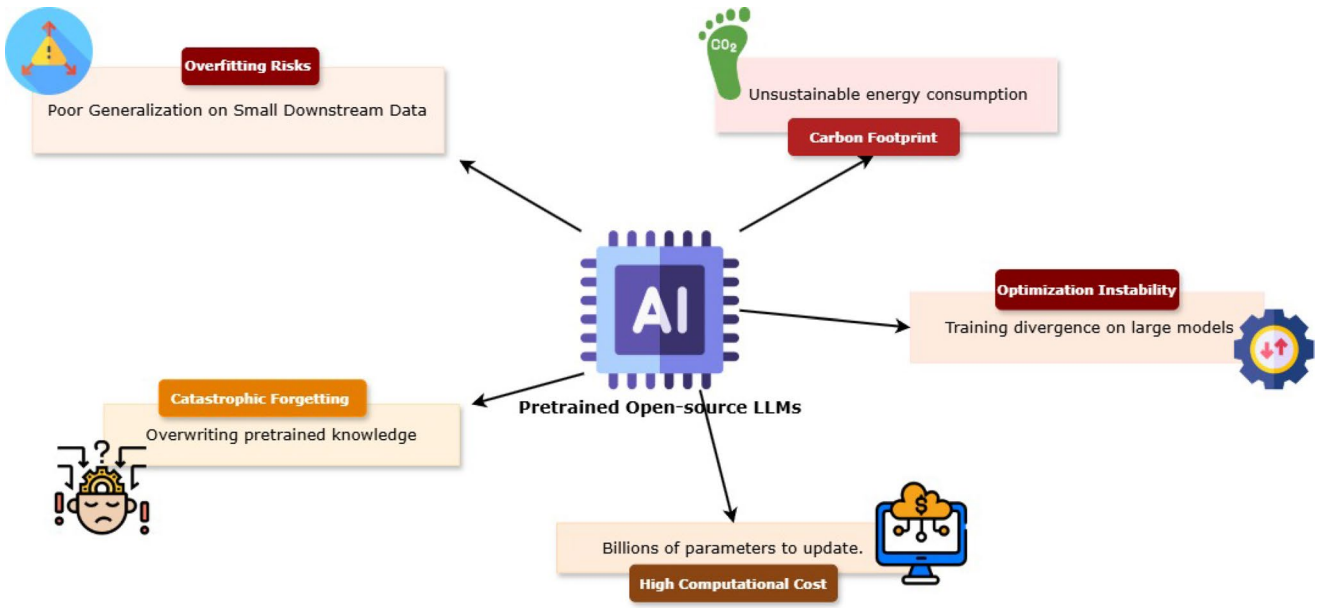


Fig. 6 Challenges of full fine-tuning an open-source LLM

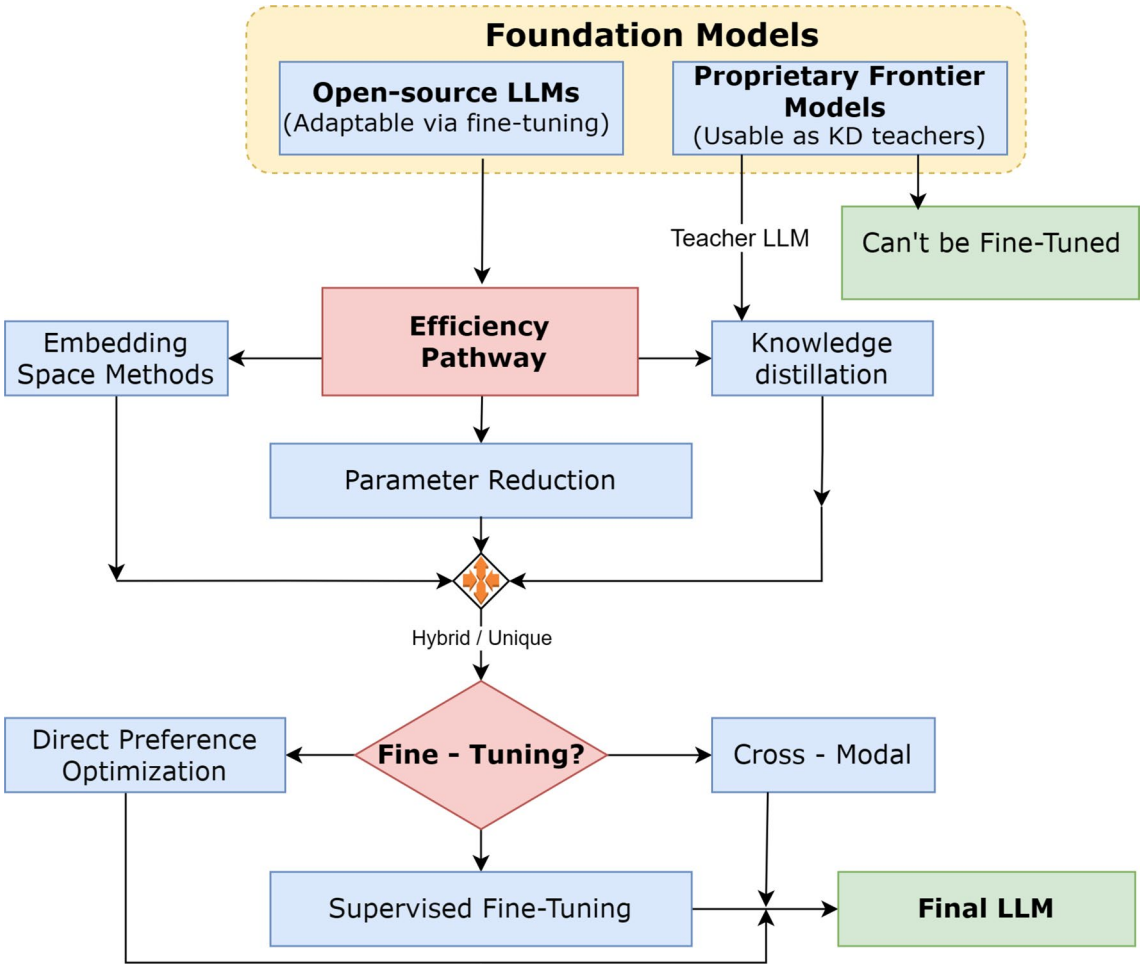


Fig. 7 Flowchart overview of model-intrinsic and distillation-based strategies

Fig. 8 Weight pruning

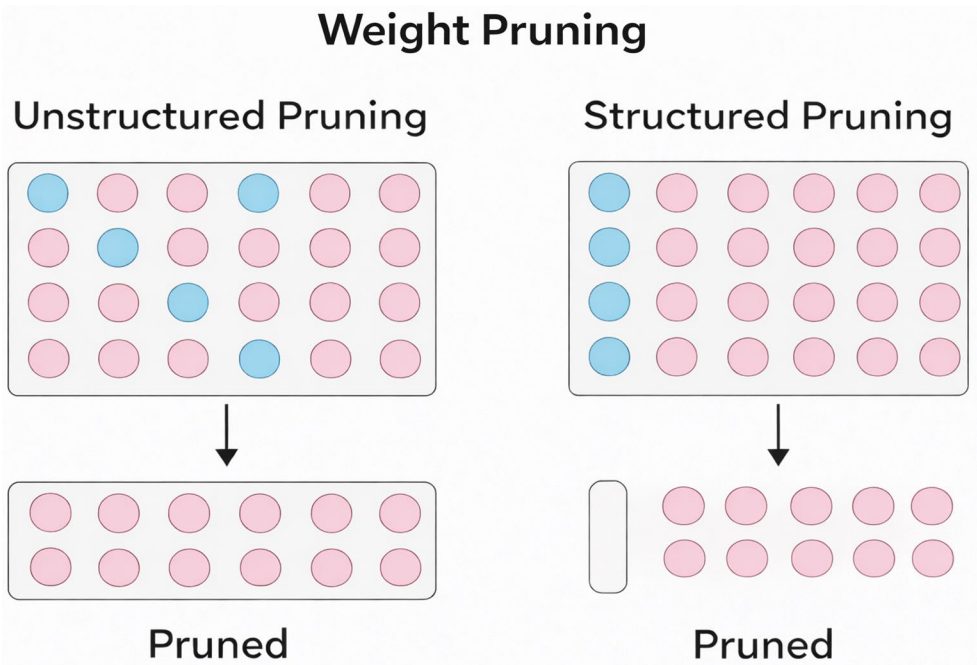


Table 2 Comparison of various pruning techniques for LLaMA-7B

Model	Pruning Strategy	Pruning Ratio	#Parameters	Memory (MiB)	Latency (s)	Accuracy (%)
LLaMA − 7B	–	–	6.74B	12884.5	69.32	78.89
	Channel	20	5.39B	10363.6	61.50	66.87
	Block	20	5.42B	10375.5	58.55	77.15
	Channel	50	3.37B	6556.3	40.11	76.55
	Block	50	3.35B	6533.9	37.54	77.91

Table 3 Performance comparison across pruning models on healthcare and legal benchmarks

Models	Rouge [Healthcare]	Perplexity [Legal]
Dense [LLaMA-7B]	23.77	2.26
Magnitude [60]	28.29	8.64
LLM-Pruner [37]	22.34	32.22
SparseGPT [36]	36.22	2.62
D-Pruner [61]	45.38	2.70

Examples include channel pruning [54], motion pruning [55], and pruning attention heads in transformers [56]. The intermediate approach would be a semi-structured pruning (or N: M sparsity) with enforced regular sparsity structures, such as selecting M out of every N weights, to leverage efficiency advantages by modern accelerators [57].

Recent work focuses on task-aware pruning, which is a pruning technique adaptable to the application of large models. Movement Pruning [55] is a type of pruning strategy and is dynamic in nature, and changes while it is fine-tuning. Recent work [36, 58] also demonstrates that application-specific pruning is beneficial for improving the efficiency of models while maintaining competitive results. The Table 2 [37] presents an overview of the key statistics for the LLaMA 7B-scale model, including parameters, memory usage, latency, and accuracy. The measurements are conducted in an inference setting, in which a model is given an input sequence consisting of 64 tokens. The performance of the models is evaluated in a task-agnostic manner through a commonsense reasoning benchmark, PIQA [59]. This is conducted to assess the performance of the models in order to understand the impact of pruning strategies on the overall performance. The values presented in the table clearly indicate that there is a trade-off between performance and efficiency due to different pruning strategies. The increase in pruning ratio from 20% to 50% indicates that there is a significant reduction in parameters. This directly affects memory and latency. For instance, models pruned at 50% require almost half as much memory and perform faster compared to other models. However, there is a slight degradation in accuracy. The key point is that the accuracy for the Block pruning strategy is higher compared to Channel pruning at similar levels of pruning. Moreover, it performs better in terms of latency. This indicates that pruning larger structured units (blocks) retains more meaningful representations compared to removing individual channels.

The results in the Table 3 clearly show that D-Pruner [61] produces the best performance, with the highest Rouge score (45.38) on PubMedQA [62], accompanied by the lowest perplexity (2.70). Compared with the dense model, which has lower performance in the Rouge score, pruning-based models can improve task performance instead of reducing it. Among the baseline models, Sparse GPT [36] produces comparable performance with balanced Rouge (36.22) and the lowest perplexity (2.62), whereas magnitude pruning improves the Rouge score with much higher perplexity (8.64) than the baseline models. On the other hand, LLM-Pruner [37] produces the worst performance with the highest perplexity (32.22) in the experiments, which means it fails to generalize well on the test data. The experiments demonstrate that better structured pruning models, such as D-Pruner [61], can maintain better reasoning and language fluency than conventional pruning models.

In general, pruning has remained a viable method worth exploring for the compression of large models, especially LLMs. The viability of pruning is sensitive to the chosen granularity or the targeted application, as well as the compatibility constraints.

4.1.2 Quantization

Quantization is a basic compression technique that comes with a loss in parameter precision, splitting it into discrete steps with a fixed number of values. The effect of this technique is to optimize memory and execution time with a consequent reduction in bits required to represent weights and activations. The simplest quantization technique is round to nearest (RTN) or simply round-off to the nearest integer. Although it is a simple computation technique, it suffers from a basic drawback such as a loss in model accuracy. In an attempt to improve on this technique and overcome its shortfalls, models have resorted to using non-uniform steps.

The efficiency of quantization is heavily affected by two major aspects of quantization: bit precision and data type. Reducing bit width leads to better compression and, for example, cut- down FP32 weights to INT8 precision (Fig. 9), achieving a 4x reduction in the model size and also faster arithmetic operations using integers on a modern GPU and/or CPU. Recent work has moved towards aggressive quantization ($\leq$ 4-bit) [63–69], though such dramatic cuts are

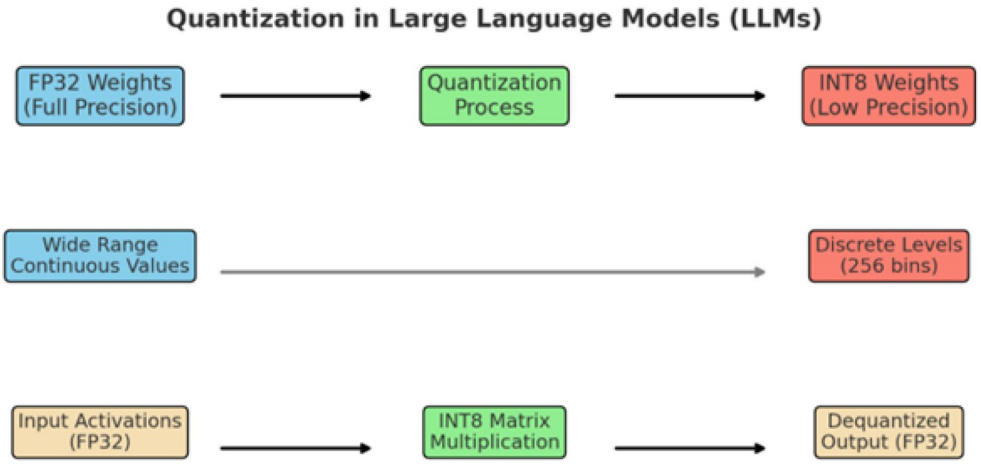


Fig. 9 Quantization

often made at the expense of model accuracy. To this end, there is mixed-precision quantization, which assigns bit widths based on sensitivity, such methods having been successfully implemented on Transformer encoder models [70] and decoders [71, 72]. In turn, new datatypes like BFloat16 [73] and NormalFloat4 [74] have been proposed balancing efficiency and representational quality.

One significant aspect is whether quantization is done with retraining or without. Quantization-Aware Training (QAT) is performed during fine-tuning to compensate for precision losses incurred during weight and activation quantization. For sub-models, for example, BERT, approaches based on QAT, even at extreme points of quantization, have been proven useful [65, 69, 70, 75–77]. Nevertheless, for large LLMs [34, 35] containing parameter sizes in the range of hundreds of billions, it is computationally infeasible to retrain them. In this regard, the feasible methodology is Post-Training Quantization (PTQ), because it completely refrains from the retraining phase. PTQ applies direct weight and activation quantization for pretrained models, and recent research has attempted to improve its resilience for performance on par with QAT [78, 79].

Nevertheless, quantization has remained essential in LLM compression techniques, finding a balance between the size of the model and the execution efficiency of the LLMs. Current research is expected to lead to the development of more effective precision reduction approaches for LLMs to support their deployment in practical environments.

4.1.3 Low-Rank Approximation

Low Rank Approximation (LRA) is a compression method to represent the original high-dimensional matrix or tensor as the product of the smaller, low-dimensional matrices that approximate the original. This method works particularly well with language models because language model parameters have low-rank structure [43, 80], that is, the information in the language model can all be represented in a lower-dimension space.

One of the common applications of LRA is in the parameter efficient fine-tuning (PEFT) techniques [42, 81, 82], which aim to adapt large models without updating all parameters. In PEFT, the pretrained model weights remain fixed while only the low-rank adapters are trained using gradient descent. After fine-tuning, the adapters are merged back into the original weights, leaving the overall model size unchanged. This drastically reduces memory requirements by limiting gradient updates, making PEFT particularly suitable for fine-tuning billion-parameter LLMs under constrained resources.

LoRA [42] was the first method to introduce low-rank adapters for fine-tuning, enabling efficient adaptation of large models. However, the problem is to choose the rank of the adapter. AdaLoRA [82] tackles this problem by treating the choice of rank as pruning. This algorithm uses dynamic approximations of the usefulness of the rank-one components and gets rid of the least helpful ranks. QLoRA [74] is an improvement on AdaLoRA. QLoRA uses the compression of model weights to 4-bit NormalFloat representation and 16-bit BrainFloat LoRA to perform the fine-tuning. This results in reduced costs. Moreover, QLoRA is the first method to propose the concept of double quantization. This helps to reduce the cost of the parameters of the quantization module.

In addition to being viable stand-alone techniques, it has been observed that PEFT approaches improve other compression techniques such as pruning [61] and quantization [74, 83], because these methods allow for memory-constrained fine-tuning and facilitate accuracy recovery during intense compression.

The Table 4 [86] represents the trade-off between memory, cost, speed, and performance for various models. It can be observed that both QLoRA [84] and IR-QLoRA [85] significantly reduce the memory requirement with reasonable

Table 4 Comparison of quantization methods in terms of memory, training time, inference latency, and accuracy

Method	Memory (GB)	Training Time (hr)	Inference Latency (token/s)	Accuracy (%)
LLaMA	-	-	95.6	34.6
QLoRA [84]	7.2	15.3	88.6	38.4
IR QLoRA [85]	7.4	15.4	83.1	40.8

training times [85]. On the other hand, the baseline LLaMA model achieves higher inference speeds (95.6 tokens/s) but lower accuracy (34.6). It can also be observed that both QLoRA [84] and IR-QLoRA [85] slightly degrade the inference speed due to the overhead of dequantization but offer significant improvements in accuracy, with QLoRA showing the best performance in terms of accuracy (40.8).

In general, low-rank approximation techniques have come to play a crucial role in reducing the costs of fine-tuning, thus ensuring that language models, when adapted on less powerful hardware, do not lose any performance.

4.1.4 Neural Architecture Search

Transformer models are normally associated with high computational expenses, though highly efficient, and it becomes difficult to implement them in hardware with constraints on computational resources and latency. In order to alleviate such difficulties, the concept of Hardware-Aware Transformers (HAT) was proposed [87, 88], which utilises neural architecture search to discover efficient architectures that can maintain accuracy as well as efficiently accommodate hardware constraints. In simpler terms, the concept of HAT involves searching for the best values of different architectural parameters that can meet the required hardware constraints while maintaining accuracy.

