Learn From Model Beyond Fine-Tuning: A Survey

Hongling Zheng, Li Shen, Anke Tang, Yong Luo, Han Hu, Bo Du, Dacheng Tao *Fellow, IEEE*

Abstract—Foundation models (FM) have demonstrated remarkable performance across a wide range of tasks (especially in the fields of natural language processing and computer vision), primarily attributed to their ability to comprehend instructions and access extensive, high-quality data. This not only showcases their current effectiveness but also sets a promising trajectory towards the development of artificial general intelligence. Unfortunately, due to multiple constraints, the raw data of the model used for large model training are often inaccessible, so the use of end-to-end models for downstream tasks has become a new research trend, which we call ***Learn From Model (LFM)*** in this article. LFM focuses on the research, modification, and design of FM based on the model interface, so as to better understand the model structure and weights (in a black box environment), and to generalize the model to downstream tasks. The study of LFM techniques can be broadly categorized into five major areas: model tuning, model distillation, model reuse, meta learning and model editing. Each category encompasses a repertoire of methods and strategies that aim to enhance the capabilities and performance of FM. This paper gives a comprehensive review of the current methods based on FM from the perspective of LFM, in order to help readers better understand the current research status and ideas. To conclude, we summarize the survey by highlighting several critical areas for future exploration and addressing open issues that require further attention from the research community. The relevant papers we investigated in this article can be accessed at https://github.com/ruthless-man/Awesome-Learn-from-Model.

Index Terms—Learn from model, Foundation model, Fine-tuning, Knowledge distillation

1 Introduction

The rapid advancement of algorithms and computing power has sparked significant development and interest in large-scale pre-training models across both industry and academia. These models, such as GPT-3 [1], LLAMA [2], and Imagen [3], leverage the power of over-parameterized transformers to effectively model natural language in a variety of ways. This infrastructure enables these models to handle large-scale language and vision tasks and exhibit impressive performance across a wide spectrum of downstream applications. The continuous growth and refinement of FM indicate a promising future for natural language processing and related fields.

Standing on the shoulders of giants, research based on FM are demonstrating unprecedented vigor and innovation in the current landscape [4], [5], [6], [7], [8]. This gives rise to a burgeoning research paradigm, which we term as ***Learn From Model (LFM)*** in this paper. As shown in Figure 2, LFM refers to the study of foundation models to understand the model's behavior, strengths, and possible shortcomings. This kind of research can help us to better optimize the performance of the model, find and fix the problems of the model, and ultimately put the model into production and work better.

There are numerous reasons to learn from the model itself rather than from the data used to train the model. From a data privacy perspective, large models are typically trained on vast amounts of data, which may contain sensitive information such as personal identity information and private communications. Directly studying this data could infringe on user privacy [9], [10]. However, by studying the model itself, we can avoid direct contact with this sensitive data. At the same time, the data used to train large models involve commercial interests. These data, as a crucial part of the competitive advantage of the model owners, are usually not disclosed [11]. Therefore, ordinary users can usually only access the API input interface and the final output of the model, which further promotes the motivation to learn from the model itself. From the perspective of model generalization ability, by studying the model itself, we can better understand the model generalization ability, that is, how the model handles new data it has not seen during training [12], [13], [14]. Both approaches, learn from data and learn from model, have their strengths and applications [15]. Learn from data is fundamental in situations where abundant labeled or unlabeled data is available, while learn from model is useful when existing models can provide valuable insights, speed up training, or enhance prediction accuracy. These approaches often complement each other in practice, enabling researchers and practitioners to build innovative and high-performing machine learning solutions.

In general, there are several advantages to learn from model: From a data perspective, due to factors such as data privacy and storage costs, it is difficult for smes and even individual users to obtain data to train high-quality models. The existing basic model highly compresses the characteristic information of large-scale raw data, and the requirement of raw data can be reduced as much as possible based on LFM paradigm. From a computing standpoint, as the parameters of the base model scale from 175B parameters of GPT-3 to 1.8T parameters of GPT-4, the computational power requirement of the retraining model increases

- *Hongling Zheng, Anke Tang, Yong Luo, and Bo Du are with the School of Computer Science, Wuhan University, China and Xiaomi - Wuhan University Joint Laboratory of Artificial Intelligence. E-mail: {hlzheng, anketang, luoyong, dubo}@whu.edu.cn*
- *Li Shen is with JD Explore Academy, China. E-mail: mathshenli@gmail.com*
- *Han Hu is with Beijing Institute of Technology, China. E-mail: hhu@bit.edu.cn*
- *Dacheng Tao is with the University of Sydney, Australia. E-mail: dacheng.tao@gmail.com*

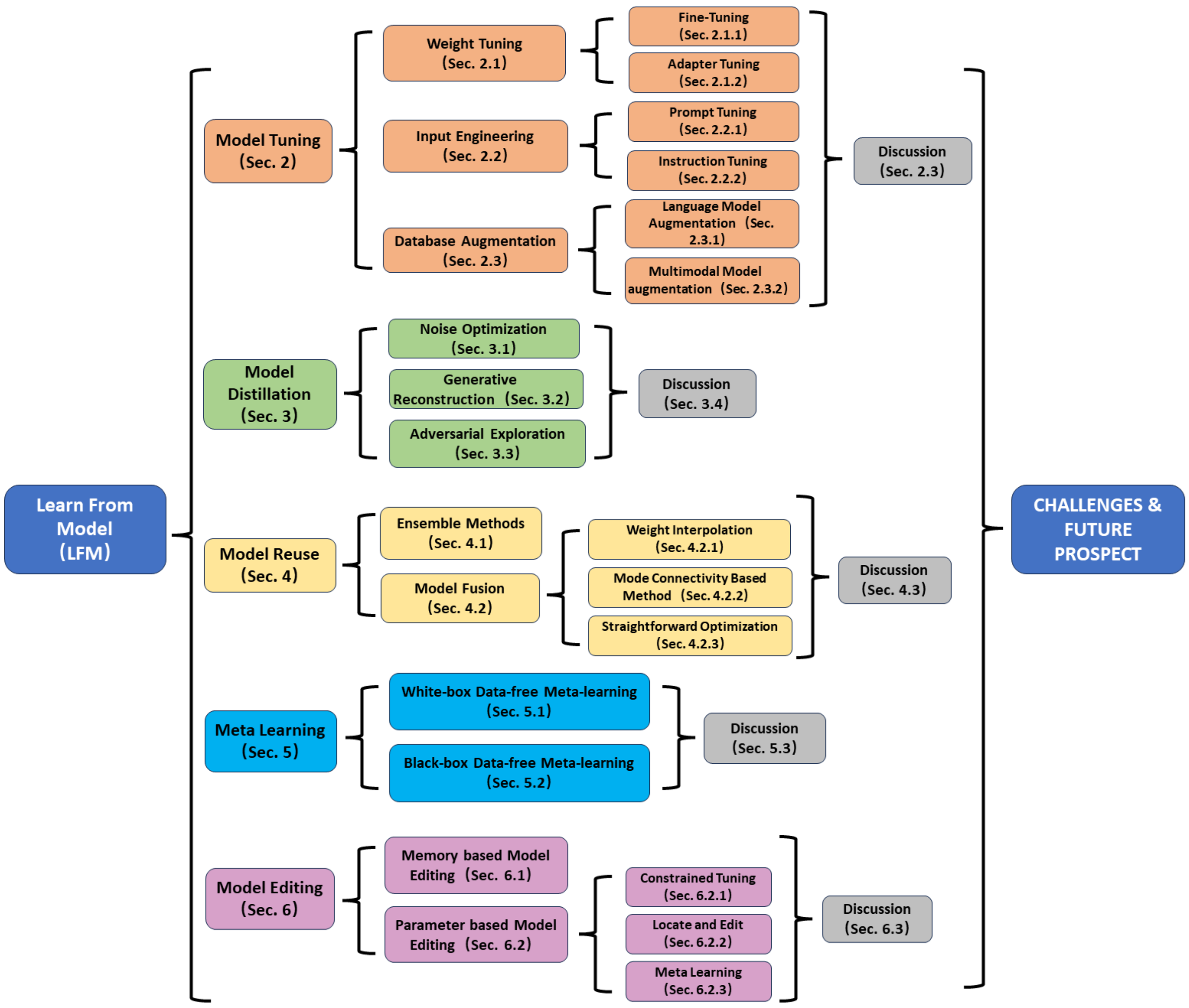

Learn From Model (LFM)
Model Tuning (Sec. 2)
Weight Tuning (Sec. 2.1)
Fine-Tuning (Sec. 2.1.1)
Adapter Tuning (Sec. 2.1.2)
Input Engineering (Sec. 2.2)
Prompt Tuning (Sec. 2.2.1)
Instruction Tuning (Sec. 2.2.2)
Database Augmentation (Sec. 2.3)
Language Model Augmentation (Sec. 2.3.1)
Multimodal Model augmentation (Sec. 2.3.2)
Discussion (Sec. 2.3)
Model Distillation (Sec. 3)
Noise Optimization (Sec. 3.1)
Generative Reconstruction (Sec. 3.2)
Adversarial Exploration (Sec. 3.3)
Discussion (Sec. 3.4)
Model Reuse (Sec. 4)
Ensemble Methods (Sec. 4.1)
Model Fusion (Sec. 4.2)
Weight Interpolation (Sec. 4.2.1)
Mode Connectivity Based Method (Sec. 4.2.2)
Straightforward Optimization (Sec. 4.2.3)
Discussion (Sec. 4.3)
Meta Learning (Sec. 5)
White-box Data-free Meta-learning (Sec. 5.1)
Black-box Data-free Meta-learning (Sec. 5.2)
Discussion (Sec. 5.3)
Model Editing (Sec. 6)
Memory based Model Editing (Sec. 6.1)
Parameter based Model Editing (Sec. 6.2)
Constrained Tuning (Sec. 6.2.1)
Locate and Edit (Sec. 6.2.2)
Meta Learning (Sec. 6.2.3)
Discussion (Sec. 6.3)
CHALLENGES & FUTURE PROSPECT

Fig. 1: The structural taxonomy for LFM. The survey is organized according to the hierarchical structure.

exponentially [16]. The paradigm of the base model combined with downstream task tuning has become an alternative, with significant computational cost savings. From the perspective of knowledge representation, the Emergent Abilities [17] displayed by the large model are able to show extraordinary performance in general purpose tasks, and the mechanism behind this is difficult to obtain using the combination of small models. At the same time, the self-monitoring capability of the basic model can provide impetus and theoretical basis for downstream applications. Finally, from the perspective of the industry, FM APIs with carefully adjusted parameters have emerged one after another, and SOTA performance on new tasks based on high-quality FM has become a recognized paradigm. With the advent of the large model era, we can foresee that LFM will replace the traditional paradigm of learning from data and become a new way of thinking.

This paper categorizes the paradigms of FM according to their application scenarios, including model tuning, model distillation, model fusion, model reuse, meta learning and model editing. In the last part of the body of the survey, we analyze and provide a forecast for the future applications of FM.

The main job of model tuning [18] is to design the FM (modify internal parameters, add additional components) to achieve efficient performance on downstream tasks. This approach allows fast adaptation of the model to new tasks and avoids the large computational cost of training the model from scratch. The above techniques are all examples of LFM. Model distillation [8] transfers knowledge from a large model (teacher model) to a small model (student model) that can operate in a resource-constrained environment while maintaining similar performance to the large model. Model Reuse [19] learns strengths from models, combining predictions from multiple models to improve overall performance. Meta-learning [20], also known as ”learning to learn from model,” as a way to design models so that they quickly adapt to new tasks, is similarly brought under the scope of LFM. With the rapid

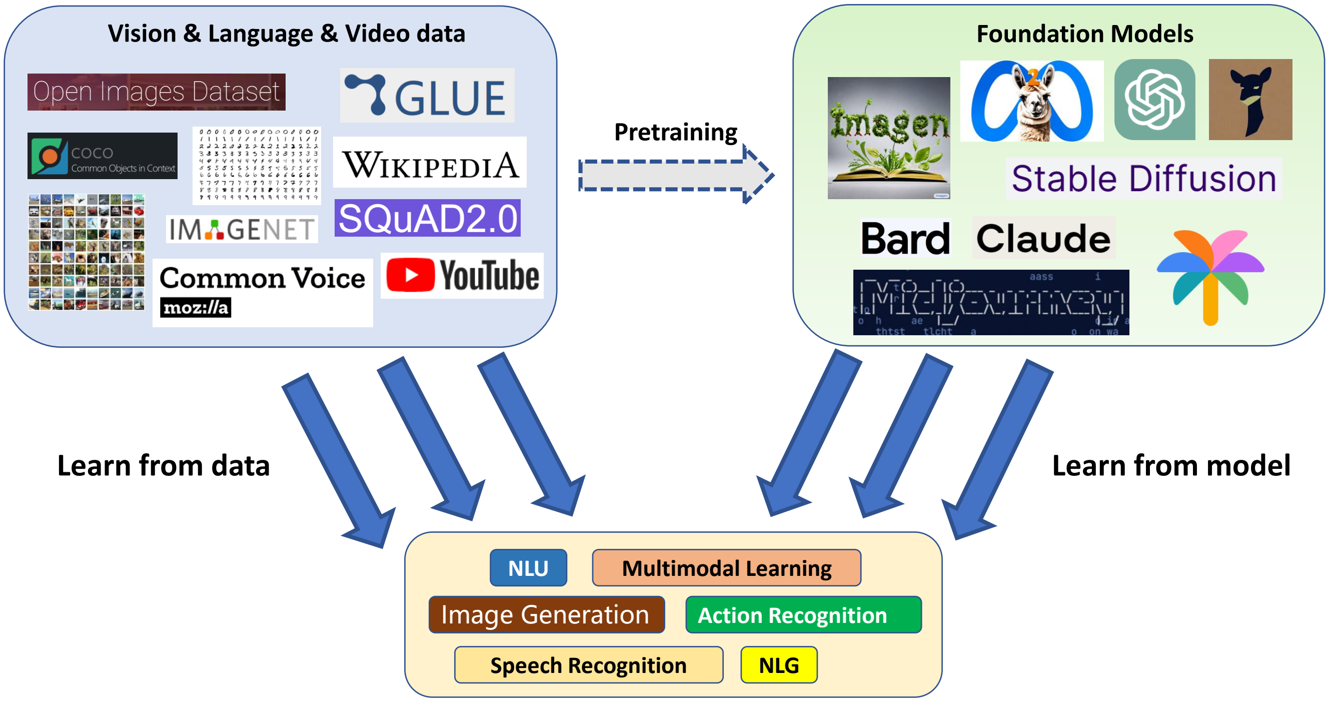


Fig. 2: **Two fundamental approaches in machine learning**. The "learn from data" and "learn from model" are two fundamental approaches that play a vital role in training models and improving their performance. Both approaches have their strengths and applications. **LFD**: The "learn from data" approach involves training models by using large amounts of labeled or unlabeled data. This approach relies on the premise that patterns and relationships within the data can be learned by the model to make accurate predictions. **LDM**: The "learn from model" approach involves leveraging the knowledge and insights gained from existing FM to improve model performance. Rather than starting from scratch with raw data, this approach utilizes FM as a foundation and builds upon them.

change of the real world, the knowledge inside the FM inevitably has the problem of insufficient accuracy. Model editing [21] solves the defect of backward model knowledge by directly adjusting the model behavior, avoiding the huge resource consumption caused by retraining the model.

