

# Towards Incremental Learning in Large Language Models: A Critical Review

Mladjan Jovanovic, Peter Voss

**Abstract** \*—Incremental learning is the ability of systems to acquire knowledge over time, enabling their adaptation and generalization to novel tasks. It is a critical ability for intelligent, real-world systems, especially when data changes frequently or is limited. This review provides a comprehensive analysis of incremental learning in Large Language Models. It synthesizes the state-of-the-art incremental learning paradigms, including continual learning, meta-learning, parameter-efficient learning, and mixture-of-experts learning. We demonstrate their utility for incremental learning by describing specific achievements from these related topics and their critical factors. An important finding is that many of these approaches do not update the core model, and none of them update incrementally in real-time. The paper highlights current problems and challenges for future research in the field. By consolidating the latest relevant research developments, this review offers a comprehensive understanding of incremental learning and its implications for designing and developing LLM-based learning systems.

**Index Terms**—Incremental learning, Large language models, Continual learning, Meta-learning, Parameter-efficient learning, Mixture-of-experts learning.

## I. INTRODUCTION

### A. Contribution and Structure

Incremental learning (IL) is the ability of a learning system to make decisions based on continuously incoming inputs rather than relying only on previously processed data. This allows the system to adapt and respond to changing conditions and user behavior immediately rather than waiting for periodic retraining.

Given the paradigm's complexity, the existing LLM solutions aim to enable IL but still haven't achieved it in practice. Therefore, this review takes a broader perspective on incremental LLM updates, considering them batch-incremental periodic model updates. It also differentiates between core model updates and external system updates.

Thus, the critical review primarily discusses IL approaches, methods, and techniques in Large Language Models (LLMs) that acquire new abilities to solve unseen tasks while preserving existing ones.

Distinct from typical systematized reviews, the paper highlights the following key factors:

- 1) *Analysis of topics related to IL in LLMs.* The paper covers fundamental aspects of continual learning, meta-learning, parameter-efficient learning, and mixture-of-experts learning, including learning from scarce, changing, or non-

existent task information, learning multimodal tasks, and learning task sequences. By exploring the IL in related learning paradigms, we highlight the challenges and opportunities in these areas.

- 2) *Comprehensible and compact exposition.* This paper recognizes the importance and complexity of IL and describes the concepts, methods, and techniques clearly and concisely. It aims to reach a broad audience, including engineers and researchers. Through brief descriptions, explanations, and references to original works, the paper helps readers grasp fundamental IL principles and their practical implications.
- 3) *Synthesis of essential information.* IL is a fast-growing field, with information scattered across various fields and sources. The paper consolidates the relevant information about IL, offering a comprehensive overview to interested readers. Synthesis of the latest practical developments aims to guide researchers and engineers looking for a deeper understanding of IL and its LLM-based applications.

By highlighting these contributions, the paper complements existing related reviews and offers a unique perspective into the current situation and outlooks for IL in LLMs. The paper consulted relevant sources, including Scopus, ACM Digital Library, and IEEE Xplore Digital Library. The inclusion criteria for each paper analyzed is the practical verification of the proposed approach describing the solution, learning process, training data, and experiment results.

This paper provides critical insights into existing IL approaches within the LLM design space. To do so, we first define the key LLM concepts and notations used throughout the paper in the current section. Section 2 delves into the current state of IL approaches from a unified perspective by categorizing them into three related topics: continual learning methods, meta-learning methods, parameter-efficient tuning methods, and a mixture of expert methods. The paper describes the representative examples from different approaches. Section 3 extracts the main findings with implications concerning the application of IL to real-world problems with an overview of the challenges ahead. The concluding section summarizes these challenges and research opportunities.

### B. Context and Motivation

LLMs are based on a transformer neural-network architecture [1] and produce text in natural language. Transformers introduced the *attention mechanism* that tracks the relations between words across lengthy text sequences in forward and reverse directions. It pays attention to (co-) appearances of words as chunks of information in terms of probabilities, not

\* This is a pre-print version of the article published by Expert Systems, Wiley, available at <http://dx.doi.org/10.1111/exsy.70127>.

the meaning of those words. Thus, it cannot *understand* since there is no reference to the "real world" or "conceptual world" to ground as explicit knowledge. Instead, they approximate complex probability distributions on a large number of variables (the context of a word in text) and predict next words by drawing from these probability distributions. That said, it represents domain-specific knowledge as a language structure relying on context-dependent long-range associations (aka parametric knowledge).

Over time, the architecture expanded for generating other media as embedding vectors have become numerical representations of text, audio, image and video (i.e., encoders for different data modalities) [2, 3].

LLMs can produce plausible human-like content in various domains. Models with these generative capabilities are called Foundation Models (FMs). They learn about existing data artifacts and use learned patterns to generate new data instances, including text, images, music, design, and motion. Massive models process vast amounts of unlabeled data, prioritizing general-purpose applications. These models are then customized to create tools and infrastructure for specific domains. This way, FMs can be adapted across a variety of scenarios. Table 1 lists fundamental LLM architectures.

TABLE 1. FUNDAMENTAL LLM ARCHITECTURES.

Type	Application	Learning	Models
Encoder (text-features)	Natural language understanding (NLU)	Masked language modeling	BERT, DeBERTa, ALBERT
Decoder (features-text)	Natural language generation (NLG)	Language modeling	GPT4, ChatGPT, Gemini, LLaMA, Falcon, Bard
Encoder-decoder (text-features-text)	Sequence-to-sequence tasks	Sequence-to-sequence modeling	GLM, T5, BART

The encoder transforms input text into a fixed-size set of features as words' positional encodings (embedding vectors) for NLU tasks, including named entity recognition and sentiment classification. The decoder accepts features to produce output text, such as dialog, question answering, and text summarization. Combined architecture encodes the input text into vectorial representation and decodes embeddings to generate output, such as machine translation.

The transformer-decoder architecture is shown in Figure 1. It consists of the embedding component, a decoder stack, and a head component with linear and softmax layers. The embedding component transforms NL text into numerical vectors (or tokens) chunks. The tokens are forwarded to the decoders comprising Multi-head Self-Attention (MSA) and Feedforward Network (FFN) modules. The MSA clusters each token by an attention map obtained from two linear mappings of the input tokens (as their dot product). An FFN then processes the tokens. LLMs stand out from other Deep Neural Network (DNN) architectures in two ways. First, they exhibit the autoregressive pattern, performing the generation task in iterations. Next, they incorporate the attention mechanism, which quadratically scales computational complexity with the input length. At the same time, LLM's inherent computations occur in the attention blocks within each decoder layer. LLMs

iteratively generate tokens (words) based on the previously generated sequence. The process repeats until the model processes and outputs the entire input sequence.

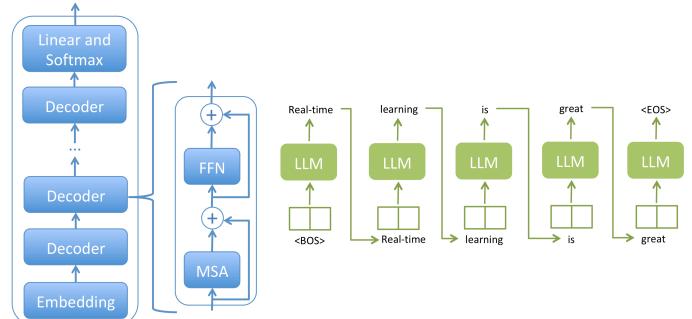


Fig. 1. The transformer-decoder architecture (left), and auto-regressive behavior (right). Adapted from [1].

Typical LLM learning strategies are as follows [3, 4]:

- 1) *Pretraining-finetuning* includes unsupervised training with a massive amount of data for general NLP tasks (also known as upstream tasks), followed by supervised training on a smaller amount of domain-specific data for specific tasks (aka downstream tasks).
- 2) *Instruction tuning* involves obtaining a new, instruction-financed model capable of following instructions. The model learns new tasks via natural language instructions. It involves actual training of the model by changing its weights. *Soft prompt* tuning is a case in which the model is frozen, and only the soft prompts (learnable tensors obtained through backpropagation) added to the input embedding are learned [5]. Unlike finetuning, which involves billions of parameters to train, in soft prompting, there are typically several thousand parameters to train. Similarly, *prefix-tuning* optimizes a small continuous task-specific vector (aka prefix) where subsequent tokens attend to the prefix [6]. It obtains state-of-the-art performance compared to full-data training and outperforms finetuning in low-data training by learning only 0.1% of the parameters [6].
- 3) *Prompting* refers to conceiving a prompt (as user input) that will yield the optimal response in a given situation. No training is involved (i.e., the LLM's weights are not changed due to no gradient updates). A prompt is a flexible task specification containing NL descriptions of intentions with examples and expected responses [7]. Various templates have been proposed as prompting techniques, including zero-shot (instructions with task-specific formatting but without task-specific examples), few-shot (instructions with a few task-specific demonstrations to guide the model's outputs), chain-of-thought (stepwise instructions with logical context to simulate reasoning), and tree-of-thought (guidance to select among multiple solutions) [7, 8, 9, 10]. *In-context learning* (ICL) is sometimes used to refer to few-shot prompting as the model is guided during inference with prompts containing task examples [7].

*Reinforcement Learning from Human Feedback* (RLHF) represents an alternative in which the model is refined based on human feedback or corrections [11]. Users express their preferences among different responses provided by the model.

Subsequently, a reward mechanism is established. The model enters a reinforcement learning loop to maximize the rewards by continuously refining its responses to better match the user's preferences. Ranking the model's responses from human evaluators offers a superior objective function compared to those utilized in pre-training. A recent study [12] highlighted practical challenges in using RLHF to customize LLM as misaligned humans (evaluators holding different goals and values), quality control with difficult oversight (humans making mistakes), representative data (biased), and challenges with reward model (reward hacking and evaluation). This often requires retraining these models, which is impractical regarding computation, time, and privacy.

*Retrieval-augmented generation* (RAG) [13] is a framework for increasing the quality of LLMs' outputs by grounding the models on external knowledge retrieval components to supplement the LLM's internal representation of information. Implementing RAG verifies LLM-based outputs against reliable facts and sources, ensuring users can check its claims for accuracy and trust. RAG helps reduce hallucinations, outdated knowledge, and non-transparent, untraceable content generation process [14, 15].

The resources required by LLMs (compute, data, environment impact) are enormous as they are based on statistical pattern matching on large quantities of human intelligence outputs. At the same time, LLMs are still out of proportion to achieving human-like intelligence, such as planning, reasoning, and self-verification [16, 17, 18].

IL of knowledge emerging in the real world is essential to achieve human-like intelligence [19]. However, standard neural network (NN) topologies are hindered by catastrophic forgetting (CF), a limitation that prevents them from learning a sequence of tasks. This issue must be addressed for an NN to successfully adapt to lifelong or continuous learning, a fundamental requirement for AI. Two common goals of continuous learning are to learn new tasks from known classes (online learning) and learn unknown classes (class-incremental learning) [20].

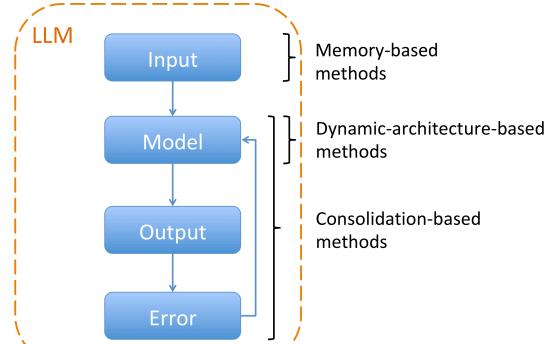
Rehearsing techniques can avoid CF [21]. This implies that when new data is introduced, the NN gets retrained on some previously learned data. However, previously learned knowledge may not be available for such retraining in general. In addition, CF may occur when compensating for concept drift [21]. Deep Learning (DL) assumes that a training set is a representative sample of the underlying unknown distribution. If the distribution space shifts, new modeling should expand this shift. If the initial space is merely shifted instead of adjusting the size of the distribution space, then CF will occur.

The current situation inspired us to look deeper into existing work on topics related to IL in LLMs, including continual learning, meta-learning, parameter-efficient learning, and mixture-of-experts learning, and the challenges posed by the extensive parameterization of LLMs and continual model adjustments to address deficiencies or undesirable behaviors.

## II. RELATED WORK

### A. Continual Learning

Continual learning (CL) learns new, emerging tasks efficiently while mitigating CF in LLMs.



**Fig. 2.** Approaches to Continual Learning.

Figure 2 illustrates approaches to CL. The existing work on CL can be classified as follows [22, 23, 24, 25]:

1) *Consolidation-based* methods, such as *regularization* [26] or *distillation* [27], protect important parameters from significant shifts. They do this by aligning the current output space with previous ones (distillation loss) or restricting the model parameters by estimating loss or averaging weights (regularization loss). *Weight regularization* concerns parameters, whereas *function regularization* influences behaviors of the existing model. *Weight regularization* selectively regularizes variation of NN's parameters. It typically penalizes each parameter's variation based on its importance in performing the previous tasks [28]. *Function regularization* is a robust approach targeting the model's in-between results or the prediction. It typically implements Knowledge Distillation (KD) [29] by learning a small student model from a large teacher model previously learned to mitigate CF. The student model can learn from new training examples, replay previous examples, and use available unlabeled data or data generated from previously learned tasks.

Aside from regularization, some methods focus on *error optimization*, such as calculating gradient propagation on the old tasks' input data [30] (e.g., rectifying the current gradient directions through parameter updates), meta-learning of arriving tasks [31] (e.g., obtaining an inductive bias for new tasks internally, rather than specifying it explicitly), and investigating loss spaces [32] (e.g., discovering a local minimum by adjusting the training variables, including learning rate decay, dropout rate, and batch size).

Self-supervised learning (generating implicit labels from unstructured data) and upstream pretraining with downstream finetuning (pre-trained representations fixed when learning downstream tasks) also belong to this class as they create and employ internal representations in LLMs. However, they require large amounts of data.

*Task-agnostic distillation* (TAD) iteratively prunes the student's model without relying on task-specific exemplars until it reaches the target size, thereby reducing the performance gap between the teacher and student models across a wider set of tasks [33]. It is also used within RL as a method for distilling a policy model in a self-supervised

manner, without external goals [34]. Some approaches combine TAD variations to distill a compact student, such as extracting teacher parameters randomly or based on a relevance metric [35].

