

Neural Network Reprogrammability: A Unified Theme on Model Reprogramming, Prompt Tuning, and Prompt Instruction

Zesheng Ye^{◇*}, Chengyi Cai^{◇*}, Ruijiang Dong^{◇*}, Jianzhong Qi[◇], Lei Feng[♠], Pin-Yu Chen[♠] and Feng Liu[◇]

◇ University of Melbourne, ♠ Southeast University, ♣ IBM Research

Abstract

As large-scale pre-trained foundation models continue to expand in size and capability, efficiently adapting them to specific downstream tasks has become increasingly critical. Despite substantial research progress, existing adaptation approaches have evolved largely in isolation, without a clear understanding of their interrelationships. This survey introduces *neural network reprogrammability* as a unifying framework that bridges mainstream model adaptation techniques—model reprogramming, prompt tuning, and prompt instruction—previously fragmented research areas yet converges on a shared principle: repurposing a pre-trained model by *manipulating information at the interfaces* while keeping the model parameters frozen. These methods exploit neural networks’ sensitivity to information manipulation on different interfaces, be it through perturbing inputs, inserting tokens into intermediate layers, or providing task-specific examples in context, to redirect model behaviors towards desired outcomes. Building on the concept of reprogrammability, we present a taxonomy that categorizes such information manipulation-based adaptation approaches across four key dimensions: manipulation format (fixed or learnable), location (interfaces where manipulations occur), operator (how they are applied), and output alignment requirement (post-processing needed to align outputs with downstream tasks). Notably, this framework applies consistently across data modalities, independent of specific model architectures. Moreover, viewing established techniques such as *in-context learning* and *chain-of-thought* prompting through this lens reveals both their theoretical connections and practical distinctions. We further analyze remaining technical challenges and ethical considerations, positioning neural network reprogrammability as a fundamental paradigm for efficient model adaptation. We lastly identify promising research directions emerging from this integrative viewpoint.

*Equal contributions.

Correspondence: fengliu.ml@gmail.com



1. Introduction

The rise of foundation models is revolutionizing the landscape of modern machine learning. By learning general-purpose data representations, these powerful pre-trained architectures have delivered unprecedented results in computer vision (Dosovitskiy et al., 2021; Zhao et al., 2023) and natural language processing (Brown et al., 2020; Devlin et al., 2019; Vaswani et al., 2017), but their massive scale—often billions of parameters—creates significant deployment challenges.

Conventionally, adapting these models to specific downstream tasks necessitates an intensive process of

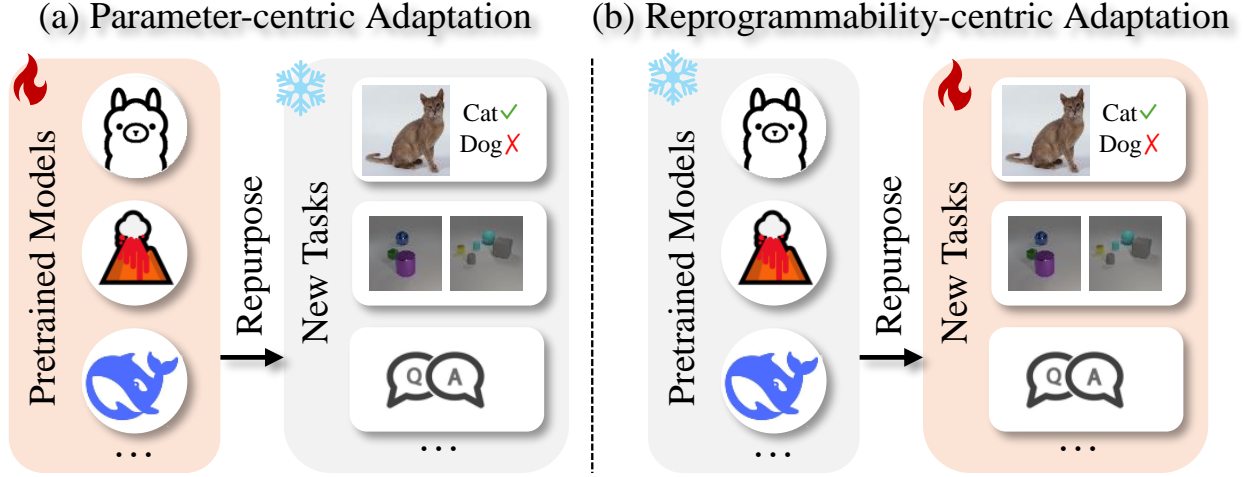


Figure 1 | Paradigm shift from conventional *parameter-centric adaptation* (i.e., modifying model parameters) to *reprogrammability-centric adaptation* (i.e., modifying input data and model output). This represents a shift in thought from *modifying the model to align with the task* to *modifying the task to align with the model*.

fine-tuning, where a substantial portion, if not all, of the model’s *parameters* are updated using task-specific data (Chen, 2024; Pan and Yang, 2009). This *parameter-centric adaptation strategy* (PCA), depicted in Figure 1 (a), directly modifies the model’s learned weights. While straightforward and often effective, PCA may come with several limitations, especially when resources are constrained. The computational demands of retraining these models often become prohibitively expensive. Access to model internals may not be available often when models are provided as services (i.e., black-box setting). Furthermore, the need for specialized hardware (e.g., high-performance GPUs) creates barriers to entry, and the significant energy footprint raises valid environmental concerns.

A more efficient paradigm is gaining traction. Rather than *modifying the internal parameters*, a growing body of work now focuses on adapting pre-trained models for downstream tasks (i.e., target tasks) *without* extensive (or, in some cases, any) *parameter updates*. As Figure 1 (b) shows, these approaches strategically manipulate the target task-specific inputs or contextual information provided to a *fixed* pre-trained model, and then align the model output to the requirement of the target task’s output space. In this paper, we term such methods *reprogrammability-centric adaptation* (RCA) and formally define the concept of *reprogrammability* in Sec. 2.1. Importantly, the shift of adaptation focus in RCA methods leads to a compelling benefit: model adaptation costs (in terms of trainable parameters) become largely independent of model size, correlating instead with the complexity of the task itself. Namely, the input and output dimensionality of the target data.