Our contributions can be summarized into these folds:

- Based on a deep understanding of current research, we first summarize and propose the concept of LFM, which aims to outline research based on FM, liberating mindsets from traditional paradigm of learning from data.
- Compared with existing surveys, we provide a more systematic survey of LFM. Our survey includes specific classifications and cutting-edge analysis of LFM methods, as well as corresponding application trends, providing a more comprehensive overview of the field.
- Our study serves as a stepping stone for the scientific community, shedding light on the diverse opportunities and challenges that lie ahead in the pursuit of learning from large pre-trained models. We hope that our work will foster a deeper understanding of the LFM paradigm, catalyzing breakthroughs that will ultimately benefit both industry and academia in the years to come.

The whole article is structured as follows: We first introduced the definition and related classification of LFM technology in Section 1, described the model tuning in Section 2, reviewed the model distillation technology in section 3, reviewed the model reuse in section 4, introduced the application of meta-learning in the LFM paradigm in section 5, and introduced the model editing in sections 6 The future direction of LFM is discussed in Section 7. Finally, the thesis is summarized in Section 8.

2 Model Tuning

As one of the paradigms of LFM, the main goal of model tuning is to design the parameters of the FM for migration to downstream tasks. Compared with retraining model based on data for new tasks (learn from data), the idea of model tuning based on transfer learning reduces the hidden dangers of insufficient data sets and high training costs, and makes use of the common sense knowledge stored by the FM itself to provide convenience for parameter initialization for new tasks. The success of model tuning requires an in-depth study of the internal structure and dynamics of the pretrained model, understanding how the FM encode input data and identifying specific components that strongly influence the predicted results. On the other hand, according to the modification degree and modification position of model parameters, model tuning can be further divided into weight tuning, input engineering and database augmentation. Weight tuning is based on the internal parameters of FM, input engineering designs better suggestions from the input level, and database augmentation updates model knowledge based on external databases to ensure model accuracy.

2.1 Weight Tuning

2.1.1 Fine-Tuning

With the dramatic changes in model structure in recent years, methods such as fine-tuning have gradually replaced supervised

Literature	Summary	Classification
Big Self-Supervised Models [22]	Semi-supervised learning for large visual datasets.	Fine-Tuning
AIFT [23]	Try to integrate active learning and transfer learning into one framework.	Fine-Tuning
GPT-1 [24]	A generative model that is pre-trained on an unsupervised corpus and fine-tuned on a supervised dataset.	Fine-Tuning
R4F [25]	An approach based on trust domain theory is proposed to replace adversarial targets with parametric noise, thereby reducing representation changes during fine-tuning without affecting performance.	Fine-Tuning
Lightweight Adapter Tuning [26]	Adapter adjustment for multilingual neural machine translation.	Adapter Tuning
MHR [27]	New methods such as Multi-Head Routing (MHR) and MHR-μ are introduced to enhance presentation capabilities and optimize multitasking adaptation.	Adapter Tuning
ADAMIX [28]	A parametric efficient method is proposed to increase adapter capacity in large pre-trained language models.	Adapter Tuning
LLM-Adapters [29]	Various adapters can be integrated into the LLM and these adapter-based LLM PEFT methods can be executed for different tasks.	Adapter Tuning
Black Box Adversarial Prompting for Foundation Models [30]	A black box framework for generating adversarial prompts for unstructured images and text generation is developed.	Prompt Tuning
Prefix-Tuning [31]	A black box framework for generating adversarial prompts for unstructured images and text generation is developed.	Prompt Tuning
Progressive Prompts [32]	A simple and effective method for continuous language model learning.	Prompt Tuning
RLPrompt [33]	Optimizing Discrete Text Prompts with Reinforcement Learning.	Prompt Tuning
GRAM [34]	A novel gradient-regulated meta-cue learning framework is introduced by using only unlabeled image-text pre-training data.	Prompt Tuning
BBTv2 [35]	The gradient-free adaptive of pre-trained models (PTMs) is enhanced by optimizing continuous prompts without accessing model parameters.	Prompt Tuning
MULTIINSTRUCT [36]	The first multimodal instruction tuning baseline dataset.	Instruction Tuning
LINGUIST [37]	Instruction fine-tuning of large-scale seq2seq models to control output generated from multilingual intents and slot markup data.	Instruction Tuning
LLaVA [38]	End-to-end trained large multimodal model that connects a vision encoder and LLM for general-purpose visual and language understanding.	Instruction Tuning
GPT4RoI [39]	Visual features extracted by the spatial instruction and the language embedding are input to LLM, and trained on the data in instruction tuning format.	Instruction Tuning.

TABLE 1: Model Tuning Literature

learning [40] as the main standard. Fine-tuning is a transfer learning technique for using the knowledge of pre-trained neural networks to solve new, relevant tasks. In the field of natural language processing (NLP), fine-tuning is often used to adjust large pre-trained language models (e.g. BERT, GPT, etc.) to specific tasks such as text classification, sentiment analysis, named entity recognition, etc. This approach can significantly reduce training time and improve the model's performance on new tasks [41]. Here we define the fine-tuning for updating all parameters of the model. By utilizing the knowledge of the pre-trained model, fine-tuning can significantly improve the model's ability to generalize on new tasks.

The idea of pre-training large models was well established before deep neural network modeling became the paradigm [42]. Nima [43] in 2016 investigated the effects of fine-tuning on the effectiveness of deep convolutional neural networks in the context of medical image analysis, demonstrating the robustness of layered fine-tuning. GPT-1 [24] first introduced fine-tuning ideas into the design of large language models, proving the effectiveness of this approach in a wide range of benchmarks for natural language understanding. BitFit [44] optimizes the bias parameters in pre-trained transformers, effectively improving performance on all evaluated GLUE tasks. This approach streamlines deployment and enhances hardware efficiency by leveraging pre-trained weights for the majority of computations, only modifying a minor fraction during inference. WiSE-FT [45] improves the robustness of fine-tuned pre-trained models by ensembling the weights of the zero-shot and fine-tuned models. This method not only maintains high accuracy on a given target distribution, but also significantly boosts the model's performance under distribution shifts.

Although there have been improvements and studies on traditional fine-tuning model methods[25], [46], [47], there are still inevitable major flaws in principle. To be specific, fine-tuning essentially requires that we can fully explore and master the internal parameters and structure of the model. On this basis, reverse gradient propagation is carried out to adjust a large number of model parameters, reuse the model, and generalize to a new task. As a result, there are several limitations in a real LFM scenario: **Computational Resource Requirement**: Full-parameter fine-tuning requires substantial computational resources because it involves optimizing all the parameters of the model. This could be problematic in a small-scale or resource-constrained environment [48]. There is a notable risk of **Overfitting Risk**: In scenarios where the amount of training data is limited, full-parameter fine-tuning can lead to overfitting, where the model may perform well on the training data but not generalize effectively to new, invisible data [49]. Another critical issue is **Catastrophic Forgetting**: Fine-tuning may cause the model to forget the pretraining knowledge it initially learned when it is trained on a new task. This phenomenon is also known as ”catastrophic forgetting” [50], [32]. Concerns about **Model Stability Issues** are also worthy of attention: Fine-tuning could disrupt the stability of the model. The patterns learned by the pretrained model on a large corpus may be distorted during the fine-tuning process, leading to a decrease in the quality of the model's generated outputs [51].

2.1.2 Adapter Tuning

Adapter tuning maintains most of the model parameters fixed and inserts new trainable parameters (referred to as ”adapters”) in between the internal layers of the model, specifically for the purpose of fine-tuning on specific tasks. This strategy was proposed by Houlsby [52] in 2019, with the aim of enhancing the fine-tuning performance of models while reducing computational costs and the risk of overfitting. Adapters are inserted layer by layer to make the least modification to the original model structure and don not require retraining a large number of parameters, hence significantly reducing the computational cost in the fine-tuning process. Specifically, one adapter module contains a down-projection and an up-projection. For an input feature h, a down-projection projects the input to a r-dimensional space with a parameter matrix W_d, after which a nonlinear function f is applied. Then the up-projection W_u maps the r-dimensional representation back to d-dimensional space. Added with a residual connection, the complete computation could be written as:

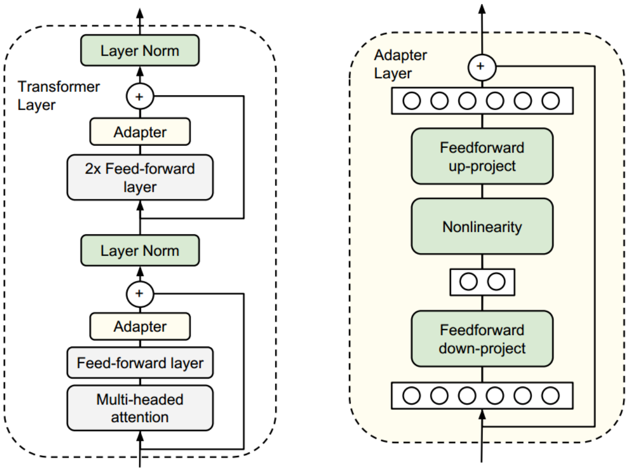


Fig. 3: The basic structure of Adapter Tuning [52].

$$\hat{h} = f(h * W_d) * W_u + h.$$

Adapters can be trained for multiple tasks and then plugged into the pre-trained model to perform new tasks [53], as shown in Figure 3.

Based on the Adapter architecture, AdapterFusion [54] realizes the maximum task migration between multiple adapter modules by separating the two stages of knowledge extraction and knowledge combination. He [55] provides an in-depth study on the effectiveness of adapter-based tuning. Not only does adapter-based tuning mitigate forgetting issues by maintaining closer adherence to the initial pretrained language model's representations, it also demonstrates superior performance on low-resource and cross-lingual tasks, exhibiting greater robustness to overfitting and changes in learning rates compared to standard fine-tuning. Although an adapter works with much fewer tunable parameters than vanilla fine-tuning, some work attempts a more rigorous saving strategy by introducing inductive biases into the structure of the adapter layer. Rabeeh [56] inserts the task-specific weight matrix into the FM's weight, which is efficiently calculated as the sum of the Kronecker product between the shared slow ”weights” and the fast ”first-ranked matrices defined for each compter layer. In multilingual speech translation (ST) tasks, Le [26] effectively integrated adapter modules for further specialisation of a fully-trained multilingual ST model for each language pair. AdaMix [28] introduces a parameter-efficient fine-tuning (PEFT) approach for large pre-trained language models. This method enhances performance on downstream tasks by leveraging a mixture of adaptation modules. Despite tuning only 0.1-0.2% of parameters, AdaMix outperforms full model fine-tuning and existing PEFT methods.

Adapter tuning maintains performance comparable to full-parameter fine-tuning while reducing computational cost. Additionally, due to the fewer parameters of adapters, the risk of overfitting is also relatively reduced. Another significant advantage of adapters is that, as they are separately added to the model, they can be easily shared or transferred among different tasks, enabling multi-task learning or transfer learning with the model [57]. On the other hand, Adapter-based approaches require adding relevant parameters to the pre-trained model for downstream tasks. Although they improve the training efficiency of the model, they also lead to the problem of inference delay. When the model is deployed in real applications, the speed drop will be very noticeable [58].

2.2 Input Engineering

2.2.1 Prompt Tuning

Downstream fine-tuning induces a high-cost burden by modifying all parameters. To alleviate the tuning burden and maintain the performance, researchers froze large pre-trained models and try to optimize the prompts. Brown [1] was the first to demonstrate the remarkable ability of prompt engineering to adapt large pre-trained models to downstream tasks. Comprehensive results [59], [60] also confirm the optimal prompts could induce comparable ability with fine-tuning models. In this part, we describe how to learn better prompts from the fixed FM. Concretely, we separate this application into two scenarios: 1) White-Box Prompt Tuning (i.e., having access to gradients and parameters); 2) Black-Box Prompt Tuning (i.e., having no access to gradients and parameters).

White-Box Prompt Tuning: White-Boxing tuning built on emerging considerable open access models in early stages of FM development. Li [61] proposed prefix-tuning, prehending upstream task-specific vectors to steer a downstream model. In this way, we could only store a small amount of parameters to achieve model custom on downstream tasks. Furthermore, Lester [60] used the special token embedding as soft prompt initialization, which need not prehend parameters in front of each layer. Qin [62] proposed to learn a mixture of soft prompts conditioned on fixed FM. Liu [63] proposed to construct inputs with anchor words and trainable vectors. They proved task-related anchor words could bring further improvement. PTR [64] composed prompts with manual templates and virtual tokens, then tuned with rules. Gu [59] applied the pre-training on soft prompt to obtain a better initialization. Liu [40] provide a detail survey of white-box tuning development. Jiang [65] rethought the past classic model tuning methods include prefix tuning, prompt tuning and adapter tuning, then unified them in a parallel form called U-Tuning. U-Tuning can incorporate off-the-shelf tuning methods and derive a framework for parameter efficient tuning. Wen [66] applied the gradient-based discrete prompt optimization on multi-modal generation models, which can automatically design hard prompt for CLIP model.

Black-Box Prompt Tuning: Currently, considerably powerful large pre-trained models are deployed on the cloud, such as ChatGPT and GPT-3 [1]. The users have no access to the parameters and gradients of models. This black-box setting secures the model owner from potential attack and misuse. As for commercial consideration, black-box service will become mainstream. However, querying cloud service through hand-crafted prompts cannot fully exploit data in many use cases. The urgent works mainly focus on black-box prompt tuning.