While these methods may only partially utilize historical data, their ability to safeguard crucial parameters is a significant advantage.

- 2) *Dynamic-architecture-based* methods [36] train a model by increasing its size (number of parameters) with the task number (also known as *parameter expansion*). These methods introduce task-specific parameters through a model adaptation by adding task-specific parameters (*parameter allocation*), creating task-specific modules (*modular network*), and differentiating shared and task-specific model components (*model decomposition*).

*Parameter allocation* dedicates a parameter set in the model to each task. The expansion of such models should be balanced with the increase in tasks [37]. *Model decomposition* separates a model into shared and task-specific elements. For instance, the parameters can be decomposed by additive decomposition, singular value decomposition, and low-rank factorization [38]. A *modular network* includes modules or sub-networks learning incremental tasks simultaneously without sharing task-specific components. Mixture-of-Experts (MOEs) architectures are examples of this method. We describe them separately in Section 2.4.

*Visual Prompt Tuning* (VPT) [39] combines previous techniques by utilizing lightweight trainable modules to adjust the pre-trained model by prepending learnable parameters (aka prompts) to the base features. The model concatenates parameters and features for the vision transformer blocks. Encoding task-specific information into the prompts while keeping pre-trained weights unchanged minimizes the cross-entropy loss.

*Growing neural networks-based approaches* propose a scalable network of NN architectures in which specific nodes encode particular tasks, such as multitask learning in robots [40]. Alternatively, self-organizing incremental NNs can learn classification tasks while mitigating CF [41].

The previous methods have a major drawback - the linear growth of training costs [22, 25] as the model size grows with new tasks linearly. This cost increase can be a deterrent despite their ability to overcome CF.

- 3) *Memory-based* methods [42] keep additional memory of examples of previous tasks, which are used (replayed) when learning new tasks. While they retain some information from previous tasks, they face challenges such as overfitting and low scalability due to frequent retraining of the model (aka *replay-based* or *rehearsal-based*).

These methods typically approximate and reconstruct old data distributions. *Experience replay* records a small amount of old training examples [43]. *Generative replay* employs a generative model to produce synthetic examples for training [44]. *Feature replay* restores the distribution of old features [45] through saving prototypes (i.e., feature distillation between the old and new model or representation shift to update the preserved old features),

statistical information (e.g., mean and covariance), or training a generative model.

*Resource-efficient replay approaches* aim to merge similar inputs and increase distances between samples to enlarge sample variability and reduce memory [46]. Similarly, in streaming learning, a reduced number of prototypical examples that capture most of the variance within different classes can be stored [47].

However, selecting exemplars from pretraining datasets may not be possible or can be difficult, and downstream data may not always be suitable for preserving the pretraining knowledge.

Table 2 gives an overview of CL methods, techniques, target model elements, and trade-offs.

TABLE 2. CL METHODS COMPARATIVE SUMMARY.

Method	Technique	Element	Trade-offs
Consolidation-based	Distillation Regularization Optimization	Parameter subsets Intermediate values and representations	Updates/extracts relevant parameters Increased computation
Dynamic-architecture-based	Parameter/network expansion Model decomposition	Task-specific parameters, modules and components	Updates portion of parameters New tasks increase model size
Memory-based	Historical and derived task examples	Parameters Components Layers	Retains model size Obtaining suitable training samples

Some definitions of IL refer to continual learning techniques in NN-based models [23] as task-incremental, domain-incremental, and class-incremental learning, depending on whether task identity is specified at test time or is inferred by the model. Task-incremental learning (TIL) learns to solve several distinct tasks. Domain-incremental learning (DIL) learns to solve the same problem in different contexts (input distribution changes). Thus, the context refers to the underlying distribution from which observations are sampled. Class-incremental learning (CIL) learns to discriminate between incrementally observed classes. The main challenges in domain-incremental and class-incremental learning are preventing forgetting and correctly identifying tasks, respectively [48].

IL in LLMs can also be classified as offline IL and online in-context learning [49], considering training data sources.

Offline IL, while allowing for specific adaptation with new annotated data sets (e.g., supervised fine-tuning and RLHF [11]), has its limitations. It requires frequent model updates for evolving tasks and scenarios, and static post-updates are inflexible to adapt to real-time changes.

Online in-context learning assumes an LLM interacts with an external AI model or agent or with an offline or online knowledge source (e.g., RAG [13]). Such methods depend on external knowledge and do not retain persistent learned information, which may lead to a loss of used knowledge.

Existing reviews examine CL as LLM adaptations across time, domains, and capabilities by focusing on the kind of information that is updated and learning strategies at their specific training stages (i.e., pre-training and fine-tuning) [22, 50, 51]. Specifically, the analysis from [51] classifies CL as:

- *Continual Pre-training* (CPT) expands the model's general NL understanding and generation capabilities [52]. It conducts self-supervised training to enrich the model's general NLP knowledge and general knowledge of new domains, such as learning new natural or programming languages, factual knowledge (e.g., most recent information), and domain knowledge (e.g., e-commerce).
- *Continual Instruction Tuning* (CIT) improves the model's response to specific user inputs [53]. It conducts supervised instruction-following finetuning to learn domain-specific tasks. For instance, learning to solve new tasks (e.g., code generation), tasks in novel domains (e.g., medicine), or to use new tools to solve problems (e.g., search engines).
- *Continual Alignment* (CA) ensures the model's outputs adhere to human values, preferences, and ethical and societal norms [54]. For example, learning to align to ethical and social norms (e.g., culture-specific norms) or users' preferences (e.g., emerging values on new users).

While it does not provide an in-depth analysis of model workings and training data, our review aggregates their principles and criticalities according to the proposed classification. Moreover, it outlines some typical cases, reveals related effects, and highlights practical limitations for a broader audience as follows. While some examples combine different methods, we classify them based on the primary method employed.

#### *Consolidation-based examples*

Prototype and Low-Rank Adaptation (PLoRA) [55] starts from a pre-trained backbone and finetunes a small number of parameters to learn new classes in Federated Class-Incremental Learning (FCIL). FCIL is a method to continuously learn new classes while addressing the challenge of forgetting old classes [56]. It is based on a distributed ML paradigm called Federated Learning (FL), where multiple data owners (or local models) collaborate to train a shared or server model. Existing methods [56, 57] propose different loss functions on the client and server sides to alleviate local and global CF but result in significant memory and communication costs. PLoRA introduces a shared prototype module for the global server, aggregating (re-weighting) the local prototypes from clients. The local prototype vectors are derived from prototypes of corresponding class features. Then LoRA [58] finetunes only trainable parameters and sends them to the global server. In particular, each client uses local data to train LoRA parameters to obtain prototypes of each class. Only the trainable parameters and the average class feature are shared with the server. The server re-weights the local prototypes it received to form the global prototype. PLoRA utilizes ViT-B/16-IN21K [59] as the backbone PTM for server and client models. While PLoRA outperforms state-of-the-art approaches, it has high training costs for data, memory, and computational resources for server and client models.

Few-Shot Class-Incremental Learning (FSCIL) [60] trains a base model with sufficient data. Later, it utilizes the knowledge of base classes to learn of new classes with limited data while preserving the understanding of old ones. FSCIL usually fine-tunes the entire model, being susceptible to overfitting and with reduced performance in learning new

classes [60]. The Attention-aware Self-adaptive Prompt (ASP) framework [61] adds specific prompts between attention blocks as attention-aware task-invariant prompts (TIP) and self-adaptive task-specific prompts (TSP). The TIP contains task information common for all classes (old and new). The prompts are constructed by encoding input images strongly correlated with semantically relevant information and weakly correlated with irrelevant image information. The TSP is constructed for a particular input image, combining the average and image prompt features. This way, it fine-tunes only the task-specific parameters. ASP uses ViT-B/16-1K [59] as the base model, pre-trained on the ImageNet1K dataset, which spans 1000 classes and 1,281,167 training images, 50,000 validation images, and 100,000 test images. The ASP outperforms state-of-the-art FSCIL methods in learning new classes. However, training costs are high, and learning new classes can drop the performance of previous ones.

The INCremental Prompting (INCPrompt) [62] utilizes flexible task-aware prompts to effectively represent task-specific information and mitigate the CF of old tasks. It implements an attention-based transformation of the token sequence whose results are passed to a group of FF layers with a non-linear activation function (named the prompter) to generate the prompt as key and value components. Experimental evaluations on Split CIFAR-100 (60,000 images in 100 classes) and Split ImageNet-R [63] (30,000 images in 200 classes) CL benchmarks showed improved performance over existing methods. However, its scalability concerning task number and data complexity and generalizability to other CL scenarios with different data distributions remain unclear.

Incremental Vision-Language Object Detection (IVLOD) [64] adapts pre-trained Vision-Language Object Detection Models (VLODMs) to different domains while preserving their general capabilities. The method employs Zero-interference Reparameterizable Adaptation (ZiRa) with Zero-interference Loss (ZiL) and reparameterization techniques for IVLOD without incurring computation and memory costs. In particular, it retains the original model parameters and adds a parallel branch component (Reparameterizable Dual Branch - RDB) for tuning on downstream tasks without overwriting existing knowledge. Based on the RDB's dual-branch structure, the ZiL component reduces the interference of new knowledge on learned knowledge by penalizing the RDB's output. The RDB learns new tasks while protecting pretraining and downstream task knowledge. The IVLOD uses a Swin Transformer [65] as a backbone VLM. The evaluation demonstrates that ZiRa preserves general capabilities while accurately learning new tasks, outperforming state-of-the-art object detection methods [66, 67]. However, the enhanced performance comes with an increased model size.

The study on *Knowledge Editing* (KE) for LLMs presents practical and efficient methods for modifying and evaluating LLMs post-training to learn new tasks or correct outdated information [68]. These methods, such as knowledge insertion, modification, and erasure, are designed to allow on-the-fly model modifications while maintaining overall performance across various inputs.

In analogy to the human learning types, KE methods are categorized as:

- 1) *Recognition* - exposing the model to new knowledge within a relevant context.
- 2) *Association* - forming connections between the model's new knowledge and previous knowledge in the model.
- 3) *Mastery* - the model acquires and utilizes the knowledge through its parameters.

However, the above methods focus on LLMs' components - memory, retriever, layers, and parameters which reflect correlations rather than explicit knowledge. In particular, recognition as resorting to external knowledge is implemented by expanding LLMs' layers (e.g., FFNs) and parameters (e.g., LoRA) [69]. The association is realized via user interactions with the LLM without retraining as additional memory of instruction examples [70] (i.e., various prompting techniques). Mastery manifests through cost-efficient alternatives to fine-tuning. For instance, meta-learning methods edit the model by learning the change in the model instead of updating the weights directly. Model Editor Networks using Gradient Decomposition (MEND) [71] uses the rank-one decomposition to split the model into separate rank-one matrices to compute the change of the weights, reducing the number of parameters. Some techniques try to locate where the knowledge was stored and edit that part of the model [72] (e.g., FFN's area or a set of model weights). However, the side effects are unclear due to underlying LLM opaqueness [73].

The study in [74] introduces a post-training adjustment method to mitigate CF and improve CL in MLLMs. The Model Tailor retains the pre-trained parameters and replaces fewer finetuned parameters (up to 10%). Building on the tailor metaphor that selects patches for a garment, it identifies and adjusts a minimal parameter set as a sparse mask (or the model patch) from the finetuned model. It extracts the patch by inspecting parameter and loss changes to identify a critical subset of finetuned parameters essential for learning a target task. Moreover, it decorates the patch by compensating weights with an inverse Hessian matrix to enhance the performance on both current and previous tasks. The method is evaluated using InstructBLIP [75] and LLaVA-1.5 [76] MLLMs for VQA and image captioning tasks with available benchmark datasets. The results show improved CL while preserving pre-trained capabilities compared to traditional finetuning. However, it remains to be verified in more complex and open domains.

An approach to ingrain learning-in-realtime capabilities is an integration of LLMs with knowledge graphs (KGs) [77]. Neurosymbolic computing [78] combines neural networks' (NNs') pattern-recognition capabilities with KGs' reasoning abilities by either compressing knowledge structures into vectorized representations suitable for NNs [79] or linking neural patterns with symbolic knowledge by extracting pertinent pattern information [78]. Currently, the LLM-KG integration is mainly implemented in learning pipelines with separate LLM and KG components, losing some of the original semantics in the produced representations in both techniques [77, 80].

Continual Optimal Policy Regularization (COPR) is a method that aligns LLMs with human preferences in a CL setting [81]. RLHF-based techniques often require retraining when encountering new queries or feedback, as human preferences vary across domains or tasks. The method computes the distribution of optimal policy bypassing the partition function and then regularizes the policy based on optimal distribution to mitigate CF and improve preference learning. Although the method outperforms existing CL methods in the task and domain CL settings, it still needs RLHF reward and policy models to start from, which may reflect the preferences of a particular user group and induce a bias when optimizing the current policy.

While LLMs' attention has a quadratic dependence on context size, it is often sparse where a query is most relevant to a subset of the keys, such that many similarity scores converge to zero and their computation can be skipped. While predefined heuristics primarily drive existing sparsity solutions, intrinsic attention adaptation [82] learns the attention sparsity directly from the LLM itself as a gate that selectively activates a subset of relevant attention blocks. The technique is computationally efficient and improves output accuracy. Yet, the accuracy drops with context size.

#### *Dynamic-architecture-based examples*

The study described in [83] employed various techniques to prevent CF of the finetuned LLaMA2 model with 7 billion parameters, including freezing specific model layers, freezing specific model modules (e.g., self-attention and FF modules), and model adapters introducing additional trainable parameters (e.g., LoRA [58]). The evaluation results showed the repetition problem (the greedy approach of choosing the next token, where the highest probability token is always selected, causes the model to generate a repetitive token sequence) and a decline in reliability. In contrast, the model's knowledge remained mainly unaffected. However, the model was evaluated on only two distinct tasks.

Excessive LLM modifications can cause forgetting, while insufficient adjustments make inadequate fits for new classes. ExpAndable Subspace Ensemble (EASE) is an approach to Pretrained Model (PTM)-based CIL addressing the above issue [48]. The expanded model initializes and trains an adapter with trainable parameters encoding task-specific information per incoming task. The model extracts embeddings by aggregating historical features to synthesize prototypes of old classes later. In addition, all adapters extract the prototypes of the current dataset for new classes. Verification using ViT-B/16-IN21K [59] (a vision transformer encoder model with 86.4 million parameters pre-trained on ImageNet-21k containing 14 million images with 21,843 classes) as a backbone PTM shows state-of-the-art performance. Although adapters are small (0.3% of the base model), model size increases with additional adapters.