The efficiency benefits can be substantial. Consider a typical neural network, such as a *multi-layer perceptron* (MLP) with L layers of dimension D processing inputs of dimension d , we typically find that $D \times L \gg d$ —the model’s complexity substantially exceeds that of the data it processes since neural networks are heavily over-parameterized (Allen-Zhu et al., 2019; Zhang et al., 2017). This implies that RCA methods can reduce the training burden as they shift the model adaptation efforts from the parameter space to a much lower-dimensional input space. Figure 2 showcases a concrete illustration, where we compare the number of trainable parameters when adapting an ImageNet-pretrained ViT/B-32 (Dosovitskiy et al., 2021) to a new remote sensing image classification task (Helber et al., 2019). The comparison is between various fine-tuning (i.e., PCA) scenarios and reprogramming (i.e., RCA), confirming that RCA consistently involves significantly fewer trainable parameters than most PCA scenarios. The implications of this efficiency are far-reaching. By significantly reducing computational demands, RCA methods promise to democratize

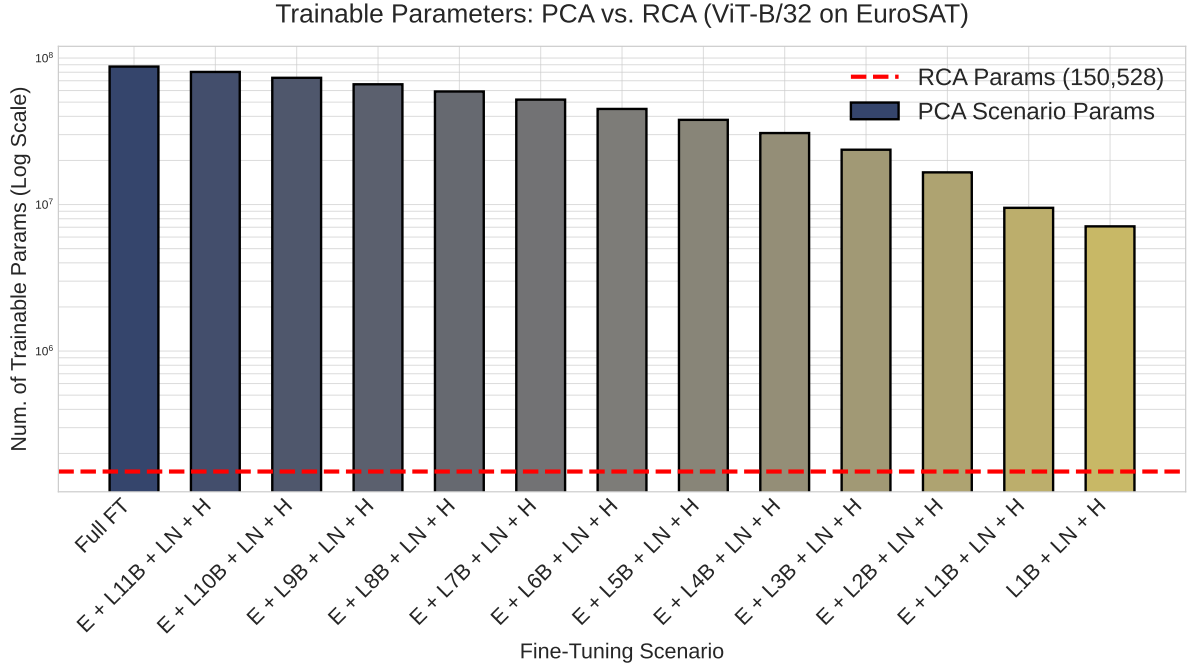


Figure 2 | Comparison of trainable parameters for fine-tuning (i.e., PCA) vs. reprogramming (i.e., RCA) an ImageNet-pretrained ViT/B-32 on a new classification task. Different PCA scenarios are shown (e.g., "E+L10B+LN+H" indicates fine-tuning the embedding layer, the last 10 Transformer blocks, layer normalization, and the classification head). The dashed line indicates the number of trainable parameters for RCA, which is consistently lower than that of most PCA configurations.

access to cutting-edge AI capabilities, enable deployment in resource-constrained environments, and reduce the carbon footprint of AI applications.

1.1. Motivation: The Need for Terminological Clarity

Despite the promise of this emerging paradigm, a significant obstacle to progress in this field is the bewildering array of inconsistent terminology. Techniques like "prompt tuning" and "model reprogramming" and variations thereof are often used interchangeably or, worse, with conflicting definitions across research communities, such as the same terms applied to fundamentally different approaches. Consider how the term "prompt tuning" operates in the literature. In some contexts, it broadly encompasses any modifications to the raw input data (Brown et al., 2020), yet in other contexts, it specifically refers to adding trainable continuous vectors to input embeddings (Lester et al., 2021; Li and Liang, 2021). Similarly, "model reprogramming" can denote input transformations (Chen, 2024; Elsayed et al., 2019) and, elsewhere, describe manipulations within embedding spaces (Neekhara et al., 2019). This terminological ambiguity creates unnecessary barriers to knowledge transfer across research communities. Accordingly, researchers who work on similar problems may struggle to identify and build upon each other's work or conduct fair comparisons, ultimately slowing progress in such a field with vast potential to broaden access to state-of-the-art machine learning capabilities.

1.2. Purpose: Unifying Disparate Approaches

To address these issues, this survey introduces *Neural Network Reprogrammability* as a *unifying conceptual framework* to bring clarity and structure to the diverse landscape of parameter-efficient adaptation methods. We define *reprogrammability* as the inherent capability of pre-trained models to be repurposed for new