To this end, Sun [67] invokes derivative-free optimization on continuous prompt tuning. Concretely, they use a projection matrix to optimize the subspace of the original prompt. Then prompts are sampled and updated from the multivariate normal distribution, called BBT. And later, Sun [35] further proposed BBTV2, constructing a divide-and-conquer method to prepend and optimize layer-specific prompts. Furthermore, Diao [68] design a policy gradient inspired framework characterizing the problem as discrete tokens selection problem. Specifically, they conduct a variance-reduced policy gradient algorithm to estimate the gradient of the categorical distribution. However, user needs to upload their to fine-tuned the cloud model, which damage the privacy. Xiao [69] formulated an emulator simulating cloud model and an adapter for fine-tuning downstream tasks. And model owner only transfer the emulator and adapter to user for user-end fine-tuning, called off-site tuning. Hou [70] paired prompts with corresponding output distribution into a large of weak learners. Then they use a gradient-free method, ADABOOST algorithm, to refine a pool of prompts. RLPROMPT [33] optimized the discrete prompts with reinforcement learning, and the reward comes from black-box models.

2.2.2 Instruction Tuning

Prior works have shown that instruction tuning, the technique of fine-tuning FM on a set of NLP tasks formatted with instructions, further enhances the ability of the model to perform invisible tasks from instructions [38], [71], [72]. Instruction tuning refers to the process of deep training of FM on a data set consisting of (instruction, output) pairs in a supervised manner. Different from prompt tuning, if prompt tuning is to guide the model to generate relevant content through prompts, then instruction tuning is to train the model to perform specific tasks through instructions.

FLAN [73] takes a pre-trained language model with 137B parameters and adjusts the instructions on more than 60 NLP datasets expressed through natural language instruction templates and evaluates them on invisible task types. T0 [74] increases the number of tasks and prompts based on FLAN, proving that implicit multi-task learning can improve model generalization and zero-shot ability. InstructGPT [75] uses a reinforcement learning (RLHF) technique via human feedback to rank multiple outputs of the model based on the results of user and API interactions, and then uses this data to fine-tune GPT-3 so that the InstructGPT model is better at following instructions than GPT-3. Peng [72] used GPT-4 to generate command follow data for LLM tuning, and experiments showed that the 52K English and Chinese command follow data generated by GPT-4 had better zero fire performance on the new mission than the command follow data generated by the previous state-of-the-art model. Jang [76] in multi-task hinted fine-tuning found that the distributed approach of training a separate expert LM for each training task has many advantages, including (1) avoiding the negative task transfer that often occurs during instruction tuning, (2) being able to continuously learn new tasks without having to retrain previous tasks to avoid catastrophic forgetting, and (3) demonstrate combinatorial capabilities when merging individual experts.

MULTIINSTRUCT [36] datasets provide the basis for multimodal instruction tuning. Liu [38] explored the design of instruction tuning in the field of multimodality, where they first attempted to use language-only GPT-4 to generate multimodal language image instruction follow data. Instruction tuning of this generated data enabled end-to-end training of a large multimodal model, LLaVA, capable of connecting visual encoders and LLMS for general visual and language understanding. Similar work is being studied in the [77]. LLaVA network architecture is shown in Figure 4.

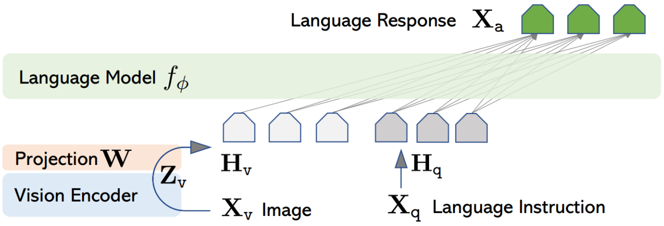


Fig. 4: LLaVA network architecture [38]

The improvements still needed for instruction tuning are (1) increasing the number of instruction tasks and model size to improve the performance and zero-shot capabilities of the model on new tasks. (2) A more precise instruction template is required, and attention should be paid to the difference between training and inference instruction template settings. (3) Pay attention to the balance of data under different tasks, and the number of training samples under each task does not need to be too large.

2.3 Database Augmentation

In the practical application of FM (ChatGPT, LLaMA), the strategy of using external knowledge bases for database augmentation to further improve the effectiveness is gaining more and more recognition [78], [79], [80]. Database augmentation improves the performance of FM in a different way than the model editing described in Section 6, especially if the internal knowledge of the model is insufficient and lagging. The model based on database augmentation allows the introduction of external knowledge from different sources including training corpus, external data, unsupervised data and other options. The database augmentation model typically comprises a retriever and a generator. The retriever retrieves relevant knowledge from external sources based on the query, while the generator combines the query with the retrieved knowledge to make model predictions.

The benefits of this strategy are numerous: (1) It enables larger models to acquire richer knowledge, especially up-to-date information. Despite the vast knowledge capacity of the large model, it does not retain all of the knowledge, especially the new knowledge that emerges after its training. (2) Alleviating the illusion problem of FM. By providing external information, we can make the output of large models more reliable and closer to the real world. For example, when using language models to comment on events, embedding causal logic between events can make the output of the larger model more reasonable. (3) Many of today's open source FM are generic, and combined with domain-specific external knowledge bases, these models can perform better in dealing with domain-specific problems. This not only improves the efficiency of the model, but is also a low-cost strategy for applying large models. By applying retrieval augmentation strategies to FM, we can give these models a greater range of knowledge, better handle complex real-world problems, and provide more accurate and in-depth responses in specific domains.

2.3.1 Language Database Augmentation

As deep learning has proven successful, some retrieval systems have now adopted dense learned representations that are based on the activations of a neural network. Continuous cache [81] assigns probability values to tokens that have similar previous and current activation vectors, thereby expanding the model's context to include local history. The kNN-LM [82] proposed by Khandelwal in 2020, expands upon this idea and applies it to transformers. Furthermore, it broadens the retrieval database to encompass the entire English Wikipedia, which has led to significant improvements in the Wikitext103 evaluation.

As shown in Figure 5, although the concept of search augmentation has been partially investigated before, REALM [83] is the first to use ”augmentation” as a whole concept for the optimization of pretrained models. By introducing a knowledge finder during the pre-training phase, the language model can explicitly use knowledge from text corpus (e.g., Wikipedia) during the pre-training, fine-tune, and prediction phase. With the introduction of REALM, retrieval augmentation is gradually emerging in the field of open-domain question answering (Open-QA). RAG [84] targets knowledge-intensive tasks, using pre-trained seq2seq models as parametric memory, and using pre-trained neural retrievers to access Wikipedia's dense vector index as non-parametric memory. RAG focuses on solving the shortcomings of previous studies that are limited to specific tasks. FID [85] focuses on how generative models handle retrieved paragraphs. This allows scaling to large numbers of documents and benefits from large amounts of evidence. Inspired by knowledge distillation technology, FID-KD [86] considers the problem of annotated supervision of retrieved documents by using the attention score of reader model based on FID. Atlas [87] works with the searcher to fine-tune the encoder-decoder model by modeling documents as latent variables. Similar to FID, RETRO [78] combines a frozen BERT searcher, a differentiable encoder, and a block-cross attention mechanism to merge retrieved text to predict tags with an order of magnitude more data than would normally be consumed during training. The difference is that RETRO performs retrieval in the pre-training phase, rather than insertion to solve a specific downstream task. REPLUG [88] adds the retrieved documents to the input of the frozen black box large model, proving that it can be used to oversee the retrieval model, which can then find documents that help the large model make better predictions.

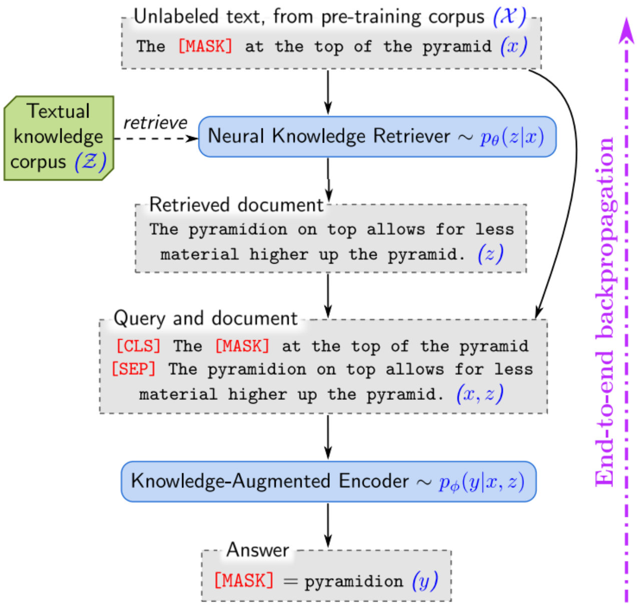

Unlabeled text, from pre-training corpus $(\mathcal{X})$
The [MASK] at the top of the pyramid (x)
Textual knowledge corpus $(\mathcal{Z})$
retrieve
Neural Knowledge Retriever $\sim p_\theta(z|x)$
Retrieved document
The pyramidion on top allows for less material higher up the pyramid. (z)
Query and document
[CLS] The [MASK] at the top of the pyramid [SEP] The pyramidion on top allows for less material higher up the pyramid. (x, z)
Knowledge-Augmented Encoder $\sim p_\phi(y|x, z)$
Answer
[MASK] = pyramidion (y)
End-to-end backpropagation

Fig. 5: REALM augments language model pre-training with a neural knowledge retriever that retrieves knowledge from a textual knowledge corpus [83].

2.3.2 Multimodal Database Augmentation

Not limited to the plain text knowledge, database augmentation is also gaining attention in the field of multimodality, which means that searchers and generators need to be designed to process multimodal documents composed of images and text. MuRAG [89] accesses external nonparametric multimodal memory to enhance language generation. Using joint contrast and generation loss, MuRAG was pre-trained using a hybrid corpus of large-scale image text and plain text to implement the first multimodal retrieval augmentation transformer. Re-Imagen [90] explores the retrieval augmentation of text image generation models, using retrieved information to generate high-fidelity and faithful images. Considering that the generators in the above work are limited to a single mode, either text generation or image generation, Yasunaga [91] proposes a retrieval model RA-CM3 which can retrieve and generate text and image simultaneously. The input data and external memory consists of a set of multimodal documents, each of which is an arbitrary sequence of text/images, providing a generic and modular retrieval augmentation framework for multimodal models.

2.4 Discussion And Application

The main purpose of model tuning is to generalize FM to a specified downstream task. Weight tuning is mainly oriented towards white-box scenarios, where the parameters and structure inside the model are known, allowing for model modification. Input engineering can deal with model generalization problems in both scenarios, and database augmentation enhance model performance by introducing external knowledge base. The common point of these methods is that the feedback given by the model is needed for parameter adjustment, and the model optimization is realized through LFM. Compared with the LFD paradigm that trains the model from scratch with sampled data, LFM requires less data and computing power, and can understand new tasks based on the existing general knowledge of the model.

In the future, we believe that the model tuning based on LFM paradigm can be studied in the following directions:

- **Loss function design** New forms of model fine-tuning loss function need to be explored. For example, self-supervised learning or other types of unsupervised learning can be used to reduce the dependence of fine-tuning on downstream task data.
- **Balancing Memory Size and Retrieval Efficiency** With a large enough retrieval memory, the probability of retrieving an example that closely resembles the query becomes higher. Unfortunately, the downside of this is that the overall inference efficiency of the search-enhanced generative model is low due to the considerable retrieval overhead. Therefore, it is urgent to consider some methods to balance the size of the retrieval memory and the retrieval efficiency, such as the retrieval memory of data compression [92].
- **Design of tailored retrieval metrics** Exploring the application of tailored metrics for retrieval could potentially enhance the control we have over text generation. Existing universal metrics, while generally effective, can inadvertently limit the diversity of retrieval results. By

assembling a more varied set of retrieval outcomes, we can enhance the breadth of useful information covered. Therefore, the incorporation of multiple, distinct retrieval metrics could pave the way for higher quality generation in the future. This shift from a universal to a more customized approach in retrieval metrics could revolutionize the way we generate text.

3 Model Distillation

Model Distillation is a knowledge distillation technique primarily used to compress deep learning models, reducing computational complexity, increasing runtime speed, and saving storage space while maintaining the performance of the original model [94]. We show three traditional knowledge techniques in Figure 6. The knowledge can be derived from various sources, such as the soft-label predictions [94], hidden layer activation [95], [96], embedding [97], [98] or relationship among embeddings [99]. With emergency of FM, researchers began to explore whether FM can be used as databases [100] to teacher student model without access to the training data. The key insight in it is how to induce informative knowledge from FM. In previous studies, these works were known as data-free knowledge distillation (DFKD) [101], [102], [103]. In a white-box scenario, we possess access to the teacher model's architecture, gradient, parameters, and weights. Conversely, in a black-box scenario, we lack prior knowledge of the underlying model's architecture and parameters. Instead, each model is offered as a service (MaaS) [104]. Building upon previous research, DFKD methods can be classified into three distinct paradigms: noise optimization, generative reconstruction, and adversarial exploration.

3.1 Noise Optimization

The core idea of noise optimization is leveraging teacher model to recover the alternative data. Specifically, these recovered data contain certain semantic information of training data (e.g., intermediate feature map). We formulate the goal of noise optimization as following:

$$\hat{x} = \arg\min R(x|P) \tag{1}$$

where the R is a regularization function combined with model P. Following the traditional knowledge distillation, the quality of recover data will affect the performance of student model, *i.e.*, the R is the key factors of whole process. And in this step, classifier loss or other auxiliary loss function can help with convergence.

The second step aims to align the prediction distributions of teacher and student models conditioned on recover data. The formulation of this step is below:

$$\min \sum_{\hat{x},\hat{y}} \mathcal{L}_{CE}(Q(\hat{x},\hat{y})) + \mathcal{L}_{KL}\left(\beta(Q(\hat{x})), \beta(P(\hat{x}))\right), \tag{2}$$

where the $\mathcal{L}_{CE}$ and $\mathcal{L}_{KL}$ indicate cross entropy loss and Kullback-Leibler (KL) divergence [105], $\beta(\cdot)$ represent the softmax function.