MiniGPT-4 [84] connects a pre-trained visual encoder and a pre-trained LLM. In particular, it combines a vision encoder (a pretrained ViT coupled with Q-Former [85]), Vicuna LLM [86], and a projection layer connecting the two models. The linear projection layer aligns the visual features with the textual ones. It does so in a two-step training method. First, it

pretrains the vision encoder on many image-text pairs to learn vision-language skills. Then, it finetunes the model using a small number of high-quality image-text examples and a predefined prompt template to elicit more natural and reliable NL outputs. MiniGPT-4 showcases compositional vision-language abilities in a variety of tasks (e.g., generating detailed image descriptions, extracting facts from movie photos, and image captioning). However, the model inherits the backbone LLM’s limitations and may hallucinate (e.g., identifying non-existent objects in the image), especially with longer captions.

The Generative Multi-modal Model (GMM) framework for CIL [87] generates detailed image descriptions by incorporating visual and textual information with a customized generative model. The model contains an encoder producing image and text embeddings image and text, which are further processed by an auto-aggressive decoder to generate image descriptions. Then, it uses a text encoder to produce textual features from the image descriptions and matches features to decide on a label that is highly similar to the prediction. The framework employs Bootstrapping Language-Image Pre-training with frozen unimodal models (BLIP) [88], a vision-language model pre-trained from existing pre-trained image encoders and LLMs with significantly fewer trainable parameters than existing methods. The model bridges modalities using a lightweight Querying Transformer learning vision-language representation with the image encoder and vision-to-language generation with the language model. GMM improves the state-of-the-art classification accuracy and CF. However, the main challenge is transferring the generated textual information into classifying distinct categories.

The study in [89] presents an Interactive Continual Learning (ICL) framework inspired by the Complementary Learning Systems theory [90]. ICL utilizes interactions among different models. Specifically, it explores the interactions between ViT (simulating System1) and MLLM (being System2). It adapts System1 parameters while keeping System2 parameters unchanged. System2 handles complex examples and interacts with System1. The Class-Knowledge-Task Multi-Head Attention (CKT-MHA) module enables continuous ViT finetuning using category features and from Class-Task collections. This way, System1 retains old parameters, avoiding ineffective updates. The von Mises-Fisher Outlier Detection and Interaction (vMF-ODI) mechanism estimates sample complexity so that System1 can select complex enough examples. The examples are used for System1 inference, whose predictions represent background knowledge for System2, thereby coordinating the two systems. System 1 uses ViT-B/16-IN21K [59] as a backbone, whereas System 2 employs MiniGPT-4 [84] as a base model. ICL was evaluated using CIFAR-10 (10 classes, with 50,000 training and 10,000 test color images per class), CIFAR-100 (100 classes, 500 training, and 100 testing images per class), and ImageNet-R (200 classes, 30,000 images). The evaluation showed CF resistance and superior TIL and CIL performance compared to existing methods. However, the limitations of the backbone models restrict the performance enhancement of System2.

Research from [91] pre-trained and evaluated the T0 model with 3 billion parameters [92] on 50 datasets related to 50 distinct textual QA, classification, and summarization tasks (100,000 examples for training per task). The model was extended to and verified through CL on 8 new language generation tasks (e.g., headline generation, text simplification, and haiku generation). On the one hand, the resulting model demonstrated acceptable performance in learning new tasks (e.g., concept drift) while maintaining nearly 100% of the initial performance. On the other hand, the study showed that CL emerges from the intensive pre-training stage.

Progressive Prompts [93] is a technique of learning a soft prompt per task, concatenating it with the learned prompts while keeping the base model unchanged. The goals are to keep the knowledge of previous tasks and allow the transfer of existing knowledge for new tasks. The model parameters consist of the fixed PLM parameters and fine-tuned prompt parameters. The study used an encoder-only BERT model and an encoder-decoder T5 model on text classification tasks. The proposed approach eliminates the need for data replay or storing many task-specific parameters. It shows superior performance on a standard text classification CL benchmark on longer CL sequences spanning 15 tasks. Nevertheless, the model size grows when the number of new tasks increases.

Visual instruction tuning uses machine-generated instruction-following data to improve LLMs in the language-to-image tasks. It extends instruction tuning to connect vision and language understanding with models like LLaVA (Large Language and Vision Assistant) [76]. LLaVa-1.5 architecture contains a pre-trained visual encoder, a pre-trained LLM to interact with users (Vicuna1.5 with 13 billion parameters [86]), and a vision-language cross-modal connector to align the vision encoder outputs to the language model. The model improved baselines on VQA (Visual question answering) tasks that answer an NL question about visual images. The results also suggest that visual instruction tuning is more critical in improving an LMM’s capabilities than pretraining. However, it requires high-quality, targeted visual instruction-tuning data and incurs high computation costs.

Contrastive Learning (CoL) learns meaningful representations from large-scale unlabeled data by maximizing the agreement of the positive pairs formed by the original instance and its corresponding augmented instance while minimizing the agreement with the negative pairs formed by other instances [94]. CoL can also use supervised learning by adding label information based on category discrimination. Some image classification tasks introduce prototypes, which force the embedding of instances closer to their corresponding prototypes in the embedding space while moving them away from other prototypes [95]. Contrastive Language-Vision Pre-training (CLIP) [96] represents a VLM pre-trained on large image-text datasets. The model contains one image and one text encoder. CoL is conducted during pre-training, where a correctly paired image and text is a positive pair. In contrast, incorrectly paired image and text (belonging to distinct image-text pairs) are a negative pair. The prediction is formed by finding the closest text embedding given the image. CLIP is known for its ability to perform zero-shot image classification,

which can classify images based on NL descriptions without specific training in those categories [96]. The study from [97] introduced Zero-Shot Continual Learning (ZSCL) to address the CF in continual learning with the VLM through feature distillation and parameters ensemble. Distilling the initial model features on a reference dataset significantly enhances its performance. Parameter ensemble among different training stages effectively mitigates the forgetting issue. The study also proposed a Multi-domain Task Incremental Learning (MTIL) benchmark to evaluate learning tasks from different domains, surpassing state-of-the-art methods. However, the limitation is that an extensive, high-quality reference dataset is needed.

Sequential fine-tuning (SEQ) described in [98] examined fine-tuning PLMs to learn a sequence of 15 downstream tasks of 4 types, including text and intent classification, named entity recognition, and relation extraction. For encoder backbones, the study used BERT (versions with 109 and 335 million parameters). Decoder backbones used GPT2 (124 and 774 million parameters) and GPT-NeoX (19 million to 1,21 billion parameters). The study revealed that the inherent anti-forgetting ability of PLMs comes from the pre-training stage and that classifiers learned in SEQ cause forgetting. Specifically, the relative position changes between the class embeddings in classifiers and the features extracted by PLMs led to decreased performance on old tasks even when PLMs do not forget. Besides, the experiments focused on IL classification tasks.

Class-incremental Learning (CIL) is employed by Vision-Language Models (VLM) to learn general representations from textual information without forgetting previously learned tasks [99]. PROjectiOn Fusion (PROOF) [100] approach alleviates CF in VLM. The model appends linear projections on the pre-trained image/text backbones. The respective projection layer encodes the task-specific information by mapping the projected features. Learning new tasks extends new projections while freezing old ones. The benchmarks indicate PROOF learns new classes while retaining the old ones, with state-of-the-art classification performance. Nevertheless, learning new tasks enlarges the model size and requires more training exemplars.

Processing long input sequences during inference is challenging for LLMs as they perform poorly on inputs whose lengths are larger than those in training sequences. SelfExtend [101] is a method that aims to handle long contexts without fine-tuning. It extends the LLMs' context window by structuring the attention component into a group and a neighbor attention. The group attention manages long-distance token relationships by tracking the dependencies among distant tokens. The neighbor or standard attention tracks dependencies among adjacent or proximate tokens based on a specified range. The attention matrices are merged, and the softmax operation is applied. This way, the method connects the distant positions at inference time to positions seen during training so that LLM can handle longer texts without additional fine-tuning. Extensive evaluations with LLaMA-2 [102] and its successors, Phi-2 [103], Mistral [104], and SOLAR [105], on short-context tasks, synthetic and real-world long-context tasks, and language modeling tasks have

consistently demonstrated the effectiveness of SelfExtend. However, this extension in context window length comes at the cost of increased computation, as it requires calculating extra attention across all query-key pairs.

The Scalable Language Model (SLM) [106] adapts the base LLM to new tasks from distinct domains while keeping its performance on the older tasks. SLM integrates the task distribution within the vector space retrieval into the language model for knowledge expansion and management. The Dynamic Task-related Knowledge Retrieval (DTKR) component identifies the most relevant knowledge for each input instance, assuming each task has a distinct data distribution in the vector space. It preserves relevant knowledge by compiling weight increments through low-rank adaptation to decrease computation. The Joint Adaptive Re-parameterization (JARe) then uses these weight increments to obtain adaptive re-parameterization of the pre-trained model, aligning it with downstream tasks based on their distribution. The SLM achieves state-of-the-art performance on different base models, including BERT [107], T5 [108], and LLaMA-2 [109]. However, the method introduces a separate retrieval component, increasing memory and computational costs.

Cross-modal CL [110] integrates multiple modalities via cross-attention, transferring information between different modalities into a common semantic and representation space. Additionally, a dynamic codebook keeps unknown information from incoming modalities as appended dictionary codes. This way, the model can generalize to new tasks and modalities while mitigating the CF of previously learned information. While it outperforms state-of-the-art models on bi-modal generalization tasks, it requires pre-training on multimodal datasets to acquire sufficient representation space.

#### *Memory-based examples*

Diffusion-based Class Incremental Learning (DiffClass) [111] uses Diffusion Models (DMs) [112], generative models that produce synthetic samples, to improve the classifier performance in CIL tasks. DMs generate images in a stochastic process by progressively denoising them [112]. They incrementally introduce Gaussian noise into the input data (a forward diffusion step). The procedure is gradually reversed to generate synthetic samples, which are used to update the classifier by removing the noise from noised input (a reverse diffusion step). Synthetic samples are similar to the real ones. The DiffClass produces artificial examples using Multi-Distribution Matching (MDM) models to balance the quality of the generated samples and their diversity. Before generating samples, Selective Synthetic Image Augmentation (SSIA) enlarges the training data distribution and enhances the model's plasticity. The method demonstrates state-of-the-art performance in example-free settings in CIFAR100 (60000 images with 100 classes) and ImageNet100 (135,000 images with 100 classes) benchmarks. However, learning new classes prolongs finetuning.

Some approaches [113] introduce and train soft prompts for each new task. However, the accumulation of prompts raises scalability issues by inducing higher training costs.

Instruction-based Continual Learning (InsCL) [114] replays data for each previous task based on their similarity calculated

as Wasserstein Distance (WD). The instruction embeddings are produced for pairs of tasks, and the proportions of instructions for each task are calculated as probability distributions. When learning a new task, a replay dataset is formed by sampling examples from previous tasks, where the size of the datasets is estimated as WD between new and previous task training data distributions. The number of previous examples to replay is determined based on their similarity with the task at hand. Moreover, it introduces the Instruction Information Metric (InsInfo) to measure the instructions' quality and diversity and guide the replay process to use high-quality data. Evaluation of 16 distinct tasks (classification, coding, sentiment analysis, QA, comprehension, dialogue and generation among others) demonstrates consistent performance improvements. However, it requires high-quality instructions since fuzzy instructions can affect the task similarity calculation and decrease its performance.

The study [115] examined how pre-trained language models (PLMs) tackle out-of-distribution (OOD) data in CL scenarios. In particular, the model without fine-tuning has poor generalization to OOD data while performing strongly on in-distribution (ID) data. Thus, it must learn from a data stream containing new, unseen examples over time. It fine-tuned two transformer-based architectures (GPT-2 decoder with 110 million parameters and RoBERTa encoder with 135 million parameters) on two downstream tasks (predicting API usage and API call). The datasets with API usage sequences were collected from GitHub as Java programs managing one of the associated APIs (10 million samples as ID to pre-train the models and 147,000 samples as OOD to fine-tune the models continually). Reply-based and regularization-based CL techniques mitigated CF to some extent while maintaining or slightly improving PLMs' performance in downstream tasks. The work builds on a carefully selected ID/OOD scenario and cannot guarantee generalizability on novel, unseen examples.

The study in [116] introduces a CoLeCLIP, a method for Open-Domain CL (ODCL) in VLMs that enables them to learn from a sequence of tasks with non-overlapping classes without forgetting previously learned knowledge. CoLeCLIP uses a class vocabulary and a prompt-based approach to learn task-specific patterns and refine class embeddings while mitigating forgetting through replay-free cross-domain vocabulary learning. It simultaneously learns task prompts and class embeddings in CLIP's text and image encoders [59]. The model outperformed state-of-the-art methods concerning classification accuracy and forgetting on 11 datasets for open-domain CL in both CIL and task-incremental learning (TIL) scenarios. However, real-world ODCL can use very different datasets representing incoming tasks, which may exhibit significant domain gaps and large variations in class correlations and data distributions.

Rehearsal-based methods use past training data, which may be unavailable or nonexistent at inference time. Self-Synthesized Rehearsal (SSR) [117] utilizes the LLM to produce synthetic rehearsal data. It finetunes the base LLM to create synthetic examples through I-CL with few-shot examples. Next, it employs the latest LLM to refine outputs using the synthetic

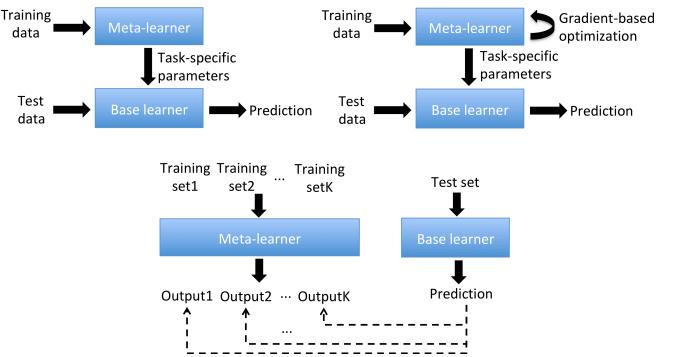
instances. Finally, it selects diverse, high-quality synthetic instances augmented with currently available data for future rehearsals. SSR employs LLaMA-2-7B, LLaMA-2-7B-Chat [109], and Alpaca-7B [118] LLMs using Super-Natural Instructions (SuperNI) [119], a comprehensive instruction tuning benchmark dataset. The results indicate that SSR is more data-efficient and achieves superior or comparable state-of-the-art performance while preventing CF. However, synthetic instances may contain unreliable content from biases in data, degrading performance on unseen tasks.