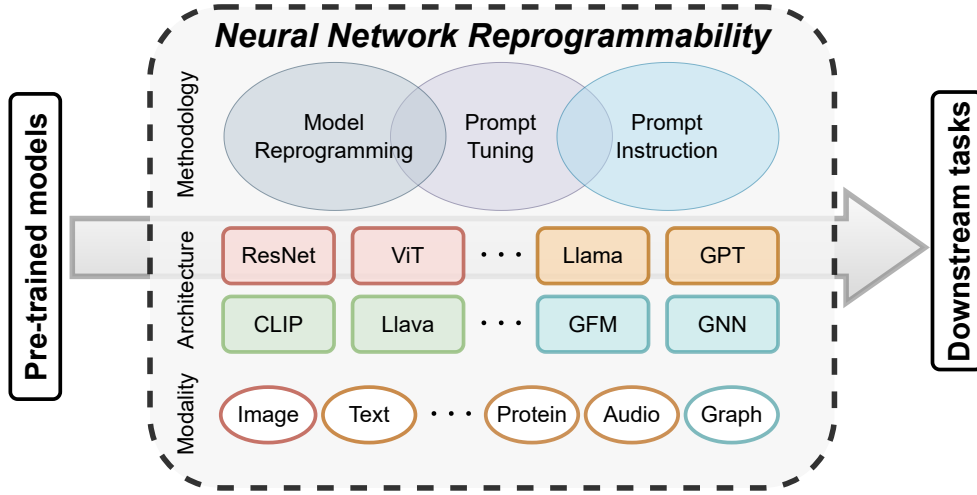


Figure 3 | We introduce *Neural Network Reprogrammability* as a unifying framework to bring coherence to a set of model adaptation techniques that have often been studied in isolation. Importantly, this shared underlying principle applies regardless of pre-trained model architectures and data modalities.

tasks *without internal parameter updates*; whilst the adaptation is achieved by strategically manipulating the inputs (and/or auxiliary) information provided to the model (Chen, 2024; Elsayed et al., 2019) and the output model produces, effectively leveraging neural networks’ inherent input-output sensitivity (Lowd and Meek, 2005) to elicit desired task-specific behaviors.

Reprogrammability provides a coherent lens through which to view three major families of work: *model reprogramming* (MR) (Chen, 2024; Elsayed et al., 2019), *prompt tuning* (PT) (Lester et al., 2021; Li and Liang, 2021), and *in-context learning* (Brown et al., 2020; Kirillov et al., 2023), which we refer to as *prompt instruction* (PI) throughout this paper to avoid confusion with PT. Although often developed in parallel across different communities, we argue that they all rely on a common underlying principle that has yet to be formally recognized and consolidated. This commonality motivates grouping them under the unifying concept of *reprogrammability-centric adaptation* (RCA). This perspective reveals the connections between seemingly disparate methods, showing them as variations on a common theme: the systematic exploitation of a neural network’s sensitivity to manipulations at its interfaces.

With this unified view (Figure 3), we aim to bridge previously disparate research efforts by uncovering the overarching principles of adaptation that hold across different techniques, data modalities, and applications. By examining these methods through the common lens of *neural network reprogrammability*, we intend to clarify their relationships, foster more precise cross-community terminology and discussions, and stimulate deeper investigations into the core mechanisms. Ultimately, this should accelerate the creation of more effective, efficient, and broadly accessible means for adapting large-scale pre-trained models.

1.3. Scope and Contributions of this survey

Before outlining our contributions, we first acknowledge the extensive body of existing work and delineate the scope of this survey with existing literature. The field has seen remarkable survey papers covering specific methodologies and applications in detail, from model reprogramming (Chen, 2024) and comprehensive reports on the prompting technique (Schulhoff et al., 2024), to surveys on textual (Luo et al., 2024) and visual (Zhang et al., 2023a) in-context learning, and prompt engineering for visual-language models (Gu et al., 2023) or even broader multi-modal architectures (Wu et al., 2024b). To avoid redundancy and maintain a clear focus, our work will not replicate these specialized reviews.

Survey Focus	Gu et al. (2023)	Schulhoff et al. (2024)	Luo et al. (2024)	Zhang et al. (2023a)	Wu et al. (2024b)	Chen (2024)	Ours
In-Modal Progress	✓	✓	✓	✓	✓		
Application-targeted	✓	✓	✓	✓	✓		
Arch-agnostic	✓				✓	✓	✓
Cross-Field		✓				✓	✓
Unifying Paradigm							✓

Table 1 | Comparison with existing surveys related to reprogramming and prompting techniques. Unlike existing surveys that focus on specific modalities, applications, model architectures, or benchmarks, this work aims to systematically organize the diverse (and sometimes miscellaneous) methodologies and terminologies under a unified conceptual framework with a clear taxonomy.

Relation to Existing Surveys. While several recent surveys have covered related areas, such as prompting or parameter-efficient fine-tuning (Chen, 2024; Gu et al., 2023; Schulhoff et al., 2024; Zhang et al., 2023a), this survey distinguishes itself by offering a distinct perspective and presenting *neural network reprogrammability* as a central concept to unite different adaptation families (MR, PT, PI) that have no internal parameter updates. Existing surveys typically focus on tracking advancements within a specific technique (e.g., prompting or model reprogramming) or application area. In contrast, we propose a more general framework (Section 2) that spans traditionally separate research communities, along with a systematic taxonomy (Section 3) that classifies RCA methods along four key dimensions. This helps reveal connections that remain potentially hidden when these approaches are studied in isolation (See Table 1 for a comparative summary).

Contributions. This paper presents a structured overview for adapting pre-trained models without updating their parameters. We introduce *neural network reprogrammability* as the core concept and a **new** perspective to unify what has become a fragmented and often terminologically confusing area of research. We connect parameter-efficient adaptation techniques used across diverse modalities, emphasizing their shared strategy of input/output manipulation rather than parameter modification. This survey makes four primary contributions:

- 1) **A Conceptual Unification.** We show that model reprogramming, prompt tuning, and prompt instruction—techniques often developed independently—share fundamental principles. They all exploit pre-trained representations and guide model behaviors by carefully crafting inputs and outputs. We provide a conceptual framework that bridges these distinct lines of study.
- 2) **A Modality-Agnostic Taxonomy.** We develop a systematic way to classify reprogramming and prompting techniques that transcends specific domains (text, images, etc.) or architectures, such as ResNet (He et al., 2016), ViT (Dosovitskiy et al., 2021), BERT (Devlin et al., 2019), and CLIP (Radford et al., 2021). Namely, this taxonomy organizes RCA methods based on *what* (i.e., whether manipulations are “hard” or “soft” ones), *how* (i.e., which operator is relied on to insert manipulations), and *where* (i.e., locations where manipulations are applied to) the input or context is manipulated, as well as *whether* post-processing is required to align pre-trained model outputs with target tasks. This taxonomy enables a clear comparison of different RCA approaches.
- 3) **A Comparative Analysis.** We place the proposed unified framework RCA within a broad context of model adaptation research, highlighting connections and differences with other approaches. Unlike surveys focused narrowly on specific techniques or domains, this paper examines how reprogrammability enables model reuse across diverse tasks and architectures and touches upon underlying theoretical aspects typically overlooked by previous studies.
- 4) **A Discussion of Open Challenges and Future Directions.** We identify fundamental open questions regarding the theoretical foundations of reprogrammability, analyze limitations of current evaluation methods, and discuss emerging ethical issues, aiming to catalyze future research in this rapidly evolving field.