In terms of first step, many works endeavored to explore a better regularization function. Since the goal of distillation is alignment, main implementation method of this regularization function is also a distance function. Lopes [106] proposed to minimize the distance of activation records (*e.g.*, means, covariance matrices). Concretely, this records come from teacher models' final layer before softmax. Furthermore, Dream Distillation [107] use the statics of activation vectors as meta-data to generate images. Although, these methods claimed basing on source data-free setting. Indeed, we also need meta-data (*i.e.*, activation records) to optimize the noise. On the other hand, Nayak [108] proposed zero-shot Knowledge Distillation, synthesizing the Data Impressions from the complex Teacher model. Specifically, they utilize a Dirichlet distribution to model the softmax space of the Teacher network, where the concentration parameter of the Dirichlet distribution is interpreted as the measure of similarity among the components in the softmax vector. Therefore, they build a similarity matrix through the weight of pre-final layer and final layer before softmax function. Then this similarity matrix can be assigned as concentration parameter of Dirichlet distribution. Then based on "data impression", they can optimize the noise to synthetic pseudo data.

DeepDream [109] conduct an image prior to drive the generated away from unrealistic images with no discernible visual information:

$$R_{prior} = \alpha_{tv} R_{TV}(\hat{x}) + \alpha_{l_2} R_{l_2}(\hat{x}), \tag{3}$$

where R_{TV} and R_{l_2} penalize the total variance and l_2 norm of $\hat{x}$, respectively, with scaling factors α_{tv} and α_{l_2}. Furthermore, DeepInversion [110] extend image regularization R_{prior} with a new feature distribution regularization term. DeepInversion proposed to minimize the distance between feature map statistics. Supposing that these features are all Gaussian-distributed, the feature distribution regularization term can be formulated as:

$$R_{BNS}(\hat{x}) = \sum ||\hat{\mu_l}(\hat{x}) - \mu_l||_2^2 + ||\hat{\sigma_l}(\hat{x}) - \sigma_l||_2^2 \tag{4}$$

where $\hat{\mu_l}(\hat{x})$ and $\hat{\sigma_l}(\hat{x})$ represent the estimated mean and variance of the feature maps, respectively, corresponding to the l-th neural layer. Average statistics stored in the widely-used BatchNorm (BN) layers are more than sufficient.

In terms of using publicly shared gradients, Zhu [111] and [112], [113] try to recover data from graidents. These gradients are derived from contemporary multi-node learning systems, wherein a key concept is to minimize the distance between the current node and the central node. The quality of synthetic data for noise optimization methods depends on the selection of knowledge priors, which usually requires a large amount of computational cost to update parameters for noise optimization.

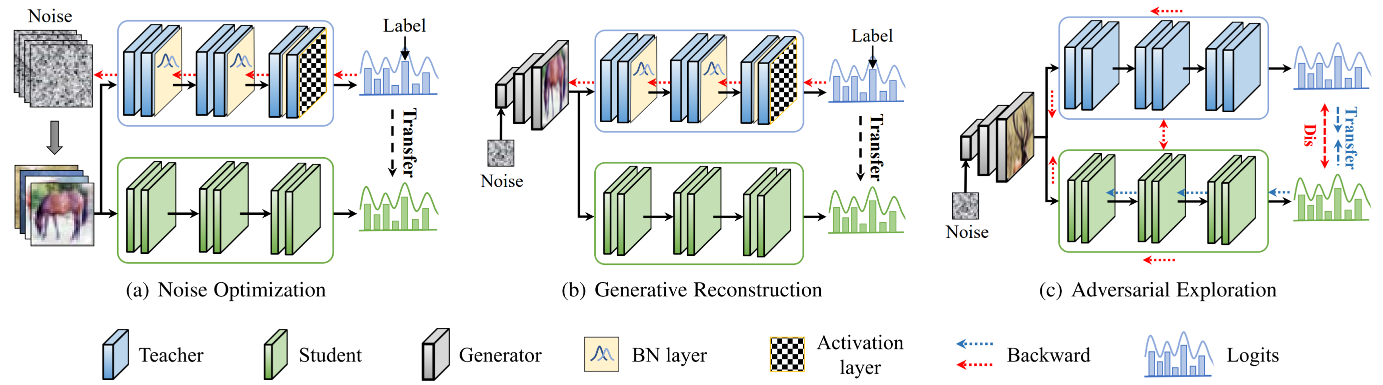
3.2 Generative Reconstruction

This line of works is based on Generative Adversarial Networks (GANs) [114] to generate data representations or feature maps with pre-defined labels. Specifically, a noise vector z is input into the generator G, which then maps z to the desired data distribution. Aforementioned, prior information can be used to constraint the probability of generative model. The optimization function can be formulated as:

$$\min \mathbb{E}_{z\sim p_z(z)}[R(G(z,\hat{y})|P) + \mathcal{L}(G(z,\hat{y}),\hat{y})] \tag{5}$$

where $p_z(z)$ is the distribution of z, P is the pre-trained model.

Chen [115] proposed DAFL based on GAN framework, which regarded the pre-trained teacher networks as a fixed discriminator. Then generator is trained to approximate images given fixed teacher network. In iteration of training GAN, compact student



(a) Noise Optimization (b) Generative Reconstruction (c) Adversarial Exploration

Fig. 6: Three kinds of data-free knowledge distillation frameworks [93].

network can efficiently learn by utilizing generated images and the guidance of the teacher network. In addition to the cross-entropy loss function, they also proposed two other very useful regularization functions to constrain the generator. The first regularization function is applied to activation values. Since intermediate representations from real images tend to receive higher activation values than those from noise, the L1 norm of the intermediate representations can be used as one of the regularization terms. Furthermore, the real data can induce lower entropy of teacher model. Therefore, the regularization terms can be formulated as:

$$R = -||f_P||_1 + -\frac{1}{N}\sum_{i}^{N}\hat{h}_i \log(\hat{h}_i) \tag{6}$$

where f_p is the representation vetor from teacher model and h_i is softmax vector from teacher model.

Ye [116] expand previous methods with group-stack GANs. Specifically, group-stack generators are used to collect data impression. Then all intermediate representations pass to corresponding discriminators with knowledge amalgamated from previous teacher networks. This can be seen as deeper version of DAFL. Yoo [117] proposed KegNet, which consisted of two trainable module. The learnable generator create alternative data then feed them into learnable decoder and fixed classifier. The generator created fake data conditioned on noise vector and pre-defined label. The decoder aimed to re-extract noise vector updating the errors of generator. The classifier was pre-trained teacher network, produced the label distribution of each fake data point. Finally, the generated images could boost knowledge distillation. DeGAN [118] leveraged data from a related domain (Proxy Data) to train GANs. Compared to typical GAN networks, DeGAN simply adds retrieval-enhanced samples to the training process of the discriminator. And following the privous work, DeGAN also adopted cross-entropy loss and a diversity loss conditioned on fixed classifier. As for proxy data, discriminator further penalized to fake data similar to them, which is a new class data.

Privious studies demonstrated that generated samples are similar in each class, which is interpreted as mode collapse issue in data reconstruction. Han [119] introduce a new regularization function for improving the sample diversity within each class, called diversity seeking regularization. Concretely, they use l_2 distance to represent the similarity between the samples, which was added in loss function to pull away the generated samples. Chen [120] filtered the large amount of wild data to further improve student networks. Firstly, they built a selection method to retrieve useful unlabeled images. Then the noisy label from teacher network are collected into a dataset. Under the noisy adaptation matrix supervised, portable student network was learned in conventional distillation manner and refined softmax space by noisy adaptation matrix.

The generation reconstruction method uses the generator to synthesize data, and reduces the training time by parameterizing the distribution of fake data in the iterative generation process. However, the quality of the synthesized data is seriously limited by the generator structure.

3.3 Adversarial Exploration

The main idea behind adversarial exploration based on adversarial generated strategies. The adversarial objective derived from on the discrepancy between the student and teacher networks, formulated as following:

$$D(P, Q; G), \tag{7}$$

where the P and Q indicates pre-trained teacher model and random initialized student model, respectively; G represents the generator.

As for generator updating, the generator creates images that continually widen the gap between the teacher and student networks, while the student and teacher networks are fixed in place, i.e, maximizing the distance in Eq.7. This objective is then used to update the generator. Subsequently, we obtained these challenging samples that mislead the student model. In traditional distillation manner, student networks try to reduce the distance from teacher network conditioning on their predicted distributions or intermediate features. And the generator is fixed.

Micaelli [121] proposed firstly a novel adversarial algorithm based on zero-shot knowledge transfer. They used KL divergence to search those samples poorly predicted by student network. Then traditional distillation is used to minimize KL divergence between teacher and student models. Based on previous methods, Fang [122] divided the generated images into hard and easy samples by estimating the true discrepancy between teacher and student models. Since hard samples always resulted in large output differences, an upper bound can be formulated for the real model discrepancy. Then they simply measures the Mean Absolute Error (MAE) as the optimizing objective. Fang [123] proposed the FastDFKD framework, which achieved even 100× acceleration. Inspired by that in-domain instances always shared common features, these features can be reused to compose different sample without repetitive generating. In this way, FastDFKD built an efficient synthesizer which captured common features for fast adaptation. He [124] provide thorough experiments to investigate synthetic data to improve image recognition.

Adversarial Exploration uses a truly adversarial framework for simultaneous knowledge extraction, the challenge being that the samples generated are not completely reliable, so some irrelevant samples are less effective in data transfer.

3.4 Discussion And Application

We have analyzed and reviewed three FM-based DFKD frameworks above, and in this section, we briefly discuss the application and future direction of DFKD methods.

- **Integration with downstream methods:** Data-free knowledge distillation based on FM can be flexibly deployed in various excellent learning solutions, such as adversarial learning [125], automatic machine learning [126], meta-learning [127], and reinforcement learning [128]. Therefore, combining knowledge distillation with other learning options will help solve the practical challenges of the future.
- **Model applicability study:** The application conditions of model distillation are still harsh, the existing distillation algorithms are not stable enough, and most of the research is conducted on small data sets, and there is a lack of unified benchmark methods. In order to achieve perfect results, trial and error costs are high. This flaw is magnified infinitely in the face of the foundation model. We think it is also important to explore the influence of FM internal knowledge representation on the performance of model distillation and further design the distillation structure.

4 Model Reuse

In the real-world training process, it is common practice to train multiple models with various hyperparameter settings. Subsequently, the best-performing model is selected or an ensemble of models is created to achieve superior performance [129]. However, ensemble models require additional computing resources at inference time, and selecting a single model often results in discarding the remaining models.

In the real-world training process, it is crucial to experiment with different hyperparameter settings to explore the model's potential. Training multiple models with diverse hyperparameter configurations allows us to capture a wider range of information and uncover optimal solutions. However, the manual selection of the best model among these trained models can be a daunting task. It often requires extensive computational resources, as training and evaluating each model can be time consuming. Ensembling, on the other hand, offers a powerful solution to leverage the knowledge from multiple models. By combining the outputs or predictions of individual models, we can create an ensemble model that exhibits improved performance.

4.1 Ensemble Methods

Ensemble learning is a widely used and effective approach in machine learning that involves combining predictions from multiple models to enhance overall performance. This paradigm unifies ”weak” models by utilizing their collective outputs to create a more robust and accurate predictive model [129], [130]. By combining the predictions of multiple models, ensemble learning can help reduce the impact of biases and errors present in individual models and improve the generalization and stability of the resulting system.

Given a finite collection of models, represented as $\mathcal{M} = \{M_i\}_{i=1}^{n}$, where n defines the total number of models in the set, the ensemble of logits is defined as follows:

$$M^* = \text{Ensemble}(\mathcal{M}, \boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i M_i(\cdot), \tag{8}$$

where λ_i corresponds to a set of model-specific weight hyperparameters. These weight hyperparameters determine the relative importance of each model in the ensemble. The predictions of each model M_i are weighted by the corresponding λ_i and summed to generate the final prediction. There has been significant research in this area, and has been applied in various domains, including computer vision [131], natural language processing [132], [133] and speech recognition [134].

Efficient Ensemble: In [135], the authors introduce BatchEnsemble whose computational and memory costs are significantly lower than typical ensembles. They accomplish this by using a shared weight for all ensemble members and combining it with a rank-one matrix specific to each member. In [136], the authors propose an ensemble method within the tangent space. The method hinges on the equivalence of the ensemble of linearized models, i.e., the first-order Taylor approximations around the pre-trained weight, to a singular linearized model. The primary strength of this framework lies in its ability to reduce the inference time of an ensemble of n tangent models from $\mathcal{O}(n)$ to $\mathcal{O}(1)$.

4.2 Model Fusion

Ensemble methods such as bagging, boosting, or stacking provide effective techniques for combining models and can lead to enhanced generalization and robustness. However, creating an ensemble involves running multiple models simultaneously, each model in the ensemble independently processes data and generates predictions, which are then aggregated to make a final decision, resulting in significant computational resources at inference time.

To address these challenges, this subsection aims to provide insights and strategies on how to effectively perform knowledge fusion. By fusing the knowledge from multiple models, we can leverage the strengths of each model and enhance overall performance. Advanced techniques for model fusion are being developed. These techniques aim to extract and combine the most useful information from each individual model to create a more powerful one [137].

4.2.1 Weight Interpolation

Weight interpolation is a simple yet effective method for model fusion. This method involves combining the weights of multiple models to create a new model of the same network structure. In this section, we will introduce the basic concepts of weight interpolation and its variants. We will also discuss some of the applications of weight interpolation.

Commonly, the subject of exploration is linear weight interpolation. It is established knowledge that weights from distinct models, when fine-tuned on an identical dataset, can be associated via linear trajectories while maintaining a constant loss [138], [139]. This concept is recognized as ”linear mode connectivity”. The basic operation of linear weight interpolation is simple. For example, suppose that we have a model M_1 and a model M_2 that have been trained on the same task. Let W_1 and W_2 be the weights of M_1 and M_2, respectively. Then, by interpolating the weights of M_1 and M_2, we can create a new model M_{new} that combines the knowledge from both M_1 and M_2. This new model M_{new} can be defined as follows:

$$W_{new} = \lambda W_1 + (1 - \lambda) W_2, \tag{9}$$

where λ is a hyperparameter that controls the contribution of each model. This simple technique can be used to combine two models and create a new model that may exhibit enhanced performance. Furthermore, this technique can be extended to combine multiple models. Suppose that we have n models $M_1, M_2, ..., M_n$, then we can create a new model M_{new} that combines the knowledge from all n models by interpolating their weights as follows:

$$W_{new} = \lambda_1 W_1 + \lambda_2 W_2 + ... + \lambda_n W_n, \tag{10}$$

where λ_i is a hyperparameter that controls the contribution of each model. This technique can be used to combine multiple models and create a new model that may exhibit enhanced performance.