The study from [120] combined learning rate (LR) re-warm, LR re-decay, and replay techniques to compare the performance with full-data re-training. The study trained the GPT-NeoX [121] decoder-only model and compared the performance by measuring final loss and LLM evaluation benchmarks. The results show that LLMs can be successfully adapted by combining the above elements. The performance matches the re-training baseline with a compute fraction. In particular, re-warming and re-decaying the LR from a large to a small value in pre-training improves adaptation to new tasks. The study also suggests 5% replay as a default value and using more replay with significant distribution shifts. However, the experiments updated the model on only two subsequent tasks.

### B. Meta-Learning

The *meta-learning* (MeL) paradigm goes beyond existing approaches in that a model first learns how to learn (meta-training phase) and then learns new tasks efficiently (meta-testing phase) [122]. It is usually concerned with efficient model adaptation to new tasks quickly, using few examples, and generalization across these tasks [123, 124]. MeL is critical when data is scarce, difficult to obtain, or its distribution constantly changes. Currently, it is analyzed by looking at how NN-based architectures learn reusable representations using a small number of training examples (aka few-shot learning) [122]. This way, tasks share specific structures, enabling models to transfer knowledge across tasks, including image recognition and sentiment analysis [125], machine translation [126], and dialog [127].

NN-based perspective on MeL introduces two components: a *meta-learner* that produces task-specific parameters using a smaller amount of labeled data from a task and a *base learner* that makes predictions on unlabeled test data from the same task [122]. The MeL components and approaches are shown in Figure 3.



**Fig. 3.** Approaches to Meta-Learning: black-box methods (top left), optimization-based methods (top right), and distance metric learning methods (bottom).

Depending on the implementation of the above components, MeL methods can be categorized as [122, 123]:

- 1) *Black-box* methods — A black-box neural network takes the entire training dataset and predicts task-specific parameters, which are then used to parameterize the base network and make predictions for test data. The Simple Neural AttentIve Learner (SNAIL) is a typical example [128]. While these methods are versatile regarding tasks and learning problems, the meta-learner architectures are complex with high computational and data requirements.
- 2) *Optimization-based* methods — the meta-learner is an efficient optimization process (e.g., gradient descent (GD)). It obtains a set of meta-parameters that are easy to learn via GD and fine-tunes them on new tasks. In other words, it learns task-specific parameters from training data and then updates the initial set of meta-parameters by optimizing the performance on the test set of the same task. Model-Agnostic Meta-Learning (MAML) technique works in the described way [129]. The meta-parameters can be inner optimizers, NN architectures, and other network hyperparameters [130, 131]. Nonetheless, backpropagating through multiple gradient adaptation steps is computationally expensive.
- 3) *Distance metric learning* methods — while the previous two methods use NNs as parametric base models, this method employs parametric meta-learners to produce non-parametric learners (base models). The methods learn an appropriate distance metric (e.g., Nearest Neighbors) for comparing instances in a low-dimensional space, enabling the model to learn an adequate embedding space for classification tasks. Prototypical Networks take the average of a class's examples' embeddings as a prototype and classify unseen examples based on their distances to the prototypes [132]. During the meta-test, each example in the training set is compared with the test instance to determine whether they belong to the same class [133]. These methods may not capture complex relationships between data points.

MeL methods are typically combined with other techniques to extract and represent knowledge from LLMs. We provide an overview of some practical examples, along with their limitations, as follows.

Other approaches to learning across multiple tasks by parameter sharing include *Multitask Learning* (MTL) and *Transfer Learning* (TL). MTL aims to improve task performance by learning them simultaneously [134]. One approach to MTL is *hard parameter* sharing, where the model parameters are split into shared and task-specific parameters [135]. Another approach is *soft parameter* sharing, which measures parameter similarity across task-specific models using regularization penalties [136]. However, choosing the appropriate approach, designing the model architecture, and determining the extent of parameter sharing across tasks depend on the problem being solved [134]. TL fine-tunes a pre-trained model on a new task using less training data [109]. The model leverages representations learned from previous

task(s) when solving new tasks. The standard transfer learning approach is fine-tuning, which starts from a pre-trained model whose parameters are then fine-tuned on the training data from the target task using GD or other optimizer functions [137]. However, fine-tuning can destroy initialized features from the pre-training phase [138]. Preventative measures include a lower learning rate for earlier layers, freezing earlier layers, and gradually unfreezing or re-initializing the last layer [139]. Fine-tuning may be ineffective when the target task dataset is insufficiently large [140].

MeL is used in some complex learning scenarios. For example, learning from diverse multimodal task distributions sampled from a task distribution with multiple unknown modes [141]; distributed learning without sharing data across client models as personalized federated learning [142]; learning that adapts to data distribution shifts between the training and testing for more effective domain adaptation-generalization tradeoff [143]; and unsupervised meta-learning using unlabeled data [144].

MeL faces some fundamental challenges. Generalization in which tasks in meta-testing are from a different distribution seen at the meta-training time remains difficult to achieve [122, 123]. Meta-training on multiple data modalities occurs by constructing separate meta-learners focused on tasks from one modality, as different modalities have different dimensionalities [145]. Learning a meta-learner that captures multiple data modalities remains unsolved. Current meta-learning methods have significant computational and memory costs [122, 123]. Thus, deeper theoretical improvements concerning sample complexity, generalization metrics, the design of optimization functions, and proper integration of domain/task-specific knowledge are necessary.

Graphs are crucial in representing real-world objects. Graph Neural Networks (GNNs) learn on graph-like representations with textual node attributes (aka text-attributed graphs) [146]. They represent graph nodes as textual embeddings, exposing limitations in interpreting causal knowledge and understanding semantics [146]. Besides, they are usually retrained to accommodate different tasks on graphs. LLMs are commonly used for graph node classification tasks by enhancing nodes' text attributes [147].

The Graph-oriented Instruction Tuning of Large Language Models for Generic Graph Mining (MuseGraph) [148] extracts compact graph descriptions using graph analysis techniques such as random walks and one-hop neighbors. It textualizes graphs with essential semantic and structural details that LLMs can process under limited input token size. These descriptions are used to produce instructions for different graph mining tasks (e.g., node classification and link prediction). In particular, by designing CoT templates [9] for selected graph tasks and prompting GPT-4 to produce a small set of CoT-based instructions. Finally, the instructions guide graph-aware instruction tuning of LLM for each target graph mining task. It employs LLaMA-7B LLM [109] with LoRA [58] for instruction tuning. The evaluation demonstrates performance improvements in graph mining tasks compared to GNN-based and LLM-based methods. However, it is verified on a limited task coverage and instruction tuning set.

The study from [149] introduces NPHardEval4V, a dynamic benchmark to understand the extent of reasoning capabilities in MLLMs. It is constructed from textual questions in NPHardEval [150] by converting them to images. The NPHardEval benchmark contains 9 different algorithmic problems. The evaluation is conducted on various MLLMs, including GPT-4V [151] Gemini 1.0 Pro [152], CogVLM 17B [153], Qwen-VL-7B [154], and Kosmos2 1.5B [155]. The study experimented with visual, text, and combined prompts. The MLLMs' reasoning performance degrades when increasing question difficulty in individual reasoning tasks. The effects of prompt design combining visual and textual inputs vary across the MLLMs, emphasizing the importance of an appropriate balance when combining visual and textual information.

### C. Parameter-Efficient Learning

Parameter-efficient tuning (PET) adjusts a subset of the pre-trained model's parameters by adding new parameters or modifying a smaller subset during the finetuning process [156, 157]. Selective adjustment of a smaller set of parameters reduces computation costs and memory consumption, uses less training data, and is less prone to overfitting. Figure 4 presents PET approaches.

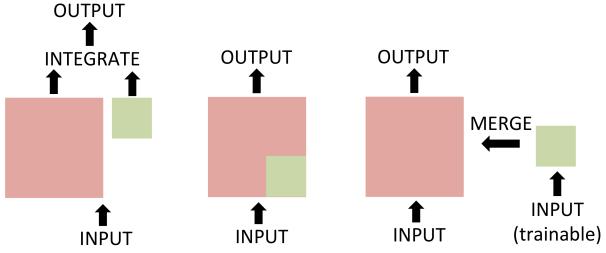


Fig. 4. Approaches to Parameter-Efficient Tuning: addition-based (left), specification-based (middle), and reparameterization-based (right). Trainable parts are in green, while the frozen parts are in red. Adapted from [157].

PET for LLMs occurs in the following principal ways [157]:

- 1) *Addition-based* methods [158] add trainable parameters while keeping the original LLM intact. Specifically, they modify the model structure by injecting trainable parameters or modules. Finetuning updates only the injected parameters, reducing memory and computational demands. Several additive methods exist.

*Adapter* approaches insert smaller adapter layers within Transformer blocks. *Serial adapters* [159] are positioned after the self-attention and FFN layers. They can reduce the model's parallelism and increase its complexity. *Parallel adapters* introduce adapter layers parallel to Transformer layers [160]. For example, CoDA [160] identifies relevant tokens processed by all layers to maintain overall model performance. The adapter only processes the 'unimportant' tokens for efficient inference without compromising overall performance.

*Soft prompts* are an alternative to adapters in that they prepend adjustable vectors (soft prompts) to input sequences [161]. Similarly, *prefix-tuning* [6] generates learnable vectors as prefixes to all Transformer keys and values. Fine-tuning optimizes the prefix vectors for inference.

2) *Specification-based* methods adjust only a portion of the parameters, such as BitFit, modifying only the model's bias terms or their subset [162]. In particular, they select and finetune a smaller parameter subset for downstream tasks. For example, Diff pruning [163] applies a binary mask by learning a task-specific difference vector, which is added to the pre-trained model parameters that remain fixed during fine-tuning. The mask identifies the finetuned parameters by selective pruning. Structured diff pruning considers dependencies between parameters to modify local groups, whereas unstructured diff pruning treats parameters independently. The former adapts better to the target model and outperforms the latter in terms of accuracy and training efficiency [163].

3) *Reparameterization-based* methods [58] transform a model's architecture by converting LLMs' adaptive parameters into parameter-efficient forms during optimization. Transforming the original model into a functionally equivalent but simplified representation optimizes inference speed. They introduce additional low-rank trainable representations for finetuning combined with the original model. The goal is an effective low-dimensional fine-tuning reparameterization. Low-Rank Adaptation (LoRA) [58] is the widely used technique introducing parameters to model the weight differences and performs better than most mainstream PET methods [156, 164]. Specifically, it constructs two trainable weight matrices. Upon fine-tuning, its weights are integrated with the pre-trained weights to maintain the inference efficiency without extra costs. For example, the PET time on the LLaMA-7B model [109] was approximately one-fourth compared to tuning all parameters [165]. The challenge is choosing an appropriate rank for trainable matrices.

4) *Mixed* methods integrate PET methods and techniques, utilizing their advantages while mitigating drawbacks. UniPELT [166] incorporates prefix-tuning, adapters, and LoRA into LLM blocks, demonstrating consistent improvements in accuracy. S4 [167] explores design spaces to combine and formulate different PET methods as design patterns. LLM-Adapters [168] builds a framework incorporating different PET methods into LLMs. The smaller LLMs utilizing PET and appropriate finetuning data show competitive performance compared to training a larger parameter set.

The PET methods, along with their affected model components and elements, as well as their trade-offs, are summarized in Table 3.

TABLE 3. PET METHODS COMPARATIVE SUMMARY.

Method	Component	Element	Trade-offs
Addition-based	Adapters Soft prompts	Parameter layers Learnable vectors	Updates injected parameters Increases model size
Specification-based	(Un)Structured Masking	Parameter subsets	Updates portion of parameters Intricate parameter selection
Reparameterization-based	Low-rank Representations	Parameter-efficient forms	Efficient parameter representation Non-trivial representation choice

PET methods are commonly part of larger CL solutions. We outline some typical examples as follows.

Continual Parameter-Efficient Tuning (ConPET) is an extension for continual LLM learning [169]. Static ConPET combines data replay with PET. Dynamic ConPET comprises lightweight, task-specific PET modules as experts tuned without changing the original LLM parameters. The results demonstrate a significant reduction in tuning costs. However, this remains to be verified using more diverse continual learning scenarios and task split strategies among experts.

The study from [170] introduces a PET framework for vision-language models for CL to alleviate CF. It expands a pre-trained CLIP model [59] by adding LoRAs [58] as sparse experts in all image and textual encoders. The Distribution Discriminative Auto-Selector (DDAS) routes inputs to the expert adapters or the pre-trained CLIP. Task-specific routers activate corresponding experts responsible for particular tasks. The framework outperforms state-of-the-art solutions on classification tasks and reduces parameter training costs. However, the number of experts increases with the number of tasks, and it does not improve the backbone CLIP.

The SSF (Scale and Shift the deep Features) method [171] scales and shifts the pre-trained model's features to learn new tasks, if upstream and downstream datasets have different data distributions. Scaled and shifted parameters fall in discriminative and unified parameter space, as variance and mean, to modulate the trainable features for downstream tasks. The method is evaluated on different pre-trained MLLM backbones, including ViT-B/16 [59] and Swin Transformer [65]. It outperforms other PET methods. However, obtaining tunable parameters by scaling and shifting original parameters needs verification in domains whose training data significantly vary from the ones used in pretraining.

Post-deployment LLM updates to deal with errors and capture changing data is known as Model Editing (ME). The MELO [69] represents a plug-in ME solution integrating LoRA modules indexed in an internal VD. It extends backbone LLM capabilities by activating specific LoRAs based on their VD indices. LoRA blocks are activated by searching inputs in the VD during inference. LoRA blocks from different layers but shared indices are active, and training updates VD clusters for prospective LoRA searches during inference. The MELO employs multiple LLM backbones for editing, including BERT, T5-Small, and T5-Large. Experiments on tasks of classifying documents, QA, and correcting hallucinations indicate state-of-the-art performance while requiring the least trainable parameters and computational cost. However, its size increases with the number of learning tasks and introduces a VD as an additional component to the backbone models.