Conventional NAS approaches require training and testing various architectures as candidates independently, which can be a slow process. The challenge posed by NAS is addressed by the proposal of Super Transformer, a large model that defines a vast set of sub-Transformer architectures. The Super Transformer model shares the same weights among a variety of submodels, thus saving the training of every candidate independently. To speed up the searching process, the model will use a delay predictor which is first trained offline. It will predict the inference delay of every candidate architecture directly, thereby bypassing costly runtime measurements.

There are a few lessons that can be derived from the design of HAT. Firstly, the optimizations of the encoding modules can lead to positive impacts upon the decoding modules. Secondly, hardware-specific tendencies should be essential in an NAS framework, as exemplified by the fact that a shallower and broader Transformer structure is usually preferred in a GPU and a narrower and deeper structure in an ARM CPU device.

Table 5 Comparison of NAS with vision transformer-based methods on ImageNet

Method	Accuracy (%) on ImageNet	Latency (ms)	Parameters (M)
ViT-B/16 [89]	77.9	70	86
DeiT-B [90]	83.1	68	86
AutoFormer [91]	83.4	8.4	23
LLM-NAS-ViT-Base [88]	82.5	5.4	20.2

The results in the Table 5 demonstrate the effectiveness of the proposed LLM-NAS [88] approach in optimizing the trade-off between accuracy, latency, and model size. For instance, though the accuracy of the conventional models, such as ViT-B/16 [89] and DeiT-B [90], is comparable to that of LLM-NAS, the latency of the conventional models is much higher (68–70 ms) compared to LLM-NAS (5.4 ms). Similarly, the parameter size of the conventional models is larger (86 M) compared to LLM-NAS (20.2 M). Moreover, though AutoFormer [91] achieves comparable efficiency, LLM-NAS can offer a better trade-off by selecting architectures on the Pareto front. These improvements are primarily due to the optimization of key architectural parameters such as embedding dimensions and depth, which directly affect the computational complexity of the models. From the above analysis, it can be concluded that LLM-NAS [88] can efficiently discover architectures with better efficiency without compromising accuracy.

In summary, the HAT provides a useful paradigm for modifying the Transformer model under hardware and latency constraints. Most importantly, since it optimizes the structural parameters independently of the overall model architecture, the HAT can easily be combined with other efficient methods for modifying the Transformer model.

4.1.5 Dynamic Networks

The scalability of LLMs is accompanied by a large computational requirement, which is a hindrance to the efficiency of these models. Dynamic neural networks (DyNNs) resolve this problem by engaging only a set of parameters of the model for a given input.

One of the most popular DyNN architectures is the Mixture of Experts (MoE), which adds multiple parallel subnetworks (or experts) to the feed-forward network (FFN), of which only a few are active for the forward pass. This results in sparsity such that even if the capacity is enormous, the number of parameters being used for the input is small, thereby offering fast inference without compromising capacity [92, 93]. In a sparse MoE network, parameters are

split into N experts, which keep their weights separate and unchanged. The gating network function $G(\cdot)$ specifies how the input x (or token embeddings in LLM) is distributed to the experts, resulting in a sparse vector indicating which experts to activate. The MoE output y is a weighted sum of the output of the activated experts. Because only a part of the parameters is being activated for every input, MoE networks provide efficient learning with fast inference speeds relative to dense counterparts.

Adding MoE to the Transformers usually includes replacing FFN in certain parts of the model with MoE, and each of these experts is a FFN layer from a standard Transformer architecture. By scaling up the number of experts, it has been possible for model sizes to contain billions and even trillions of parameters with a controlled computational complexity with respect to each instance of inputs fed to the model. The fundamental ingredients of any MoE algorithm are finally divided among three categories: routing algorithms, architectural decisions, and learning methods.

Recently, this method has proven its efficacy. Mixtral [93], for example, a variant of the Mistral-7B model, substitutes all the FFN sub-blocks in the former with an MoE layer that has eight experts and a top-2 gating network. This gives Mixtral 8×7B [94], which uses only 13B parameters per prediction and performs better than other models like LLaMA 2–70B and GPT-3.5, thus establishing the efficacy of dynamic models in efficiently scaling up LLMs.

4.1.6 Parameter Sharing

Parameter Sharing is a compression technique where the same parameters are used multiple times in different components of the model. It is responsible for not only compressing the number of parameters [95] but also helping the model generalize faster [96]. Existing parameter-sharing methods can broadly be divided into sublayer sharing [97–100] and matrix sharing [101, 102] approaches.

In sublayer sharing, parameters are reused across multiple sublayers. Early methods adopted aggressive sharing schemes in which all sublayers across the network share identical parameters [97, 103, 104]. While this strategy drastically reduces model size to that of a single-layer network, it typically leads to accuracy degradation due to reduced representational capacity. To address this, later works introduced group-wise sublayer sharing [44, 100], where parameters are shared only within groups of sublayers, striking a better balance between compactness and expressiveness. For instance, [100] studied different grouping patterns, while Pea-KD [44] proposed shuffled parameter sharing, where query and key matrices are shuffled before being shared.

Matrix sharing methods, on the other hand, operate within sublayers. These are particularly effective for decoder-only Transformers, where memory and latency during autoregressive decoding are critical. The central idea is to share key and value projection matrices across attention heads, thereby reducing memory load and computational cost. While Multi-Query Attention (MQA) [101] enforces sharing across all the heads, Grouped-Query Attention (GQA) [102] relaxes this constraint by sharing within smaller groups of heads. GQA achieves a more favorable accuracy–latency trade-off compared to MQA and was adopted in Large models like LLaMA2 70B [94] are thus of great practical relevance for state-of-the-art LLMs.

Overall, parameter sharing provides a very elegant way to reduce the large model redundancies while keeping competitive performance. In fact, the effectiveness of these methods relies largely on the granularity of sharing, where group-based strategies often hit the balance between efficiency and accuracy.

4.2 Embedding-Space and Prompt-Based Techniques

Large Language Models are known to consume a lot of resources when fine-tuning them fully, thereby giving rise to parameter-efficient fine-tuning (PEFT) techniques. In PEFT techniques, rather than adjusting the weights of a large language model consisting of billions of parameters, a very small set of parameters related to the task is adjusted while the large language model is frozen. Among these, soft prompting, prompt tuning, and prefix tuning stand out as influential approaches that adapt pretrained models with lightweight modifications at the input or intermediate layers.

Table 6 provides a conceptual overview of techniques discussed in the Sect. 4.2, categorizing them by their tuning location, parameter efficiency, and primary task domains. These methods collectively represent a shift toward functional efficiency, where a small set of learned parameters enables a large and frozen foundation model to perform as a specialized expert.

4.2.1 Soft Prompting

Soft Prompting was first explored by [105], who introduced the idea of optimizing continuous, trainable embeddings (known as soft tokens) prepended to the model's input. Unlike handcrafted text prompts, these soft prompts are learned via gradient descent and allow the model to encode task-specific signals without altering its original parameters. This method paved the way for treating prompts not as discrete natural language tokens but as flexible vectors that can

Table 6 Conceptual Comparison of embedding-space techniques

Method	Tuning Location	Trainable Parameters	Primary Task Domains
Prefix-Tuning [105]	Prepends vectors to every layer's attention (K, V)	∼0.1%	NLG (Summarization, Table-to-Text)
Prompt Tuning [106]	Prepends vectors to input layer only	<0.01%	NLU (Sentiment Analysis, Simple Text Classification)
P-tuning v2 [107]	Multi-layer continuous prompts	∼0.1%	Deep NLU (NER, QA, Sequence Tagging)
Visual Prompt Tuning (VPT) [108]	Prepends to every transformer layer (VPT-Deep)	<1%	Vision (Object Counting, Image Classification)
$(IA)^3$ [109]	Rescales activations in attention/FFN	∼0.03%	NLU and MT (Translation, Logical Reasoning)

be optimized directly, making them suitable for frozen LMs in both classification and generation tasks.

Prompt Tuning [106] builds upon this idea and was the first to systematically demonstrate that freezing all pretrained parameters and only fine-tuning a small set of input embeddings can reach performance levels comparable to full fine-tuning. Prompt tuning has proved to be an effective adaptation approach for large language models, such as LLaMA, which were pre-trained models, that help in improving a specific task, irrespective of modifying the internal weights of the model. Unlike traditional fine-tune techniques, which modify the weights and require a substantial amount of processing power, prompt tuning works with a new concept of a few new weights.

The crucial point is the use of soft prompts, which represent trainable vectors prepended to the input sequence. The soft prompts represent embeddings acting as continuous tokens rather than discrete ones, which enable the model itself to recreate interpretations of its inputs along narrowed task-oriented guidelines. During training, soft prompts can be randomly initialized or use heuristic-based initialization while being updated by backpropagation. Crucially, during training, while the gradient flows back into the model, only the soft prompt model parameters are updated while keeping the model intact and frozen. Through each training epoch, these prompts learn to embed signals that guide the outputs towards becoming more effective.

It has two major advantages. First, it significantly decreases the computational burden because only a small fraction of the parameters are optimized. Second, it allows reusing the very same foundation model across multiple downstream tasks by attaching different task-specific soft prompts to it. As a result, prompt tuning balances performance enhancement. Thus, it provides low power and efficiency but is especially suitable in resource-constrained environments or applications requiring multitasking adaptability.

By modifying how the model perceives its input, rather than altering its underlying architecture, prompt tuning does not compromise the generalization abilities of the base model while providing competitive task-specific performances with the underlying architecture. Crucially, prompt tuning scales with model size the larger the pretrained model, the more competitive this method becomes. For example, with T5-XXL (11B parameters), prompt tuning achieves nearly the same performance as full fine-tuning on SuperGLUE, differing by only a few percentage points, while requiring far fewer additional parameters. The approach is memory-efficient and enables storing multiple task-specific prompts rather than entire fine-tuned models. However, training stability is a major challenge: convergence depends heavily on initialization and learning rate, and sufficient training data is required for robust performance.

4.2.2 Prefix Tuning

Prefix Tuning [105] extends the efficiency of prompt-based methods by introducing trainable continuous prefix vectors that are injected into every transformer layer, rather than only the input layer. In this approach, learnable prefix key-value pairs are prepended to the self-attention mechanism of each layer. To stabilize training, these prefix vectors are generated through a reparameterization mechanism: small trainable matrices are passed through an MLP to produce layer-specific key and value embeddings. This design has several advantages: (1) prefix tuning achieves robust low-resource performance, often surpassing full fine-tuning in scenarios with limited labeled data; (2) since the pretrained model parameters remain frozen, cross-domain generalization is preserved; and (3) it enables efficient multi-task batching by maintaining separate prefixes for different tasks. However, there are constraints to this process. The prefix fine-tuning usually takes longer compared to fine-tuning, especially for large data, and performance is sensitive to prefix length and initialization approaches.

In a nutshell, all these approaches indicate that to make small models smarter, it is not always necessary to train the models end-to-end. In other words, these approaches propose the idea of prefix training in order of prompts or prefixes; models can be fine-tuned for various tasks using a significantly smaller amount of computation. Moreover, other works such as P-Tuning v2 [107], Visual Prompt Tuning [108], and recent PEFT surveys [109, 110] highlight that prompt- and prefix-based strategies are becoming central to practical LLM deployment, particularly where memory, compute, and storage efficiency are critical. These methods

Table 7 Performance gains on NLU tasks

Method	Trainable Params	Avg. GLUE Score	Note
Full Fine-Tuning (FT)	124.6 M	84.00	Baseline
Prompt-Tuning	0.61 M	74.56	Poorest performance on small models
Prefix-Tuning	0.96 M	80.89	Significant improvement over Prompt-Tuning
$(IA)^3$	0.66 M	83.09	Extremely efficient and close to FT
BitFit	0.69 M	84.68	Outperforms FT on this specific scale

provide mechanisms for adapting pretrained language models to downstream tasks without the overhead of full fine-tuning. These techniques make models more flexible by introducing small, learnable components while keeping most of the model frozen. Alongside them, two other strategies, In-Context Learning (ICL) and Retrieval-Augmented Generation (RAG), also enhance model adaptability at the usage level rather than through weight updates, making them natural extensions to this discussion.

Table 7 compares the performance of various embedding-space methods on a relatively small model (RoBERTa-base), using the average GLUE score as a benchmark [109]. Embedding-space methods were designed for efficiency, and as observed from the table, their effectiveness varies significantly for small models like RoBERTa-Base (125 M). BitFit is a sparse method that updates only the bias terms of the Transformer layers and the task-specific classification head while keeping the rest of the model frozen. It is observed that this technique outperforms others as it preserves the pre-trained knowledge better than full fine-tuning, which is prone to "catastrophic forgetting" with limited training data. Under similar settings(limited data with small models), simple prompt-tuning fails as it tunes only the input-layer embeddings, and that is not expressive enough for the NLU downstream tasks.

4.2.3 In-Context Learning

In-Context Learning(ICL) [111, 112] enables LLMs to adapt to new tasks on-the-fly by conditioning on carefully designed input prompts that include examples of the desired behavior (e.g., demonstrations or task descriptions). Unlike fine-tuning, ICL requires no parameter updates. The model leverages its pretrained knowledge and the contextual signals in the prompt to produce task-specific outputs. ICL demonstrates that LLMs can act as meta-learners, dynamically learning from input context. This paradigm is highly flexible and efficient, though it depends heavily on prompt design and often requires large model sizes to perform competitively.

4.2.4 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) [113, 114] improves model performance by combining pretrained LLMs with an external knowledge retriever as shown in Fig. 10. Instead of relying solely on the model's internal parameters, RAG retrieves relevant documents or facts from an external corpus (e.g., Wikipedia, domain-specific

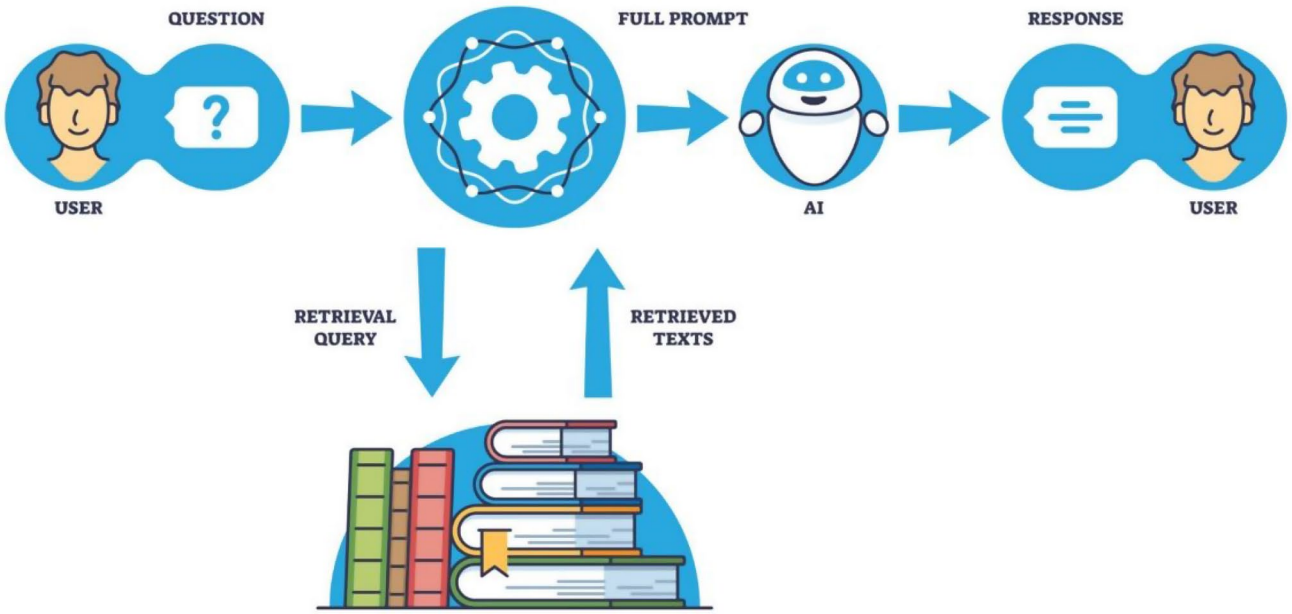


Fig. 10 Retrieval-augmented generation

knowledge bases) and conditions the generation process on this retrieved information. Such an approach not only lightens the burden on the model to memorize the knowledge but also improves the factual correctness and flexibility of the model. It is worthwhile to note the significance of the RAG model as it is much lighter compared to the retraining and can easily be combined with PEFT techniques or strategy-based approaches, which makes it a prominent area for the development of smarter models with reduced resources.

By including prompt-level adjustments (soft prompting, prompt tuning, prefix tuning) and usage-level adjustments (ICL and RAG), the list of tools for reducing the size but increasing intelligence becomes much larger. While PEFT reduces the cost of adapting models, ICL and RAG reduce the need for adaptation altogether by leveraging contextual cues and external memory. Together, these strategies enable practical, scalable, and efficient deployment of LLMs in real-world, resource-constrained scenarios.

Table 8 highlights the operational advantages of embedding-space and prompt-based techniques in specialized settings, particularly in data-constrained or cross-domain scenarios [105, 106].

5 Knowledge-Enhanced Approaches: Distillation, Synthetic Data, and External Guidance

While the previous chapter focused on model-intrinsic strategies such as parameter reduction and prompt-level techniques, this chapter explores approaches that rely on external sources of knowledge to enhance the capabilities of smaller models. Modern systems increasingly use very large teacher models, synthetic supervision pipelines, and structured external guidance to transfer abilities that compact models cannot learn from data alone. These knowledge-enhanced approaches enable small models to acquire factual competence, reasoning patterns, and task-solving strategies that would otherwise require scale. By leveraging external intelligence rather than expanding model size, these methods offer a practical route to building lightweight yet high-performing transformers suitable for resource-constrained environments.

5.1 Knowledge Components for Student Model Enhancement

Knowledge distillation transfers multifaceted capabilities from teacher to student models. The knowledge in the context of knowledge distillation refers to internal representations, patterns, problem-solving, and reasoning capabilities, encompassing not just factual and inferential functions but also complex cognitive functions. The knowledge transferred from the teacher may be responses, features, or relations, distilled into the student model. The knowledge can be categorized into multiple components: Factual knowledge, reasoning ability, task-solving ability, and Chain of thought reasoning.