2. A Unified View of Reprogrammability

Here, we define reprogrammability as the capacity of *fixed neural network* to be redirected toward tasks outside its original training task, by *manipulations* within its *input and output spaces*, such that the new task’s context can fit in the pre-trained model’s learned feature space. This concept unifies a range of parameter-efficient adaptation techniques by highlighting their shared reliance on the model’s intrinsic sensitivity to input and/or output transformations. Later, in Sec. 3, we will show how existing model adaptation strategies fit this definition and can be categorized along four dimensions: (1) the *format* of the manipulation, (2) its *location* within the model, (3) the *operator* used, and (4) the need for *output alignment*.

2.1. From Vulnerability to Adaptation

Input Sensitivity. The remarkable capabilities in end-to-end representation learning of neural networks are juxtaposed by a paradoxical property: their susceptibility to adversarial manipulation (Carlini and Wagner, 2017a; Chen et al., 2018; Goodfellow et al., 2014). Studies in adversarial machine learning have consistently shown that even high-performing models can be misled by carefully crafted input perturbations that remain imperceptible to humans (Carlini and Wagner, 2017b; Lowd and Meek, 2005). This vulnerability is not merely an artifact of specific architectures or training procedures but appears to be rooted in the fundamentally discontinuous way neural networks map high-dimensional input spaces to outputs (Szegedy et al., 2014).

Several theories have attempted to explain this phenomenon. One hypothesis attributes it to the locally linear behavior of neural components, where small input changes can accumulate across layers, leading to large shifts in the final output (Goodfellow et al., 2014). Another perspective points to the geometry of high-dimensional spaces, where decision boundaries, while potentially far from data points in low dimensions, can become unexpectedly close in high dimensions, making them fragile to even subtle perturbations in the input space (Fawzi et al., 2016; Gilmer et al., 2018). Additionally, recent work suggests models might rely heavily on “non-robust features”—patterns statistically correlated with labels but easily altered adversarially—which contribute significantly to standard generalization but also to adversarial fragility (Ilyas et al., 2019). Such inherent input sensitivity means the learned function, while accurate on the training data distribution, possesses complex and sensitive decision boundaries when exploring regions slightly off the training data manifold (Chen, 2024; Elsayed et al., 2019).

Repurposing Sensitivity for Adaptation. This inherent input sensitivity, conventionally framed as a weakness, can however be reinterpreted as the *enabler of neural network reprogrammability*. The very mechanisms that allow adversarial examples to drastically alter model predictions—namely, the ability to find specific input directions that maximally change the output—can be repurposed. Thus, a seemingly detrimental vulnerability can be transformed beneficially. Instead of perturbing inputs to make a model fail (e.g., via gradient ascent on the loss function), we can engineer input transformations to guide a pre-trained model to perform tasks it was not originally designed or trained for. Such transformations eventually form the foundation of neural network reprogrammability.

Specifically, reprogrammability allows us to reuse a model’s established computational graph and learned feature hierarchy. Consider an image classifier pre-trained on ImageNet (Deng et al., 2009), which has learned a set of visual features including edges, textures, parts, and objects. Reprogrammability seeks to leverage this existing knowledge by learning transformation function(s) at the model’s interfaces—input, intermediate representations, and/or output—that map data from the new task (say, classifying medical images) into a format that effectively makes use of these pre-learned features and (potentially) then map model predictions to align with new label spaces. For instance, one might learn an input transformation that adds a particular pattern to medical images, causing the classifier to activate representations useful for the

new task, combined with a mapping from the ImageNet classes to the desired medical labels (Chen, 2024).

This viewpoint marks a paradigm shift in model adaptation. While traditional transfer learning methods, e.g., fine-tuning (Guo et al., 2022; Pan and Yang, 2009) and domain adaptation (Long et al., 2016, 2018), which focus on modifying weights of the source model, have been the de-facto practices, these approaches, especially the latter ones, find challenging source-free domain adaptation setting (Chi et al., 2021; Liang et al., 2020), where adaptation to a new target domain must be performed using only the pre-trained source model and target domain data, without any access to the original source data¹. By contrast, with reprogrammability, models can be instead adapted by manipulating the data fed to a *fixed* function. This implies that the emphasis shifts from updating the model to transforming the data. Through careful design of transformations, we can effectively “reprogram” a model’s input-output mapping, without touching its core learned representations.

2.2. Formulation of Reprogrammability

We now formalize neural network reprogrammability to provide a precise framework for understanding diverse RCA approaches. This formulation captures how a fixed model can be repurposed for new tasks through strategic transformations at its interfaces.

Definition 1 (Neural Network Reprogrammability). Let $f(\mathbf{x}^S; \theta)$ be a pre-trained neural network with parameters θ , representing a learned mapping $f : \mathcal{X}^S \rightarrow \mathcal{Y}^S$ from the input space \mathcal{X}^S to the output space \mathcal{Y}^S in a source (i.e., pre-trained) domain $\mathcal{D}^S \subseteq \mathcal{X}^S \times \mathcal{Y}^S$. We consider *neural network reprogrammability* as the realization of $f(\mathbf{x}; \theta)$ that induces a *target functionality*, jointly defined over the target input and output spaces $\mathcal{X}^T \times \mathcal{Y}^T$. The reprogramming process comprises two *configurable mappings*. Namely, an *input manipulation* mapping $I_{\lambda, \tau, \ell}$ applied prior to f , and an *output alignment* mapping O_ω applied after f .