Uniform Averaging: Uniform averaging is the special case where $\forall_{i \in [n]} \lambda_i = 1/n$, and is also known as 'model soups' or 'adapter soups' for parameter-efficient fine-tuned models [140], [141]. The practice of uniform weight averaging is a straightforward but potent technique, which has found utilization in the domain of deep learning. This method involves combining the weights of multiple models to create a new model with the same network structure, thereby raising the effective model reuse. Its application helps improve generalization, as evidenced in [142], while also providing superior accuracy by averaging the weights of models that are fine-tuned with diverse hyperparameter configurations [140]. Additionally, it increase robustness against distribution shifts [143]. Even just utilizing weight-space averaging of adapters trained on different domains, rather than full parameter space, can bolster out-of-distribution capability of model [144]. In addition to averaging on trained models, checkpoint averaging along the trajectory of a training run speeds up convergence [145] and improves test generalization [146]. The Federated Learning community offers solutions such as FedAvg, a decentralized learning method, where models are collectively learned by averaging their weights [147].

Weighted Averaging: In addition to the same weight value, different models can have different weight values, and even different layers of different models can have their own weight values. For model-specific weighted averaging, the model-specific scalar hyperparameters $\lambda_i, i \in [n]$ sum up to 1, i.e. $\sum_{i=1}^{n} \lambda_i = 1$. There has been significant research in this area, with recent studies including [148], [149], [150]. Checkpoint averaging methods are also model-specific average methods but along the training trajectory. For example, latest weight averaging (LAWA) [145], [146].

Task Arithmetic: Gabriel [151] edited large pre-trained model with task vectors, which can be formulated as

$$W_{new} = W_{pre} + \sum_{i=1}^{n} \lambda_i \underbrace{\left(W_{ft}^{(i)} - W_{pre}\right)}_{\tau_i}, \tag{11}$$

where W_{pre} is the parameters of a pre-trained model, $W_{ft}^{(i)}$ is the parameters of model fine-tuned on i-th downstream task, $\tau_i := W_{ft}^{(i)} - W_{pre}$ is named as task vector, and λ_i are not explicitly bounded. They found that the interpolated model's performance on specific downstream target task can be manipulated to be better via task vector addition or reduced via task vector negation.

4.2.2 Mode Connectivity Based Method

Neural network loss landscapes refer to the relationship between the model's parameters and the corresponding loss function. Understanding these landscapes is crucial for optimizing neural networks.

The idea of mode connectivity refers to the findings that different local minimal in model weight space might be connected by simple paths along which the model's training or validation performance is nearly constant [152], [138], [139]. These paths could potentially be found using optimization methods. Finding mode connectivity is a relatively new field of research and several methods have been proposed.

Especially, if finetuned from a same pre-trained model, the neural network often lies in a single loss basin in weight space.

Git Re-Basin is a technique for aligning and merging models trained with different random initializations or hyperparameters by permuting units to match a reference model and merging in weight space, creating a convex basin of functionally equivalent weights near the reference [153]. ZipIt builds on model merging techniques by allowing partial zipping of models up to a specified layer to account for differences in models trained on disjoint tasks. It also introduces a general "zip" operation to merge weights within each model. These changes enable feasible merging of arbitrary models trained on completely different tasks [154].

4.2.3 Straightforward Optimization

Straightforward optimization methods for model fusion involve simple and direct techniques to combine models. These methods typically focus on finding the optimal weightings or parameters for the fused model by minimizing an objective function. These methods provide a straightforward way to combine models without requiring complex algorithms or architectures.

In [155], a layer-wise model fusion algorithm that utilizes optimal transport to (soft-) align neurons across the model before averaging their associated parameters is proposed. Various techniques aim to minimize the distance (such as the l_2 norm distance) between the merged and individual models. Some methods offering closed-form solutions [156] and others employing gradient-based optimization approaches [157], [149].

4.3 Discussion And Application

In real-world training, it is common to train multiple models with different hyperparameters and select the best model or ensemble them. However, ensembles require more inference resources and selecting a single model often discards useful knowledge. Ensemble methods like bagging and boosting combine models to improve generalization and robustness. However, they require running multiple models, increasing computational costs. Efficient ensemble methods like BatchEnsemble reduce costs by using a shared weight plus a low rank matrix per member [135]. Model fusion extracts and combines useful knowledge from individual models into a unified, improved model. This avoids ensemble computational costs. Some researches on model reuse have been carried out in transfer learning [158], [159], model compression [160], [135], contunual learning [161], domain adaptation, personalization [162], efficient search [140] and multi-task leanring [154]. The key goals of model reuse methods include:

- **Model complementarity research:** Improving overall predictive performance by combining complementary knowledge from different models.
- **Reduce computing costs:** Reducing computational costs compared to ensembling multiple models.
- **Knowledge representation extraction:** Extracting essential knowledge from multiple models into a distilled representation.
- **Robustness improvement:** Building robustness by merging diverse models into a unified whole.
- **Enable efficient model training and editing:** Accelerating model training by transfer and reuse of knowledge from pre-trained models.

5 Meta Learning

Meta-learning improves the performance of a model on a new task by extracting experience (usually data distribution) from multiple related tasks. The core idea of meta-learning is that learning becomes more efficient with increased experience, facilitated by the acquisition of inductive biases or knowledge that enhances future learning [163], [164]. The abstract formula for meta-learning can be expressed as follows:

$$\min_{\omega} \mathop{\mathbb{E}}_{\mathcal{T}\sim p(\mathcal{T})} \mathcal{L}(\mathcal{D};\omega) \tag{12}$$

where ω represents meta-knowledge, $p(\mathcal{T})$ corresponds to the distribution of tasks, and $\mathcal{D}$ represents the dataset associated with each task. Our goal is to acquire a universal meta-knowledge that minimizes the loss for various tasks, striving for improved performance across different tasks.

This 'learning-to-learn' [165] method can lead to a variety of benefits such as improved data and compute efficiency, and it is better aligned with human and animal learning, where learning strategies improve both on a lifetime and evolutionary timescales [166], [167], [168].

The generalized use of meta-learning can be traced back to the [169], proposed by Philip as a general technique for fusing the results of multiple learning algorithms. Subsequently, meta-learning paradigm has been introduced into many fields and widely used. In the field of biology, Bengio [170] proposes an original approach to neural modeling based on meta-learning to find biologically credible synaptic learning rules, which enable networks to learn to perform difficult tasks. The learning rules involved can be optimized in the parameter learning rule space [171]. Thrun [172] provides the first clear definition and analysis of learn to learnin the field of algorithms, with significant practical implications for the field of machine learning and beyond. Vilalta [173] believes that the goal of meta-learning is to build an adaptive learner (that is, a learning algorithm that dynamically improves its bias through experience by accumulating meta-knowledge), and points out how to use meta-knowledge to improve the performance of learning algorithms key to progress in the field.

In essence, meta-learning is also based on the model itself to extend to downstream tasks, and is therefore one of the important paradigms of LFM. This chapter focuses on the research progress and classification of meta-learning related to pretrained large model transfer learning. In particular, meta-learning has the potential to mitigate many of the major shortcomings of contemporary deep learning, such as improving data efficiency, facilitating knowledge transfer, and unsupervised learning. Meta-learning has been shown to be useful in multi-tasking scenarios, for example in areas such as small lens image recognition, reinforcement learning (RL), hyperparameter optimization, and neural structure search (NAS).

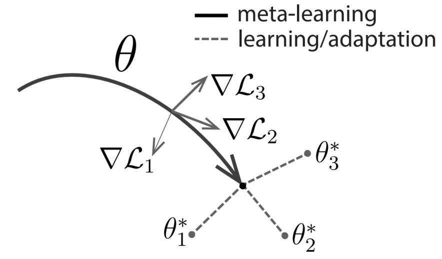


Fig. 7: Schematic of the MAML algorithm, which optimizes a representation θ that can quickly adapt to new tasks [177]

From the perspective of the LFM paradigm, meta-learning can be further referred to as data-free meta-learning (DFML) [174], which achieves effective generalization to new tasks by meta-learning from a collection of pre-trained models without access to training data. According to the degree of understanding of the internal parameters and structure of the model, DFML is specifically divided into white-box data-free learning and black-box data-free learning. White-box data-free meta-learning focuses on exploiting the model based on the known underlying architecture and parameters of the model. Black-box data-free learning takes into account the generalization of tasks under the premise that the internal structure of the model is unknown. This method is more suitable for the background of modern industrial systems that only open large-scale model APIs for access, such as OpenAI's GPT-4 [175], Google' Lamda [176], etc., which is one of the main problems solved by the LFM paradigm.

5.1 White-box Data-free Meta-learning

As shown in Figure 7, MAML [177] is compatible with any model trained with gradient descent and can be applied to a variety of different learning problems, including classification, regression, and reinforcement learning, setting a precedent for optimization-based meta-learning. The key idea of MAML is to use a set of source tasks $\{T_1, \ldots, T_k\}$ to find the initialization of parameters θ_0 from which learning a target task T_0 would require only a small number of training samples.

Continual-MAML [178] addresses the problem of fast online adaptation to new tasks while maintaining acquired knowledge on previously learned tasks. Since then, a large number of MAML variants require access to the original training data of the model [179], [180], [181], which restricts the application of meta-learning in large model migration. Existing white-box data-free learning methods solve this problem in parameter space. Meta-transfer learning (MTL) [182] learning transfers large-scale pre-trained DNN weights to solve a small number of learning tasks, which opens the paradigm of white box learning. A similar technique is proposed in [183], also for small sample learning. Meta-DETR [184] performs meta-learning object localization and classification at the image level in a small sample environment, using a semantic alignment mechanism (SAM) to encode and query supporting images into features of a specific class, and then input these features into a class-independent decoder to directly generate predictions for the specific class. CDCS [185] adaptes a large common programming language pre-trained model CodeBERT [186] based on MAML to suit domain specific languages, and the experimental results are significantly better than the traditional pre-trained code model directly fine tuned.

5.2 Black-box Data-free Meta-learning

A lot of DFML work can only deal with pre-trained models in white-box scenarios, but this assumption is often difficult to meet, especially when the training cost of commercial large models is high, and meta-learning requires that downstream models also have the same architecture, limiting the scenarios that can be applied.

How to use the method of meta-learning to generalize the black box model for the downstream task is the primary consideration. Wang [174] proposed to predict the meta initialization through a meta-trained black-box neural network. For each task identity T_i, the task identifier is first embedded as the corresponding task embedding e_i. A black-box network is used to fuse different pre-trained models. Assuming the parameters of the black-box model are ϕ, the black-box function $f_\phi(e_i)$ is utilized for fitting the parameters of task T_i with task embedding e_i as input. The objective is the optimization of the following:

$$\max_\phi \sum_{i=1}^{i=N} \log P\left(\boldsymbol{\theta}_i \mid f_\phi\left(\boldsymbol{e}_i\right)\right). \qquad (13)$$

The likelihood function $P\left(\boldsymbol{\theta} i \mid f \phi\left(\boldsymbol{e}_i\right)\right)$, where σ is a standard deviation constant, quantifies the deviation of the black-box network prediction from the true pre-trained task parameters.

$$P\left(\boldsymbol{\theta}_i \mid f_\phi\left(\boldsymbol{e}_i\right)\right) = \exp\left(-\frac{\left\|f_\phi\left(\boldsymbol{e}_i\right) - \boldsymbol{\theta}_i\right\|^2}{\sigma^2}\right). \qquad (14)$$

Reptile [187] extends MAML's results by repeatedly sampling a task for training and moving the initialization toward the task's training weights. This approach still has some limitations, such as merging models only in the parameter space and ignoring the underlying data knowledge that can be extracted from the pre-trained model. At the same time, the way in which neural networks are used to predict model parameters determines that they can only be applied to small-scale pre-trained models. BiDf-MKD [188] integrates model distillation and meta-learning, inverting each API to recover its training data only through a query API and a zero-step estimation, and then transferring more general meta-knowledge from a series of black-box apis to a single meta-model through a two-layer meta-knowledge distillation structure. PURER [189] further overcomes the limitations of data representation and model architecture on the basis of DFML and proposes a unified framework to synthesize a sequence of pseudo episodes.

In the field of computer vision, the small sample optimization method goes far with the help of meta-learning [190]. RN [191] learns to learn a deep distance metric to compare a small number of images within episodes, each of which is designed to simulate the few-shot setting. FILM [192] uses a pre-trained language model based on comparative learning, and adopts MAML to train the model through double-layer optimization to achieve model generalization with few samples.

Another important application of meta-learning in black box scenarios is in the field of model security (Meta Attack [193]). Du [193] proposes a meta-attack approach to learn generalizable prior abstractions from previously observed attack patterns to help infer attack patterns from a small number of queries and outputs. MGAA [194] improves the transferability of adversarial examples in black box Settings by narrowing the gap in gradient direction between white box attacks and black box attacks. In order to perform adductive attacks on black box models without allowing queries, Qin [195] designed a Meta-Surrogate Model (MSM), which is mathematically expressed as a two-layer optimization problem, and designed a differentiated attacker to make training feasible. This makes the adversarial examples generated on the MSM extremely transferable. The model extraction attack framework D-DAE [196] uses a meta-learning-based corruption detection module to learn the basic differences between the distribution of corrupted and uncorrupted query results.

5.2.1 Discussion And Application

Meta-learning can be seen as transfer learning in a two-tier optimization scenario, i.e. the embodiment of LFM. The study of meta-learning without data is of great significance to the field of large models, especially when the concept of model as a service is widely mentioned. The pre-trained model interface provided by various major manufacturers can not only be used as a tool to solve specific tasks, but also as a training resource for white/black box no-data learning to realize effective learning of new unseen tasks. The point of this is to eliminate the need to perform meta-learning on large amounts of labeled data and to reliably protect data privacy and security. At present, under the complementarity of FM and meta-learning, some researches have been carried out in few-shot image classification [197], robotic control policy learning [198], hyperparameter optimization [199], meta-learning learning rules [200] and abstract reasoning [201] have made excellent progress.

From the perspective of learning from models, We believe that there are still the following problems in DFML to be studied:

- **Effective black box DFML algorithm** The primary focus of meta-learning research remains on the issue of data knowability, and studies on Data-Free Meta Learning (DFML) are still relatively limited. Given the current trend of pre-training model proliferation, the development of efficient and fast black-box DFML algorithms has become particularly crucial.
- **Design of general DFML algorithm** White-box DFML requires precise parameters for each pre-trained model and requires that all pre-trained models share the same architecture and cannot be adapted to large pre-trained models. Therefore, the research of general white box DFML algorithm is also one of the promising directions.
- **Cost optimization scheme** Meta-learning usually requires a highly complex optimization process, which makes DFML computationally complex and expensive. Especially in scenarios based on FM, the iteration of a large number of parameters will exponentially increase the computational cost. A considerable part of the work has put forward the cost optimization method of the meta-learning structure [202], [203], [204], and the follow-up will be a direction worthy of efforts.
- **Zero-shot learning/Few-shot learning** In the few shot scenario, the performance of FM meta-learning is still weaker than the fine tuning paradigm, which restricts the landing deployment in the real scene [205]. Improving the performance of meta-learning under small or even zero samples is also one of the research directions.