While existing PET methods focus on updating a small number of LLM parameters, the study from [172] develops a family of Representation Finetuning (ReFT) methods updating hidden LLM representations. They learn task-specific functions, manipulating a smaller fraction of hidden representations to solve new tasks during inference. ReFT instance called Low-rank Linear Subspace ReFT (LoReFT) finds internal representations from a low-rank projection

matrix using the Distributed Alignment Search (DAS) [173]. DAS finds an optimal alignment between an interpretable high-level causal model and a low-level DL system it simplifies using GDs. LoReFT is evaluated on LLMs of different sizes, from RoBERTa 125M [174] to LLaMA 13B [109], and benchmarked against existing PET methods. It uses ten times fewer parameters than LoRA, while showing comparable performance.

The Shared Attention Framework for Parameter-efficient conTinual learning (SAPT) [175] introduces the Shared Attentive Learning & Selection module (SALS). Specifically, the SALS pre-processes training samples to find the optimal PET block combinations for completing the current task. It does so by obtaining an attention weight via instance-level shared attention operation. Inputs follow the same shared attention operation to select the appropriate PET blocks. The Attentive Reflection Module (ARM) assists SALS in recalling the shared attention operation of samples of earlier tasks that should be performed initially through generative replay with synthesized samples to prevent forgetting. The SAPT has been evaluated on the SuperNI CL NLP benchmark [119], including dialogue generation, information extraction, QA, summarization and sentiment analysis tasks, and Long Sequence [93], a CL benchmark with 15 classification tasks. It achieved superior performance compared to other PET methods with different model sizes (from 770M to 13B) and architectures, including the T5 [108] and the LLaMA-2 [109]. However, increasing the learning sequence (number of tasks) requires additional PET blocks.

#### D. Mixture of Experts

MoE architectures extend LLM capabilities by combining multiple layers or components as experts. The router mechanism determines which experts are activated for a particular task. Figure 5 shows typical MoE architecture.

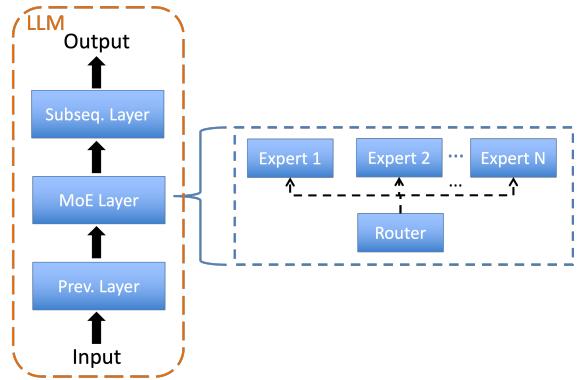


Fig. 5. Mixture of Experts in LLMs.

Experts are mainly model layers' components (such as FFNs and attention heads) [176]. Specific inference tasks are delegated to one or more expert FFNs in a respective model layer through a gating mechanism (i.e., router) for each input. Specifically, Dense MoE activates all experts from the layer, whereas Sparse MoE selects a subset of experts (e.g., top-k) [176].

The MoE approaches, along with their respective model elements and trade-offs, are outlined in Table 4.

TABLE 4. MOE APPROACHES COMPARATIVE SUMMARY.

Expert	Element	Trade-offs
LLM layer component (FFN, Attention)	Parameters Activation functions Attention heads	Straightforward task delegation More experts affect routing efficiency New tasks increase model size and computation
PET (FFN and Attention low-rank decomposition)	Efficient representations (parameter, activation, attention)	Smaller size Accuracy depends on expert selection quality (i.e., routing) Routing increases latency New tasks increase model complexity

MoEs are employed in various tasks and combined with previously described CL strategies, as illustrated by the following examples and their trade-offs.

Domain expert mixture (DEMix) [177] is a collection of domain-specialized expert FFNs. The experts are added, removed, or mixed after pretraining. However, introducing additional layers modifies existing structures in Transformer-based architectures [1] (e.g., FFN layer). Besides, it keeps all experts in memory and can face scalability issues.

The study from [178] employed instruction tuning on MoE models consisting of FFN experts. While instruction tuning on downstream tasks improves the model capabilities, it requires high-quality instruction examples and larger models.

The research in [179] combined MoE with other methods, such as PET and instruction tuning on 'lightweight' experts (updating less than 1% of an 11B parameter model). However, the parameter percentage (or expert size) varies with the pre-trained model size and the number of experts involved.

Another study trained domain-specific LoRA modules as Mixture-of-LoRAs (MoA) [180]. The modules are aligned using a predefined routing strategy. However, expert training can require labeled data at scale to specialize in a specific domain and the adaptation of the routing algorithm.

Similarly, Mixture of LoRA Experts (MoLE) [181] models experts as layers containing trained LoRAs. Moreover, it learns a gating function within each expert (layer) for efficient and flexible selection of multiple LoRAs. Although it outperforms existing LoRAs compositions, adding new LoRAs (up to 128) decreases composition performance.

The Mixture of Prompt Experts (MoPE) [182] is a conditional multimodal prompt tuning method. MoPE conducts multimodal input fusion by introducing multiple prompt experts rather than extending individual prompt lengths. It learns multiple fixed-length prompts as experts and an expert router per layer. The router selects the appropriate prompt for each input instance based on its modality-specific embeddings. Its results are comparable to fine-tuning, requiring less parameters. However, increasing the number of tasks increases the number of experts, and different routing strategies affect expert learning.

Finally, the study in [183] extended Mistral 7B LLM such that each layer comprises eight FFN blocks as experts. Two experts process the current state and combine their outputs for every token at each layer. Each token can access 47 billion

parameters but only uses 13 billion active parameters during inference. Although it demonstrates performance and efficiency gains concerning other LLMs, its capability to learn new tasks depends on the model size (number of parameters).

### III. PRINCIPAL FINDINGS

#### A. Continuous Learning

CL learns incremental tasks with dynamic data distributions in various domains, including object detection and semantic segmentation (computer vision), NLP tasks, conditional generation (generative models), and autonomous driving (perception, motion planning, motion control, and user-vehicle interaction) [2].

One of the key challenges in CL is to achieve an appropriate balance between stability and plasticity while ensuring adequate generalizability intra-task (from training to test data) and inter-task (accommodating data distribution changes) [22, 157]. Stability concerns LLM's memory as the difference between the task's current and maximum past performance (aka forgetting measure) and how learning a new task affects learned tasks (aka backward transfer). Plasticity is an LLM learning capacity, defined as the extent to which the model cannot learn new tasks or the joint influence of learned tasks on learning the current task.

The CL techniques working in embedding space (e.g., soft prompting, adapters and prefix-tuning) are more expressive and effective than prompting which operates in the discrete token space. However, the existing work focuses on learning plasticity and inter-task generalizability [20, 22, 23, 25].

The challenges of continuous learning of LLMs include:

- 1) *Multimodal Learning* — LLMs are pre-trained with enormous datasets and can pick up very complex and unexpected behaviors (e.g., hallucinations). Real-world data are not restricted to a specific modality but a random combination of multiple ones. Integrating and merging various modalities into a coherent CL workflow requires efficient modalities encoding, multi-modal databases, assigning weights during fusions, and comprehensive benchmarks, which inevitably leads to increased computational costs without general performance guarantees [184].
- 2) *Data Scaling and Quality* — CL methods require task-specific datasets using sources similar to old tasks [185]. The size and quality should adapt to obtain an appropriate trade-off between the efficiency and the effectiveness of learning incremental tasks. Inherent biases in training data can manifest in harmful or incorrect model decisions [186]. Besides, the transferability of the task- and domain-specific training data and their compositionality with other tasks and domains without extra inference cost and scalability issues in large-scale training remain unsolved. Finally, effective and efficient CL methods may differ for smaller and larger models. Exploring scaling strategies that adapt and remain effective as the model size varies is crucial due to the ever-evolving landscape of LLM architectures.
- 3) *Sustainability* — The environmental impact of training and deploying LLMs is still at stake due to enormous resources

- required. Future CL models and pipelines should be more energy-efficient and operate in low-resource environments to reduce environmental footprint and increase their accessibility [187].
- 4) *Trustworthiness* — LLMs are susceptible to adversarial attacks. Modifying inputs can produce unexpected and harmful outputs [188]. CL system design should include encryption protocols for personal data and intermediate training and inference results without compromising performance. Improving CL models' robustness against such attacks is important for safety-critical applications.
  - 5) *Few-Shot Generalization* — LLMs struggle to learn tasks from few examples or domain-specific knowledge [189]. CL should improve its performance with limited training data. These data should be carefully selected as their distribution shift can degrade performance compared to previous data or cause poor model adaptation.
  - 6) *Multilingual Learning* — Making CL more accessible and effective in underrepresented languages with limited training data [190].
  - 7) *Responsiveness* — The inference in CL should generate predictions in real time for applications such as chatbots or recommenders, where low-latency responses are critical [191]. LLM inference efficiency is affected by the model size, increased computational costs of longer inputs, the number of requests, and dependencies on external knowledge sources. Rethinking what constitutes efficiency is necessary as additional parameters and representations are computed and stored. Model compression techniques [192] (e.g., pruning and quantization) focus on the LLM's parameters representation, not their architectural elements.
  - 8) *Context Size* — CL uses a limited amount of input tokens to generate subsequent words due to LLMs' constrained context window [193]. This can cause difficulties processing lengthy sequences concerning coherence and relevance since the model can neglect or lose track of the relevant information.
  - 9) *Up-to-date Knowledge* — LLMs learn from data snapshots, and their knowledge is restricted to what is available on a particular date. Consequently, CL needs access to the most recent information. At the same time, updating LLMs with up-to-date knowledge may outdate their internal knowledge [194]. Besides, managing conflicting knowledge from one or more sources to choose the correct one is non-trivial for LLMs and CL. For example, different datasets can contain contradictory information about the same fact or carry conflicting biases and inaccuracies. Currently, external resolution strategies must assess the data's reliability.
  - 10) *Unified benchmarking* — Various benchmark tools and datasets make it difficult to reliably compare the performance of different methods and algorithms [195]. There are no guarantees that benchmark data were not used for LLM training directly or indirectly. Moreover, the performance against research benchmarks does not necessarily imply usefulness in a real-world setting. These may be small for industry standards, and data can be very different [195]. Open-ended evaluation is the biggest bottleneck to AI adoption, and current efforts focus on solutions that can be evaluated [195].
- A recent study [196] discovered that popular context-based parameter-efficient fine-tuning methods, including prompting, in-context learning, prompt tuning, and prefix-tuning, are potentially less effective than full fine-tuning, even with the same number of learnable parameters. In particular, they have structural limitations when learning new attention patterns since they cannot change the attention pattern over the content and can only drive the outputs of an attention layer in a fixed direction. Therefore, despite their success in learning tasks on the fly, they may be unable to learn novel tasks that require new attention patterns.

Table 5. compares and exemplifies previously analyzed IL approaches and their methods against the given criteria. Task-incremental support denotes the capability of learning a sequence of tasks, each with a known identity that is leveraged during training and inference. Parameter update volume is the fraction of the parameters that are updated. Real-time adaptability refers to the ability to adapt promptly to unseen data or tasks. Historical data dependency indicates the extent to which methods utilize past training examples or representations when learning new tasks.

TABLE 5. IL APPROACHES COMPARATIVE SUMMARY.

Approach	Method	Task-incremental support	Parameter update volume	Historical data dependency	Real-time adaptability	Example
Continual Learning	Consolidation-based	Medium	< 5% – 10%	Low	Low	PLoRA [55], FSCIL [60], IVLOD [64]
	Dynamic-architecture-based	High		Medium	Medium	MiniGPT-4 [84], SEQ [98], SLM [106]
	Memory-based	High	~ 10% – 100%	High	High	DiffClass [111], InsCL [114], SSR [117]
Meta-Learning	Black-box	Low	< 5%	Low	High	SNAIL [128]
	Optimization-based	High	~ 100%	High	Medium	MAML [129]
	Distance metric learning	Medium – High	< 10%	Medium	Medium – High	Prototypical Networks [132]
Parameter-Efficient Learning	Addition-based	Medium	~ 0,1% – 3%	Medium – High	Medium – High	CoDA [160], Soft prompts [161]
	Specification-based	Medium	~ 0,1% – 1%	Medium	Medium	BitFit [162], Diff pruning [163]
	Reparameterization-based	High		Medium	Medium – High	LoRA [58], LoReFT [172]
Mixture of Experts	LLM layer component	High	~ 0,1% – 1%	Medium – High	High	DEMIX [177], Mixtral of Experts [183]
	Low-rank component					MoA [180], MoLE [181]

### B. LLMs versus Humans

LLMs engage in an iterative communication process, generating streams of tokens. They rely on underlying probabilistic models to produce token arrays in response to token arrays. When presented with an input token string, they utilize previously calculated weights to iteratively generate and return a token stream that aligns with the input based on a certain probability threshold. This is constrained by the maximum number of tokens LLMs accept at once (context size) to generate the output. External data can be incorporated as vector embeddings and query vector databases (VDs) to augment the trained model during inference (e.g., RAG).

Mainstream LLM research focuses on improving capabilities within existing knowledge boundaries. Understanding the boundaries of their knowledge, referred to as self-knowledge, is essential. Evaluating 20 available LLMs' self-knowledge [197] by measuring their ability to identify questions they do not know or cannot answer revealed a considerable gap between their capabilities and the human ability to recognize the limitations of their knowledge. While humans can quickly acquire new skills by leveraging prior experience (or mental models) as generalists, LLMs are still specialists performing well specific tasks.

Hyperparameters (e.g., adapter size, LoRA's rank, placement of added components, optimizer choice, and number of fine-tuned layers) influence the CL performance. Performance ultimately depends on the manual tuning of hyperparameters, which requires considerable effort. Automated optimization of hyperparameters is still an issue for LLMs, lacking simpler solutions with better performance-efficiency tradeoffs [198].

Current ME approaches aim to make the existing model architectures more flexible to provide cost-efficient targeted updates without retraining the entire model. However, changing one model component can affect other components. Ensuring partial modifications do not reduce general performance or introduce algorithmic- and data-related biases is challenging. VDs are helpful as a cost-efficient information retrieval solution within LLMs' frameworks [15]. However, they are still not optimized for retrieving the information with the intended meaning, such as incremental query adjustment and free-text search using keywords in established retrieval paradigms (e.g., relational and graph databases).