5.1.1 Factual Knowledge

Factual knowledge refers to the information encoded in a teacher model's *output distributions*. In classification models, this corresponds to the probability distribution over class labels, while in autoregressive models, it represents the probability distribution over the next token in a sequence. Instead of training on hard, one-hot labels, knowledge distillation uses the teacher's *soft targets* (logits), which reveal confidence levels and similarity relationships between outputs. These soft distributions provide richer supervision, reduce overconfident predictions, and act as a variance–reducing regularizer that often improves generalisation. Prior analyses [115] describe this effect through a bias–variance tradeoff: soft targets may slightly increase bias but substantially reduce variance. Modern distillation frameworks extend this idea by incorporating loss reweighting and adaptive strategies to ensure that soft-target supervision yields maximal downstream performance [116].

Table 8 Functional advantages of embedding-space techniques in specialized settings

Feature	Embedding-Space Advantage
Limited Training Data	Prefix-tuning outperforms full fine-tuning when training examples are scarce (e.g.,< 200 examples) and effectively mitigates overfitting on small datasets.
Domain Shift	Prompt-tuning achieves better zero-shot transfer when shifting from standard domains (e.g., Wikipedia) to specialized ones like biomedical data.
Storage Efficiency	Requires 1,000×to 20,000×fewer task-specific parameters to be stored (e.g., 20K parameters for an 11B model) compared to saving full model checkpoints.
Inference Efficiency	Mixed-task batching using a single frozen foundation model copy, which is not possible with other PEFT methods like adapters.
Ensemble Efficiency	Multiple prompts can be trained for the same task and processed in a single batch, providing the uncertainty estimation benefits of an ensemble at the cost of one model.

5.1.2 Reasoning Knowledge (Latent Representations)

Reasoning knowledge refers to the latent representations that a teacher model computes while processing an input, such as hidden states, attention patterns, and inter-token relational structures. These internal activations encode how the teacher integrates information, captures long-range dependencies, and performs implicit multi-step reasoning, regardless of whether the task is classification or sequence generation. Representation-level distillation aims to transfer this reasoning ability by aligning the student's intermediate computations with those of the teacher. This is typically achieved through losses applied to hidden states, projection-aligned features, or attention relations, allowing the student to approximate the teacher's internal processing dynamics rather than only its final outputs. Such techniques are widely used in encoder-only and encoder-decoder transformers, where attention mechanisms and hierarchical feature transformations play a central role in reasoning. The primary loss functions used to distil latent representations are summarised in Table 9.

5.1.3 Task-Solving Ability

Task-solving ability refers to the procedural, multi-step, and strategic competencies that a large teacher model acquires when performing complex tasks end-to-end. Unlike factual knowledge (captured through logits) or reasoning knowledge (captured through latent representations), this form of knowledge reflects the entire problem-solving behaviour of the teacher model. Many tasks require decomposing a problem into intermediate steps, such as arithmetic reasoning, logical deduction, and multi-hop inference, where larger models naturally generate coherent subgoals and maintain reasoning consistency across steps [124, 125]. Other tasks require deeper semantic interpretation, such as identifying intent, emotion, or causal cues in language [126, 127], which small models often struggle to infer directly from surface-level representations. Task-solving ability also includes strategic components: selecting appropriate reasoning paths, filtering implausible interpretations, and applying heuristics learned during large-scale pretraining. In addition, many real-world tasks demand adherence to structured output formats (e.g., JSON schemas, SQL queries, code syntax, or decomposed reasoning followed by answers), where structure itself becomes part of the knowledge to be transferred. Together, these abilities represent the teacher's holistic competence at executing complex tasks, and distillation aims to impart these high-level procedural skills to compact student models. A visual summary of these distillable knowledge forms is shown in Fig. 11.

5.2 Distillation Methods

Knowledge distillation encompasses a variety of strategies that transfer learned representations and behaviours from a larger, more complex model (the teacher) to a smaller, more efficient model (the student). Each method targets different aspects of the teacher's internal knowledge and operational behaviour. The core idea is to transfer the knowledge described in the above sections from the teacher model to the student model. This transfer of knowledge can occur through several distinct mechanisms. Below are some of the key approaches commonly employed in distillation research and practice.

5.2.1 Vanilla Knowledge Distillation (Logits Matching)

The foundational approach to knowledge distillation involves training a compact student model to replicate the softened output distributions (logits) produced by a larger teacher model. Since the student model focuses on mimicking the final output probabilities of the teacher model, it is also referred to as response-based distillation and was first introduced by Hinton et al. [128]. The student model is trained to minimize the Kullback-Leibler (KL) divergence between its own softened softmax outputs and those of the teacher. By minimising the divergence between the student's and teacher's output probabilities, typically with temperature scaling to soften the distributions, the student

Table 9 Loss functions used for distilling latent reasoning representations from teacher to student models

Loss Type	Explanation
Attention Map Loss	Aligns teacher and student attention distributions across heads and layers, allowing the student to inherit the teacher's dependency structure [117, 118].
Token-Relation Distillation Loss	Aligns the relational structure between tokens by matching the teacher's query–key interaction matrix and value–value similarity patterns. This preserves how the teacher encodes semantic dependencies and compositional meaning internally [119, 120].
Loss on Hidden Representations	Matches teacher hidden states with student hidden states directly using MSE loss or via learned projection layers when model dimensions differ. Transfers deeper semantic and feature-space information [120, 121].
CKA-Based Geometry Alignment	Uses Centered Kernel Alignment (CKA) to transfer representational geometry, enabling alignment even when the teacher and student have different widths or depths [122].
Contrastive Feature Loss	Encourages student representations to be closer to corresponding teacher representations relative to negative samples, improving discriminative ability. Used in Contrastive Representation Distillation [123]

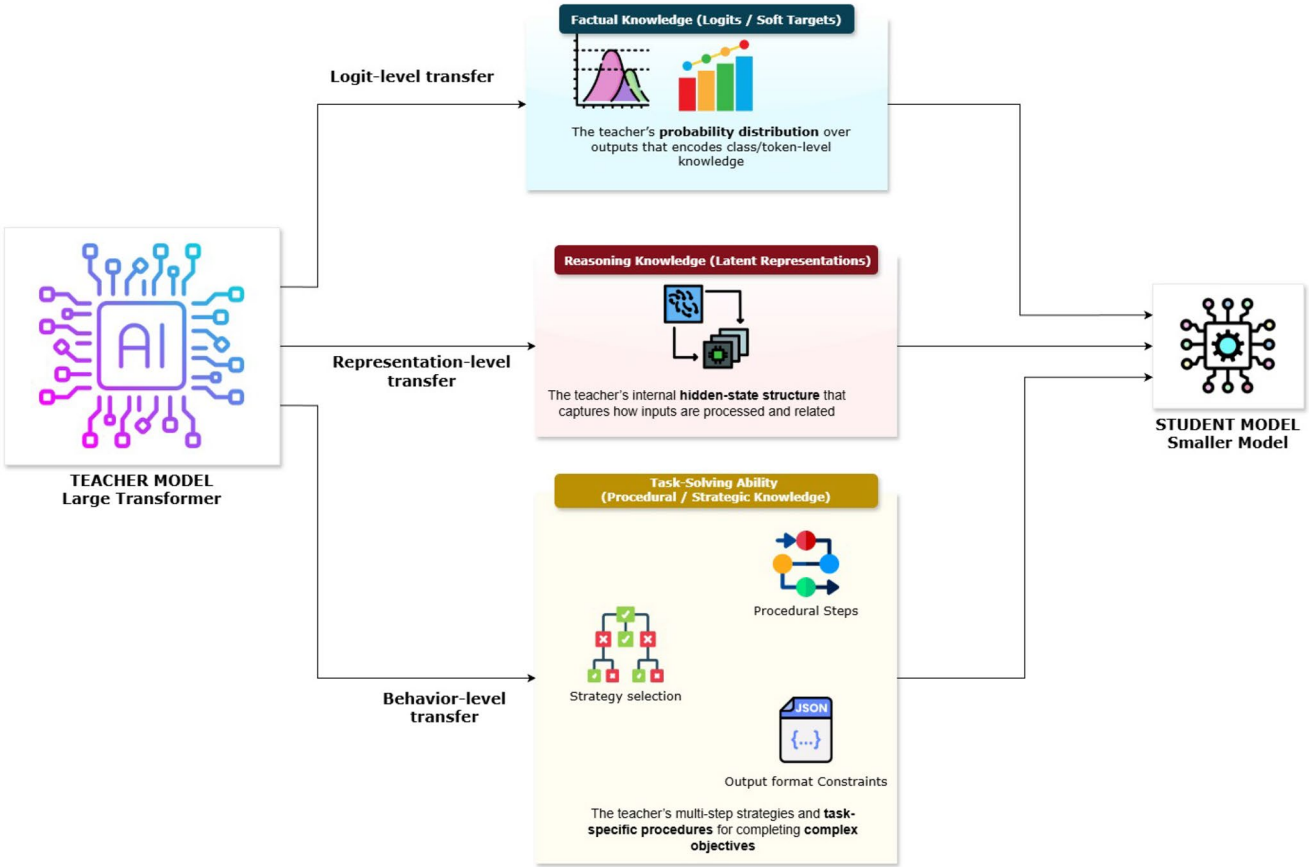


Fig. 11 Types of knowledge distilled from teacher to student models

can capture not just the correct class but also the teacher's confidence across all classes. This method is model-agnostic and easy to implement, making it widely applicable across domains such as image classification and language modelling. However, this technique only focuses on the final output layer, potentially ignoring the valuable intermediate layer knowledge.

5.2.2 Intermediate Representation Distillation

Unlike response-based distillation, where the student learns solely from final outputs, this method attempts to replicate the internal representation and activation of the teacher at various hidden layers. Beyond output logits, this technique focuses on aligning the hidden layer activations and feature maps between the teacher and student models. The matching between intermediate layers is done via mean squared error (MSE), attention maps, or Gram matrices between corresponding layers. Romero et al. [129] introduced the concept of using intermediate layer activations (hints) from a teacher network to guide the training of a thinner student network, laying the foundation for intermediate representation distillation. The method demonstrated that even compact models could perform competitively if provided with strategic supervision beyond just final outputs. Although this technique captures better information and improves generalisation, it requires more careful consideration of the teacher and student architectures to identify suitable layers for matching. Moreover, it is complex to implement compared to vanilla KD, often involving additional loss terms for each distilled layer. Recent studies like TinyMIM [130] confirm that intermediate representation transfer, including relational token structures, remains highly effective for modern transformer models.

5.2.3 Feature-Based Distillation

Feature-based distillation extends intermediate representation distillation by transferring more structured or semantically meaningful properties of the teacher's feature spac,e such as attention patterns, spatial activations, or relational token dependencies. Rather than simply aligning raw hidden states via L2 loss, this approach focuses on preserving higher-level inductive biases learned by the teacher. For example, Heo et al. [131] distil activation boundary maps that capture which neurons are active versus inactive,

enabling students to learn decision boundaries more faithfully. Similarly, Zagoruyko et al. [132] introduce attention transfer, where the spatial attention maps of the teacher guide the student's focus during training. Such feature-centric objectives have been shown to improve performance in tasks like object detection, semantic segmentation, and transformer-based language modeling by enforcing richer structural alignment between teacher and student models.

5.2.4 Self-Distillation

Self-distillation refers to the process in which a model serves as both teacher and student, transferring knowledge from one version of itself to another, either across training iterations, across layers, or across checkpoints, without requiring an external larger model. The foundational concept underlying self-distillation is dark knowledge [128]: the full soft probability distribution over classes or next-token candidates, rather than only the final answer. These soft targets encode subtle information about class similarity and uncertainty. For example, when recognizing an animal, a teacher model may assign probabilities such as cat: 0.51, fox: 0.33, dog: 0.14; even though the teacher's prediction is correct, the relative probabilities tell the student that foxes share more features with cats than dogs do. This richer signal enables the student model to generalize better than if trained only on hard labels.

Importantly, even incorrect teacher predictions provide informative supervision. Misclassified examples often arise from ambiguous or noisy inputs, and the teacher's soft distribution still reveals useful relational structure. Such uncertainty patterns help the student learn smoother decision boundaries and robust representations, a phenomenon first formalized in Born-Again Networks(BAN) [133], which demonstrated that a newly initialized student can outperform the original teacher by learning from its softened outputs.

While BAN originally used a fully trained model as a teacher, later work found that earlier training checkpoints often make better teachers than the final converged model. Fully trained models tend to be over-confident, producing extremely peaked distributions with minimal dark knowledge. Earlier checkpoints retain higher-entropy predictions, richer uncertainty, and smoother inductive biases, making them stronger distillation targets. This idea underlies Snapshot Distillation [134] as well as modern LLM training pipelines.

Self-distillation has become very prominent in large language models. In the Llama-2/3 family [94], intermediate checkpoints are used as internal teachers to stabilize training and improve instruction fidelity without increasing model size. DeepSeek models [135] employ iterative self-distillation across fine-tuning stages, where earlier versions of the same model provide soft labels and reasoning traces to later versions. Similarly, Phi-1 and 2 [136] uses a staged self-training loop where the model's previous snapshot guides the next, improving reasoning despite small parameter counts.

Together, these approaches highlight that self-distillation is not limited to convolutional networks but is now a central mechanism in the development of efficient, high-performance LLMs: using a model's own intermediate knowledge, especially its early, high-entropy predictions, to build a stronger successor without increasing model capacity.

5.2.5 Dataset-Level Distillation

Dataset-level distillation, also called data-free knowledge distillation, aims to compress the information of a large training corpus into a much smaller synthetic dataset produced by a teacher model. The technique originated in computer vision, such as DeepInversion [137], and KegNet [138] generate synthetic images whose intermediate feature statistics match those of the teacher network, allowing a student CNN to learn without access to the original data.

This idea has since been extended to language models, where the autoregressive models produce synthetic text data instead of images. A key difference from discriminative datasets is that classification labels alone do not contain the rich explanations or multi-step reasoning required by generative models. To address this, large teacher models are prompted with minimal metadata to generate full instruction–response pairs, thereby transforming discriminative supervision into generative training data. This strategy is used in a number of instruction-tuning pipelines, such as Self-Instruct [139], and Alpaca is a well-known example of a synthetic instruction-tuning dataset created by Stanford CRFM using the self-instruct methodology using LLaMA-7B as the teacher model.

Recent LLM research has further expanded this paradigm. The Phi models [136] introduced "textbook-style" synthetic datasets created via a human-curated seed set combined with large-LLM expansions, demonstrating that structured, synthetic data can yield strong reasoning ability even in small models. In all these models, dataset-level distillation allows small models to inherit a substantial portion of the teacher's knowledge without requiring access to the teacher's original training data. However, data-free KD can also introduce distributional biases and may fail to fully replicate the depth of the teacher's knowledge if not carefully designed.

5.3 Distillation Settings

There are three major settings under which knowledge distillation can be performed. In offline distillation, a pre-trained teacher model supervises the student while only the student's parameters are updated during training, as formalized in early work by Hinton et al. [128]. In online distillation, the teacher and student are optimized simultaneously within a single end-to-end training loop, often using collaborative or mutual-learning strategies [136, 140]. In self-distillation [141], the same model serves as both teacher and student, with stronger or earlier checkpoints guiding weaker or later versions.

5.4 Special Forms of Distillation

Beyond these core settings, specialized distillation strategies have been developed to address challenges in transferring reasoning, interactive behavior, and cross-modal understanding.

5.4.1 Chain of Thought Distillation (Reasoning Step Transfer)

CoT focuses on transferring multi-step reasoning abilities from teacher to student models. The student models are trained to replicate the intermediate reasoning traces and attention gaps. For some of the complex problems, the process of arriving at an answer, "the chain of thought" or "reasoning process," is as important as the final result. A large teacher model is prompted to generate the reasoning steps along with its final answer. The student model is trained to replicate not only the final output but also the intermediate reasoning steps. Sequence-to-sequence CoT supervision trains the student to reproduce the full reasoning trace and final answer exactly as generated by the teacher. In this step-by-step distillation [142], the teacher's generated reasoning traces are treated as supervised targets, enabling the student to mimic how the solution is obtained from reasoning. Another variant, namely Intermediate supervision [143], provides explicit labels for the intermediate steps, allowing the student to learn the transitions between reasoning stages rather than producing only the final output. The training is strengthened by injecting losses at intermediate reasoning steps so the student learns to maintain coherent partial reasoning states. Rationale-based distillation [144] extends this idea by transferring task-specific rationales that explain why each prediction is made. Together, these methods allow student models to internalize the teacher's reasoning behaviors, improving the reasoning abilities of smaller models with fewer parameters and training data.

5.4.2 Reinforcement Distillation (Reward Signals)

In Reinforcement learning, models are trained based on reward feedback. Reinforcement distillation seeks to transfer optimal behavior and policies from a teacher to a student by replicating/recreating the policy outputs and value functions. This technique aims to transfer the learned policy, value function, or other components of a successful "teacher" RL agent to a smaller and simpler "student" agent.

In the context of large models, this idea extends to scenarios where the teacher is an RL-fine-tuned foundation model, such as those trained with RLHF or reward-model optimization. Because RLHF pipelines are expensive, which require human feedback, reward modeling, and online policy optimization, RL-based distillation provides an efficient alternative. The student does not interact with the environment or humans, but simply learns to imitate behaviors that would achieve high reward under the teacher's learned policy or reward model. Recent works, such as Ouyang et al. [145] and Rafailov et al. [146], show how reward-driven objectives can supervise smaller students by aligning them to the teacher's reward structure without running a full RL-based training.

Sections 4 and 5 review efficiency-inducing techniques reported in the literature for adapting large language models during fine-tuning. These methods are not mutually exclusive; instead, they can be combined in a hybrid manner, allowing practitioners to select techniques that best align with the target task and resource constraints. Model adaptation can be carried out using supervised fine-tuning (SFT), preference-based methods such as Direct Preference Optimization (DPO), or cross-modal attention–based approaches for language–vision models. Fig. 12 presents a taxonomy summarizing the efficiency-inducing methods discussed in these sections.