6 Model Editing

Under the huge computational overhead of pre-trained models, updating the knowledge inside the model is not a simple "learning task". Ideally, with the complex transformation of various situations in the world, large models should also keep up with the pace of The Times anytime and anywhere [206], [207], but the computational burden of training new large models cannot make large models achieve instant updates, so the algorithm for knowledge updating on the basis of the original pre-trained model has received widespread attention. Model editing, as one of the paradigms of LFM, can effectively change the knowledge of a pre-trained model in a specific domain without adversely affecting the results of other inputs. Model editing can better serve other LFM paradigms, because all LFM paradigms have high requirements for knowledge storage and output accuracy of the model itself.

To be clear, although both involve parameter and structure adjustments, model editing and model fine-tuning are not the same task oriented. The primary purpose of model fine-tuning is to use the pre-trained model for downstream tasks, while model editing focuses on updating the overall knowledge base or domain knowledge base within the model from a macro perspective, and does not need to be designed specifically for one task. Model editing can be considered a prerequisite for model fine-tuning.

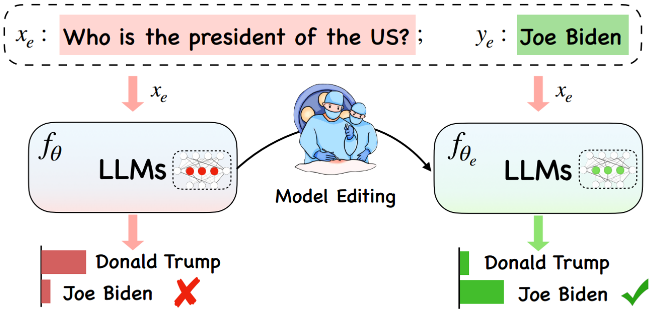


Fig. 8: Model editing techniques for updating large pretrained models [208].

As shown in Figure 8, $f_\theta(x) \neq y$, our intention is to adjust the original pretrained model f_θ by using the edit descriptor (x_e, y_e) to get the modified model f_{θ_e}, and $f_{\theta_e}(x) = y$. On the other hand, model editing also requires that editing in the current domain does not affect the normal output results of inputs in other domains [208], which can be formally expressed as:

$$f_{\theta_e}(x_e) = \begin{cases} y_e & \text{if } x_e \in I_{x_e} \\ f_\theta(x_e) & \text{if } x_e \in O_{x_e} \end{cases} \tag{15}$$

When the input x_e is in the domain I_{x_e} that needs to be updated, the output is adjusted to y_e, otherwise the original model output $f_\theta(x_e)$ remains.

The concept of model editing was first defined in 2020 [209], KnowledgeEditor [210] implements model editing on BERT architecture language models. Mitchell [21] brings model editing to the field of large pre-trained models. According to the concrete implementation of current model editing in the field of large-scale pre-trained models, we mainly divide it into four categories: (1) adding additional parameters while freezing the original model parameters, and (2) directly modifying the internal parameters of the model. The first method is also called memory based model editing, and the core idea is to keep the original parameters of the model unchanged, by adding an additional set of independent parameters to modify the results of the model output. The second method acts as parameter based model editing, modifying the original model parameters directly to achieve the desired effect.

6.1 Memory Based Model Editing

Memory based model editing Uses additional editors (such as a new network) to influence the behavior of the model and keep the original parameters unchanged. An external learning editor was developed that modifies the fine-tuning gradient of the edits, but does not change the basic model that must process the edits [211].

SERAC [21] wraps a black-box base model with an explicit cache of user-supplied edit descriptors (arbitrary discursions of the language model), as well as a small auxiliary range classifier and counterfactual model. Instead of making changes to the underlying model in the parameter space, SERAC simply stores the edits in the cache, using a range classifier to assess the likelihood that new input will fall within the scope of the stored edits. The drawback of SERAC as a learnable editor is that it relies excessively on editing datasets to train classifiers and counterfactual models. Mem-Prompt [212] introduces user feedback on PLMs error correction. Specifically, they keep a memory of errors in the model and user feedback, thereby enhancing the model to produce update prompts and avoid similar errors. Transformer-Patchcher [213] extends model editing to sequential model editing (SME), borrowing the idea of adapter tuning to change the behavior of Transformer-based models by simply adding and training a few neurons in the last feedforward network layer. CALINET [214] also uses a specific FFN in the Calibrated FFN extended PLM for the addition of additional parameters, which consists of multiple calibration memory slots. These parameters are trained on the modified fact data set to achieve the effect of model editing. Model editing for black box scenarios is still worth exploring, and a recent article shows us the way [215], [216].

6.2 Parameter Based Model Editing

An update matrix is applied to edit the model to update some internal parameters of the model [208]. Parameter-based model editing can be finely divided into constrained tuning, locate and edit, and meta-learning.

6.2.1 Constrained Tuning

In order to combine new data and update the knowledge inside the model, a very natural and simple idea is to use new data for fine-tuning. In order to reduce the forgetting of knowledge, some restrictions need to be added in the process of fine-tuning, such as updating parameters as little as possible and only updating part of the structure of the model. Zhu [217] formulates the model modification as a constraint optimization problem, constrines the loss of unmodified facts, and explores a better baseline method to approximate the execution of this constraint. In the implementation process, considering the maintenance cost of constraints, constraints are approximately expressed as:

$$\text{minimize}_{\theta \in \Theta} \frac{1}{m} \sum_{x \in \mathcal{D}_\mathcal{M}} L(x;\theta) \quad \text{subject to} \quad \|\theta - \theta_0\| \leq \delta, \tag{16}$$

The original model is fine-tuned on the modified fact dataset $\mathcal{D}_M$, while using any appropriate norm explicitly constrained in the parameter space to minimize interference with the unmodified facts. This constraint is approximated by using the local continuity of the loss around θ_0.

6.2.2 Locate And Edit

By introducing the concept of knowledge neuron, Dai [218] makes a preliminary study on how fact knowledge is stored in pre-trained Transformers, and proposes a knowledge attribution method to identify neurons that express this fact. The activation of these knowledge neurons is positively correlated with the expression of the corresponding facts, and knowledge neurons can be directly utilized to edit (such as updating and deleting) specific factual knowledge without fine-tuning. MEMIT [219] uses explicitly calculated parameter updates to insert new memories for weights of transformer modules identified as causal mediators of factual knowledge recall, which can be expanded to mass store thousands of memories. Ilharco [220] proposes a new paradigm for guiding neural network model editing centered on task vectors. A task vector is constructed by subtracting the weight of the pre-trained model from the weight of the same model after fine-tuning the task. Task vectors can be modified and combined by arithmetic operations such as negative numbers and additions, and the resulting model's behavior is controlled accordingly.

6.2.3 Meta Learning

The use of meta-learning methods to modify model parameters is also a feasible direction. Meta-learning methods use hypernetworks to learn the weight matrices needed to edit large pre-trained models. KnowledgeEditor (KE) [210] trains a supernetwork with constraint optimization to modify a fact without affecting other knowledge, and then uses the trained supernetwork to predict weight updates at test time. MEND [221] integrates a network of small gradient-decomposed model editors to make fast local edits to the behavior of a pre-trained model using a single desired input/output pair. MEND learns to transform gradients fine-tuned by standard, using low-rank decomposition of gradients to make the parameterization of such transformations easy to handle, and trained MEND can quickly apply new edits to pre-trained models. This method directly modifies the parameters of the model, which is good for editing a single knowledge or a small amount of knowledge, but easy to cause model parameter collapse when applied to a large amount of knowledge editing.

6.3 Discussion And Application

Model editing is necessary to learn and use models (LFM) more rationally, helping to keep them in sync with the real world in real time. Therefore, efficient, reliable and accurate algorithm design is essential. Many of the methods mentioned above still have great limitations waiting for further research. Model editing technology is currently mainly targeted at tasks such as neural machine translation [222], text style transfer [223] and text-image generation [224] based on FM.

- **Model editing object** Current model editing research focuses on the editing of factual knowledge, which is relatively easy to formalize and evaluate, but other types of knowledge included in editing models, such as preferences, require more sophisticated algorithm design. Developing a unified editing framework for all types of knowledge of the model is also a feasible direction. The realization of editing these contents requires a sufficient understanding of the knowledge representation of the pre-trained model, and some work can provide beneficial guidance for the development of these works [7], [225].
- **Black box editting** Previous model editing work was still based on the premise that the internal structure of the pre-trained model was known. Quite a few large pre-trained models show excellent performance (ChatGPT, GPT4), but can only be accessed through apis. How to edit the model in the black box environment and use it for downstream tasks still needs a lot of research.
- **Unified evaluation benchmark** Most knowledge editing research utilizes metrics such as the success rate of editing target knowledge, the invariance rate of predictions on unrelated knowledge (to assess generality), and the accuracy of paraphrasing target knowledge (to assess consistency). However, we find these assessments limited in their ability to comprehensively evaluate the knowledge editing capabilities of different strategies. For example, many evaluations only sample unrelated knowledge from the same distribution as the target knowledge [226]. Therefore, it is crucial to design comprehensive benchmark tests that can more effectively assess the capabilities of editing strategies.

7 CHALLENGES AND FUTURE PROSPECT

7.1 Summary and Research Trends

To provide a visual representation of the evolution of large language models, we have created a timeline in Figure 9. This timeline showcases the names and key contributions of prominent large models proposed by the academic community in recent years. By visually presenting this information, readers can gain a comprehensive understanding of the advancements and innovations taking place within the large-scale pre-training domain. Building upon the foundation laid by these models, researchers can further enhance their capabilities, explore new methods, and unlock new avenues to solve complex language understanding and generation challenges.

Interdisciplinary research on LFM Interdisciplinary research is essential to advancing the science of machine learning, with various fields providing valuable insights and innovations through interdisciplinary research. The LFM paradigm itself provides specific and feasible generalization ideas for each discipline, and each discipline in turn provides the overall direction for LFM. In future work, we believe that actively promoting cooperation in different research fields and facilitating the exchange of ideas will be key to overcoming disciplinary barriers.

Multimodal large model Learning Large language models exhibit surprising Zero/Few Shot reasoning performance on most NLP tasks, but they are inherently ”blind” to vision because they can only understand discrete text. Large visual base models progress rapidly in perception and slowly in reasoning. Given this complementarity, the resulting multimodal architecture of language models and visual models running simultaneously toward each other is expected to make LFM more intelligent, including the advantages of having a more user-friendly interface, being a more comprehensive task solver, and better fitting the way humans perceive the world.

Model upgrade With the supplement of data and the upgrade of algorithms, we believe that the perceptual pre-training model in the previous stage has completed the progress to the cognitive pre-training model, and the next step is to move towards the general pre-training model, that is, to use a more unified way to make the ability of the pre-training model land in the industry. From

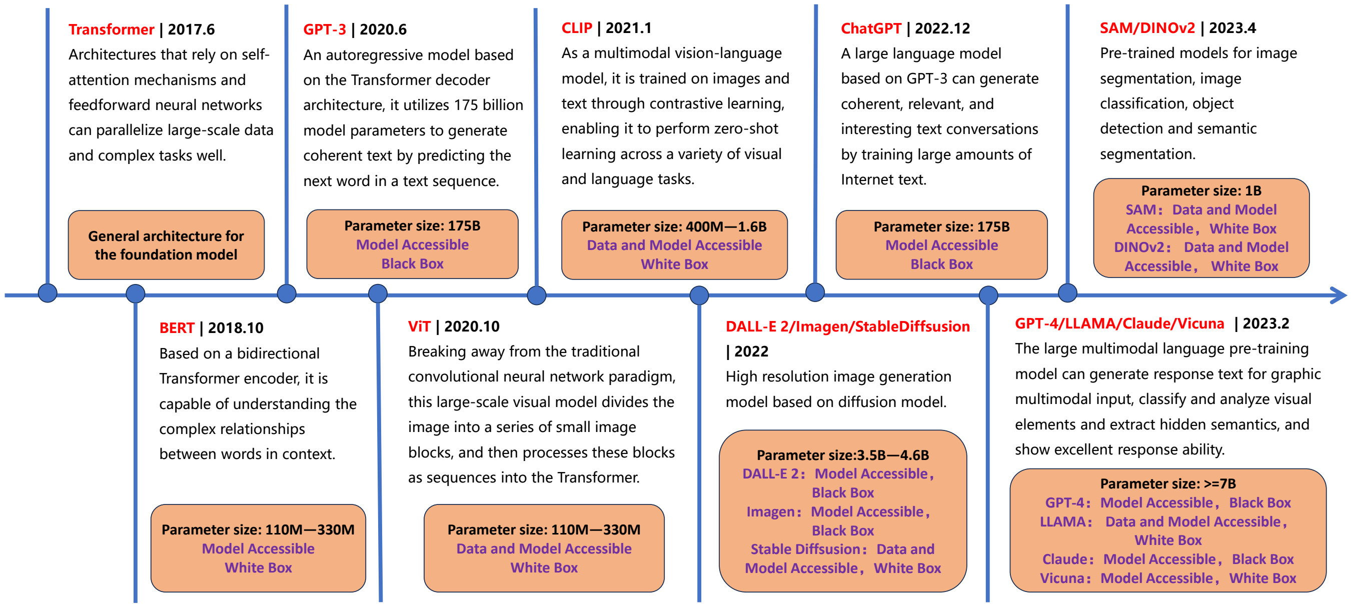

Transformer | 2017.6
Architectures that rely on self-attention mechanisms and feedforward neural networks can parallelize large-scale data and complex tasks well.
General architecture for the foundation model
GPT-3 | 2020.6
An autoregressive model based on the Transformer decoder architecture, it utilizes 175 billion model parameters to generate coherent text by predicting the next word in a text sequence.
Parameter size: 175B
Model Accessible
Black Box
CLIP | 2021.1
As a multimodal vision-language model, it is trained on images and text through contrastive learning, enabling it to perform zero-shot learning across a variety of visual and language tasks.
Parameter size: 400M—1.6B
Data and Model Accessible
White Box
ChatGPT | 2022.12
A large language model based on GPT-3 can generate coherent, relevant, and interesting text conversations by training large amounts of Internet text.
Parameter size: 175B
Model Accessible
Black Box
SAM/DINOv2 | 2023.4
Pre-trained models for image segmentation, image classification, object detection and semantic segmentation.
Parameter size: 1B
SAM：Data and Model Accessible，White Box
DINOv2：Data and Model Accessible，White Box
BERT | 2018.10
Based on a bidirectional Transformer encoder, it is capable of understanding the complex relationships between words in context.
Parameter size: 110M—330M
Model Accessible
White Box
ViT | 2020.10
Breaking away from the traditional convolutional neural network paradigm, this large-scale visual model divides the image into a series of small image blocks, and then processes these blocks as sequences into the Transformer.
Parameter size: 110M—330M
Data and Model Accessible
White Box
DALL-E 2/Imagen/StableDiffsuion | 2022
High resolution image generation model based on diffusion model.
Parameter size:3.5B—4.6B
DALL-E 2：Model Accessible，Black Box
Imagen：Model Accessible，Black Box
Stable Diffusion：Data and Model Accessible，White Box
GPT-4/LLAMA/Claude/Vicuna | 2023.2
The large multimodal language pre-training model can generate response text for graphic multimodal input, classify and analyze visual elements and extract hidden semantics, and show excellent response ability.
Parameter size: >=7B
GPT-4：Model Accessible，Black Box
LLAMA：Data and Model Accessible，White Box
Claude：Model Accessible，Black Box
Vicuna：Model Accessible，White Box

Fig. 9: **The evolution of large language models and multimodal models**. This figure highlights the key milestones in the development of large pre-trained language models and multimodal models over the past several years. It traces the progression from bidirectional Transformer-based models like BERT to autoregressive models like GPT-3 and diffusion models for image generation like DALL-E 2.

the perceptual intelligence of ”listening, speaking, and seeing”, to the cognitive intelligence of ”thinking, answering questions, summarizing, translating, and creating”, and even to the level of ”decision-making and reasoning”. The combination of pre-trained models and reinforcement learning is also a new trend.