LLMs can store massive amounts of real-world knowledge through intensive training. However, even with the existing IL approaches, they still cannot update their knowledge fast enough to keep pace with constant external, real-world knowledge change and emergence. Accordingly, the IL approaches described and existing solutions [199, 200] still need improvement concerning computational demands and effectiveness. Moreover, KGs mainly rely on textual graph structures to represent knowledge, whereas real-world knowledge inherently comprises different modalities. Effective graph representations for multimodal knowledge that represent and align different modalities into coherent entities remain an issue for KGs [201]. MLLMs encode and align distinct modalities in a vectorial form. This gap between multimodal KG and LLM representations hampers their

synergy.

### C. LLMs and Metacognition

LLMs can answer questions and generate useful content on a wide range of topics.

Prompting techniques guide them to generate outputs in response to specific types of inputs. However, they expose shallow reasoning and are prone to errors as their complexity increases. They fail to address complex knowledge because they do not actually know how they learned to produce outputs. LLMs generate statistical responses emerging from numerical weights calculated at training time.

Adapting LLMs to evolving knowledge is challenging LLMs. This requires comprehensive and responsible guidance to incorporate emerging knowledge in a timely manner while conforming to ethical norms to ensure safe usage.

The language modeling paradigm is based on the next token prediction, but its underlying structures and processes remain opaque. Thus, it is unclear whether current KE methods constitute meta-learning stemming from variable probability distributions of training data. Therefore, achieving meaningful and intentional KE with LLM is questionable. For example, a failure in LLM's logical deduction is known as the reversal curse [202], where if a model is trained on a pattern 'A is B,' it will not necessarily generalize in the reverse direction 'B is A.' KE in LLMs may be more effective in task-specific scenarios, where the consequences of model changes can be anticipated.

Knowledge injection usually leads to overfitting LLMs as they struggle to obtain recent factual knowledge via unsupervised fine-tuning [203]. A key challenge is the timely injection of relevant and updated knowledge in LLMs to develop user trust. Maintaining knowledge persistency in LLMs is another challenge, as their current knowledge is represented by generally nonpersistent parametric memory and derived internal representations influenced by inherent CL-CF tradeoff.

LLMs can sometimes solve reasoning problems, and they do so because of the token prediction process. In other words, reasoning problems are translated into pattern completion problems. Building trustworthy reasoning systems using LLMs means integrating them with a logic tool that implements causal mechanisms [16, 17]. Otherwise, we should be able to reverse-engineer their responses to understand an internal process that leads to the reasoning-driven response [204]. Relatedly, the amount of LLMs generalizability corresponds to the extent of token prediction, which ultimately depends on the number of trained parameters and dataset size.

## IV. CONCLUSION

Significant advancements have been made in developing specialized LLM-based systems to solve specific problems. However, the goals of generality and adaptability across multiple tasks remain an open problem. IL aims to learn new knowledge over time, transfer knowledge across tasks, and efficiently adapt to novel domains.

The review investigated various IL approaches in LLMs that have demonstrated promising results. Nonetheless, open

problems and challenges remain, calling for further investigation. Periodic batch updates of LLMs still fail to achieve the IL in real-time. Foundational advancements in IL will enable more versatile algorithms that learn multiple tasks, generalize across domains, and continuously accumulate knowledge.

The broader contribution of the review lies in framing IL beyond merely as a technical phenomenon of LLM tuning, but as a system-level challenge that spans architectural, optimization, and knowledge representation concerns. Accordingly, we see our work as a step forward in encouraging research in this direction. By analyzing the current state of IL in LLMs and highlighting the challenges and opportunities, we aim to inspire researchers and practitioners to explore synergies between and within AI and non-AI research fields for the future progress of IL.

## V. REFERENCES

- [1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., et al., (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30.
- [2] Cui, C., Ma, Y., Cao, X., Ye, W., Zhou, Y., et al., (2024). A Survey on Multimodal Large Language Models for Autonomous Driving. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 958-979.
- [3] Raiaan, M. A. K., Mukta, M. S. H., Fatema, K., Fahad, N. M., et al., (2024). A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges. *IEEE Access*.
- [4] Wei, J., Bosma, M., Zhao, V. Y., Guu, et al., (2021). Finetuned Language Models Are Zero-shot Learners. *arXiv preprint arXiv:2109.01652*.
- [5] Lester, B., Al-Rfou, R., & Constant, N. (2021). The Power of Scale for Parameter-Efficient Prompt Tuning. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 3045-3059.
- [6] Li, X. L., & Liang, P. (2021). Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, 4582-4597.
- [7] Brown, T., Mann, B., Ryder, N., Subbiah, et al., (2020). Language Models Are Few-shot Learners. *Advances in neural information processing systems*, 33, 1877-1901.
- [8] White, J., Fu, Q., Hays, S., Sandborn, M., et al., (2023). A Prompt Pattern Catalog To Enhance Prompt Engineering With Chatgpt. *arXiv preprint arXiv:2302.11382*.
- [9] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., & Zhou, D. (2022). Chain Of Thought Prompting Elicits Reasoning In Large Language Models. *arXiv preprint arXiv:2201.11903*.
- [10] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2023). Tree of Thoughts: Deliberate Problem Solving With Large Language Models. *arXiv preprint arXiv:2305.10601*.
- [11] Ouyang, L., Wu, J., Jiang, X., Almeida, D., et al., (2022). Training Language Models To Follow Instructions With Human Feedback. *Advances In Neural Information Processing Systems*, 35, 27730-27744.
- [12] Casper, S., Davies, X., Shi, C., Gilbert, T. K., et al., (2023). Open Problems And Fundamental Limitations Of Reinforcement Learning From Human Feedback. *arXiv preprint arXiv:2307.15217*.
- [13] Lewis, P., Perez, E., Piktus, A., Petroni, F., et al., (2020). Retrieval-Augmented Generation For Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, 33, 9459-9474.
- [14] Gao, Y., Xiong, Y., Gao, X., Jia, K., et al., (2023). Retrieval-Augmented Generation For Large Language Models: A Survey. *arXiv preprint arXiv:2312.10997*.
- [15] Jing, Z., Su, Y., Han, Y., Yuan, B., et al., (2024). When Large Language Models Meet Vector Databases: A Survey. *arXiv preprint arXiv:2402.01763*.
- [16] Guan, L., Valmeekam, K., Sreedharan, S., & Kambhampati, S. (2023). Leveraging Pre-Trained Large Language Models To Construct And Utilize World Models For Model-Based Task Planning. *Advances in Neural Information Processing Systems*, 36, 79081-79094.
- [17] Kambhampati, S., Valmeekam, K., Guan, L., Stechly, K., et al., (2024). LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. *arXiv preprint arXiv:2402.01817*.
- [18] Stechly, K., Valmeekam, K., & Kambhampati, S. (2024). On the Self-Verification Limitations of Large Language Models on Reasoning and Planning Tasks. *arXiv preprint arXiv:2402.08115*.
- [19] Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., & Fu, Y. (2019). Large Scale Incremental Learning. In Proceedings of The IEEE/CVF Conference on Computer Vision and Pattern Recognition, 374-382.
- [20] Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., et al., (2019). Continual Lifelong Learning With Neural Networks: A Review. *Neural Networks*, 113, 54-71.
- [21] Ramasesh, V. V., Dyer, E., & Raghu, M. (2020). Anatomy of Catastrophic Forgetting: Hidden Representations and Task Semantics. In International Conference on Learning Representations.
- [22] Wang, L., Zhang, X., Su, H., & Zhu, J. (2024). A Comprehensive Survey Of Continual Learning: Theory, Method And Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8), 5362 – 5383.
- [23] Van de Ven, G. M., Tuytelaars, T., & Tolias, A. S. (2022). Three Types of Incremental Learning. *Nature Machine Intelligence*, 4(12), 1185-1197.
- [24] Khan, M. G. Z. A., Naeem, M. F., Van Gool, L., Stricker, D., et al., (2023). Introducing Language Guidance In Prompt-Based Continual Learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 11463-11473.
- [25] Zhou, D. W., Sun, H. L., Ning, J., Ye, H. J., & Zhan, D. C. (2024). Continual Learning with Pre-Trained Models: A Survey. *arXiv preprint arXiv:2401.16386*.
- [26] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, et al., (2017). Overcoming Catastrophic Forgetting In Neural Networks. *Proceedings Of The National Academy Of Sciences*, 114(13), 3521-3526.
- [27] Zhang, J., Zhang, J., Ghosh, S., Li, D., et al., (2020). Class-Incremental Learning Via Deep Model Consolidation. In Proceedings of the IEEE/CVF winter conference on applications of computer vision, 1131-1140.
- [28] Wang, L., Zhang, X., Li, Q., Zhang, M., et al., (2023). Incorporating Neuro-Inspired Adaptability For Continual Learning In Artificial Intelligence. *Nature Machine Intelligence*, 5(12), 1356-1368.
- [29] Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge Distillation: A Survey. *International Journal of Computer Vision*, 129(6), 1789-1819.
- [30] Liu, H., & Liu, H. (2022). Continual Learning with Recursive Gradient Optimization. In International Conference on Learning Representations.
- [31] Wu, Y., Chi, Z., Wang, Y., & Feng, S. (2023). MetaGCD: Learning To Continually Learn In Generalized Category Discovery. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 1655-1665.
- [32] Gupta, K., Thérien, B., Ibrahim, A., Richter, M. L., et al., (2023). Continual Pre-Training of Large Language Models: How to re-warm your model?. In Workshop on Efficient Systems for Foundation Models@ICML2023.
- [33] Liang, C., Jiang, H., Li, Z., Tang, X., Yin, B., & Zhao, T. (2023). HomoDistil: Homotopic Task-Agnostic Distillation of Pre-trained Transformers. In The Eleventh International Conference on Learning Representations.
- [34] Hafez, M. B., & Erekmen, K. (2024). Continual deep reinforcement learning with task-agnostic policy distillation. *Scientific Reports*, 14(1), 31661.
- [35] Chen, C., Yin, Y., Shang, L., Wang, Z., Jiang, X., Chen, X., & Liu, Q. (2021). Extract then Distill: Efficient and Effective Task-Agnostic BERT Distillation. In International Conference on Artificial Neural Networks, 570-58.
- [36] Gu, X., Liu, L., Yu, H., Li, J., Chen, C., & Han, J. (2021). On the Transformer Growth for Progressive BERT Training. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 5174-5180.
- [37] Qin, Q., Hu, W., Peng, H., Zhao, D., & Liu, B. (2021). BNS: Building Network Structures Dynamically For Continual Learning. *Advances in Neural Information Processing Systems*, 34, 20608-20620.
- [38] Mehta, N., Liang, K., Verma, V. K., & Carin, L. (2021). Continual Learning Using A Bayesian Nonparametric Dictionary Of Weight Factors. In International Conference on Artificial Intelligence and Statistics, 100-108.
- [39] Jia, M., Tang, L., Chen, B. C., Cardie, C., et al., (2022). Visual Prompt Tuning. In European Conference on Computer Vision, 709-727.