Input Manipulation. The central mechanism for adapting the pre-trained model f is *input manipulation* $I_{\lambda, \tau, \ell} : \mathcal{X}^T \times \mathcal{C} \rightarrow \mathcal{M}$, which transforms target input $\mathbf{x}^T \in \mathcal{X}^T$ before it, or its subsequent representations, are processed by parts of f . The manipulation occurs at one or multiple of the model’s interfaces $\mathcal{M} = \{\mathcal{X}^S, \mathcal{E}, \mathcal{H}\}$, which can be its raw input space \mathcal{X}^S , its embedding space \mathcal{E} , or its deeper hidden representation spaces \mathcal{H}^2 . Moreover, the manipulation can potentially be guided by contextual information \mathcal{C} , which refers to task-specific contexts and constraints guiding how the pre-trained model should be repurposed. This transformation is characterized by

- **Manipulation Configuration** $\lambda \in \Lambda$ that specifies the transformation parameters. We partition $\Lambda = \Lambda_{\text{fixed}} \cup \Lambda_{\text{learnable}}$, distinguishing between fixed configurations (i.e., “hard prompts”) and learnable parameters (i.e., “soft prompts”). For learnable configurations, λ is typically optimized by minimizing a task-specific objective function $\mathcal{L} : \mathcal{Y}^T \times \mathcal{Y}^T \rightarrow \mathbb{R}^+$.
- **Manipulation Location** $\ell \in \mathcal{M}$ that specifies where in the computational graph the manipulation occurs, including three options: Input layer ℓ_0 , i.e., modifications applied directly to raw input space \mathcal{X}^S ; embedding layer ℓ_1 , i.e., modifications applied to the embedding space \mathcal{E} ; intermediate layers ℓ_i for $i = \{2, \dots, n\}$, i.e., modifications applied to the hidden representation space \mathcal{H} .
- **Manipulation Operator** $\tau : \mathcal{X}^T \times \lambda \rightarrow \mathcal{M}$ that defines how manipulation is incorporated along with the target task data, and $\tau \in \{\text{add}, \text{concat}, \text{param}\}$. Generally, the operator can be addition $\tau_{\text{add}}(\mathbf{x}^T, \lambda) = \mathbf{x}^T + \lambda$ in cases where \mathbf{x}^T and λ are appropriately dimensioned, or concatenation

¹Many source-free methods still rely on updating parts of the model, such as batch normalization statistics or select layers, often through techniques like pseudo-labeling or entropy minimization (Dong et al., 2023b). This places them in a distinct category from the reprogrammability-based methods we discussed here.

²When manipulations occur at embedding or hidden layers, it is helpful to view these layers as processing units whose own inputs are being altered.

$\tau_{\text{concat}}(\mathbf{x}^T, \lambda) = [\mathbf{x}^T; \lambda]$, denoting the concatenation along a specified dimension, or parametric transformation $\tau_{\text{param}}(\mathbf{x}^T, \lambda) = \lambda(\mathbf{x}^T)$ with a function $\lambda(\cdot)$ that maps target input to a space compatible with the input space of the corresponding interface.

Moreover, multiple manipulations can be composed sequentially as $I_{\lambda, \tau, \ell} \triangleq I_{\lambda_n, \tau_n, \ell_n} \circ \dots \circ I_{\lambda_1, \tau_1, \ell_1}$, enabling sophisticated manipulation strategies that leverage different model interfaces. For this composition to be well-defined, the output space of $I_{\lambda_i, \tau_i, \ell_i}$ needs to be compatible with the input space of $I_{\lambda_{i+1}, \tau_{i+1}, \ell_{i+1}}$ for all $i \in \{1, \dots, n-1\}$ throughout the computational graph. Formally, $I_{\lambda, \tau, \ell}$ induces a mapping that transforms target domain input $\mathbf{x}^T \in \mathcal{X}^T$ and context $c \in \mathcal{C}$ to a sub-manifold $\mathcal{M}' \subset \mathcal{M}$ that intersects meaningfully with the effective feature spaces learned by f during its pre-training.

Output Alignment. Once the pre-trained model f processes the manipulated input to produce a source-domain output $y^S \in \mathcal{Y}^S$, this output often needs to be mapped to the format compatible with the target task. The output alignment mapping $O_\omega : \mathcal{Y}^S \rightarrow \mathcal{Y}^T$, parametrized by configuration $\omega \in \Omega$, this function performs this step, establishing necessary correspondences between source and target output spaces \mathcal{Y}^S and \mathcal{Y}^T . Key alignment methods include:

- **Identity Mapping.** The simplest approach, used when \mathcal{Y}^S is directly compatible with \mathcal{Y}^T , i.e., $\mathcal{Y}^S = \mathcal{Y}^T$, requiring no explicit alignment transformation. This is common in generative tasks where the model directly produces outputs in the desired format.
- **Structured Alignment.** This involves a mapping $O_\omega(\mathbf{y}^S) = \omega(\mathbf{y}^S)$, where $\omega(\cdot)$ is typically a pre-defined, fixed function (e.g., a rule-based parser or extractor), aiming to isolate or reformat task-relevant information from the raw output of the pre-trained model, often employed for tasks requiring specific output structures.
- **Statistical Alignment.** This involves a mapping $O_\omega(\mathbf{y}^S) = \omega \mathbf{y}^S$, where $\omega \in \{0, 1\}^{|\mathcal{Y}^T| \times |\mathcal{Y}^S|}$ is a binary matrix. The matrix ω is determined non-parametrically based on statistical correspondences between the model’s source predictions and the target labels, known as *label mapping* (Chen et al., 2023a; Tsai et al., 2020).
- **Linear Alignment.** This also uses a matrix transformation $O_\omega(\mathbf{y}^S) = \omega \mathbf{y}^S$, but here $\omega \in \mathbb{R}^{|\mathcal{Y}^T| \times |\mathcal{Y}^S|}$ is a real-valued matrix. The matrix ω can be obtained either by gradient-based backpropagation (Tsao et al., 2024), e.g., optimizing a linear probe trained on top of the frozen pre-trained model’s outputs; or statistically estimated without backpropagation, for example, by estimating the optimal probabilistic mapping matrix from observed statistics of source predictions and target labels (Cai et al., 2024a)³.

The choice of output alignment method depends heavily on the nature of the target task and the relationship between the source and target output spaces. linear projections and class remappings are often set for downstream classification tasks, whilst structured extraction is utilized for complex tasks like language modeling.