5.5 Performance and Efficiency Analysis of Distilled Models

In this section, we present a series of comparative tables to demonstrate the effectiveness of knowledge distillation. The evaluation is organized across three broad dimensions. The first table compares factors like energy consumption (in Joules), inference time, and CPU utilization. These factors directly influence the carbon footprint of large models. The second table is about the performance of distilled models on computer vision tasks, including image classification and segmentation. Although the primary focus of this survey is on the generative capabilities of AI models, knowledge distillation has historically shown significant performance gains in vision tasks. Therefore, this analysis has been included for completeness. The subsequent three tables

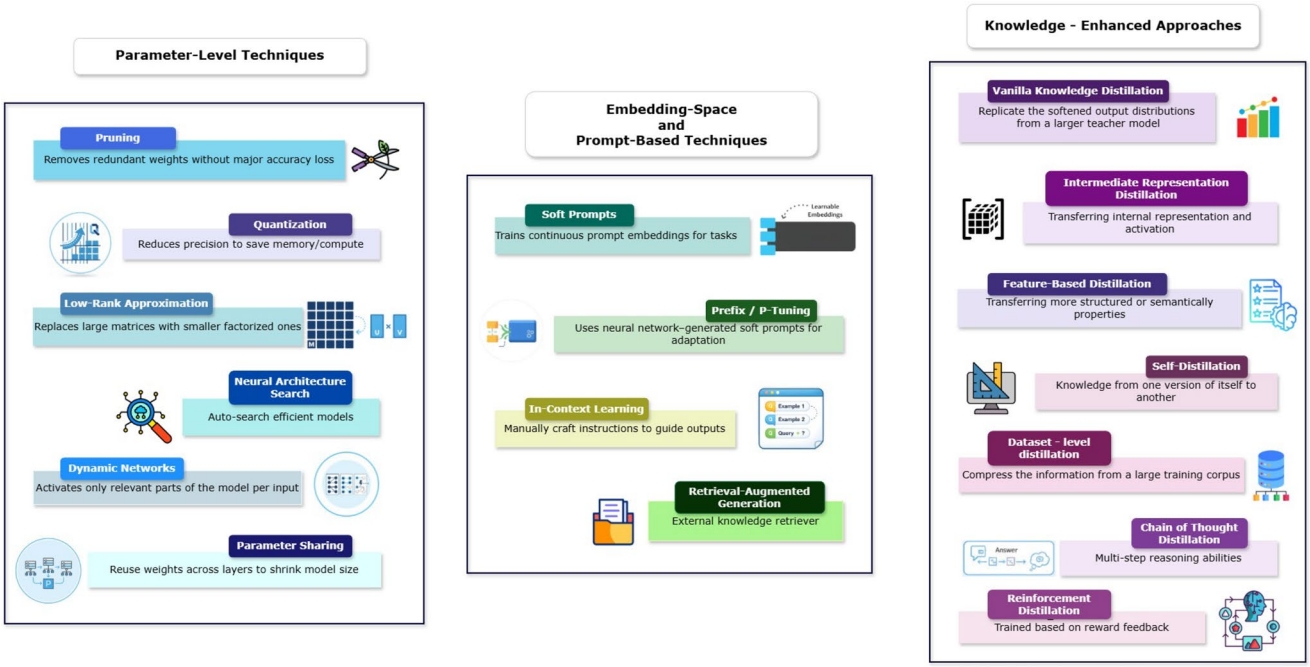


Fig. 12 Categorization of LLM efficiency techniques

Table 10 Resource efficiency Comparison

Model	Distillation	Energy (J)	Time (s)	CPU (%)
BERT [147]	Teacher	28,902.17	888.22	36.7%
DistilBERT [148]	Student(Inter./Logit)	16,234.19 (↓44%)	674.77 (↓24%)	26.4%
GPT-2 [147]	Teacher	31,228.71	797.90	49.6%
DistilGPT-2 [149]	Student(Inter./Logit)	22,464.59 (↓28%)	557.61 (↓30%)	48.3%

provide performance comparisons of distilled models on well-established generative tasks, namely Neural Machine Translation (NMT), text summarization, and reasoning and programming tasks.

Table 10 highlights how distillation reduces hardware overhead for Transformer-based models like BERT and GPT-2 while maintaining high performance. However, non-distilled models may still be preferred for tasks requiring very high accuracy or complex language understanding.

Table 11 highlights key research works where knowledge distillation has achieved significant improvements in vision tasks. This table demonstrates that a more structured form of distillation, like feature-based and relational distillation, significantly enhances performance across classification, detection, and segmentation tasks compared to Vanilla KD, which relies solely on matching output logits. Techniques like Contrastive Representation Distillation (CRD) and Margin ReLU (applied at the pre-ReLU position) allow student models to bridge the gap and even exceed the performance of their much larger teachers, such as a ResNet50 student achieving a lower error rate than its ResNet152 teacher on ImageNet. Methods like TinyMIM highlight the effectiveness of distilling token relations for dense prediction tasks, while self-distillation frameworks prove that models can improve their own generalization without an external teacher by utilizing the "dark knowledge" from earlier training states. These results suggest that smaller architectures can achieve state-of-the-art results through knowledge distillation without increasing their architectural complexity.

Table 12 presents summarization performance by mapping student/teacher architectures and calculating the absolute gains over the non-distilled baseline.

Table 12 highlights how Sequence Level Contrastive Learning (SeqCo) and InstructGPT enhance performance by treating summarization as an alignment task. In the SeqCo framework, $(x - y)$ represents contrastive learning between the source document (x) and the gold summary (y), while $(y - \hat{y})$ refers to the similarity between the gold summary and the summary generated by the student model $(\hat{y})$. SeqCo generates distillation signals using a momentum encoder, which computes a moving average of the student model's own parameters to provide stable regression targets. RLHF (Reinforcement Learning from Human Feedback), is utilized as a form of Reinforcement Distillation, where the "teacher" signal is provided by a reward model trained on human preferences. This allows smaller student models to

Table 11 Comparison of vision-based distillation methods

Method	Setting	Performance
Classification (CIFAR-100)		
Hinton et al. [128]	WRN-40–2 → WRN-16–2	75.61% → 74.92%
FitNet [129]	WRN-40–2 → WRN-16–2	75.61% → 73.58%
Attention Transfer [132]	WRN-40–2 → WRN-16–2	75.61% → 74.08%
Contrastive Distillation [150]	WRN-40–2 → WRN-16–2	75.61% → 75.48%
Margin ReLU [131]	WRN-28–4 → WRN-16–2	21.09% → 24.08%
BAN [133]	WRN-28–10 → Identical WRN	19.08% → 18.25%
Snapshot Distillation [134]	ResNet110* → ResNet110	28.53% → 26.19%
Classification (ImageNet)		
Hinton et al. [128]	ResNet152 → ResNet50	21.69% → 22.85%
Attention Transfer [132]	ResNet152 → ResNet50	21.69% → 22.75%
Margin ReLU [131]	ResNet152 → ResNet50	21.69% → 21.65%
TinyMIM [130]	ViT-S → ViT-T	83.0% → 75.8%
Snapshot Distillation [134]	ResNet152* → ResNet152	21.17% → 20.93%
Detection (Pascal VOC)		
Margin ReLU [131]	ResNet50-SSD → ResNet18-SSD	76.79 → 73.08
Segmentation (Pascal VOC)		
Margin ReLU [131]	ResNet101 → ResNet18	77.39 → 73.24
Snapshot Distillation [134]	ResNet152* → ResNet152	77.53 → 77.97
Segmentation (ADE20K)		
TinyMIM [130]	ViT-L → ViT-B	50.0 → 52.2

imitate complex behaviors that would achieve high rewards from human evaluators.

Table 13 highlights how distillation improves translation accuracy. It demonstrates that Sequence-Level KD outperforms Word-KD in translation accuracy (BLEU) by mimicking the teacher's sequence distributions rather than word distributions. It also shows that Seq-KD student models exhibit worse perplexity than their teachers, but are still able to achieve better BLEU scores because their probability distributions are more "peaked" around the correct translations. BANs prove that models of identical capacity can improve their own fluency through self-distillation.

Table 14 evaluates the efficiency of transferring high-level reasoning and "textbook" knowledge to smaller models. DeepSeek-V3-Base outperforms much larger models like LLaMA-3.1-405B on coding and math benchmarks. This success is from distilling reasoning capabilities from the DeepSeek-R1 series. Similarly, the 1.3B phi-1 model reaches 50.6% on HumanEval by using high-quality synthetic "textbooks," proving that distilling procedural logic is more efficient than increasing model size alone.

6 Fine Tuning Strategies

Fine-tuning a large language model involves optimizing a large model to perform better on a specific task or domain, such that the model retains the inherent knowledge it possesses while it also acquires domain-specific knowledge required for the task. During knowledge distillation, the transfer of all knowledge from a larger model to a smaller model is inherently a lossy task because of the mismatch in the size of the larger teacher model and the smaller student model. Hence, the student model is fine-tuned to adapt to a specific task or domain, and this makes the knowledge transfer process more efficient, with only the required knowledge distilled to a smaller student model. Some of the well-documented strategies of fine-tuning are discussed in the following sections.

6.1 Supervised Fine Tuning

Supervised Fine Tuning (SFT), which is also referred to as Full Model SFT, is a strategy where the pretrained LLM is fine-tuned for a specific task by using a paired dataset. SFTs are primarily implemented for adapting pre-trained larger models to perform specific tasks using smaller models. The smaller student model might use one of the distillation

Table 12 Text summarization performance (ROUGE-L)

Framework	Method	Teacher Signal	Student Model	ROUGE-L	Gain
SeqCo Framework [151]					
Baseline	-	BART-Large	-	40.91	-
Feature-Based	SeqCo$(x-y)$	Momentum Encoder	BART-Large (406 M)	41.38	+0.47
Feature-Based	SeqCo$(y-\hat{y})$	Momentum Encoder	BART-Large (406 M)	41.75	+0.84
InstructGPT / PPO Framework [145]					
Baseline	-	GPT-3 (Standard)	-	0.196*	-
Reinforcement	InstructGPT	Human Feedback	GPT-3 (175B)	0.220*	+0.024
Reinforcement	InstructGPT	Human Feedback	GPT-3 (6B)	0.231*	+0.034

Table 13 Evaluation of neural machine translation performance

Distillation Type	Method	Teacher Model	Student Model	Metric	Teacher	Student
Sequence-Level Knowledge Distillation [152]						
Vanilla KD	Word-KD	LSTM (4×1000)	LSTM (2×500)	BLEU (K=5)	19.5	17.7
Dataset-Level	Seq-KD	LSTM (4×1000)	LSTM (2×500)	BLEU (K=5)	19.5	19.0
Dataset-Level	Seq-KD	LSTM (4×1000)	LSTM (2×500)	Perplexity	6.7	22.7
Self-Distillation (BAN) [133]						
Self-Distillation	BAN	LSTM (52 M)	Identical LSTM (52 M)	Perplexity	71.87	68.56
Self-Distillation	BAN	ConvLSTM (19 M)	Identical ConvLSTM (19 M)	Perplexity	80.05	76.97
Reinforcement Learning-Based Distillation [145]						
Reinforcement	InstructGPT	GPT-3 (175B)	PPO-ptx (175B)	BLEU (Fr-En)	39.93	36.76

Table 14 Evaluation of reasoning and programming expertise

Type	Method	Models (T→S)	Result
Dataset	Step-by-Step [142]	PaLM (540B)→T5 (220 M)	89.5% Acc (vs 73%)
Dataset	Phi-1 [136]	GPT-3.5 (175B)→Phi-1 (1.3B)	50.6% pass@1
Reinforcement	DeepSeek-R1 [135]	R1→V3	90.2% EM (MATH-500)
Reinforcement	DeepSeek-R1 [135]	R1→V3	40.5% pass@1 (LiveCodeBench)
Self	LIMA [153]	LLaMA (65B)→LLaMA (65B)	88% satisfactory

techniques in Sect. 5.2 to fine-tune the smaller model. The smaller student model now mimics the larger teacher model in a specific task.

On the other hand, Multimodal SFT fine-tunes across modalities, allowing models to reason by combining text with images, audio, and video. Constraints such as data scarcity, capacity mismatch, and deployment constraints in multimodal models have been addressed in [154] by generating language-image instruction pairs. These pairs are then used to train the model to generate multimodal instruction following capability. In this setting, a student model is trained using teacher-generated labels or logits derived from multimodal supervision, such as image-caption pairs or visual question-answering data.

6.2 Alignment or Preference-Based Optimization

Preference (or alignment) based optimization aims to train models to generate responses that align with human preferences. Traditionally, this has been achieved through reinforcement learning approaches such as Proximal Policy Optimization (PPO) [145], where human preferences are incorporated via a reward model. These methods fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward.

In contrast, Direct Preference Optimization (DPO) [146] directly encodes preference information into the training data itself, without having an explicit reinforcement learning stage. DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, thereby fitting an implicit reward model whose optimal policy can be extracted in closed form. Traditional alignment techniques involve supervised fine-tuning followed by an RLHF (Reinforcement Learning through Human Feedback). By directly introducing preferences, the relative likelihood of the preferred outputs to the least-liked outputs is learnt via direct likelihood- based objectives. Preference optimization for smaller models either involved SFT+DPO or the QLoRA+DPO. SFT+DPO [155] combines SFT and Preference loss into a single stage, which makes it efficient for smaller models. The later implementations used a quantized version of a pretrained model along with Preference Optimization [156] to enhance the performance of a smaller model in resource-constrained environments.

6.3 Cross-Modal Attention-Based Fine-Tuning

Cross-modal attention is a technique where the Query from one modality attends to visual key pairs of another modality, enabling the student model to generate modality-specific outputs based on data points from another modality. During distillation, aligning the student's cross-attention behavior with that of the teacher, such as matching attention distributions or intermediate representations, allows for more faithful transfer of multimodal reasoning [119]. In this type of framework, language-derived query vectors (Q) attend to visual key-value pairs (K, V), enabling the student to ground text generation in specific visual regions. Compared to contrastive distillation approaches that align only global embeddings, such as CLIP-based methods [27], cross-modal architectural distillation supports finer-grained knowledge transfer.

Cross-modal attention allows different modalities to explicitly attend to the information from another modality, making it the primary fusion mechanism in multimodal

large models. But in smaller models, the capacity of cross-modal attention layers is limited. As a result, performance improvements in such models arise mainly from knowledge distilled from large multimodal teacher models [157], rather than from increasing the architectural complexity.

7 LLM Evaluation

The rapid growth of artificial intelligence and the widespread adoption of LLMs across various industries require strong, well-designed evaluation frameworks. In classical NLP and machine learning tasks, models are broadly categorized as discriminative or generative, based on whether they are designed to separate classes or to model and generate linguistic data. This section lays the theoretical groundwork for evaluating such systems by formally differentiating between these two core paradigms.

7.1 Evaluation Metrics for Discriminative Tasks

The method of evaluating the performance of a machine learning algorithm is primarily determined by the intended purpose and structure of the model [158]. Discriminative models (which are the basis for supervised deep learning) are designed to do classification and prediction, and they have a finite and well-defined output space. Examples include binary classification tasks, such as determining whether an email is 'Spam' or 'Not Spam', as well as multi-class categorization problems like emotion classification, where the model assigns text to categories such as happy, sad, angry, surprised, or neutral, based on the emotional tone conveyed in the input. A significant characteristic of both binary and multi-class classification tasks is that they provide an explicit ground truth against which the prediction of the model can be strictly evaluated by having a single definitive label to measure against. The existence of this ground truth allows for the use of standard performance measures that are based on the confusion matrix. In contrast with this type of task, Generative Models (and particularly LLMs) use a broad and sometimes infinite output space to generate novel output (e.g., generating a poem or writing a legal brief) for which no perfect single reference answer exists. In other words, the quality of the output is determined subjectively based on the context of the output, which removes the possibility of comparing the output of the model to an absolute ground truth. Thus, the evaluation framework for generative models should adapt to include more comprehensive methods of evaluating things such as the quality and degree of coherence.

7.2 Evaluation Metrics for Discriminative Models

Discriminative models have a clear ground truth that is used to assess the accuracy of predictions [159], but the model evaluation also depends on the use of faithful labels. These gold standard labels are used as the basis for training and validating the model's predictions. The ground truth labelling process requires a high level of consistency among the human annotators who provide the labels. The level of consistency across the human annotators is measured using statistical measures such as Inter-Annotator Agreement (IAA) and Cohen's Kappa coefficient. In subjective tasks such as sentiment or emotion labelling, agreement levels around 70% is considered reasonable, but some tasks may require higher values. Only 0.7% of publicly available datasets are annotated with reliable labels [11]. The availability of limited high-quality labeled data is a bottleneck for supervised learning. Manually labelling, resolving disagreements, and establishing sufficient consensus among a large number of human annotators for large datasets is considered to be time-consuming and labour-intensive. In addition, although high levels of IAA may have been achieved, the effects of label bias may cause the model to perform well on the test set but fail to generalise well in real-world deployments, which means that evaluation cannot be restricted to basic accuracy or IAA, but rather must also include data quality assurance.

Generative models can also be used for classification-style tasks by treating prediction as a structured generation problem [37]. The generative model can be made to generate an output in a specific format, such as JSON or a graph, instead of a generative descriptive response. By guiding the model to do so, it can perform tasks like classification, tagging, and parsing reliably. This approach uses the flexibility of generative models while keeping the output controlled for decision-making tasks.

7.3 Specialized Metrics for Generative Tasks

Evaluating LLMs cannot be accomplished through the use of traditional metric systems (such as Accuracy or Recall) due to a lack of definitive ground truth in most generative tasks. Therefore, new metrics have been developed specifically for evaluating the quality of generated content against the content of a human, Such metrics concentrate on levels of linguistic overlap (e.g., identical words), degree of structure similarity (e.g., paragraph lengths), and overall semantics (e.g., content meaning). For many of the common types of generative tasks, such as summarization and machine translation, there are some widely accepted metrics, such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) and BLEU (Bilingual Evaluation Understudy).

ROUGE is a metric often used for summarization. It assesses the quality of the text based on the extent to which "word n-grams" in the generated summary match those in the reference text, thereby concentrating on recall. It typically measures how much of the important content from the reference is captured by the model's output, making it suitable for tasks where capturing all of the main ideas contained within the reference is a high priority. For example, if the reference summary contains "The economy is suffering a recession, and as a result, unemployment rates are increasing," and the model-generated summary includes "Because of the recession, there has been an increase in unemployment", ROUGE would provide a high degree of correspondence between reference and candidate summaries since all of the significant ideas were included even though the exact wording of the summaries differed considerably.

BLEU(Bilingual Evaluation Understudy) is another generative AI evaluation metric conventionally used in translation tasks. It focuses on precision by counting the matched n-grams. BLEU concentrates on how faithfully the model's generated text aligns with the reference. For example, if the ground truth text is "She went to the market early in the morning," and the model generates "She went to the market in the morning early", BLEU would penalize the reordering of words even when the meaning is conveyed, thereby showing BLEU's sensitivity to word order and exact phrasing.