- [40]Hafez, M. B., & Wermter, S. (2023). Continual Robot Learning using Self-Supervised Task Inference. *IEEE Transactions on Cognitive and Developmental Systems*, 16(3), 947-960.
- [41]Wiwatcharakoses, C., & Berrar, D. (2021). A Self-Organizing Incremental Neural Network for Continual Supervised Learning. *Expert Systems with Applications*, 185, 115662.
- [42]Zhao, K., Xu, H., Yang, J., & Gao, K. (2022). Consistent Representation Learning for Continual Relation Extraction. In *Findings of the Association for Computational Linguistics: ACL 2022*, 3402-3411.
- [43]Boschini, M., Bonicelli, L., Buzzega, P., Porrello, A., & Calderara, S. (2022). Class-Incremental Continual Learning Into the eXtended DER-VeRse. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 45(5), 5497-5512.
- [44]Wang, L., Lei, B., Li, Q., Su, H., Zhu, J., & Zhong, Y. (2021). Triple-Memory Networks: A Brain-Inspired Method For Continual Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5), 1925-1934.
- [45]Petit, G., Popescu, A., Schindler, H., Picard, D., & Delezoide, B. (2023). Fetril: Feature Translation For Exemplar-Free Class-Incremental Learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 3911-3920.
- [46]Hafez, M. B., Immisch, T., Weber, T., & Wermter, S. (2023). Map-based experience replay: a memory-efficient solution to catastrophic forgetting in reinforcement learning. *Frontiers in Neurorobotics*, 17, 1127642.
- [47]Hayes, T. L., Cahill, N. D., & Kanan, C. (2019). Memory efficient experience replay for streaming learning. *International Conference on Robotics and Automation (ICRA)*, 9769-9776.
- [48]Zhou, D. W., Sun, H. L., Ye, H. J., & Zhan, D. C. (2024). Expandable Subspace Ensemble for Pre-Trained Model-Based Class-Incremental Learning. *arXiv preprint arXiv:2403.12030*.
- [49]Liang, J., Wang, Z., Ma, Z., Li, J., et al., (2024). Online Training of Large Language Models: Learn while Chatting. *arXiv preprint arXiv:2403.04790*.
- [50]Shi, H., Xu, Z., Wang, H., Qin, W., et al., (2025). Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*.
- [51]Wu, T., Luo, L., Li, Y. F., Pan, S., et al., (2024). Continual Learning for Large Language Models: A Survey. *arXiv preprint arXiv:2402.01364*.
- [52]Jin, X., Zhang, D., Zhu, H., Xiao, W., et al., (2022). Lifelong Pretraining: Continually Adapting Language Models to Emerging Corpora. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4764-4780.
- [53]Zhang, Z., Fang, M., Chen, L., & Rad, M. R. N. (2023). CITB: A Benchmark for Continual Instruction Tuning. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.
- [54]Zhang, H., Gui, L., Zhai, Y., Wang, H., et al., (2024). COPR: Continual Learning Human Preference Through Optimal Policy Fitting. *arXiv preprint arXiv:2310.15694*.
- [55]Guo, H., Zhu, F., Liu, W., Zhang, X. Y., et al., (2024). Federated Class-Incremental Learning with Prototype Guided Transformer. *arXiv preprint arXiv:2401.02094*.
- [56]Dong, J., Wang, L., Fang, Z., Sun, G., et al. (2022). Federated Class-Incremental Learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10164-10173.
- [57]Dong, J., Li, H., Cong, Y., Sun, G., et al., (2023). One Left Behind: Real-World Federated Class-Incremental Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [58]Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., et al., (2021). LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.
- [59]Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., et al., (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.
- [60]Tian, S., Li, L., Li, W., Ran, H., et al., (2024). A Survey on Few-Shot Class-Incremental Learning. *Neural Networks*, 169, 307-324.
- [61]Liu, C., Wang, Z., Xiong, T., Chen, R., et al., (2024). Few-Shot Class Incremental Learning with Attention-Aware Self-Adaptive Prompt. *arXiv preprint arXiv:2403.09857*.
- [62]Wang, Z., Qu, X., Xiao, J., Chen, B., et al., (2024). INCprompt: Task-Aware incremental Prompting for Rehearsal-Free Class-incremental Learning. *arXiv preprint arXiv:2401.11667*.
- [63]Hendrycks, D., Basart, S., Mu, N., Kadavath, S., et al., (2021). The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, 8340-8349.
- [64]Deng, J., Zhang, H., Ding, K., Hu, J., et al., (2024). Zero-shot Generalizable Incremental Learning for Vision-Language Object Detection. *arXiv preprint arXiv:2403.01680*.
- [65]Liu, Z., Lin, Y., Cao, Y., Hu, H., et al., (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012-10022.
- [66]Liu, Y., Schiele, B., Vedaldi, A., & Rupprecht, C. (2023). Continual Detection Transformer for Incremental Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 23799-23808.
- [67]Dong, N., Zhang, Y., Ding, M., & Lee, G. H. (2023). Incremental-DETR: Incremental Few-Shot Object Detection via Self-Supervised Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(1), 543-551.
- [68]Zhang, N., Yao, Y., Tian, B., Wang, P., et al., (2024). A Comprehensive Study of Knowledge Editing for Large Language Models. *arXiv preprint arXiv:2401.01286*.
- [69]Yu, L., Chen, Q., Zhou, J., & He, L. (2024). MELO: Enhancing Model Editing With Neuron-Indexed Dynamic Lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 19449-19457.
- [70]Zhong, Z., Wu, Z., Manning, C. D., Potts, C., et al., (2023). MQUAKE: Assessing Knowledge Editing in Language Models via Multi-Hop Questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 15686-15702.
- [71]Mitchell, E., Lin, C., Bosselut, A., Finn, C., & Manning, C. D. (2022). Fast Model Editing at Scale. In *International Conference on Learning Representations*.
- [72]Ma, J. Y., Gu, J. C., Ling, Z. H., Liu, Q., & Liu, C. (2023). Untying the Reversal Curse Via Bidirectional Language Model Editing. *arXiv preprint arXiv:2310.10322*.
- [73]Pinter, Y., & Elhadad, M. (2023). Emptying the Ocean with a Spoon: Should We Edit Models?. In *the 2023 Conference on Empirical Methods in Natural Language Processing*.
- [74]Zhu, D., Sun, Z., Li, Z., Shen, T., et al., (2024). Model Tailor: Mitigating Catastrophic Forgetting in Multi-modal Large Language Models. *arXiv preprint arXiv:2402.12048*.
- [75]Dai, W., Li, J., Li, D., Tiong, A. M. H., et al., (2023). InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning. *Advances in Neural Information Processing Systems*, 36.
- [76]Liu, H., Li, C., Li, Y., & Lee, Y. J. (2023). Improved Baselines with Visual Instruction Tuning. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.
- [77]Jovanović, M., & Campbell, M. (2023). Connecting AI: Merging Large Language Models and Knowledge Graph. *Computer*, 56(11), 103-108.
- [78]Garcez, A. D. A., & Lamb, L. C. (2023). Neurosymbolic AI: The 3rd Wave. *Artificial Intelligence Review*, 56(11), 12387-12406.
- [79]Wang, X., Gao, T., Zhu, Z., Zhang, Z., Liu, Z., Li, J., & Tang, J. (2021). KEPLER: A Unified Model For Knowledge Embedding And Pre-Trained Language Representation. *Transactions of the Association for Computational Linguistics*, 9, 176-194.
- [80]Sheth, A., Roy, K., & Gaur, M. (2023). Neurosymbolic Artificial Intelligence (Why, What, And How). *IEEE Intelligent Systems*, 38(3), 56-62.
- [81]Zhang, H., Gui, L., Zhai, Y., Wang, H., et al., (2024). COPR: Continual Learning Human Preference through Optimal Policy Fitting. *arXiv preprint arXiv:2310.15694*.
- [82]Gao, Y., Zeng, Z., Du, D., Cao, S., et al., (2024). SeerAttention: Learning Intrinsic Sparse Attention in Your LLMs. *arXiv preprint arXiv:2410.13276*.
- [83]Li, C. A., & Lee, H. Y. (2024). Examining Forgetting in Continual Pre-training of Aligned Large Language Models. *arXiv preprint arXiv:2401.03129*.
- [84]Zhu, D., Chen, J., Shen, X., Li, X., & Elhoseiny, M. (2023). MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models. In *The Twelfth International Conference on Learning Representations*.
- [85]Fang, Y., Wang, W., Xie, B., Sun, Q., et al., (2023). EVA: Exploring the Limits of Masked Visual Representation Learning at Scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19358-19369.
- [86]Chiang, W. L., Li, Z., Lin, Z., Sheng, Y., et al., (2023). Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%\* ChatGPT Quality.

- [87] Cao, X., Lu, H., Huang, L., Liu, X., et al., (2024). Generative Multi-modal Models are Good Class-Incremental Learners. arXiv preprint arXiv:2403.18383.
- [88] Li, J., Li, D., Savarese, S., & Hoi, S. (2023). BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In International Conference on Machine Learning, 19730-19742.
- [89] Qi, B., Chen, X., Gao, J., Liu, J., et al., (2024). Interactive Continual Learning: Fast and Slow Thinking. arXiv preprint arXiv:2403.02628.
- [90] Kumaran, D., Hassabis, D., & McClelland, J. L. (2016). What Learning Systems do Intelligent Agents Need? Complementary Learning Systems Theory Updated. Trends in Cognitive Sciences, 20(7), 512-534.
- [91] Scialom, T., Chakrabarty, T., & Muresan, S. (2022). Fine-Tuned Language Models Are Continual Learners. arXiv preprint arXiv:2205.12393.
- [92] Sanh, V., Webson, A., Raffel, C., Bach, S. H., et al., (2021). Multitask Prompted Training Enables Zero-Shot Task Generalization. arXiv preprint arXiv:2110.08207.
- [93] Razdaibiedina, A., Mao, Y., Hou, R., Khabsa, M., et al., (2023). Progressive Prompts: Continual Learning for Language Models. In The Eleventh International Conference on Learning Representations.
- [94] Gao, T., Yao, X., & Chen, D. (2021). SimCSE: Simple Contrastive Learning of Sentence Embeddings. In Conference on Empirical Methods in Natural Language Processing, 6894-6910.
- [95] Li, J., Zhou, P., Xiong, C., & Hoi, S. (2021). Prototypical Contrastive Learning of Unsupervised Representations. In International Conference on Learning Representations.
- [96] Thengane, V., Khan, S., Hayat, M., & Khan, F. (2022). CLIP Model Is An Efficient Continual Learner. arXiv preprint arXiv:2210.03114.
- [97] Zheng, Z., Ma, M., Wang, K., Qin, Z., et al., (2023). Preventing Zero-Shot Transfer Degradation In Continual Learning Of Vision-Language Models. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 19125-19136.
- [98] Zheng, J., Qiu, S., & Ma, Q. (2023). Learn or Recall? Revisiting Incremental Learning with Pre-trained Language Models. arXiv preprint arXiv:2312.07887.
- [99] Zhou, D. W., Wang, Q. W., Qi, Z. H., Ye, H. J., Zhan, D. C., & Liu, Z. (2023). Deep Class-incremental Learning: A Survey. arXiv preprint arXiv:2302.03648.
- [100] Zhou, D. W., Zhang, Y., Ning, J., Ye, H. J., et al., (2023). Learning Without Forgetting for Vision-Language Models. arXiv preprint arXiv:2305.19270.
- [101] Jin, H., Han, X., Yang, J., Jiang, Z., et al., (2024). LLM Maybe LongLM: SelfExtend LLM Context Window Without Tuning. arXiv preprint arXiv:2401.01325.
- [102] Touvron, H., Martin, L., Stone, K., Albert, P., et al., (2023). LLaMA 2: Open Foundation and Fine-Tuned Chat Models. arXiv preprint arXiv:2307.09288.
- [103] Javaheripi, M., Bubeck, S., Abdin, M., Aneja, J., et al., (2023). Phi-2: The Surprising Power of Small Language Models. Microsoft Research Blog.
- [104] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., et al., (2023). Mistral 7B. arXiv preprint arXiv:2310.0682.
- [105] Kim, D., Park, C., Kim, S., Lee, W., et al., (2023). SOLAR 10.7B: Scaling Large Language Models with Simple yet Effective Depth Up-Scaling. arXiv preprint arXiv:2312.15166.
- [106] Peng, B., Tian, Z., Liu, S., Yang, M., et al., (2024). Scalable Language Model with Generalized Continual Learning. arXiv preprint arXiv:2404.07470.
- [107] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: 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, 4171-4186.
- [108] Raffel, C., Shazeer, N., Roberts, A., Lee, K., et al., (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research, 21(140), 1-67.
- [109] Touvron, H., Lavril, T., Izacard, G., Martinet, X., et al., (2023). LLaMA: Open and Efficient Foundation Language Models. arXiv preprint arXiv:2302.13971.
- [110] Xia, Y., Huang, H., Fang, M., & Zhao, Z. (2025). Continual Cross-Modal Generalization. arXiv preprint arXiv:2504.00561.
- [111] Meng, Z., Zhang, J., Yang, C., Zhan, Z., et al., (2024). DiffClass: Diffusion-Based Class Incremental Learning. arXiv preprint arXiv:2403.05016.
- [112] Nichol, A. Q., & Dhariwal, P. (2021). Improved Denoising Diffusion Probabilistic Models. In International Conference on Machine Learning, 8162-8171.
- [113] Han, J., Na, J., & Hwang, W. (2024). Semantic Prompting with Image-Tokens for Continual Learning. arXiv preprint arXiv:2403.11537.
- [114] Wang, Y., Liu, Y., Shi, C., Li, H., et al., (2024). InsCL: A Data-efficient Continual Learning Paradigm for Fine-tuning Large Language Models with Instructions. arXiv preprint arXiv:2403.11435.
- [115] Weyssow, M., Zhou, X., Kim, K., Lo, D., & Sahraoui, H. (2023). On The Usage Of Continual Learning For Out-of-distribution Generalization In Pre-trained Language Models Of Code. In Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 1470-1482.
- [116] Li, Y., Pang, G., Suo, W., Jing, C., et al., (2024). CoLeCLIP: Open-Domain Continual Learning via Joint Task Prompt and Vocabulary Learning. arXiv preprint arXiv:2403.10245.
- [117] Huang, J., Cui, L., Wang, A., Yang, C., et al., (2024). Mitigating Catastrophic Forgetting in Large Language Models with Self-Synthesized Rehearsal. arXiv preprint arXiv:2403.01244.
- [118] Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., et al., (2023). Stanford Alpaca: An Instruction-following LLaMA Model.
- [119] Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., et al., (2022). Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, 5085-5109.
- [120] Ibrahim, A., Thérien, B., Gupta, K., Richter, M. L., et al., (2024). Simple and Scalable Strategies to Continually Pre-train Large Language Models. arXiv preprint arXiv:2403.08763.
- [121] Andonian, A., Anthony, Q., Biderman, S., Black, S., et al., (2021). GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch.
- [122] Vettoruzzo, A., Bouguila, M. R., Vanschoren, J., Rognvaldsson, T., & Santosh, K. C. (2024). Advances and Challenges in Meta-Learning: A Technical Review. IEEE Transactions on Pattern Analysis and Machine Intelligence.
- [123] Hospedales, T., Antoniou, A., Micaelli, P., & Storkey, A. (2022). Meta-Learning in Neural Networks: A Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(9), 5149-5169.
- [124] FiWang, Y., Ji, Y., Wang, W., & Wang, B. (2024). Bi-channel Attention Meta Learning for Few-shot Fine-grained Image Recognition. Expert Systems with Applications, 242, 122741.
- [125] Liang, B., Li, X., Gui, L., Fu, Y., He, Y., Yang, M., & Xu, R. (2023). Few-shot Aspect Category Sentiment Analysis Via Meta-learning. ACM Transactions on Information Systems, 41(1), 1-31.
- [126] Sherborne, T., & Lapata, M. (2023). Meta-learning a Cross-lingual Manifold for Semantic Parsing. Transactions of the Association for Computational Linguistics, 11, 49-67.
- [127] Lee, H. Y., Li, S. W., & Vu, T. (2022). Meta Learning for Natural Language Processing: A Survey. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 666-684.
- [128] Mishra, N., Rohaninejad, M., Chen, X., & Abbeel, P. (2018). A Simple Neural Attentive Meta-Learner. In International Conference on Learning Representations
- [129] Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In International Conference on Machine Learning, 1126-1135.
- [130] Xu, S., Li, Y., Lin, M., Gao, P., et al., (2023). Q-DETR: An Efficient Low-Bit Quantized Detection Transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 3842-3851.
- [131] Qin, X., Song, X., & Jiang, S. (2023). Bi-Level Meta-Learning For Few-Shot Domain Generalization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 15900-15910.
- [132] Laenen, S., & Bertinetto, L. (2021). On Episodes, Prototypical Networks, and Few-shot Learning. Advances in Neural Information Processing Systems, 34, 24581-24592.
- [133] Zhao, Y., Zhang, T., Li, J., & Tian, Y. (2023). Dual Adaptive Representation Alignment for Cross-Domain Few-Shot Learning. IEEE Transactions on Pattern Analysis and Machine Intelligence.
- [134] Zhang, Y., & Yang, Q. (2021). A Survey on Multi-Task Learning. IEEE Transactions on Knowledge and Data Engineering, 34(12), 5586-5609.
- [135] Wang, J., Zheng, Y., Ma, J., Li, X., et al., (2023). Information Bottleneck-Based Interpretable Multitask Network For Breast Cancer Classification And Segmentation. Medical Image Analysis, 83, 102687.