Without loss of generality, the *reprogrammed neural network* can be represented as

$$f'(\mathbf{x}^T; \theta, \lambda, c, \ell, \tau, \omega) \triangleq O_\omega \circ f \circ I_{\lambda, \tau, \ell}(\mathbf{x}^T, c), \quad (1)$$

by bringing together both transformation components.

Importantly, this composition achieves the target functionality *without modifying* θ . In practice, the adaptation process focuses entirely on determining optimal values for contextual guidance $c \in \mathcal{C}$, manipulation configurations $\lambda \in \Lambda$, and output alignment parameters ω . These configuration elements may be

³We note that Cai et al. (2024a) also relies on statistics and does not rely on gradient-based optimization. Still, here we distinguish it from the “statistical alignment” category as ω is not restricted to binary values and can represent more precise linear transformations.

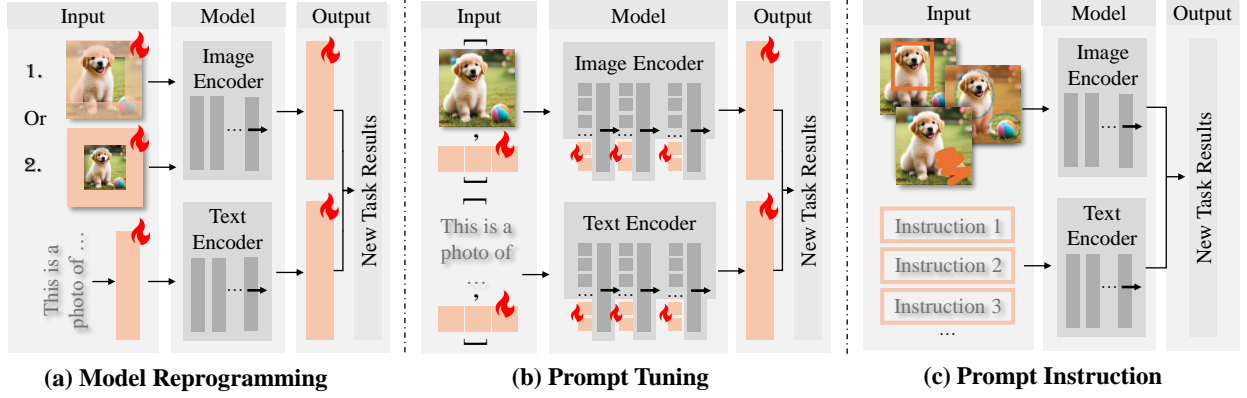


Figure 4 | Comparisons of implementations between MR, PT, and PI (using a pre-trained vision-language model as the example), distinguishing from each other by different manipulation forms, locations, operators, and output alignment strategies.

manually specified using domain expertise or learned through optimization against task-specific objectives. The implementation varies across adaptation method families, which we will detail in Sec. 3.

2.3. Revisit Existing Manifestations of Reprogrammability

The proposed framework (Definition 1) provides a lens through which we can understand several popular efficient adaptation families—*model reprogramming* (MR), *prompt tuning* (PT), and *prompt instruction* (PI)—as variations on a central theme. Despite emerging from different research communities and contexts, they primarily diverse in how they implement input manipulation $I(\cdot; \lambda, \tau, \ell, c)$ and output alignment $O(\cdot; \omega)$. Figure 4 visualizes these differences, and Table 2 summarizes a comparison of these approaches.

Model Reprogramming (MR). This represents an early structured approach (Chen, 2024; Elsayed et al., 2019) to repurpose pre-trained models without changing their weights, redirects model processing pathways through task-specific input perturbations to achieve desired outputs on a new target task. MR instantiates Eq. (1) by applying manipulation at either the input (for continuous data) or embedding interfaces (for discrete data), formally $\ell = \{\mathcal{X}^S, \mathcal{E}\}$, with optimizing $\lambda \in \Lambda_{\text{learnable}}$ against task-specific objective functions.

For continuous inputs such as images, MR commonly equips three kinds of manipulations, namely (1) *additive perturbation* $I_{\lambda, \text{add}, \mathcal{X}^S} = \text{reshape}(\mathbf{x}^T) + \lambda$ that superimposes learnable patterns onto reshaped inputs, preserving dimensionality while modifying feature activations; (2) *concatenative augmentation* $I_{\lambda, \text{concat}, \mathcal{X}^S} = [\mathbf{x}^T; \lambda_{\text{pad}}]$, extending the input dimensionality by concatenating learnable elements alongside the original data; (3) *transformative projection* $I_{\lambda, \text{transform}, \mathcal{X}^S} = \lambda(\mathbf{x}^T)$, a parameterized function λ that projects target inputs into a representation aligned with the pre-trained model’s input distribution. For discrete inputs such as text, MR typically operates at the embedding level by integrating learnable token representations into the embedding space. These learnable parameters λ serve as domain-bridging elements that activate relevant features within the model’s representation space, influenced by $c \in \mathcal{C}$ that encodes contextual information on perturbations.

Since MR usually adapts f to a new task with a different output space, the output alignment $O_\omega : \mathcal{Y}^S \rightarrow \mathcal{Y}^T$ is critical. In the case of classification tasks for example, O_ω is often implemented as a linear mapping $\omega \in \mathbb{R}^{|\mathcal{Y}^T| \times |\mathcal{Y}^S|}$ or a remapping permutation matrix $\omega \in \{0, 1\}^{|\mathcal{Y}^S| \times |\mathcal{Y}^T|}$, translating the model’s output categories to target task categories and making pre-trained model’s predictions meaningful in new tasks.

Prompt Tuning (PT). Initially developed for language models, now been extended to multimodal architectures, particularly vision-language models, PT (Lester et al., 2021; Li and Liang, 2021) also heavily employs learnable manipulations λ , but differ from MR in the manipulation interfaces. Often, PT involves prepending learnable tokens to the model’s embedding $\mathbf{e} \in \mathcal{E}$ or hidden representations $\mathbf{h} \in \mathcal{H}$.