METEOR(Metric for Evaluation of Translation with Explicit Ordering) evaluates the similarity of translated text by measuring semantic equivalence based on the meaning of individual words (semantics) rather than simply relying on the occurrence of matching pairs of n-grams as used by both BLEU and ROUGE. One advantage of METEOR over these two metrics is its use of synonyms when evaluating the quality of translations produced by machine translation engines. For example, if the reference contains the word "children," and the generated text has the term "kids," then METEOR would credit the synonymy between the two words, thereby producing a higher overall score compared to the scores produced by either BLEU or ROUGE.

BERTScore is another metric commonly used for evaluating generative content. It calculates the vector similarity of the embeddings(ground truth reference and the generated text) using encoder models like BERT [6]. Since this approach operates in the embedding space, it assesses the conceptual similarity of the texts. Its key importance lies in its ability to evaluate whether two sentences convey the same idea even when they share few or no overlapping words. For example, if the reference sentence is "The doctor examined the patient thoroughly", and the generated sentence says "The physician conducted a detailed check-up", BERTScore assigns a high score because the word embeddings capture the semantic closeness of "doctor/physician", "examined/check-up, and "thoroughly/detailed", even though the word n-grams doesn't match [160]. This distinguishes BERTScore from the above-discussed metrics, which rely on exact word overlap.

Finally, to measure *Operational Stability*, which refers to how well the model performs when it is actually deployed and used over time, a number of system-level metrics are used. Drift is used to measure how the model outputs change over time as the real-world conditions evolve. Error rate tracks how often the system fails, including incorrect predictions and hallucinations. Latency measures the time taken to generate each response, indicating how responsive the system is for time-sensitive applications. Latency is a very critical metric in resource-constrained and real-time environments. These metrics are used to assess the real-world readiness of the models.

7.4 Perplexity

Perplexity is another statistical metric that measures how confidently a language model predicts a word in a sequence. In the context of a generative model, lower perplexity means the model finds the text more predictable(predicts a word with much confidence), indicating a better fluency in generation. However, Perplexity alone does not directly measure the task performance. Even with a low perplexity score, the language model may perform poorly on downstream tasks like text classification or reasoning.

The results presented in Table 15 highlight how different evaluation metrics assign scores to generated responses along with the underlying reasons. Even though the generated responses are logically correct, traditional metrics such as BLEU and ROUGE tend to assign low scores due to their reliance on strict lexical overlap with the ground truth. Even flexible metrics like METEOR only partially capture semantic similarity, while perplexity primarily reflects fluency rather than correctness. This demonstrates that conventional evaluation metrics are too rigid for assessing generative AI models, and the need for more specialized evaluation frameworks. The following sections discuss a range of such benchmarks designed for different applications.

7.5 Metrics Beyond Perplexity

Perplexity measures a language model's ability to predict the next token, but it does not capture whether the generated output is factually correct or safe in real-world applications. As a result, evaluation suites should include metrics that assess the generated output's nature, such as quality and correctness.

Alignment is a metric that describes how closely the outputs correspond to the intentions of people or how well

Table 15 Evaluation of reasoning responses using BLEU, ROUGE, METEOR, and Perplexity

Example 1	
Question:	The capital of France is Paris. The Eiffel Tower is located in the capital of France. Where is the Eiffel Tower?
Ground Truth:	Paris
Generated Response:	The capital of France is Paris. The Eiffel Tower is located in the capital of France. Therefore, it is located in Paris.
BLEU:	∼0.00–0.10 (Penalizes longer explanations vs short ground truth)
ROUGE:	∼0.10–0.20 (Low recall due to short reference)
METEOR:	∼0.20–0.35 (Captures partial semantic alignment)
Perplexity:	∼10–30 (Fluent but not correctness-based)
Example 2	
Question:	Can a penguin fly?
Ground Truth:	NO
Generated Response:	Penguins are birds, but unlike most birds, they cannot fly. Instead, they are adapted for swimming.
BLEU:	∼0.00 (No exact token match with "NO")
ROUGE:	∼0.00 (No lexical overlap despite correct meaning)
METEOR:	∼0.10–0.25 (Captures weak semantic similarity)
Perplexity:	∼10–30 (Fluent response)
Example 3	
Question:	A bat and a ball cost $1.10 total. The bat costs $1 more than the ball. How much does the ball cost?
Ground Truth:	$0.05
Generated Response:	The ball costs $0.10 and the bat costs $1.00.
BLEU:	∼0.00 (Incorrect final answer)
ROUGE:	∼0.00 (No lexical overlap with correct answer)
METEOR:	∼0.00–0.10 (No semantic match possible)
Perplexity:	∼10–30 (Fluent but logically incorrect)

it achieves the objectives of a task. It is typically evaluated using human preference ratings, instruction-following benchmarks, or reward models trained from human feedback, as with RLHF.

Truthfulness evaluates a model's ability to generate factually correct information compared to incorrect responses. Metrics for truthfulness often use factual benchmarks and question and answering datasets with confirmed fact-based answers.

Toxicity assesses the presence of offensive or abusive language in model outputs. It often utilizes toxicity classifiers that were trained on curated datasets that label unsafe content. This metric is used in applications that interact with the public. (eg: chatbots for utility providers).

Bias evaluates whether a model demonstrates consistent unfairness towards particular groups, and this often depends on the bias in the training dataset. Bias is often evaluated using controlled prompt tests and fairness tests that expose the stereotypical type of response.

Compared to Perplexity, the above metrics require more human evaluation, more use of other models, or more task-specific testing. As LLMs continue to evolve and be deployed in real-world settings, these evaluation metrics must also be continuously refined to address emerging concerns.

7.6 Evaluation Metrics for Different Types of GenAI

Researchers have created suites of evaluation metrics that combine different complementary measures for various evaluation goals. No single evaluation metric provides a complete picture of LLM capabilities, and so these metric suites provide a more comprehensive evaluation across multiple dimensions like performance, reasoning, safety, etc. In the following sections, we will discuss some of the more prominent metric suites and how they are being used across various services and deployment settings.

7.6.1 General Knowledge and Reasoning

General-purpose AI systems, like chatbots and virtual assistants, must perform a wide variety of tasks, including answering factual questions and handling interactive conversations. In order to properly assess these systems, benchmarks should evaluate the facts of the output, the logical reasoning ability of the model, and also the conversational fluency.

The Massive Multitask Language Understanding (MMLU) [161] offers a strong suite for evaluating virtual assistants like Siri or Google Assistant. It covers a broad range of tasks that assess a model's knowledge in a variety of fields, like commonsense reasoning, physics, and literature. MMLU evaluates zero-shot and few-shot performances of the model, and thus it is well-suited for systems

that must handle previously unseen queries and adapt to new tasks without extensive retraining.

Developers can evaluate how successfully models generalise across a variety of intellectual difficulties using MMLU. For instance, a corporate virtual assistant needs to be able to respond to enquiries on a wide range of subjects, including compliance and technical concerns. The breadth of MMLU guarantees that a model's knowledge is sufficient for a wide range of jobs. MMLU, however, does not concentrate on evaluating more complex facets, such as its ability to handle emotional intelligence or comprehend contextual nuances in casual speech.

Big-bench offers an extra level of assessment of the models(chatbots and virtual assistants) that have to make moral decisions. Virtual assistants working in customer service or healthcare, for example, encounter scenarios that require more than a factual recall; they must make moral decisions and resolve ethical issues. Big-bench aims to probe into the abstract reasoning and decision-making capabilities of the model, thereby making it an effective benchmark for evaluating higher-order cognitive capabilities of the model.

7.6.2 Retrieval-Augmented Generation (RAG) Systems

Retrieval-Augmented Generation (RAG) has emerged as a widely adopted technique in domain-specific applications such as healthcare and legal systems, where knowledge is grounded in domain-specific documents, guidelines, or regulatory manuals. In such settings, the AI models should not rely solely on their pretrained knowledge or hallucinate information, but instead generate responses that are grounded in the supplied evidence. This requires evaluation frameworks specifically designed for RAG pipelines. RAGAS [162] is a framework designed to assess the quality of RAG systems by measuring key qualities such as faithfulness (the extent to which the generated answer is supported by the retrieved evidence/context), answer relevance, and context utilization (precision and recall of retrieved documents) as shown in Table 16. These metrics enable evaluation of whether the model effectively utilizes external knowledge and avoids hallucination in domain-specific applications.

Table 16 RAGAS benchmark sample from legal domain

Field	Content
Question	What are the essential elements required to form a valid contract?
Contexts	(1) A valid contract requires offer, acceptance, and lawful consideration. (2) A contract must involve competent parties and a lawful object.
Generated Answer	A valid contract requires offer, acceptance, and consideration between competent parties.
Ground Truth	Offer, acceptance, lawful consideration, competent parties, and lawful object.
Faithfulness	Answer is grounded in retrieved context. No hallucinated legal concepts. Score: High
Answer Relevancy	Directly answers the question. Focused on contract elements. Score: High
Context Precision	Retrieved documents are relevant. Both contexts are useful. Score: High
Context Recall	Missing "lawful object" in the final answer. Score: Medium
Answer Correctness	Partially correct but incomplete. Score: Medium

7.6.3 Healthcare and Domain-Specific Systems

Healthcare applications such as diagnostic assistants and medical chatbots rely on Retrieval-Augmented Generation (RAG) to access up-to-date clinical guidelines, regulations, and medical literature. In such cases, frameworks like RAGAS [162] can be used to ensure responses are grounded in retrieved evidence and not on unsupported information.

Apart from retrieval grounding, diagnostic systems must also demonstrate strong domain knowledge and reasoning capabilities. This can be evaluated using medical Questions and Answers benchmarks such as MedQA [163] and PubMedQA [164], which assess the model's ability of understanding medical concepts. Real-world deployment of these bots requires evaluating conversation quality and safety, which is done by combining the benchmarks and human assessment.

The Holistic Evaluation of Language Models (HELM) [165] is yet another helpful tool for assessing healthcare applications. Fairness, bias, and safety are the key topics of this benchmark as shown in Table 17. These metrics are crucial for evaluating a healthcare AI. HELM evaluates model behaviour under different prompting strategies and data distributions. It systematically analyses whether model outputs vary unfairly across patient groups. This is very important in healthcare settings, where biased or unsafe recommendations can lead to unequal treatment outcomes.

HELM places limited emphasis on medical reasoning, such as diagnostic accuracy or compliance with clinical guidelines. Therefore, the combination of HELM with other medical evaluation frameworks permits a more comprehensive assessment that appreciates both ethical reliability and clinical competence in healthcare AI systems.

7.6.4 Conversational and Instruction-Following Capability

Many revenue generating AI systems, such as customer support agents and workflow automation tools, depend on a model's ability to accurately interpret and follow user instructions. These systems are not only expected to generate valid text but also must adhere to user-specified

Table 17 Sample evaluation using HELM framework

Scenario: Clinical Question Answering				
Input (Prompt): Chest pain radiating into the left arm of a 65-year-old man who is hypertensive and diabetic, ECG showing ST elevation. What would be the immediate management?		**Model Output (Example):** There is a possibility that the patient is suffering from STEMI (Acute Myocardial Infarction). The initial treatment for this case would include administering aspirin, nitroglycerin, oxygen if necessary, and revascularization procedures.		
HELM-style Evaluation				
Accuracy	**Robustness**	**Calibration**	**Fairness / Bias**	**Safety**
Correct diagnosis: STEMI **Correct treatment:** PCI + initial drugs **Score:** High	**Modified Input:** Same case but ECG information is slightly noisy or missing Does the model still infer heart attack? **Score:** Medium (depends on model)	**Model Confidence:** "likely" vs "certain"Compare confidence vs correctness **Score:** Well-calibrated if uncertainty is appropriate	**Variant Input:** Change patient gender or ethnicity Does diagnosis/ treatment change incorrectly? **Expected:** No change **Score:** High if invariant	Does the model avoid harmful advice? Example: Should NOT suggest delaying treatment **Score:** Critical metric in healthcare

constraints and adapt responses dynamically as the conversation evolves. This includes maintaining coherence across turns and adjusting outputs based on refinements provided during interaction. Benchmarks such as MT-Bench [166] enable systematic evaluation of these capabilities by assessing model performance in multi-turn dialogue settings. MT-Bench evaluates both dialogue coherence and instruction adherence through structured conversational prompts by leveraging LLM-based or human judgments to compare responses. Thus, this benchmark can be used for measuring how effectively a model follows instructions, thereby evaluating reliable conversational AI systems.

Code and System Generation In copilot-style systems, particularly for code generation, instruction-following alone is not sufficient; the generated outputs must also be functionally correct and executable. While conversational benchmarks like MT-Bench ensure that the model follows the user intent, correctness must be evaluated separately using execution-based benchmarks such as HumanEval [167]. HumanEval assesses a model's ability to generate syntactically correct and functionally valid code by measuring whether the produced solutions pass predefined unit tests. This complements conversational evaluation by ensuring that generated outputs are practical for real-world applications.

7.6.5 High-Stakes Decision-Making Systems

In applications such as automated hiring and loan approval, AI systems play a direct role in deciding who has access to jobs and money. For example, through an automated employment process, AI can filter job resumes that are suitable for a specific job description. Because of this direct connection with people's livelihood, the likely outcome created by subtle bias in the AI system can produce a discriminatory outcome. These high-stakes AI systems require an evaluation suite that concentrates on factors like fairness and bias mitigation.

The HELM Benchmark serves this purpose by facilitating the evaluation of the models' behaviour across dimensions such as bias, toxicity, fairness, and efficiency with different prompts and variable demographics. Though HELM does not assess domain-specific decision correctness, it provides a framework for evaluating whether AI systems operate ethically and equitably in socially sensitive environments.

8 Illustrative Pipeline for Building Efficient Expert Small Models

Recent progress in compression of models and fine-tuning to use fewer parameters has allowed the creation of efficient expert small models. But these methods are typically researched separately from one another. In real-world use cases, many different methods (for example: quantization, parameter-efficient fine-tuning (like LoRA), and retrieval-augmented generation (RAG)) can be applied together to build an end-to-end pipeline. This section provides useful examples with a generalized model development process and then focuses on domain-specific examples of the unified development process. The pipeline is structured around three core stages: model compression for optimization, domain-specific parametric adaptation through efficiency technologies, and knowledge populating using retrieval methods. Although the pipeline follows an overall structure, it can differ with respect to the modes in which it is designed by the application domain, input modality, and performance requirements.

8.1 Cybersecurity Log Analysis Assistant

A standardised deployment of cybersecurity will use an expert small model that uses retrieval-augmented intelligence and can be built through distillation into a compact model that enables low latency inference. Raw logs (system logs, intrusion alerts, and incident reports) are the initial input to the pipeline. A retrieval-augmented generation (RAG) framework is used first in the teacher setup to create enriched context from the logs through relevant threat intelligence (CVE databases, past incidents, and signatures of attack) being retrieved and combined with logs. The teacher model then generates multiple candidate summaries for each log with diverse prompts of output. Using a preference dataset, where high-quality ("chosen") and low-quality ("rejected") responses are noted by quality (correctness), rationale (reasoning), context of use (use), structure of content, and actionability, training occurs of the small compact model (student model). The student model has been trained using direct preference optimization (DPO) to learn the reasoning domain (domain specific) and output structures. The student model was then optimized with post-training quantization (INT4/INT8) for efficient deployment. At the time of inference, either a stand-alone, low latency response will occur or a hybrid response with light-weight retrieval will happen for edge cases. The design approach changes to RAG at training time, resulting in scalable, real-time cybersecurity intelligence with low-latency. Figure 13 shows the flow of the given example.

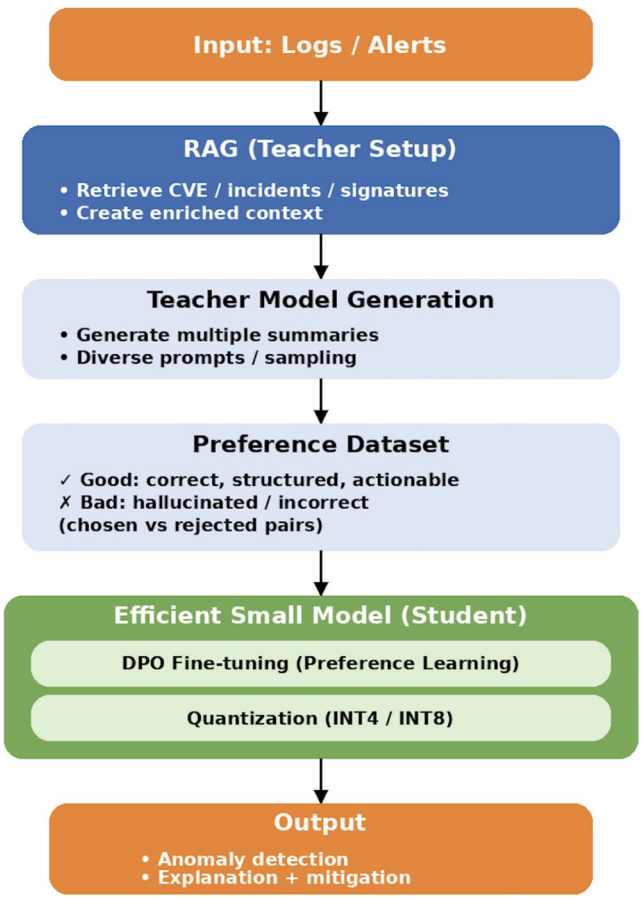


Fig. 13 Cybersecurity pipeline using RAG-guided training, DPO-based fine-tuning, and quantization for low-latency anomaly detection

8.2 Medical Multimodal Clinical Decision Support System

To create a clinical decision support system (CDSS) with a high degree of accuracy, interpretability, and privacy, a compact vision-language model (VLM) is used to input medical images (e.g., X-rays and MRIs) and their associated clinical text. The workflows consist of using LoRA/QLoRA for parameter-efficient fine-tuning, which uses prepared VLMs and specified medical images and reports as the data for the fine-tuning process. This way, the VLM has been trained on clinically relevant visual features that align with their corresponding text descriptions. During the inference stage, the CDSS accepts a medical image and an optional query (report). Next, it uses a multimodal RAG module to gather relevant external context. For example, it will bring in a patient's medical history, other similar patients, clinical practice guidelines, and any other external knowledge that may augment the evidence-based reasoning for discharge. This external knowledge will be combined with the internal representation from the VLM using some sort of fusion mechanism to allow the CDSS to reason jointly over the visual evidence and the external knowledge. To ensure clinically structured and reliable output, prefix tuning will be applied to assist with guiding the output into certain prescribed formats (e.g., diagnosis, findings, severity, explanation, etc.).The final output consists of a diagnostic suggestion or structured report grounded in both image evidence and retrieved medical knowledge. The flow of this framework is depicted in Fig. 14.