Solving LFM problems with basic models With the increasing complexity of parameters and structures of pre-trained models, the treatment of basic models shows great potential for solving large model problems in various research fields, making it an active and promising research area.

7.2 Open Research Questions And Future Prospect

In the following, we prospect several potential research directions for future study.

Focus on the quality of training data Some of the biases in current models are actually caused by the training data itself. If the training data contains a disproportionate number of examples from a particular group, the model may learn to overpredict the outcome of that group. Conversely, if a certain group is underrepresented in the training data, the model may have difficulty accurately predicting the outcomes of members of that group. This uneven representation in the training data causes the model to be biased against certain outcomes, and this bias can perpetuate or even amplify existing social biases when the model is deployed. Ensuring the quality and representativeness of training data is critical. This includes carefully collating the data to ensure that it is comprehensive and representative of diverse groups and expected predicted outcomes. In addition, techniques such as data enhancement, rebalancing, and bias correction can be used to mitigate the effects of bias in training data.

In-depth analysis of knowledge representation inside large models Models can be regarded as high-dimensional representations of data. With the increasing complexity of parameters and structures of pre-trained models, it has become an important research trend to explore how knowledge is represented inside models. Only when we fully understand the weight representation of the model and strengthen the research on the algorithm itself and the internal operation mechanism of the model can we save the cost better and apply the large-scale pre-training model to the downstream task more reasonably.

Addressing Security Concerns in LFM The strength of LFMs comes with its own set of security challenges, which are not dissimilar to those faced by smaller language models. These large, pre-trained models can be manipulated with unique instructions to generate harmful, biased, or offensive content that could be used maliciously, thereby posing a potential misuse risk. One of the primary methods to circumvent these issues is the application of reinforcement learning from human feedback (RLHF). This approach integrates human judgment into the training process, allowing for the creation of better-aligned models. Enhancing model safety can also be achieved by incorporating safety-centric guidelines during the RLHF phase. Despite its benefits, RLHF is heavily reliant on superior quality input from human annotators, making its practical implementation challenging. It is crucial, therefore, to refine the RLHF framework to lessen the burden on human annotators and explore more efficient annotation techniques to guarantee data quality. For instance, the use of models to facilitate annotation tasks could be a viable solution.

8 Conclusion

In this review, we provide a comprehensive review of the concepts of learning from models, covering methods such as model fine tuning, model distillation, model reuse, meta-learning, and model editing. Each approach uses a pre-trained model in a different way and in a different context for better performance or to solve a specific task. We systematically summarize relevant studies from early explorations to the present, not only providing a detailed overview of how each approach works, but also discussing their advantages and limitations. We took an in-depth look at how these approaches perform when dealing with real-world problems and the challenges that can be encountered during implementation. In addition, we also analyze the major challenges facing today, including issues such as interpretability, security, and fairness of models, and how to effectively fine-tune and edit models to adapt to new tasks or domains. Finally, we suggest some future directions, explore how these challenges can be addressed through further research, and how new technologies and approaches might advance the field. We believe this survey will benefit researchers in the fields of language models, knowledge graphs, knowledge repositories, and more, providing them with a comprehensive reference resource to understand current state-of-the-art methods and how they can be applied to improve their research and applications.

References

[1] T. Brown et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] H. Touvron et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[3] C. Saharia et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[4] J. Zamfirescu-Pereira et al. Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–21, 2023.

[5] I. L. Alberts et al. Large language models (llm) and chatgpt: what will the impact on nuclear medicine be? *European journal of nuclear medicine and molecular imaging*, 50(6):1549–1552, 2023.

[6] D. Ghosal et al. Text-to-audio generation using instruction-tuned llm and latent diffusion model. *arXiv preprint arXiv:2304.13731*, 2023.

[7] Z. Jiang et al. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.

[8] J. Gou et al. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[9] Y. Liu et al. Summary of chatgpt/gpt-4 research and perspective towards the future of large language models. *arXiv preprint arXiv:2304.01852*, 2023.

[10] Y. Wang et al. Model robustness meets data privacy: Adversarial robustness distillation without original data. *arXiv preprint arXiv:2303.11611*, 2023.

[11] N. Carlini et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

[12] C. Anil et al. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556, 2022.

[13] T. Wang et al. What language model architecture and pretraining objective works best for zero-shot generalization? In *International Conference on Machine Learning*, pp. 22964–22984. PMLR, 2022.

[14] Q. Zhong et al. Improving sharpness-aware minimization with fisher mask for better generalization on language models. *arXiv preprint arXiv:2210.05497*, 2022.

[15] D. Misra et al. Reprogramming under constraints: Revisiting efficient and reliable transferability of lottery tickets. *arXiv preprint arXiv:2308.14969*, 2023.

[16] E. Dupuis et al. A heuristic exploration of retraining-free weight-sharing for cnn compression. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 134–139. IEEE, 2022.

[17] J. Wei et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[18] M. Kuhn et al. Over-fitting and model tuning. *Applied predictive modeling*, pp. 61–92, 2013.

[19] D. Jiang et al. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*, 2023.

[20] T. Hospedales et al. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[21] E. Mitchell et al. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

[22] T. Chen et al. Big self-supervised models are strong semi-supervised learners. In H. Larochelle et al., editors, *Advances in Neural Information Processing Systems*, volume 33, pp. 22243–22255. Curran Associates, Inc., 2020.

[23] Z. Zhou et al. Fine-tuning convolutional neural networks for biomedical image analysis: Actively and incrementally. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[24] A. Radford et al. Improving language understanding by generative pre-training. 2018.

[25] A. Aghajanyan et al. Better fine-tuning by reducing representational collapse. *arXiv preprint arXiv:2008.03156*, 2020.

[26] H. Le et al. Lightweight adapter tuning for multilingual speech translation. *arXiv preprint arXiv:2106.01463*, 2021.

[27] L. Caccia et al. Multi-head adapter routing for cross-task generalization, 2023.

[28] Y. Wang et al. Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models. *arXiv preprint arXiv:2205.12410*, 2022.

[29] Z. Hu et al. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*, 2023.

[30] N. Maus et al. Black box adversarial prompting for foundation models, 2023.

[31] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021.

[32] A. Razdaibiedina et al. Progressive prompts: Continual learning for language models. *arXiv preprint arXiv:2301.12314*, 2023.

[33] M. Deng et al. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*, 2022.

[34] J. Li et al. Gradient-regulated meta-prompt learning for generalizable vision-language models, 2023.

[35] T. Sun et al. BBTv2: Towards a gradient-free future with large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[36] Z. Xu et al. Multiinstruct: Improving multi-modal zero-shot learning via instruction tuning, 2023.

[37] A. Rosenbaum et al. Linguist: Language model instruction tuning to generate annotated utterances for intent classification and slot tagging. *arXiv preprint arXiv:2209.09900*, 2022.

[38] H. Liu et al. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[39] S. Zhang et al. Gpt4roi: Instruction tuning large language model on region-of-interest. *arXiv preprint arXiv:2307.03601*, 2023.

[40] P. Liu et al. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[41] B. Lester et al. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[42] B. Min et al. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 2021.

[43] N. Tajbakhsh et al. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

[44] E. B. Zaken et al. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *CoRR*, abs/2106.10199, 2021.

[45] M. Wortsman et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7959–7971, 2022.

[46] A. Kumar et al. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

[47] N. Ruiz et al. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22500–22510, 2023.

[48] C. Chen et al. Deep learning on computational-resource-limited platforms: a survey. *Mobile Information Systems*, 2020:1–19, 2020.

[49] Y. Sun et al. Singular value fine-tuning: Few-shot segmentation requires few-parameters fine-tuning. *Advances in Neural Information Processing Systems*, 35:37484–37496, 2022.

[50] T. L. Hayes et al. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pp. 466–483. Springer, 2020.

[51] Z. Fu et al. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 12799–12807, 2023.

[52] N. Houlsby et al. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

[53] N. Ding et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

[54] J. Pfeiffer et al. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020.

[55] R. He et al. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164*, 2021.

[56] R. Karimi Mahabadi et al. Compacter: Efficient low-rank hypercomplex adapter layers. In M. Ranzato et al., editors, *Advances in Neural Information Processing Systems*, volume 34, pp. 1022–1035. Curran Associates, Inc., 2021.

[57] Z.-C. Chen et al. Exploring efficient-tuning methods in self-supervised speech models. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pp. 1120–1127. IEEE, 2023.

[58] S. Vander Eeckt and H. Van Hamme. Using adapters to overcome catastrophic forgetting in end-to-end automatic speech recognition. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

[59] Y. Gu et al. PPT: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8410–8423, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[60] B. Lester et al. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[61] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. 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)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics.

[62] G. Qin and J. Eisner. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5203–5212, Online, June 2021. Association for Computational Linguistics.

[63] X. Liu et al. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021.

[64] X. Han et al. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192, 2022.

[65] Z. Jiang et al. Rethinking efficient tuning methods from a unified perspective. *arXiv preprint arXiv:2303.00690*, 2023.

[66] Y. Wen et al. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *arXiv preprint arXiv:2302.03668*, 2023.

[67] T. Sun et al. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pp. 20841–20855. PMLR, 2022.

[68] S. Diao et al. Black-box prompt learning for pre-trained language models. *arXiv preprint arXiv:2201.08531*, 2022.

[69] G. Xiao et al. Offsite-tuning: Transfer learning without full model. *arXiv preprint arXiv:2302.04870*, 2023.

[70] B. Hou et al. Promptboosting: Black-box text classification with ten forward passes. *arXiv preprint arXiv:2212.09257*, 2022.

[71] S. Longpre et al. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*, 2023.

[72] B. Peng et al. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.

[73] J. Wei et al. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

[74] V. Sanh et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

[75] L. Ouyang et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[76] J. Jang et al. Exploring the benefits of training expert language models over instruction tuning, 2023.

[77] B. Li et al. Otter: A multi-modal model with in-context instruction tuning. *arXiv preprint arXiv:2305.03726*, 2023.

[78] S. Borgeaud et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

[79] M. Yasunaga et al. Deep bidirectional language-knowledge graph pretraining. *Advances in Neural Information Processing Systems*, 35:37309–37323, 2022.

[80] B. Wang et al. Shall we pretrain autoregressive language models with retrieval? a comprehensive study. *arXiv preprint arXiv:2304.06762*, 2023.

[81] E. Grave et al. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*, 2016.

[82] U. Khandelwal et al. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

[83] K. Guu et al. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.

[84] P. Lewis et al. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[85] G. Izacard and E. Grave. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*, 2020.

[86] G. Izacard and E. Grave. Distilling knowledge from reader to retriever for question answering. *arXiv preprint arXiv:2012.04584*, 2020.

[87] G. Izacard et al. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*, 2022.

[88] W. Shi et al. Replug: Retrieval-augmented black-box language models, 2023.

[89] W. Chen et al. Murag: Multimodal retrieval-augmented generator for open question answering over images and text. *arXiv preprint arXiv:2210.02928*, 2022.

[90] W. Chen et al. Re-imagen: Retrieval-augmented text-to-image generator. *arXiv preprint arXiv:2209.14491*, 2022.

[91] M. Yasunaga et al. Retrieval-augmented multimodal language modeling. 2023.

[92] H. Li et al. A survey on retrieval-augmented text generation, 2022.

[93] Y. Liu et al. Data-free knowledge transfer: A survey. *arXiv preprint arXiv:2112.15278*, 2021.

[94] G. Hinton et al. Distilling the Knowledge in a Neural Network, March 2015. arXiv:1503.02531 [cs, stat].

[95] A. Romero et al. FitNets: Hints for Thin Deep Nets, March 2015. arXiv:1412.6550 [cs].

[96] A. Koratana et al. LIT: Learned Intermediate Representation Training for Model Compression. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3509–3518. PMLR, May 2019. ISSN: 2640-3498.

[97] H. Chen et al. Learning Student Networks via Feature Embedding, December 2018. arXiv:1812.06597 [cs, stat].

[98] S. Ahn et al. Variational Information Distillation for Knowledge Transfer, April 2019. arXiv:1904.05835 [cs].

[99] H.-J. Ye et al. Distilling Cross-Task Knowledge via Relationship Matching. pp. 12396–12405, 2020.

[100] F. Petroni et al. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics.

[101] Y. Liu et al. Data-Free Knowledge Transfer: A Survey, December 2021. arXiv:2112.15278 [cs].

[102] S. Braun et al. Deep Classifier Mimicry without Data Access, June 2023. arXiv:2306.02090 [cs].