PT instantiates Eq. (1) with $\tau = \text{concat}$ and $\ell \in \mathcal{E} \cup \mathcal{H}$. A canonical formulation of input manipulation is $I_{\lambda, \text{concat}, \mathcal{E}} = [\lambda_c; \mathbf{E}(\mathbf{x}^T)]$, where $\mathbf{E}(\mathbf{x}^T)$ is the embedding of target input \mathbf{x}^T and λ_c are task-specific prompt embeddings conditioned on the adaptation directive c , prepended to the input embeddings in the embedding space. PT can extend beyond a single interface, manipulating representations across multiple layers throughout the model architecture, such that $I_{\lambda, \tau, \ell}(\mathbf{x}^T, c) = I_{\lambda_n, \text{concat}, \mathcal{H}_n} \circ \dots \circ I_{\lambda_1, \text{concat}, \mathcal{E}}$. This cascading manipulation allows more fine-grained control over the model’s internal processing path. Output alignment O_ω varies with task specifications, from selective token extraction for generation tasks (Wang et al., 2023a), to parameterized classification heads for discriminative tasks (Li and Liang, 2021). When source and target output spaces are naturally aligned $\mathcal{Y}^S = \mathcal{Y}^T$ (e.g., in text generation with language models), O_ω defaults to an identity mapping, requiring no additional transformation.

Prompt Instruction (PI). Also known as in-context learning, PI (Brown et al., 2020; Kirillov et al., 2023) guides model behavior through carefully designed textual or visual instructions, without explicit parameter updates (i.e., λ is non-trainable). PI demonstrates that sufficiently large pre-trained models, e.g., large-language models, can perform novel tasks when provided with appropriate instructional contexts, such as task descriptions or input-output examples. The term “in-context learning” arises from the model’s ability to learn from examples provided directly within the input context window.

PI instantiates Eq. (1) with non-trainable configuration $\lambda \in \Lambda_{\text{fixed}}$ by employing $\tau = \{\text{concat}, \text{add}\}$ applied at the input level $\ell = \mathcal{X}^S$. The defining characteristic of PI is its reliance on rich contextual information c containing explicit task instructions, demonstrations, or examples. For language models, a typical implementation of input manipulation is $I_{\lambda, \text{concat}, \mathcal{X}^S} = [\lambda_c; \mathbf{x}^T]$, where λ_c formats adaptation directives into appropriate instructional text. For visual tasks, instructions may be incorporated through visual templates or conditional inputs like markers and bounding boxes highlighted on the target input image. Usually, the output alignment mappings O_ω in PI tasks involve rule-based post-processing such as structured parsing, template filling, or format enforcement. These operations can be generally expressed as $O_\omega(\mathbf{y}^S) = \omega(\mathbf{y}^S)$, where $\omega(\cdot)$ denotes extraction or transformation functions that isolate task-relevant information from the model’s output. It can also be identity mapping when no additional output constraints are applied.

By viewing these methods through Eq. (1), we see they are *not* fundamentally different paradigms but rather specific implementations of reprogrammability, primarily distinguished by their choices for the learnability and location (ℓ) of λ , the nature of c , and the structure of ω .

2.4. Reprogrammability as a Unified Paradigm

The previous sections reveal how MR, PT, and PI share fundamental mechanisms despite their disparate origins. Building on this, we establish reprogrammability as a broader paradigm that transcends specific techniques, data modalities, and network architectures, bringing conceptual clarity to what have often been disconnected research threads. Our framework positions MR, PT, and PI as specific instantiations of this general principle.

Shared Principles Across Diverse Adaptation Methods. Three fundamental observations highlight the common ground that unites these approaches. First, the mathematical formulation of Eq. (1) provides a

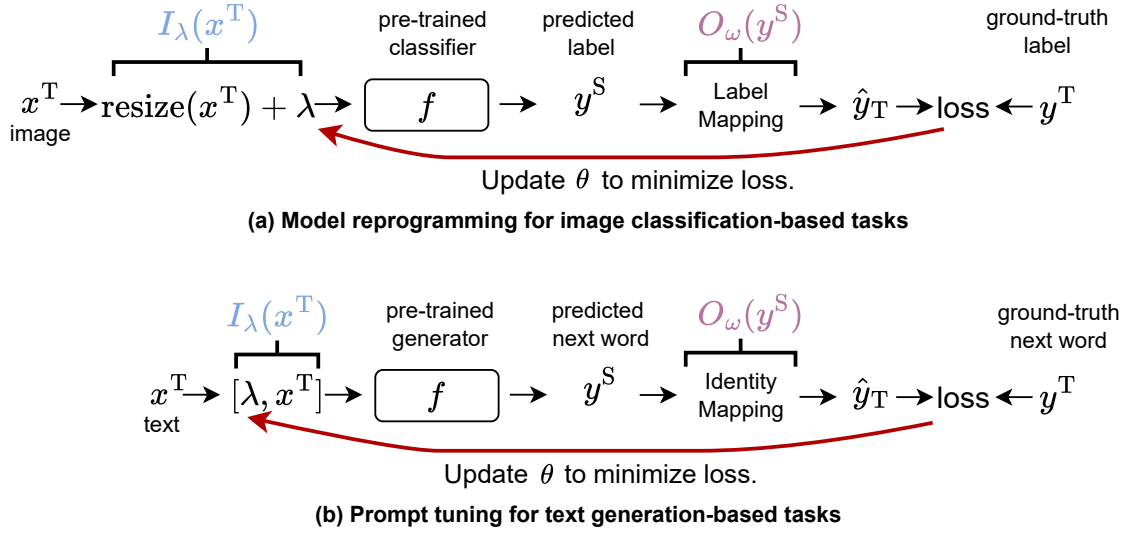


Figure 5 | Examples of how *reprogrammability* manifests across different adaptation methodologies, data modalities, and downstream tasks. (a) MR that repurposes a pre-trained image classifier for a new image classification task. (b) PT that repurposes a pre-trained language generator for a new text generation task.

universal framework regardless of task domain or modality. The specific forms of $I_{\lambda,\tau,\ell}$ and O_ω may vary, but their functional roles in transforming inputs and aligning outputs apply universally.

Second, as we will detail in the taxonomy (Sec. 3), the ways I and O can be designed in terms of manipulation formats, locations, operators, and alignment methods form a complete vocabulary for describing *any* method that adapts a model by manipulating its interfaces. Techniques for different modalities are simply different “settings” within this general design space, rather than fundamentally different mechanisms. For instance, the concept of “soft prompt” (continuous learnable tokens) in textual domain has a visual counterpart: one can learn a noise pattern that, when overlaid or concatenated to an image input, guides a pre-trained vision model to a new task (sometimes was termed visual prompt (Bahng et al., 2022)).