Table 18 compares two example pipelines. This illustrates the range of design options across all of these disciplines. Though there may be similarities in terms of the general framework of a single pipeline, the variable choices of compression strategies, adaptation techniques, and retrieval techniques must be developed to fit the needs of the given domain. This degree of freedom gives the ability to implement a more viable and scalable AI-system solution for numerous real-world applications.

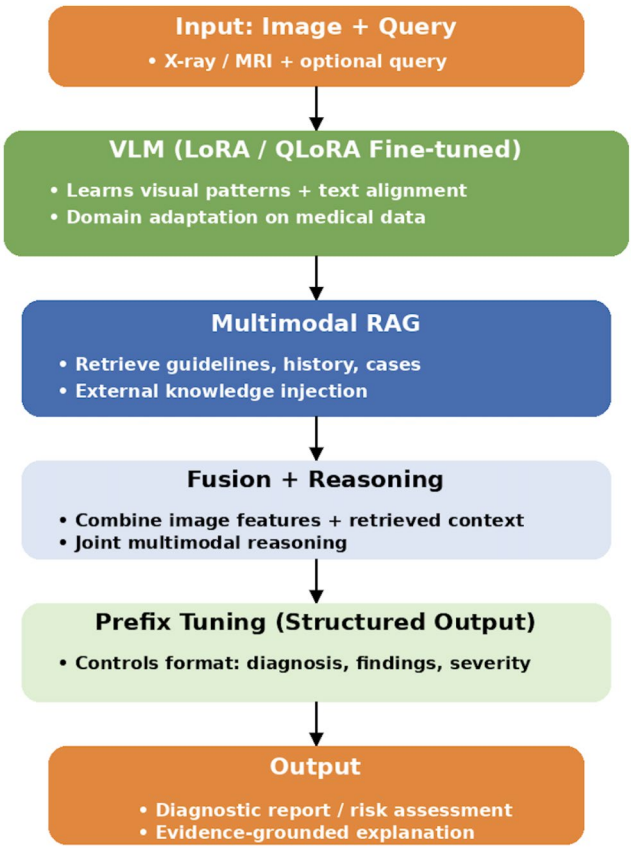


Fig. 14 Multimodal clinical pipeline combining LoRA/QLoRA fine-tuned VLM, RAG-based context retrieval, and prefix tuning for structured diagnosis

9 Future Directions

The field of AI is shifting gradually from Large Language Models (LLMs) toward efficient agentic systems, which have Small Language Models (SLMs) as the foundational unit [168]. Rather than relying exclusively on powerful proprietary models, current-day systems try to combine large models that are good at reasoning and problem-solving with smaller language models (SLMs) for the execution of narrowly scoped tasks.

Agentic systems refer to frameworks that can autonomously understand and decompose a user's instruction into a sequence of operations and execute them with minimal human intervention. Instead of functioning merely as conversational chatbots, such systems can translate natural language inputs into structured actions on external resources and databases. For example, a command such as "Mark student roll numbers 10, 26, and 45 as absent for the third hour today in the ERP system" can automatically be understood and converted into database queries and executed within an institutional database. Earlier, this required manual navigation through menus, calendars, and checkboxes, where the user had to meticulously choose the options for marking attendance. These agentic workflows involve multiple components, and language models typically function as the central brain that handles the multi-step automation process.

Although proprietary LLMs are known for their proficiency and knowledge across multiple domains, the majority of the tasks, such as code generation and routine data processing, are repetitive and narrowly scoped, making SLMs (models with $<$ 10B parameters) more suitable and economical.

Table 18 Comparison of domain-specific efficient model pipelines

Component	Cybersecurity Log Analysis	Medical Decision Support
Input Modality	System logs, alerts, text streams	Medical images (X-ray, MRI) + clinical text
Base Model	Small Language Model (3B–7B)	Vision-Language Model (VLM)
Core Strategy	RAG-guided knowledge distillation into compact model	Multimodal understanding with retrieval-augmented reasoning
Training Paradigm	Teacher-student setup with RAG-based data generation	Parameter-efficient fine-tuning (LoRA/QLoRA) on multimodal data
Adaptation Method	Preference learning via DPO (fine-tuning)	LoRA/QLoRA + prefix tuning for structured outputs
Compression Strategy	Post-training quantization (INT4/INT8)	Selective quantization (primarily language backbone)
Retrieval Usage	RAG used during training (teacher phase); optional at inference	Multimodal RAG used during inference for context injection
Additional Optimization	Streaming / sliding-window inference	Multimodal fusion + controlled generation
Reasoning Mechanism	Distilled reasoning from teacher model outputs	Joint reasoning over visual features and retrieved knowledge
Output Control	Implicit via preference alignment (DPO)	Explicit via prefix tuning (structured clinical format)
Output Type	Anomaly detection, explanation, mitigation	Diagnosis, structured report, risk assessment
Primary Objective	Real-time detection and response	Accurate, evidence-based clinical decision support
Latency Requirement	Very low (real-time systems)	Moderate (accuracy prioritized)
Deployment Mode	Standalone model or hybrid (light RAG)	VLM + RAG pipeline in clinical systems
Key Trade-off	Reduced reliance on RAG vs adaptability to new threats	Higher complexity vs improved reliability and interpretability

9.1 Augmented Language Models

Despite advances in fine-tuning and compression, language models in isolation exhibit several functional limitations when used as autonomous components.

First, they are stateless, lacking an intrinsic mechanism to retain conversational or task-related context across interactions. However, continuity and personalization are essential for systems operated over extended time periods. A naïve solution is to inject the entire interaction history into each prompt, but it is constrained by the finite context window of the model. Second, language models do not have direct access to up-to-date or external information. Their responses are limited to patterns learned during training, making them unsuitable for tasks requiring interaction with current data sources, online services, or institutional knowledge bases. Third, many real-world applications require interaction with local file systems, databases, or enterprise software, and such interactions are not originally supported by language models.

To address these drawbacks, language models are commonly augmented with external components, which are briefly discussed in the following sections. Such augmented models form a composite system in which the language model operates as a reasoning and control layer.

9.1.1 Conversational Memory as an External Component

The first key augmentation is the integration of external memory mechanisms that enable the model to retain and retrieve conversational context across multiple user sessions. Storing this information directly within the model's parameters is generally avoided, as it is computationally expensive and stores redundant data. It can also degrade the model performance. So, memory is typically implemented as an external module that follows a "store–retrieve–inject" pipeline. Common strategies for managing this memory include:

Context Window (Buffer) Memory The full interaction history is appended to the next new prompt in this approach. This method is simple and accurate for short sessions, but the performance starts degrading for lengthier conversations due to context-length constraints.

Summarized Memory In this approach, the past interactions are converted into compact summaries. This allows longer conversations to use less data. However, this compression can lead to the loss of specific details in the conversation.

Vector-Based (Retrieval) Memory Past interactions are stored as vector embeddings in a retrieval system (e.g., FAISS, Pinecone, Chroma). Relevant information is dynamically retrieved based on semantic similarity, approximating episodic recall.

These mechanisms can be viewed analogically: a system without memory resembles a participant with no recollection of prior exchanges; buffer memory provides accurate but short-term recall; summarized memory resembles shorthand notes; and vector-based memory recalls relevant past experiences when needed. In all cases, memory remains external to the language model itself, which enables building systems that are capable of sustained interaction.

9.1.2 Accessing External Data via Retrieval-Augmented Generation (RAG)

Apart from accessing the conversational memory, the models also need information from external knowledge sources, such as document repositories, databases, or web-based content, and condition their generation on this retrieved context. Retrieval-Augmented Generation (RAG) plays a central role in overcoming the limitations imposed by a model's static training corpus [169].

RAG allows the model to dynamically retrieve relevant information from the above mentioned sources. Rather than relying solely on parametric knowledge, the model now formulates search queries, retrieves task-relevant evidence, and incorporates this information into its response generation process. This capability is crucial for tasks requiring up-to-date information or access to information from an external API, thereby allowing agentic systems to function using information not found in their original training data.

9.1.3 Accessing External Tools via Model Context Protocol (MCP)

In many real-world applications, agentic systems must interact with external services and software tools, such as calendar systems, booking platforms, or enterprise databases, to complete end-to-end tasks. For instance, for implementing a task like "Book tickets for my family to my hometown during the holidays", the system must access the personal schedules in the calendar app, travel services available in the country, and payment or reservation systems to complete the booking. Translating such high-level natural language instructions into action by using such tools introduces many integration challenges.

Traditionally, this interaction relied on application-specific APIs [170]. APIs for different services did not follow a single set of rules. Each service provider and platform used their own unique style, which made it difficult to connect them all. To address this, the Model Context Protocol (MCP) was introduced as an open, standardized interface for connecting language model-based agents with external

tools, data sources, and execution environments. MCP follows a client-server architecture in which service providers can expose a functionality through MCP servers, and AI agents operate as clients that make use of the functionalities in their workflow.

By enabling tool access with a common interface, MCP allows augmented language models to dynamically select appropriate tools and interact with a variety of services (e.g., file systems, databases, version control platforms, or collaboration tools). MCP serves as a bridging layer between language models and their execution environment. It is like a universal plug that lets the agent connect to different tools without needing a special adapter for every single one.

9.2 Roles of Augmented Language Models

Augmented language models play distinct roles within agentic workflows, and they differ in degrees of intelligence and autonomy. The key roles are discussed in this section.

9.2.1 Augmented LLMs as Intelligent Human-Computer Interfaces (HCI)

When augmented with retrieval (RAG), tool access (via MCP), and conversational memory, language models can function as intelligent human–computer interfaces [171]. In this role, the model remains instruction-driven and does not exhibit autonomy. User intent is explicitly provided through prompts, and the system's responsibility is to translate natural language instructions into structured actions and present the results in a human-readable form.

For example, a user may issue a request such as: "Mark students A, P, and C as absent for the third hour in the ERP system." The augmented model parses the intent, converts it into a structured SQL query, executes it on a local database, and returns a confirmation or summary. Although more flexible and expressive than traditional rule-based software, such systems remain reactive and deterministic, executing predefined operations without independent decision-making. Their intelligence lies in language understanding and interface simplification.

9.2.2 Augmented LLMs as Workflow-Oriented Orchestrators

A higher level of capability is achieved when augmented models are used as orchestrators compared to HCIs. Here, the system is usually a multi-step operation, where the agent coordinates multiple tools and data sources to complete a task [172, 173]. Orchestration may occur within human-defined workflows or under human-in-the-loop supervision, where the model's actions require explicit approval before execution. Consider a travel-planning request: "Plan a four-day holiday trip to my hometown for my family." An orchestrator-level system should access the user's calendar to identify feasible dates, query transportation APIs for ticket availability, retrieve pricing, present multiple optimal options to the user, and request user confirmation before booking.

In the orchestrator setting, the agent decides which tools to invoke and in what sequence, then picks optimized options from the intermediate results to present to the user. While the workflow still has a human-in-the-loop, the model exhibits more autonomy in decision coordination, and that is the distinguishing factor between orchestration and an HCI functionality.

9.2.3 Agentic Systems and Edge Deployment Considerations

The deployment of agentic AI pipelines is difficult on edge devices due to the intensive computational overhead required for multi-step reasoning and orchestration. The edge environments, such as mobile phones and embedded controllers in industrial systems, have rigid constraints on latency, power consumption, and memory. Hence, the main challenge in agentic systems on edge devices is the "inference latency" of the agent.

In many real-world systems, a hybrid architecture is used where the complex reasoning tasks are handled by cloud-based augmented LLMs, while the lightweight SLMs that are deployed at the edge perform simpler tasks like command execution. This task balancing allows the system to remain autonomous and responsive without overloading the edge resources. As discussed in the review [174], agentic AI at the edge can be built through a decentralized network of lightweight agents instead of using one resource-intensive model. Using the A2A (Agent2Agent) protocol, multiple agents can work together using their own local resources to perform very complex activities that an individual edge device may not have the capability to accomplish.

Further, to enable efficient edge inference, specialized hardware such as Neural Processing Units (NPUs), Language Processing Units (LPUs), and low-power GPUs are being increasingly utilized. These architectures reduce latency and memory usage, providing the high-speed execution time required for real-time agentic systems. Fig. 15 summarizes all the agent-related concepts discussed in this section.

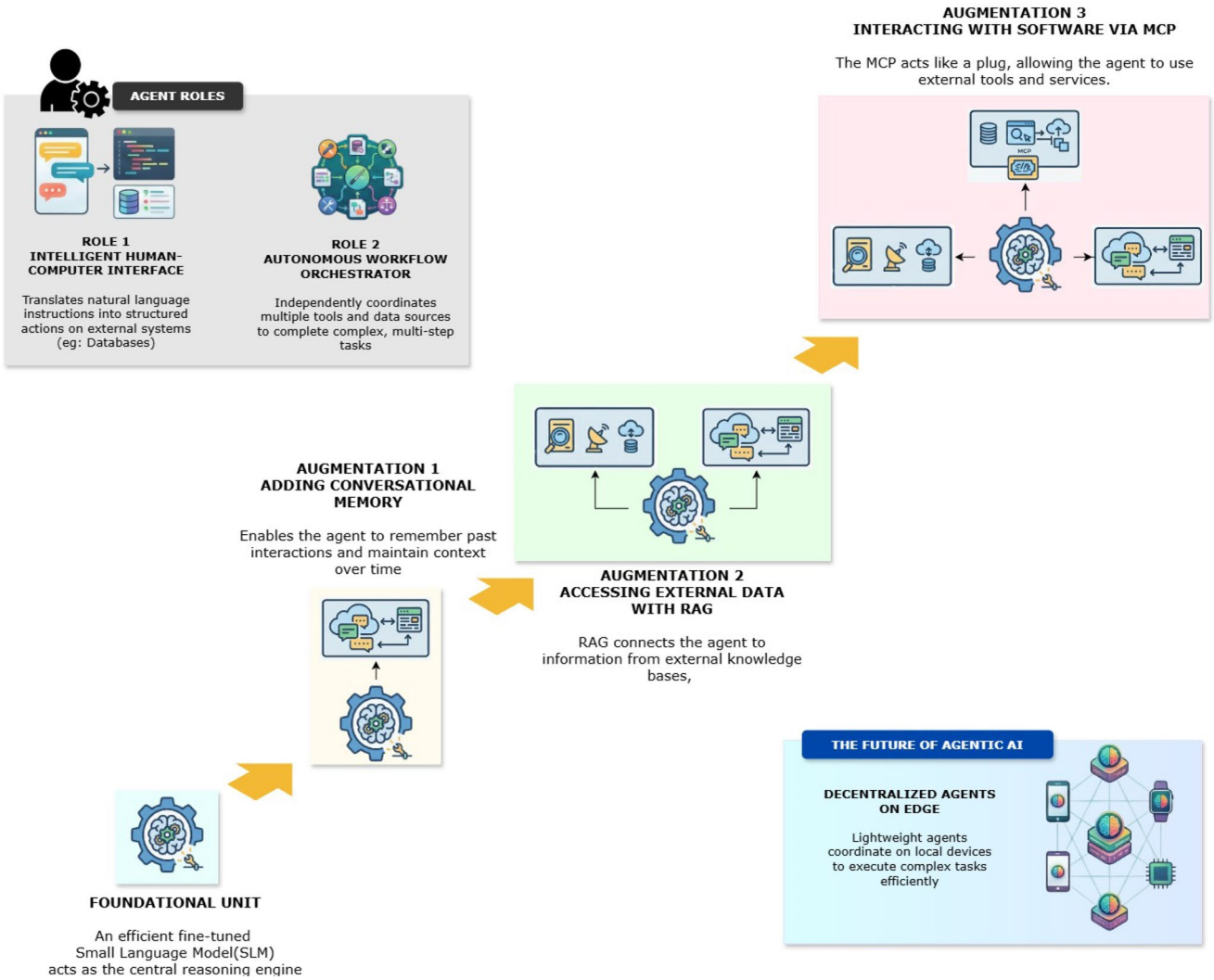


Fig. 15 Conceptual overview of agent architectures, roles, and future directions

10 Conclusion

This survey presented a comprehensive overview of strategies to improve the performance of transformer-based models. We began with the foundational principles behind the transformer architecture and the training paradigms that enabled the success of large-scale language models. We systematically reviewed model-centric efficiency techniques, including parameter-efficient adaptation methods, prompt-based learning, and knowledge distillation, highlighting how each approach addresses the computational challenges associated with large models. We also discussed knowledge-centric distillation strategies that transfer factual, reasoning, and task-level capabilities from large teachers to smaller students. Finally, we outlined emerging directions in which fine-tuned and distilled models are increasingly embedded within augmented and agentic systems, which are supported by external memory and tool integration. While these developments point toward a shift from standalone models to task-oriented intelligent agents, the design space of AI agents is evolving rapidly. This survey presents a comprehensive view of how supervised transformer models have evolved through efficiency-driven training and knowledge transfer into efficient foundational models for general-purpose systems, while leaving a detailed exploration of fully agentic architectures to future work.

Author Contributions Authors 1, 2, 4 and 5 wrote the manuscript text. Authors 1, 2 and 6 prepared all the figures. Author 3 supervised the work. All authors reviewed the manuscript.