[103] Z. Hu et al. Learning to Learn from APIs: Black-Box Data-Free Meta-Learning. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 13610–13627. PMLR, July 2023. ISSN: 2640-3498.

[104] D. Roman et al. Model as a service (maas). In *AGILE Workshop: Grid Technologies for Geospatial Applications, Hannover, Germany*, 2009.

[105] J. M. Joyce. Kullback-leibler divergence. In *International encyclopedia of statistical science*, pp. 720–722. Springer, 2011.

[106] R. G. Lopes et al. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017.

[107] K. Bhardwaj et al. Dream distillation: A data-independent model compression framework. *arXiv preprint arXiv:1905.07072*, 2019.

[108] G. K. Nayak et al. Zero-shot knowledge distillation in deep networks. In *International Conference on Machine Learning*, pp. 4743–4751. PMLR, 2019.

[109] A. Mordvintsev et al. Inceptionism: Going deeper into neural networks. 2015.

[110] H. Yin et al. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8715–8724, 2020.

[111] L. Zhu et al. Deep leakage from gradients. In H. Wallach et al., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[112] J. Geiping et al. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[113] H. Yin et al. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16337–16346, 2021.

[114] I. Goodfellow et al. Generative adversarial nets. In Z. Ghahramani et al., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[115] H. Chen et al. Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3514–3522, 2019.

[116] J. Ye et al. Data-free knowledge amalgamation via group-stack dualgan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12516–12525, 2020.

[117] J. Yoo et al. Knowledge extraction with no observable data. In H. Wallach et al., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[118] S. Addepalli et al. Degan: Data-enriching gan for retrieving representative samples from a trained classifier. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3130–3137, 2020.

[119] P. Han et al. Robustness and diversity seeking data-free knowledge distillation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2740–2744. IEEE, 2021.

[120] H. Chen et al. Learning student networks in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6428–6437, 2021.

[121] P. Micaelli and A. J. Storkey. Zero-shot knowledge transfer via adversarial belief matching. In H. Wallach et al., editors, *Advances in Neural Information Processing Systems 32*, pp. 9551–9561. Curran Associates, Inc., 2019.

[122] G. Fang et al. Data-free adversarial distillation. *arXiv preprint arXiv:1912.11006*, 2019.

[123] G. Fang et al. Up to 100x faster data-free knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6597–6604, 2022.

[124] R. He et al. IS SYNTHETIC DATA FROM GENERATIVE MODELS READY FOR IMAGE RECOGNITION? In *The Eleventh International Conference on Learning Representations*, 2023.

[125] G. Patel et al. Learning to retain while acquiring: Combating distribution-shift in adversarial data-free knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7786–7794, 2023.

[126] P. Rani et al. Machine learning and deep learning based computational approaches in automatic microorganisms image recognition: methodologies, challenges, and developments. *Archives of Computational Methods in Engineering*, 29(3):1801–1837, 2022.

[127] H.-y. Lee et al. Meta learning for natural language processing: A survey. *arXiv preprint arXiv:2205.01500*, 2022.

[128] Y. Matsuo et al. Deep learning, reinforcement learning, and world models. *Neural Networks*, 152:267–275, 2022.

[129] T. G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pp. 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[130] O. Sagi and L. Rokach. Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1249, 2018. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1249.

[131] Z. Zhou and J. Feng. Deep forest: Towards an alternative to deep neural networks. *CoRR*, abs/1702.08835, 2017.

[132] A. Aniol et al. Ensemble Approach for Natural Language Question Answering Problem. pp. 180–183. IEEE Computer Society, November 2019.

[133] B. Wang et al. A Chinese Question Answering Approach Integrating Count-Based and Embedding-Based Features. In C.-Y. Lin et al., editors, *Natural Language Understanding and Intelligent Applications*, Lecture Notes in Computer Science, pp. 934–941, Cham, 2016. Springer International Publishing.

[134] L. Deng and J. C. Platt. Ensemble Deep Learning for Speech Recognition.

[135] Y. Wen et al. BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning, February 2020.

[136] T. Y. Liu and S. Soatto. Tangent Model Composition for Ensembling and Continual Fine-tuning, July 2023.

[137] W. Li et al. Deep model fusion: A survey. *arXiv preprint arXiv:2309.15698*, 2023.

[138] J. Frankle et al. Linear Mode Connectivity and the Lottery Ticket Hypothesis, July 2020.

[139] D. Yunis et al. On convexity and linear mode connectivity in neural networks. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*, 2022.

[140] M. Wortsman et al. Model soups: averaging weights of multiple finetuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pp. 23965–23998. PMLR, 2022.

[141] A. Chronopoulou et al. AdapterSoup: Weight Averaging to Improve Generalization of Pretrained Language Models, March 2023. arXiv:2302.07027 [cs] Read_Status: Read Read_Status_Date: 2023-08-31T09:57:45.371Z.

[142] P. Izmailov et al. Averaging Weights Leads to Wider Optima and Better Generalization, February 2019.

[143] M. Wortsman et al. Robust fine-tuning of zero-shot models, June 2022.

[144] P. Lu et al. Improving generalization of pre-trained language models via stochastic weight averaging. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 4948–4954, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[145] J. Kaddour. Stop Wasting My Time! Saving Days of ImageNet and BERT Training with Latest Weight Averaging, October 2022.

[146] S. Sanyal et al. Understanding the Effectiveness of Early Weight Averaging for Training Large Language Models, June 2023.

[147] H. B. McMahan et al. Communication-Efficient Learning of Deep Networks from Decentralized Data, January 2023.

[148] G. Ilharco et al. Patching open-vocabulary models by interpolating weights, October 2022. arXiv:2208.05592 [cs].

[149] M. Matena and C. Raffel. Merging Models with Fisher-Weighted Averaging, August 2022. arXiv:2111.09832 [cs].

[150] A. Jolicoeur-Martineau et al. PopulAtion Parameter Averaging (PAPA), May 2023. arXiv:2304.03094 [cs].

[151] G. Ilharco et al. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[152] T. Garipov et al. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs, October 2018. arXiv:1802.10026 [cs, stat].

[153] S. K. Ainsworth et al. Git Re-Basin: Merging Models modulo Permutation Symmetries, March 2023. arXiv:2209.04836 [cs] Read_Status: Read Read_Status_Date: 2023-08-31T10:29:51.597Z.

[154] George Stoica et al. ZipIt! Merging Models from Different Tasks without Training, May 2023. arXiv:2305.03053 [cs].

[155] S. P. Singh and M. Jaggi. Model Fusion via Optimal Transport, May 2023.

[156] X. Jin et al. Dataless Knowledge Fusion by Merging Weights of Language Models, April 2023. arXiv:2212.09849 [cs].

[157] Y. Lou et al. Towards Efficient Front-End Visual Sensing for Digital Retina: A Model-Centric Paradigm. *IEEE Transactions on Multimedia*, 22(11):3002–3013, 2020. ISBN: 1901014738.

[158] C. Wu et al. π-Tuning: Transferring Multimodal Foundation Models with Optimal Multi-task Interpolation, May 2023. arXiv:2304.14381 [cs].

[159] D. Yang et al. Boosting the Adversarial Transferability of Surrogate Models with Dark Knowledge, September 2023. arXiv:2206.08316 [cs].

[160] T. Wei et al. NTK-approximating MLP Fusion for Efficient Language Model Fine-tuning, July 2023. arXiv:2307.08941 [cs].

[161] T. Y. Liu and S. Soatto. Tangent Model Composition for Ensembling and Continual Fine-tuning, July 2023. arXiv:2307.08114 [cs] Read_Status: Read Read_Status_Date: 2023-08-31T09:58:40.830Z.

[162] H.-Y. Chen et al. Federated Learning of Shareable Bases for Personalization-Friendly Image Classification, April 2023. arXiv:2304.07882 [cs].

[163] J. Schmidhuber et al. Simple principles of metalearning. *Technical report IDSIA*, 69:1–23, 1996.

[164] J. X. Wang. Meta-learning in natural and artificial intelligence. *Current Opinion in Behavioral Sciences*, 38:90–95, 2021.

[165] M. Wortsman et al. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6750–6759, 2019.

[166] J. Zhang et al. Progressive meta-learning with curriculum. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(9):5916–5930, 2022.

[167] Z. Chi et al. Metafscil: A meta-learning approach for few-shot class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14166–14175, 2022.

[168] T. Nguyen and A. Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. *arXiv preprint arXiv:2207.04179*, 2022.

[169] P. K. Chan and S. J. Stolfo. Experiments on multistrategy learning by meta-learning. In *Proceedings of the second international conference on information and knowledge management*, pp. 314–323, 1993.

[170] Y. Bengio et al. *Learning a synaptic learning rule*. Citeseer, 1990.

[171] S. Bengio et al. On the search for new learning rules for anns. *Neural Processing Letters*, 2:26–30, 1995.

[172] S. Thrun and L. Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[173] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18:77–95, 2002.

[174] Z. Wang et al. Meta-learning without data via wasserstein distributionally-robust model fusion. In *Uncertainty in Artificial Intelligence*, pp. 2045–2055. PMLR, 2022.

[175] OpenAI. Gpt-4 technical report, 2023.

[176] R. Thoppilan et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[177] C. Finn et al. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

[178] M. Caccia et al. Online fast adaptation and knowledge accumulation (osaka): a new approach to continual learning. *Advances in Neural Information Processing Systems*, 33:16532–16545, 2020.

[179] C. Simon et al. Meta-learning for multi-label few-shot classification. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 3951–3960, 2022.

[180] Z. Wang et al. Meta-learning with less forgetting on large-scale non-stationary task distributions. In *European Conference on Computer Vision*, pp. 221–238. Springer, 2022.

[181] Z. Wang et al. Learning to learn and remember super long multi-domain task sequence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7982–7992, 2022.

[182] Q. Sun et al. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 403–412, 2019.

[183] Q. Sun et al. Meta-transfer learning through hard tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1443–1456, 2020.

[184] G. Zhang et al. Meta-detr: Few-shot object detection via unified image-level meta-learning. *arXiv preprint arXiv:2103.11731*, 2(6), 2021.

[185] Y. Chai et al. Cross-domain deep code search with meta learning. In *Proceedings of the 44th International Conference on Software Engineering*, pp. 487–498, 2022.

[186] Z. Feng et al. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020.

[187] A. Nichol et al. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[188] Z. Hu et al. Learning to learn from apis: Black-box data-free meta-learning. *arXiv preprint arXiv:2305.18413*, 2023.

[189] Z. Hu et al. Architecture, dataset and model-scale agnostic data-free meta-learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7736–7745, 2023.

[190] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2016.

[191] F. Sung et al. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1199–1208, 2018.

[192] Z. Jiang et al. Film: How can few-shot image classification benefit from pre-trained language models? *arXiv preprint arXiv:2307.04114*, 2023.

[193] J. Du et al. Query-efficient meta attack to deep neural networks. *arXiv preprint arXiv:1906.02398*, 2019.

[194] Z. Yuan et al. Meta gradient adversarial attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7748–7757, 2021.

[195] Y. Qin et al. Training meta-surrogate model for transferable adversarial attack. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9516–9524, 2023.

[196] Y. Chen et al. D-dae: Defense-penetrating model extraction attacks. In *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 382–399, 2023.

[197] X. Zhu and S. Li. Mgml: Momentum group meta-learning for few-shot image classification. *Neurocomputing*, 514:351–361, 2022.

[198] Z. Tang et al. Meta-learning-based optimal control for soft robotic manipulators to interact with unknown environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 982–988. IEEE, 2023.

[199] S. Watanabe et al. Speeding up multi-objective hyperparameter optimization by task similarity-based meta-learning for the tree-structured parzen estimator. *arXiv preprint arXiv:2212.06751*, 2023.

[200] K. M. Stewart and E. O. Neftci. Meta-learning spiking neural networks with surrogate gradient descent. *Neuromorphic Computing and Engineering*, 2(4):044002, 2022.

[201] Y. Jiang et al. The role of deconfounding in meta-learning. In *International Conference on Machine Learning*, pp. 10161–10176. PMLR, 2022.

[202] X. Zhang et al. Distributed reptile algorithm for meta-learning over multi-agent systems. *IEEE Transactions on Signal Processing*, 70:5443–5456, 2022.

[203] X. Song et al. Efficient and effective multi-task grouping via meta learning on task combinations. *Advances in Neural Information Processing Systems*, 35:37647–37659, 2022.

[204] K. Gao and O. Sener. Modeling and optimization trade-off in meta-learning. *Advances in Neural Information Processing Systems*, 33:11154–11165, 2020.

[205] P. H. T. Gama et al. An overview on meta-learning approaches for few-shot weakly-supervised segmentation. *Computers & Graphics*, 2023.

[206] Y. Elazar et al. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031, 2021.

[207] A. Lazaridou et al. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34:29348–29363, 2021.

[208] Y. Yao et al. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*, 2023.

[209] A. Sinitsin et al. Editable neural networks. *arXiv preprint arXiv:2004.00345*, 2020.

[210] N. De Cao et al. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*, 2021.

[211] P. Hase et al. Do language models have beliefs? methods for detecting, updating, and visualizing model beliefs. *arXiv preprint arXiv:2111.13654*, 2021.

[212] A. Madaan et al. Memory-assisted prompt editing to improve gpt-3 after deployment, 2023.

[213] Z. Huang et al. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*, 2023.

[214] Q. Dong et al. Calibrating factual knowledge in pretrained language models. *arXiv preprint arXiv:2210.03329*, 2022.

[215] Y. Onoe et al. Can lms learn new entities from descriptions? challenges in propagating injected knowledge. *arXiv preprint arXiv:2305.01651*, 2023.

[216] S. Murty et al. Fixing model bugs with natural language patches. *arXiv preprint arXiv:2211.03318*, 2022.

[217] C. Zhu et al. Modifying memories in transformer models, 2020.

[218] D. Dai et al. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

[219] K. Meng et al. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022.

[220] G. Ilharco et al. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[221] K. Meng et al. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

[222] V. Raunak and A. Menezes. Rank-one editing of encoder-decoder models. *arXiv preprint arXiv:2211.13317*, 2022.

[223] G. Luo et al. Prompt-based editing for text style transfer. *arXiv preprint arXiv:2301.11997*, 2023.

[224] Y. Zhu et al. Conditional text image generation with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14235–14245, 2023.

[225] R. Cohen et al. Crawling the internal knowledge-base of language models, 2023.

[226] B. Cao et al. The life cycle of knowledge in big language models: A survey. *arXiv preprint arXiv:2303.07616*, 2023.