- [136] Rathnayake, H., Sumanapala, J., Rukshani, R., & Ranathunga, S. (2024). AdapterFusion-Based Multi-Task Learning for Code-Mixed and Code-Switched Text Classification. *Engineering Applications of Artificial Intelligence*, 127, 107239.
- [137] Chowdhery, A., Narang, S., Devlin, J., et al., (2023). PaLM: Scaling Language Modeling with Pathways. *Journal of Machine Learning Research*, 24(240), 1-113.
- [138] Kumar, A., Raghunathan, A., Jones, R. M., Ma, T., et al., (2022). Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution. In *International Conference on Learning Representations*.
- [139] Lee, Y., Chen, A. S., Tajwar, F., et al., (2023). Surgical Fine-Tuning Improves Adaptation to Distribution Shifts. In *The Eleventh International Conference on Learning Representations*.
- [140] Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 328-339.
- [141] Jiang, W., Kwok, T. Y., & Zhang, Y. (2022). Subspace Learning for Effective Meta-Learning. *Proceedings of Machine Learning Research*.
- [142] Yang, L., Huang, J., Lin, W., & Cao, J. (2023). Personalized Federated Learning On Non-Iid Data Via Group-Based Meta-Learning. *ACM Transactions on Knowledge Discovery from Data*, 17(4), 1-20.
- [143] Yao, H., Wang, Y., Li, S., Zhang, L., et al., (2022). Improving Out-Of-Distribution Robustness via Selective Augmentation. In *International Conference on Machine Learning*, 25407-25437.
- [144] Lee, D. B., Lee, S., Kawaguchi, K., Kim, Y., et al., (2023). Self-Supervised Set Representation Learning for Unsupervised Meta-Learning. In *The Eleventh International Conference on Learning Representations*.
- [145] Alayrac, J. B., Donahue, J., Luc, P., Miech, A., et al., (2022). Flamingo: A Visual Language Model for Few-Shot Learning. *Advances in Neural Information Processing Systems*, 35, 23716-23736.
- [146] Wu, Z., Pan, S., Chen, F., Long, G., et al., (2020). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1), 4-24.
- [147] Chen, Z., Mao, H., Li, H., Jin, W., et al., (2024). Exploring the Potential of Large Language Models (LLMs) in Learning on Graphs. *ACM SIGKDD Explorations Newsletter*, 25(2), 42-61.
- [148] Tan, Y., Lv, H., Huang, X., Zhang, J., et al., (2024). MuseGraph: Graph-oriented Instruction Tuning of Large Language Models for Generic Graph Mining. *arXiv preprint arXiv:2403.04780*.
- [149] Fan, L., Hua, W., Li, X., Zhu, K., et al., (2024). NPHardEval4V: A Dynamic Reasoning Benchmark of Multimodal Large Language Models. *arXiv preprint arXiv:2403.01777*.
- [150] Fan, L., Hua, W., Li, L., Ling, H., et al., (2023). NPHardEval: Dynamic Benchmark on Reasoning Ability of Large Language Models via Complexity Classes. *arXiv preprint arXiv:2312.14890*.
- [151] OpenAI. GPT-4V(ision) System Card, (2023).
- [152] Team, G., Anil, R., Borgeaud, S., Wu, Y., et al., (2024). Gemini: A Family of Highly Capable Multimodal Models *arXiv preprint arXiv:2312.11805*.
- [153] Wang, W., Lv, Q., Yu, W., Hong, W., et al., (2024). CogVLM: Visual Expert for Pretrained Language Models. *arXiv preprint arXiv:2311.03079*.
- [154] Bai, J., Bai, S., Yang, S., Wang, S., et al., (2023). Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond. *arXiv preprint arXiv:2308.12966*.
- [155] Peng, Z., Wang, W., Dong, L., Hao, Y., Huang, S., Ma, S., & Wei, F. (2023). Kosmos-2: Grounding Multimodal Large Language Models to the World. *arXiv preprint arXiv:2306.14824*.
- [156] Ding, N., Qin, Y., Yang, G., Wei, F., et al., (2022). Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.
- [157] Han, Z., Gao, C., Liu, J., & Zhang, S. Q. (2024). Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey. *arXiv preprint arXiv:2403.14608*.
- [158] Gao, T., Fisch, A., & Chen, D. (2021). Making Pre-trained Language Models Better Few-shot Learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 3816-3830.
- [159] Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., et al., (2022). AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, 487-503.
- [160] Lei, T., Bai, J., Brahma, S., Ainslie, J., et al., (2023). Conditional Adapters: Parameter-efficient Transfer Learning with Fast Inference. In *37th Conference on Neural Information Processing Systems*.
- [161] Petrov, A., Torr, P., & Bibi, A. (2024). When Do Prompting and Prefix-Tuning Work? A Theory of Capabilities and Limitations. In *The Twelfth International Conference on Learning Representations*.
- [162] Zaken, E. B., Ravfogel, S., & Goldberg, Y. (2021). BitFit: Simple Parameter-Efficient Fine-Tuning for Transformer-Based Masked Language-Models. *arXiv preprint arXiv:2106.10199*.
- [163] Guo, D., Rush, A. M., & Kim, Y. (2021). Parameter-Efficient Transfer Learning with Diff Pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 4884-4896.
- [164] Liu, S. Y., Wang, C. Y., Yin, H., Molchanov, P., et al., (2024). DoRA: Weight-Decomposed Low-Rank Adaptation. *arXiv preprint arXiv:2402.09353*.
- [165] Sun, X., Ji, Y., Ma, B., & Li, X. (2023). A Comparative Study Between Full-Parameter and Lora-Based Fine-Tuning on Chinese Instruction Data for Instruction Following Large Language Model. *arXiv preprint arXiv:2304.08109*.
- [166] Mao, Y., Mathias, L., Hou, R., Almahairi, A., et al., (2022). UniPELT: A Unified Framework for Parameter-Efficient Language Model Tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 6253-6264.
- [167] Chen, J., Zhang, A., Shi, X., Li, M., et al., (2023). Parameter-Efficient Fine-Tuning Design Spaces. In *The Eleventh International Conference on Learning Representations*.
- [168] Hu, Z., Wang, L., Lan, Y., Xu, W., et al., (2023). LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.
- [169] Song, C., Han, X., Zeng, Z., Li, K., et al., (2023). ConPET: Continual Parameter-efficient Tuning for Large Language Models. *arXiv preprint arXiv:2309.14763*.
- [170] Yu, J., Zhuge, Y., Zhang, L., Wang, D., et al., (2024). Boosting Continual Learning of Vision-Language Models via Mixture-of-Experts Adapters. *arXiv preprint arXiv:2403.11549*.
- [171] Lian, D., Zhou, D., Feng, J., & Wang, X. (2022). Scaling & Shifting Your Features: A New Baseline for Efficient Model Tuning. *Advances in Neural Information Processing Systems*, 35, 109-123.
- [172] Wu, Z., Arora, A., Wang, Z., Geiger, et al., (2024). ReFT: Representation Finetuning for Language Models. *arXiv preprint arXiv:2404.03592*.
- [173] Geiger, A., Wu, Z., Potts, C., Icard, T., & Goodman, N. (2024). Finding Alignments Between Interpretable Causal Variables and Distributed Neural Representations. In *Causal Learning and Reasoning*, 160-187.
- [174] Liu, Y., Ott, M., Goyal, N., Du, J., et al., (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.
- [175] Zhao, W., Wang, S., Hu, Y., Zhao, Y., et al., (2024). SAPT: A Shared Attention Framework for Parameter-Efficient Continual Learning of Large Language Models. *arXiv preprint arXiv:2401.08295*.
- [176] Cai, W., Jiang, J., Wang, F., Tang, et al., (2025). A Survey on Mixture of Experts in Large Language Models. *IEEE Transactions on Knowledge and Data Engineering*, 37(7), 3896-3915.
- [177] Gururangan, S., Lewis, M., Holtzman, A., Smith, N. A., & Zettlemoyer, L. (2022). DEMIX Layers: Disentangling Domains for Modular Language Modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 5557-5576.
- [178] Shen, S., Hou, L., Zhou, Y., Du, et al., (2023). Mixture-Of-Experts Meets Instruction Tuning: A Winning Combination for Large Language Models. *arXiv preprint arXiv:2305.14705*.
- [179] Zadouri, T., Üstün, A., Ahmadian, A., Ermiş, B., et al., (2023). Pushing Mixture of Experts to the Limit: Extremely Parameter Efficient MoE for Instruction Tuning. *arXiv preprint arXiv:2309.05444*.
- [180] Feng, W., Hao, C., Zhang, Y., Han, Y., & Wang, H. (2024). Mixture-of-LoRAs: An Efficient Multitask Tuning for Large Language Models. *arXiv preprint arXiv:2403.03432*.
- [181] Wu, X., Huang, S., & Wei, F. (2024). Mixture of LoRA Experts. In *The Twelfth International Conference on Learning Representations*.
- [182] Jiang, R., Liu, L., & Chen, C. (2024). MoPE: Parameter-Efficient and Scalable Multimodal Fusion via Mixture of Prompt Experts. *arXiv preprint arXiv:2403.10568*.

- [183] Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., et al., (2024). Mixtral of Experts. arXiv preprint arXiv:2401.04088.
- [184] Cao, X., Lu, H., Huang, L., Liu, X., & Cheng, M. M. (2024). Generative Multi-modal Models are Good Class-Incremental Learners. arXiv preprint arXiv:2403.18383.
- [185] Harun, M. Y., Gallardo, J., Hayes, T. L., & Kanan, C. (2023). How Efficient Are Today's Continual Learning Algorithms?. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2430-2435.
- [186] Truong, T. D., Nguyen, H. Q., Raj, B., & Luu, K. (2024). Fairness Continual Learning Approach to Semantic Scene Understanding in Open-World Environments. Advances in Neural Information Processing Systems, 36.
- [187] Yıldız, Ç., Ravichandran, N. K., Punia, P., Bethge, M., et al., (2024). Investigating Continual Pretraining in Large Language Models: Insights and Implications. arXiv preprint arXiv:2402.17400.
- [188] Zou, A., Wang, Z., Kolter, J. Z., & Fredrikson, M. (2023). Universal and Transferable Adversarial Attacks on Aligned Language Models. arXiv preprint arXiv:2307.15043.
- [189] Song, Y., Wang, T., Cai, P., Mondal, S. K., & Sahoo, J. P. (2023). A Comprehensive Survey of Few-Shot Learning: Evolution, Applications, Challenges, and Opportunities. ACM Computing Surveys, 55(13), 1-40.
- [190] Cahyawijaya, S., Lovenia, H., & Fung, P. (2024). LLMs Are Few-Shot In-Context Low-Resource Language Learners. arXiv preprint arXiv:2403.16512.
- [191] Liu, Z., Wang, J., Dao, T., Zhou, T., et al., (2023). Deja Vu: Contextual Sparsity for Efficient LLMs at Inference Time. In International Conference on Machine Learning, 22137-22176.
- [192] Zhu, X., Li, J., Liu, Y., Ma, C., & Wang, W. (2023). A Survey on Model Compression for Large Language Models. arXiv preprint arXiv:2308.07633.
- [193] Jin, H., Han, X., Yang, J., Jiang, Z., et al., (2024). LLM Maybe LongLM: Self-extend LLM Context Window Without Tuning. arXiv preprint arXiv:2401.01325.
- [194] Onoe, Y., Zhang, M., Choi, E., & Durrett, G. (2022). Entity Cloze By Date: What LMs Know About Unseen Entities. In Findings of the Association for Computational Linguistics, 693-702.
- [195] Jovanović, M., & Campbell, M. (2025). Benchmarking AI: Towards Inclusive Evaluation of Language Models. Computer, 58(8), 153-159.
- [196] Petrov, A., Torr, P., & Bibi, A. (2024). When Do Prompting and Prefix-Tuning Work? A Theory of Capabilities and Limitations. In The Twelfth International Conference on Learning Representations.
- [197] Yin, Z., Sun, Q., Guo, Q., Wu, J., Qiu, X., & Huang, X. J. (2023, July). Do Large Language Models Know What They Don't Know?. In Findings of the Association for Computational Linguistics: ACL 2023, 8653-8665.
- [198] Zhou, H., Wan, X., Vulić, I., & Korhonen, A. (2024). AUTOPEFT: Automatic Configuration Search for Parameter-Efficient Fine-Tuning. arXiv preprint arXiv:2301.12132.
- [199] Li, Z., Zhang, N., Yao, Y., Wang, M., et al., (2024). Unveiling the Pitfalls of Knowledge Editing for Large Language Models. In The Twelfth International Conference on Learning Representations.
- [200] Cohen, R., Biran, E., Yoran, O., Globerson, A., & Geva, M. (2024). Evaluating The Ripple Effects of Knowledge Editing in Language Models. Transactions of the Association for Computational Linguistics, 12, 283-298.
- [201] Pan, S., Luo, L., Wang, Y., Chen, C., et al., (2024). Unifying Large Language Models and Knowledge Graphs: A Roadmap. IEEE Transactions on Knowledge and Data Engineering, 36(7), 3580 – 3599.
- [202] Berglund, L., Tong, M., Kaufmann, M., Balesni, M., et al., (2023). The Reversals Curse: LLMs trained on “A is B” fail to learn “B is A”. In The Twelfth International Conference on Learning Representations.
- [203] Ovadia, O., Brief, M., Mishaeli, M., & Elisha, O. (2023). Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLM. arXiv preprint arXiv:2312.05934.
- [204] Shanahan, M. (2024). Talking about Large Language Models. Communications of the ACM, 67(2), 68-79.



**Mladjan Jovanovic** is an associate professor of Computer Science at Singidunum University, Belgrade, Serbia. He is a visiting scientist at the University of Trento, Italy, and Vienna University of Technology, Austria. He has over 19 years of research and industrial experience in designing, developing, and testing intelligent interactive computing systems. His primary interests include language modeling and knowledge technologies.



**Peter Voss** is the CEO and Chief Scientist at AIGO.ai, Austin, Texas, USA. He is one of the founders of the term AGI (Artificial General Intelligence) in 2002. Since then, he has been steadily developing, commercializing, and improving the technology geared towards human-level AI. His main interests are cognitive architectures based on a deep understanding of the requirements of human intelligence.