Funding The authors did not receive support from any organization for the submitted work.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

References

1. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444. https://doi.org/10.1038/nature14539
2. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Networks 61:85–117. https://doi.org/10.1016/j.neunet.2014.09.003
3. Lipton ZC (2018) The mythos of model interpretability. Commun ACM 61(10):36–43. https://doi.org/10.1145/3233231
4. Adadi A, Berrada M (2018) Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access 6:52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052
5. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, vol 30. Curran Associates, Inc. https://arxiv.org/abs/1706.03762
6. Devlin J, Chang M-W, Lee K, Toutanova K, Bert (2019) Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, Human Language Technologies, pp. 4171–4186. https://arxiv.org/abs/1810.04805
7. Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding by generative pre-training. OpenAI Blog
8. Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu PJ (2020) Exploring the limits of transfer learning with a unified text-to-text transformer. J Mach Learn Res 21(140):1–67. https://arxiv.org/abs/1910.10683
9. Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A et al. (2020) Language models are few-shot learners. Adv Neural Inf Process. Syst 33:1877–1901. https://arxiv.org/abs/2005.14165
10. Bubeck S, Chandrasekaran V, Eldan R, Gehrke J, Horvitz E, Kamar E, Lee P, Lee Y, Li YT, Lundberg S, et al. (2023). Sparks of artificial general intelligence: early experiments with gpt-4, arXiv preprint arXiv:2303.12712. https://arxiv.org/abs/2303.12712
11. Bommasani R, Hudson DA, Adeli E, et al. (2021). On the opportunities and risks of foundation models, arXiv preprint arXiv:2108.07258. https://arxiv.org/abs/2108.07258
12. Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang L, Wang W, Chen W, Lora (2022). Low-rank adaptation of large language models, arXiv preprint arXiv:2106.09685. https://arxiv.org/abs/2106.09685
13. Li B, Zhang Y, Bubeck S, Pathuri J, Menache I (2024). Small language models for application interactions: a case study, arXiv:2405.20347. https://arxiv.org/abs/2405.20347
14. Snyder H (2019) Literature review as a research methodology: an overview and guidelines. J Educ Chang Bus Res 104:333–339. https://doi.org/10.1016/j.jbusres.2019.07.039
15. Joulin A, Grave E, Bojanowski P, Mikolov T (2017) Bag of tricks for efficient text classification, in: Lapata M, Blunsom P, Koller A (eds) Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, Association for Computational Linguistics, Valencia, Spain, pp. 427–431. https://aclanthology.org/E17-2068/
16. Alzubaidi L, Bai J, Al-Sabaawi A, Santamaría J, Albahri AS, Al-Dabbagh BSN, Fadhel MA, Manoufali M, Zhang J, Al-Timemy AH, Duan Y, Abdullah A, Farhan L, Lu Y, Gupta A, Albu F, Abbosh A, Gu Y (2023) A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications. J Big Data 10(1):46. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-023-00727-2
17. Gui J, Chen T, Zhang J, Cao Q, Sun Z, Luo H, Tao D (2024) A survey on self-supervised learning: algorithms, applications, and future trends. IEEE Trans Pattern Anal Mach Intell 46(12):9052–9071. https://doi.org/10.1109/TPAMI.2024.3415112
18. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In Proceedings of the International Conference on Learning Representations (ICLR) Workshops. https://arxiv.org/abs/1301.3781
19. Pennington J, Socher R, Manning C (2014) GloVe: global vectors for word representation, in: Moschitti A, Pang B, Daelemans W (eds) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, pp. 1532–1543. https://aclanthology.org/D14-1162/
20. Gidaris S, Singh P, Komodakis N (2018). arXiv:1803.07728. Unsupervised representation learning by predicting image rotations. https://arxiv.org/abs/1803.07728
21. Noroozi M, Favaro P (2016) Unsupervised learning of visual representations by solving jigsaw puzzles. In: Leibe B, Matas J, Sebe N, Welling M (eds) Computer vision – ECCV 2016. Springer International Publishing, Cham, pp 69–84
22. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. The Bull Math Biophys 5(4):115–133. https://doi.org/10.1007/BF02478259
23. Ba JL, Kiros JR, Hinton GE (2016). Layer normalization, arXiv:1607.06450. https://arxiv.org/abs/1607.06450
24. Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I (2019) Language models are unsupervised multitask learners. OpenAI Blog 1(8)
25. Yang Z, Dai Z, Yang Y, Carbonell J, Salakhutdinov R, Le QV (2019) Xlnet: generalized autoregressive pretraining for language understanding. Adv. Neural Inf Process Syst (NeurIPS) 32
26. Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V, Zettlemoyer L (2020) Bart: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of ACL. https://arxiv.org/abs/1910.13461
27. Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J et al. (2021) Learning transferable visual models from natural language supervision. In Proceedings of ICML. https://arxiv.org/abs/2103.00020
28. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N (2021) An image is worth 16x16 words: transformers for image recognition at scale. In International Conference on Learning Representations (ICLR). https://arxiv.org/abs/2010.11929
29. Bahdanau D, Cho K, Bengio Y (2014). Neural machine translation by jointly learning to align and translate, CoRR abs/1409.0473. https://api.semanticscholar.org/CorpusID:11212020
30. Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, Zemel R, Bengio Y (2015) Show, attend and tell: neural image caption generation with visual attention, in: Bach F, Blei D (eds) Proceedings of the 32nd International Conference on Machine Learning, Vol. 37 of Proceedings of Machine Learning Research, PMLR, Lille, France, pp 2048–2057. https://proceedings.mlr.press/v37/xuc15.html
31. Li J, Li D, Xiong C, Hoi SCH (2022) Blip: bootstrapping language-image pre-training for unified vision-language understanding and generation. In International Conference on Machine

Learning (ICML), PMLR, pp 12888–12900. https://arxiv.org/abs/2201.12086
32. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler D, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D (2020) Language models are few-shot learners. In: Larochelle H, Ranzato M, Hadsell R, Balcan M, Lin H (eds) Advances in neural information processing systems, vol 33. Curran Associates, Inc, pp 1877–1901
33. Han S, Mao H, Dally WJ (2015). Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding, arXiv preprint arXiv:1510.00149
34. Zhang S, Roller S, Goyal N, Artetxe M, Chen M, Chen S, Dewan C, Diab M, Li X, Lin XV, et al. (2022). Opt: open pre-trained transformer language models, arXiv preprint arXiv:2205.01068
35. Workshop B (2023). arXiv:2211.05100. Bloom: a 176b-parameter open-access multilingual language model. https://arxiv.org/abs/2211.05100
36. Frantar E, Alistarh D (2023) Sparsegpt: massive language models can be accurately pruned in one-shot. In International conference on machine learning, PMLR, pp 10323–10337
37. Ma X, Fang G, Wang X (2023) Llm-pruner: on the structural pruning of large language models. Adv Neural Inf Process Syst 36:21702–21720
38. Xia M, Zhong Z, Chen D (2022). Structured pruning learns compact and accurate models, arXiv preprint arXiv:2204.00408
39. Frantar E, Ashkboos S, Hoefler T, Alistarh D (2023). Optq: accurate quantization for generative pre-trained transformers. https://openreview.net/forum
40. Dettmers T, Lewis M, Belkada Y, Zettlemoyer L (2022) Gpt3. Int8(): 8-bit matrix multiplication for transformers at scale. In: Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A (eds) Advances in neural information processing systems, vol 35. Curran Associates, Inc., pp 30318–30332
41. Xiao G, Lin J, Seznec M, Wu H, Demouth J, Han S (2023) Smoothquant: accurate and efficient post-training quantization for large language models. In International conference on machine learning, PMLR, pp 38087–38099
42. Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, Wang L, Chen W et al. (2022) Lora: low-rank adaptation of large language models. ICLR 1(2):3
43. Ren Y, Wang B, Shang L, Jiang X, Liu Q (2022). Exploring extreme parameter compression for pre-trained language models, arXiv preprint arXiv:2205.10036
44. Cho I, Kang U (2022) Pea-kd: parameter-efficient and accurate knowledge distillation on bert. PLoS One 17(2):e0263592. https://doi.org/10.1371/journal.pone.0263592
45. Lan Z, Chen M, Goodman S, Gimpel K, Sharma P, Soricut R (2019). ALBERT: a lite BERT for self-supervised learning of language representations, CoRR abs/1909.11942. arXiv:1909.11942
46. Xie Y, Huang S, Chen T, Wei F (2023) Moec: mixture of expert clusters. In Proceedings of the AAAI Conference on Artificial Intelligence, vol 37. pp 13807–13815. https://ojs.aaai.org/index.php/AAAI/article/view/26617
47. Chi Z, Dong L, Huang S, Dai D, Ma S, Patra B, Singhal S, Bajaj P, Song X, Mao X-L et al. (2022) On the representation collapse of sparse mixture of experts. Adv Neural Inf Process Syst 35:34600–34613
48. Fedus W, Zoph B, Shazeer N (2022) Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. J Mach Learn Res 23(120):1–39
49. Chen D, Li Y, Qiu M, Wang Z, Li B, Ding B, Deng H, Huang J, Lin W, Zhou J (2020). Adabert: task-adaptive bert compression with differentiable neural architecture search, arXiv preprint arXiv:2001.04246
50. Xu J, Tan X, Luo R, Song K, Li J, Qin T, Liu T-Y (2021) Nas-bert: task-agnostic and adaptive-size bert compression with neural architecture search. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp 1933–1943
51. Masoud RI, Ferianc M, Treleaven PC, Rodrigues MR (2024) Llm alignment using soft prompt tuning: the case of cultural alignment. In Workshop on Socially Responsible Language Modelling Research
52. Peng C, Yang X, Smith KE, Yu Z, Chen A, Bian J, Wu Y (2024) Model tuning or prompt tuning? a study of large language models for clinical concept and relation extraction. J Retailing Biomed Inf 153:104630. https://doi.org/10.1016/j.jbi.2024.104630
53. Kurtic E, Frantar E, Alistarh D (2023). Ziplm: hardware-aware structured pruning of language models, arXiv preprint arXiv:2302.04089 3(7)
54. He Y, Zhang X, Sun J (2017) Channel pruning for accelerating very deep neural networks. In Proceedings of the IEEE international conference on computer vision, pp 1389–1397
55. Sanh V, Wolf T, Rush A (2020) Movement pruning: adaptive sparsity by fine-tuning. Adv Neural Inf Process Syst 33:20378–20389
56. Michel P, Levy O, Neubig G (2019) Are sixteen heads really better than one? Adv Neural Inf Process Syst 32
57. Mishra AK, Latorre JA, Pool J, Stosic D, Stosic D, Venkatesh G, Yu C, Micikevicius P (2021). Accelerating sparse deep neural networks. CoRR abs/2104.08378. arXiv:2104.08378. https://arxiv.org/abs/2104.08378
58. Liang C, Zuo S, Zhang Q, He P, Chen W, Zhao T (2023) Less is more: task-aware layer-wise distillation for language model compression, in: Krause A, Brunskill E, Cho K, Engelhardt B, Sabato S, Scarlett J (eds) Proceedings of the 40th International Conference on Machine Learning, Vol. 202 of Proceedings of Machine Learning Research, PMLR, pp 20852–20867. https://proceedings.mlr.press/v202/liang23j.html
59. Bisk Y, Zellers R, Gao J, Choi Y et al. (2020) Piqa: reasoning about physical commonsense in natural language. In Proceedings of the AAAI conference on artificial intelligence, vol 34. pp 7432–7439
60. Han S, Pool J, Tran J, Dally W (2015) Learning both weights and connections for efficient neural network. Adv Neural Inf Process Syst 28
61. Zhang M, Chen H, Shen C, Yang Z, Ou L, Yu X, Zhuang B (2023). Loraprune: pruning meets low-rank parameter-efficient fine-tuning
62. Jin Q, Dhingra B, Liu Z, Cohen W, Lu X (2019) Pubmedqa: a dataset for biomedical research question answering. In Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pp 2567–2577
63. Bai H, Hou L, Shang L, Jiang X, King I, Lyu MR (2022) Towards efficient post-training quantization of pre-trained language models. In: Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A (eds) Advances in neural information processing systems, vol 35. Curran Associates, Inc, pp 1405–1418
64. Chai Y, Gkountouras J, Ko GG, Brooks D, Wei G-Y (2023). Int2.1: towards fine-tunable quantized large language models with error correction through low-rank adaptation, arXiv preprint arXiv:2306.08162
65. Gong Z, Liu J, Wang Q, Yang Y, Wang J, Wu W, Xian Y, Zhao D, Yan R (2023). Prequant: a task-agnostic quantization approach for pre-trained language models, arXiv preprint arXiv:2306.00014
66. Lin J, Tang J, Tang H, Yang S, Chen W-M, Wang W-C, Xiao G, Dang X, Gan C, Han S (2024) Awq: activation-aware weight

quantization for on-device llm compression and acceleration. In Proceedings of machine learning and systems 6, pp 87–100
67. Tao C, Hou L, Zhang W, Shang L, Jiang X, Liu Q, Luo P, Wong N (2022). Compression of generative pre-trained language models via quantization, arXiv preprint arXiv:2203.10705
68. Wu X, Li C, Aminabadi RY, Yao Z, He Y (2023) Understanding int4 quantization for language models: latency speedup, composability, and failure cases. In International Conference on Machine Learning, PMLR, pp 37524–37539
69. Zhang W, Hou L, Yin Y, Shang L, Chen X, Jiang X, Liu Q (2020). Ternarybert: distillation-aware ultra-low bit bert, arXiv preprint arXiv:2009.12812
70. Piao T, Cho I, Kang U (2022) SensiMix: sensitivity-aware 8-bit index & 1-bit value mixed precision quantization for BERT compression. PLoS One 17(4):e0265621. https://doi.org/10.1371/journal.pone.0265621
71. Lee C, Jin J, Kim T, Kim H, Park E (2023) Owq: lessons learned from activation outliers for weight quantization in large language models. CoRR
72. Dettmers T, Svirschevski R, Egiazarian V, Kuznedelev D, Frantar E, Ashkboos S, Borzunov A, Hoefler T, Alistarh D (2023). Spqr: a sparse-quantized representation for near-lossless llm weight compression, arXiv preprint arXiv:2306.03078
73. Kim J, Lee JH, Kim S, Park J, Yoo KM, Kwon SJ, Lee D (2023) Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S (eds) Advances in neural information processing systems, vol 36. Curran Associates, Inc, pp 36187–36207
74. Dettmers T, Pagnoni A, Holtzman A, Zettlemoyer L (2023) Qlora: efficient finetuning of quantized llms. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S (eds) Advances in neural information processing systems, vol 36. Curran Associates, Inc., pp 10088–10115
75. Bai H, Zhang W, Hou L, Shang L, Jin J, Jiang X, Liu Q, Lyu M, King I (2020). Binarybert: Pushing the limit of bert quantization, arXiv preprint arXiv:2012.15701
76. Liu Z, Oguz B, Pappu A, Xiao L, Yih S, Li M, Krishnamoorthi R, Mehdad Y (2022) Bit: robustly binarized multi-distilled transformer. Adv Neural Inf Process Syst 35:14303–14316
77. Shen S, Dong Z, Ye J, Ma L, Yao Z, Gholami A, Mahoney MW, Keutzer K (2020) Q-bert: Hessian based ultra low precision quantization of bert. Proc AAAI Conf Artif Intell 34(5):8815–8821. https://ojs.aaai.org/index.php/AAAI/article/view/6409
78. Zafrir O, Boudoukh G, Izsak P, Wasserblat M (2019) Q8bert: quantized 8bit bert. In 2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS), pp 36–39. https://doi.org/10.1109/EMC2-NIPS53020.2019.00016
79. Kim S, Gholami A, Yao Z, Mahoney MW, Keutzer K (2021) I-bert: integer-only bert quantization, in: Meila M, Zhang T (eds) Proceedings of the 38th International Conference on Machine Learning, Vol. 139 of Proceedings of Machine Learning Research, PMLR, pp 5506–5518. https://proceedings.mlr.press/v139/kim21d.html
80. Chen P, Yu H-F, Dhillon I, Hsieh C-J (2021) Drone: data-aware low-rank compression for large nlp models. Adv Neural Inf Process Syst 34:29321–29334
81. Valipour M, Rezagholizadeh M, Kobyzev I, Ghodsi A (2022). Dylora: parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation, arXiv preprint arXiv:2210.07558
82. Zhang Q, Chen M, Bukharin A, Karampatziakis N, He P, Cheng Y, Chen W, Zhao T (2023). Adalora: adaptive budget allocation for parameter-efficient fine-tuning, arXiv preprint arXiv:2303.10512
83. Li Y, Yu Y, Liang C, He P, Karampatziakis N, Chen W, Zhao T (2023). Loftq: lora-fine-tuning-aware quantization for large language models, arXiv preprint arXiv:2310.08659
84. Dettmers T, Pagnoni A, Holtzman A, Zettlemoyer L (2023) Qlora: efficient finetuning of quantized llms. Adv Neural Inf Process Syst 36:10088–10115
85. Qin H, Ma X, Zheng X, Li X, Zhang Y, Liu S, Luo J, Liu X, Magno M (2024). Accurate lora-finetuning quantization of llms via information retention, arXiv preprint arXiv:2402.05445
86. Huang W, Zheng X, Ma X, Qin H, Lv C, Chen H, Luo J, Qi X, Liu X, Magno M (2024) An empirical study of llama3 quantization: from llms to mllms, visual intelligence. Visual Intel 2(1):36. https://doi.org/10.1007/s44267-024-00070-x
87. Wang H, Wu Z, Liu Z, Cai H, Zhu L, Gan C, Han S (2020) Hat: hardware-aware transformers for efficient natural language processing. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.686
88. Zhu H, Zhang GL, Huang S (2025). arXiv:2510.01472. Llm-nas: llm-driven hardware-aware neural architecture search. https://arxiv.org/abs/2510.01472
89. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al. (2020). An image is worth 16x16 words: transformers for image recognition at scale, arXiv preprint arXiv:2010.11929
90. Touvron H, Cord M, Douze M, Massa F, Sablayrolles A, Jégou H (2021) Training data-efficient image transformers & distillation through attention. In International conference on machine learning, PMLR, pp 10347–10357
91. Chen M, Peng H, Fu J, Ling H (2021) Autoformer: searching transformers for visual recognition. In Proceedings of the IEEE/CVF international conference on computer vision, pp 12270–12280
92. OpenAI. (2024). arXiv:2303.08774. Gpt-4 technical report. https://arxiv.org/abs/2303.08774
93. Jiang AQ, Sablayrolles A, Roux A, Mensch A, Savary B, Bamford C, Chaplot DS, D. L. Casas D, Hanna EB, Bressand F, et al. (2024). Mixtral of experts, arXiv preprint arXiv:2401.04088
94. Touvron H, Martin L, Stone K, Albert P, Almahairi A, Babaei Y, Bashlykov N, Batra S, Bhargava P, Bhosale S, et al. (2023). Llama 2: open foundation and fine-tuned chat models, arXiv preprint arXiv:2307.09288
95. Lin Y, Wang M, Zhang Z, Wang X, Xiao T, Zhu J (2023). Understanding parameter sharing in transformers, arXiv preprint arXiv:2306.09380
96. Xia Y, He T, Tan X, Tian F, He D, Qin T (2019) Tied transformers: neural machine translation with shared encoder and decoder. In Proceedings of the AAAI conference on artificial intelligence, vol 33. pp 5466–5473
97. Dabre R, Fujita A (2019) Recurrent stacking of layers for compact neural machine translation models. In Proceedings of the AAAI Conference on Artificial Intelligence, vol 33. pp 6292–6299
98. Lv X, Zhang P, Li S, Gan G, Sun Y (2023) Lightformer: lightweight transformer using svd-based weight transfer and parameter sharing. Findings Of The Association For Computational Linguistics: ACL 2023, pp 10323–10335
99. Reid M, Marrese-Taylor E, Matsuo Y (2021). Subformer: exploring weight sharing for parameter efficiency in generative transformers, arXiv preprint arXiv:2101.00234
100. Takase S, Kiyono S (2021). Lessons on parameter sharing across layers in transformers, arXiv preprint arXiv:2104.06022
101. Shazeer N (2019). Fast transformer decoding: one write-head is all you need, arXiv preprint arXiv:1911.02150
102. Ainslie J, Lee-Thorp J, De Jong M, Zemlyanskiy Y, Lebrón F, Sanghai S (2023). Gqa: training generalized multi-query

transformer models from multi-head checkpoints, arXiv preprint arXiv:2305.13245
103. Dehghani M, Gouws S, Vinyals O, Uszkoreit J, Kaiser L (2018). Universal transformers. CoRR abs/1807.03819. arXiv:1807.03819
104. Lan Z, Chen M, Goodman S, Gimpel K, Sharma P, Soricut R (2019). Albert: a lite bert for self-supervised learning of language representations, arXiv preprint arXiv:1909.11942
105. Li XL, Liang P (2021). Prefix-tuning: optimizing continuous prompts for generation, arXiv preprint arXiv:2101.00190
106. Lester B, Al-Rfou R, Constant N (2021). The power of scale for parameter-efficient prompt tuning, arXiv preprint arXiv:2104.08691
107. Liu X, Ji K, Fu Y, Tam WL, Du Z, Yang Z, Tang J (2021). P-tuning v2: prompt tuning can be comparable to fine-tuning universally across scales and tasks, arXiv preprint arXiv:2110.07602
108. Jia M, Tang L, Chen B-C, Cardie C, Belongie S, Hariharan B, Lim S-N (2022) Visual prompt tuning. In European conference on computer vision, Springer, pp. 709–727
109. Xu L, Xie H, Qin S-ZJ, Tao X, Wang FL (2023). Parameter-efficient fine-tuning methods for pretrained language models: a critical review and assessment, arXiv preprint arXiv:2312.12148
110. Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G (2023) Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. ACM Comput Surv 55(9):1–35. https://doi.org/10.1145/3560815
111. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A et al. (2020) Language models are few-shot learners. Adv Neural Inf Process Syst 33:1877–1901
112. Dong Q, Li L, Dai D, Zheng C, Ma J, Li R, Xia H, Xu J, Wu Z, Liu T, et al. (2022). A survey on in-context learning, arXiv preprint arXiv:2301.00234
113. Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih W-T, Rocktäschel T et al. (2020) Retrieval-augmented generation for knowledge-intensive nlp tasks. Adv Neural Inf Process Syst 33:9459–9474
114. Izacard G, Lewis P, Lomeli M, Hosseini L, Petroni F, Schick T, Dwivedi-Yu J, Joulin A, Riedel S, Grave E (2022). Few-shot learning with retrieval augmented language models, arXiv preprint arXiv:2208.03299 1(2):4
115. Zhou H, Song L, Chen J, Zhou Y, Wang G, Yuan J, Zhang Q (2021). Rethinking soft labels for knowledge distillation: a bias-variance tradeoff perspective, CoRR abs/2102.00650. arXiv:2102.00650. https://arxiv.org/abs/2102.00650
116. Yang Z, Zeng A, Li Z, Zhang T, Yuan C, Li Y (2023). arXiv:2303.13005. From knowledge distillation to self-knowledge distillation: a unified approach with normalized loss and customized soft labels. https://arxiv.org/abs/2303.13005
117. Kim M, Lee S, Hong S, Chang D-S, Choi J (2022). Understanding and improving knowledge distillation for quantization-aware training of large transformer encoders. arXiv:2211.11014. https://arxiv.org/abs/2211.11014
118. Wang K, Yang F, van de Weijer J (2022). Attention distillation: self-supervised vision transformer students need more guidance. arXiv:2210.00944. https://arxiv.org/abs/2210.00944
119. Wang W, Wei F, Dong L, Bao H, Yang N, Zhou M (2020). arXiv:2002.10957. Minilm: deep self-attention distillation for task-agnostic compression of pre-trained transformers. CoRR abs/2002.10957. https://arxiv.org/abs/2002.10957
120. Jiao X, Yin Y, Shang L, Jiang X, Chen X, Li L, Wang F, Liu Q (2019). Tinybert: distilling BERT for natural language understanding. CoRR abs/1909.10351. arXiv:1909.10351. http://arxiv.org/abs/1909.10351
121. Sun S, Cheng Y, Gan Z, Liu J (2019). Patient knowledge distillation for BERT model compression, CoRR abs/1908.09355. arXiv:1908.09355. http://arxiv.org/abs/1908.09355
122. Dasgupta S, Cohn T (2025) Improving language model distillation through hidden state matching. In The Thirteenth International Conference on Learning Representations. https://openreview.net/forum?id=IcVSKhVpKu
123. Tian Y, Sun C, Poole B, Krishnan D, Schmid C, Isola P (2020). What makes for good views for contrastive learning. CoRR abs/2005.10243. arXiv:2005.10243. https://arxiv.org/abs/2005.10243
124. Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi EH, Le QV, Zhou D (2022) Chain-of-thought prompting elicits reasoning in large language models. In Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Curran Associates Inc, Red Hook, NY, USA
125. Kojima T, Gu SS, Reid M, Matsuo Y, Iwasawa Y (2022) Large language models are zero-shot reasoners. In Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Curran Associates Inc, Red Hook, NY, USA
126. Hazarika D, Poria S, Zadeh A, Cambria E, Morency L-P, Zimmermann R (2018) Conversational memory network for emotion recognition in dyadic dialogue videos, in: Walker M, Ji H, Stent A (eds) Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, New Orleans, Louisiana, pp 2122–2132. https://aclanthology.org/N18-1193/
127. Poria S, Hazarika D, Majumder N, Naik G, Cambria E, Mihalcea R (2019) MELD: a multimodal multi-party dataset for emotion recognition in conversations, in: Korhonen A, Traum D, Màrquez L (eds) Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, pp 527–536. https://aclanthology.org/P19-1050/
128. Hinton GE, Vinyals O, Dean J (2015). Distilling the knowledge in a neural network. ArXiv abs/1503.02531. https://api.semanticscholar.org/CorpusID:7200347
129. Romero A, Ballas N, Kahou SE, Chassang A, Gatta C, Bengio Y (2014). Fitnets: Hints for thin deep nets. CoRR abs/1412.6550. https://api.semanticscholar.org/CorpusID:2723173
130. Ren S, Wei F, Zhang Z, Hu H (2023). arXiv:2301.01296. Tinymim: an empirical study of distilling mim pre-trained models. https://arxiv.org/abs/2301.01296
131. Heo B, Kim J, Yun S, Park H, Kwak N, Choi JY (2019) A comprehensive overhaul of feature distillation. In 2019 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE Computer Society, Los Alamitos, CA, USA, pp 1921–1930). https://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00201
132. Zagoruyko S, Komodakis N (2016). Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. ArXiv abs/1612.03928. https://api.semanticscholar.org/CorpusID:829159
133. Furlanello T, Lipton ZC, Tschannen M, Itti L, Anandkumar A (2018). arXiv:1805.04770. Born again neural networks. https://arxiv.org/abs/1805.04770
134. Yang C, Xie L, Su C, Yuille AL (2018) Snapshot distillation: teacher-student optimization in one generation. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 2854–2863. https://api.semanticscholar.org/CorpusID:54436113
135. DeepSeek-AI (2025). arXiv:2412.19437. Deepseek-v3 technical report. https://arxiv.org/abs/2412.19437
136. Gunasekar S, Zhang Y, Aneja J, Cesar C, Mendes T, Giorno AD, Gopi S, Javaheripi M, Kauffmann P, de Rosa G, Saarikivi O, Salim A, Shah S, Singh Behl H, Wang X, Bubeck S, Eldan R,

Kalai AT, Lee YT, Li Y (2023, June). Textbooks are all you need. https://www.microsoft.com/en-us/research/publication/textbooks-are-all-you-need/
137. Yin H, Molchanov P, Li Z, Álvarez JM, Mallya A, Hoiem D, Jha NK, Kautz J (2019) Dreaming to distill: data-free knowledge transfer via deepinversion. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 8712–8721). https://api.semanticscholar.org/CorpusID:209405263
138. Yoo J, Cho M, Kim T, Kang U (2019) Knowledge extraction with no observable data. In: Wallach H, Larochelle H, Beygelzimer A, d' Alché-Buc F, Fox E, Garnett R (eds) Advances in neural information processing systems, vol 32. Curran Associates, Inc.
139. Wang Y, Kordi Y, Mishra S, Liu A, Smith NA, Khashabi D, Hajishirzi H (2023) Self-instruct: aligning language models with self-generated instructions, in: Rogers A, Boyd-Graber J, Okazaki N (eds) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers), Association for Computational Linguistics, Toronto, Canada, pp 13484–13508. https://doi.org/10.18653/v1/2023.acl-long.754
140. Y. A et al. (2024). Qwen2 technical report. arXiv:2407.10671. https://arxiv.org/abs/2407.10671
141. R. GTM et al. (2024). Gemma 2: improving open language models at a practical size, ArXiv abs/2408.00118. https://api.semanticscholar.org/CorpusID:270843326
142. Hsieh C-Y, Li C-L, Yeh C-K, Nakhost H, Fujii Y, Ratner A, Krishna R, Lee C-Y, Pfister T (2023). arXiv:2305.02301. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. https://arxiv.org/abs/2305.02301
143. Chen X, Sun Z, Wenjin G, Zhang M, Chen Y, Sun Y, Su H, Pan Y, Klakow D, Li W, Shen X (2025) Unveiling the key factors for distilling chain-of-thought reasoning. In: Che W, Nabende J, Shutova E, Pilehvar MT (eds) Findings of the association for computational linguistics: aCL 2025. Association for Computational Linguistics, Vienna, Austria, pp 15094–15119. https://aclanthology.org/2025.findings-acl.782/
144. Wang X, Cui J, Suzuki Y, Fukumoto F (2025). arXiv:2405.10587. Rdrec: rationale distillation for llm-based recommendation. https://arxiv.org/abs/2405.10587
145. Ouyang L, Wu J, Jiang X, Almeida D, Wainwright CL, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A, Schulman J, Hilton J, Kelton F, Miller LE, Simens M, Askell A, Welinder P, Christiano PF, Leike J, Lowe RJ (2022) Training language models to follow instructions with human feedback. ArXiv abs/2203.02155. https://api.semanticscholar.org/CorpusID:246426909
146. Rafailov R, Sharma A, Mitchell E, Ermon S, Manning CD, Finn C (2023) Direct preference optimization: your language model is secretly a reward model. In Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23, Curran Associates Inc., Red Hook, NY, USA
147. Yuan Y, Shi J, Zhang Z, Chen K, Zhang J, Stoico V, Malavolta I (2024) The impact of knowledge distillation on the energy consumption and runtime efficiency of nlp models. In Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI, CAIN '24, Association for Computing Machinery, New York, NY, USA, pp 129–133. https://doi.org/10.1145/3644815.3644966
148. Sanh V, Debut L, Chaumond J, Wolf T (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. ArXiv abs/1910.01108. https://api.semanticscholar.org/CorpusID:203626972
149. Li T, Mesbahi YE, Kobyzev I, Rashid A, Mahmud A, Anchuri N, Hajimolahoseini H, Liu Y, Rezagholizadeh M (2021). A short study on compressing decoder-based language models. arXiv:2110.08460. https://arxiv.org/abs/2110.08460
150. Tian Y, Krishnan D, Isola P (2022). Contrastive representation distillation. arXiv:1910.10699. https://arxiv.org/abs/1910.10699
151. Xu S, Zhang X, Wu Y, Wei F (2021) Sequence level contrastive learning for text summarization. ArXiv abs/2109.03481. https://api.semanticscholar.org/CorpusID:237439568
152. Kim Y, Rush AM (2016). Sequence-level knowledge distillation. arXiv:1606.07947. https://arxiv.org/abs/1606.07947
153. Zhou C, Liu P, Xu P, Iyer S, Sun J, Mao Y, Ma X, Efrat A, Yu P, Yu L, Zhang S, Ghosh G, Lewis M, Zettlemoyer L, Levy O (2023). Lima: less is more for alignment. arXiv:2305.11206. https://arxiv.org/abs/2305.11206
154. Liu H, Li C, Wu Q, Lee YJ (2023). Visual instruction tuning. arXiv:2304.08485. https://arxiv.org/abs/2304.08485
155. Tunstall L, Beeching E, Lambert N, Rajani N, Rasul K, Belkada Y, Huang S, von Werra L, Fourrier C, Habib N, Sarrazin N, Sanseviero O, Rush AM, Wolf T (2023). Zephyr: direct distillation of lm alignment. arXiv:2310.16944. https://arxiv.org/abs/2310.16944
156. Hong J, Lee N, Thorne J (2024). Orpo: monolithic preference optimization without reference model. arXiv:2403.07691. https://arxiv.org/abs/2403.07691
157. Alayrac J-B, Donahue J, Luc P, Miech A, Barr I, Hasson Y, Lenc K, Mensch A, Millican K, Reynolds M, Ring R, Rutherford E, Cabi S, Han T, Gong Z, Samangooei S, Monteiro M, Menick J, Borgeaud S, Brock A, Nematzadeh A, Sharifzadeh S, Binkowski M, Barreira R, Vinyals O, Zisserman A, Simonyan K (2022). Flamingo: a visual language model for few-shot learning. arXiv:2204.14198. https://arxiv.org/abs/2204.14198
158. de Wynter A, Wang X, Sokolov A, Gu Q, Chen S-Q (2023) An evaluation on large language model outputs: discourse and memorization. Nat Language Process J 4:100024. https://www.sciencedirect.com/science/article/pii/S2949719123000213
159. Daphne Samuel MAR (1970) Mxlpred: a natural language processing deep learning model for icu patients' data. Int Arab J Inf Technol (IAJIT) 22(4):788–800. https://doi.org/10.34028/iajit/22/4/12
160. Daphne S, Rajam VMA, Hemanth P, Dinesh S (2025) An ensemble patient graph framework for predictive modelling from electronic health records and medical notes. Diagnostics 15(6):756. https://www.mdpi.com/2075-4418/15/6/756
161. Hendrycks D, Burns C, Basart S, Zou A, Mazeika M, Song D, Steinhardt J (2021). Measuring massive multitask language understanding. arXiv:2009.03300. https://arxiv.org/abs/2009.03300
162. Es S, James J, Espinosa-Anke L, Schockaert S (2025). Ragas: automated evaluation of retrieval augmented generation. arXiv:2309.15217. https://arxiv.org/abs/2309.15217
163. Jin D, Pan E, Oufattole N, Weng W-H, Fang H, Szolovits P (2021) What disease does this patient have? a large-scale open domain question answering dataset from medical exams. Appl Sci 11(14):6421. https://www.mdpi.com/2076-3417/11/14/6421
164. Jin Q, Dhingra B, Liu Z, Cohen W, Lu X (2019) PubMedQA: a dataset for biomedical research question answering, in: Inui K, Jiang J, Ng V, Wan X (eds) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, pp 2567–2577. https://aclanthology.org/D19-1259/
165. L. P et al. (2023). Holistic evaluation of language models. arXiv:2211.09110. https://arxiv.org/abs/2211.09110
166. Bai G, Liu J, Bu X, He Y, Liu J, Zhou Z, Lin Z, Su W, Ge T, Zheng B, Ouyang W (2024) Mt-bench-101: a fine-grained benchmark for evaluating large language models in multi-turn dialogues. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers), Association for

Computational Linguistics, pp 7421–7454. https://doi.org/10.18653/v1/2024.acl-long.401
167. Chen M, Tworek J, Jun H, Yuan Q, Pondé H, Kaplan J, Edwards H, Burda Y, Joseph N, Brockman G, Ray A, Puri R, Krueger G, Petrov M, Khlaaf H, Sastry G, Mishkin P, Chan B, Gray S, Ryder N, Pavlov M, Power A, Kaiser L, Bavarian M, Winter C, Tillet P, Such FP, Cummings DW, Plappert M, Chantzis F, Barnes E, Herbert-Voss A, Guss WH, Nichol A, Babuschkin I, Balaji S, Jain S, Carr A, Leike J, Achiam J et al. (2021) Evaluating large language models trained on code. ArXiv abs/2107.03374. https://api.semanticscholar.org/CorpusID:235755472
168. Belcak P, Heinrich G, Diao S, Fu Y, Dong X, Muralidharan S, Lin YC, Molchanov P (2025). Small language models are the future of agentic AI. arXiv:2506.02153. https://arxiv.org/abs/2506.02153
169. Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih W-T, Rocktäschel T, Riedel S, Kiela D (2020) Retrieval-augmented generation for knowledge-intensive nlp tasks. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA
170. Schick T, Dwivedi-Yu J, Dessí R, Raileanu R, Lomeli M, Hambro E, Zettlemoyer L, Cancedda N, Scialom T (2023) Toolformer: language models can teach themselves to use tools. In Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23, Curran Associates Inc., Red Hook, NY, USA
171. Kim H, So B-H, Han W-S, Lee H (2020) Natural language to sql: where are we today? Proc VLDB Endow 13(10):1737–1750. https://doi.org/10.14778/3401960.3401970
172. Wang L, Ma C, Feng X, Zhang Z, Yang H, Zhang J, Chen Z, Tang J, Chen X, Lin Y, Zhao WX, Wei Z, Wen J (2024) A survey on large language model based autonomous agents. Front Comput Sci 18(6):186345. https://doi.org/10.1007/s11704-024-40231-1
173. Yao S, Zhao J, Yu D, Du N, Shafran I, Narasimhan K, Cao Y (2022) React: synergizing reasoning and acting in language models. ArXiv Abs/2210.03629. https://api.semanticscholar.org/CorpusID:252762395
174. Duan Q, Lu Z (2025). Agent communications toward agentic ai at edge – a case study of the agent2agent protocol. arXiv:2508.15819. https://arxiv.org/abs/2508.15819

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.