Third, the defining characteristic across all implementations is computational efficiency through interface manipulation rather than parameter modification, which holds regardless of the modalities and architectures involved. In addition, this principle becomes increasingly valuable as models grow in size. The adaptation cost remains relatively constant regardless of model scale, correlating instead with the complexity of the adaptation task itself.

Underlying Mechanisms Enabling Reprogrammability. What mechanisms allow a fixed neural network to perform tasks beyond its original training objective? Several complementary perspectives illuminate this phenomenon.

Neural networks’ sensitivity to interface perturbations appears to be a universal property and is not tied to a specific architecture. Both Convolutional Neural Networks (He et al., 2016) and transformers (Dosovitskiy et al., 2021; Vaswani et al., 2017), despite their structural differences, can be repurposed through strategic input manipulations (Cai et al., 2024a,b; Chen et al., 2023a). This sensitivity, related to the high-dimensional, non-linear functions these networks learn (also seen in adversarial examples, Sec. 2.1), means that slight shifts in the input space can traverse significant distances in the representation and output spaces. Thus, techniques developed for one architecture can often be conceptually adapted to others.

Second, early analyses (Chen, 2024; Yang et al., 2021) suggest that successful reprogramming occurs when sufficient alignment exists between source and target domain representations. Input manipulations

project target data into regions of the model’s representation space where existing computational pathways can be repurposed for new tasks. Large pre-trained models, in particular, develop high-dimensional vector representations that often exhibit structural similarities and can act as universal function approximators (Lu and Lu, 2020). Despite processing different modalities, these models develop generalizable computational capabilities that can be activated through appropriate input conditioning.

Lastly, while a complete theory is still an open research problem, the surprising versatility of large models in diverse tasks through “prompting” (e.g., from arithmetic to commonsense reasoning) suggests that they do not just learn one function, but potentially many sub-functions or “circuits” relevant to different tasks. The RCA approaches, especially prompting, can be seen as ways to activate the specific internal pathways needed for the current task, often guided by mechanisms such as attention, which provide input-dependent activation pathways and dynamically route information based on context (Weiss et al., 2022).

Reprogrammability Beyond Text and Vision. The reprogrammability framework extends naturally beyond standard visual and textual domains to emerging modalities. As a larger range of sensory modalities has been incorporated into modern AI systems, e.g., audio, tactile (Hung et al., 2023; Yen et al., 2023), the same principles continue to hold. Yang et al. (2021) studied how to reprogram acoustic models for time-series data, and Fang et al. (2023); Jing et al. (2023) adapted pre-trained graph neural networks for molecular property prediction. In addition, language models are shown to perform protein infilling tasks (Melnyk et al., 2023; Vinod et al., 2025) and time-series forecasting tasks (Jin et al., 2024). These examples illustrate how the underlying formulation of reprogrammability generalizes across fundamentally different data structures and representations.

Moreover, the framework accommodates novel architectures that may emerge in the future. As long as a network maintains sensitivity to manipulations at its interfaces—a property that appears intrinsic to neural networks rather than architecture-specific—the reprogrammability principles remain applicable. This generality suggests that reprogrammability captures fundamental properties of neural network computations.

In particular, the paradigm’s applicability becomes valuable in multimodal contexts, where manipulations can target specific modality pathways or cross-modal integration points. Conceptually, this extends our framework to encompass manipulation locations that span modality boundaries $I_{\lambda, \tau, \ell}^{\text{multi}}(\mathbf{x}_1^T, \mathbf{x}_2^T, c) = \{I_{\lambda_1, \tau_1, \ell_1}(\mathbf{x}_1^T, c), I_{\lambda_2, \tau_2, \ell_2}(\mathbf{x}_2^T, c)\}$, where \mathbf{x}_1^T and \mathbf{x}_2^T represent different data modalities, such as text and images in vision-language models and multi-modal large-language models.

Thus, we argue that neural network reprogrammability emerges not merely as a collection of domain-specific adaptation techniques but as a paradigm for adapting modern machine learning models. It provides a unifying perspective on model adaptation that spans modalities, architectures, and application domains, offering both theoretical insights into neural network behavior and practical tools for efficient deployment of pre-trained models across diverse tasks.

3. A Taxonomy of Reprogrammability

Having established the mathematical framework for neural network reprogrammability, we now present a systematic taxonomy that categorizes adaptation approaches based on their defining characteristics. This taxonomy organizes *reprogrammability-centric adaptation* (RCA) approaches along four key dimensions corresponding to design choices in implementing the information manipulation function $I(\cdot; \lambda, \tau, \ell, c)$ and output alignment function O_ω from Definition 1. Table 2 shows a comparative analysis of how these dimensions manifest across existing MR, PT, and PI studies in the literature. These dimensions are not mutually exclusive but provide a multifaceted structure to understand and compare different approaches.

Setup	Input Manipulation (I)								Output Alignment (O)			
	interface (ℓ)			configuration (λ)		operator (τ)			mapping (ω)			
	\mathcal{X}^S	\mathcal{E}	\mathcal{H}	OP	FX	AD	CO	PR	SA	LA	RA	ID
MR	✓	✓		✓		✓	✓	✓	✓	✓		
PT		✓	✓	✓			✓	✓		✓	✓	✓
PI	✓				✓	✓	✓	✓			✓	✓

Table 2 | A comparative analysis of existing RCA literature through the lens of reprogrammability. Under interface (ℓ), setup \mathcal{X}^S , \mathcal{E} , and \mathcal{H} denote manipulations that occur in input space, embedding space, and hidden space, respectively. The columns under configuration (λ): setup OP and FX means optimizable and fixed manipulations. The columns under operator (τ): setup AD, CO, and PR refer to additive, concatenative, and parametric operators. The columns under mapping (ω): setup SA, LA, RA, and ID represent statistical alignment, linear alignment, rule-based (i.e., structured) alignment, and identity mapping.

3.1. Manipulation Format: Fixed vs. Learnable λ

The first dimension distinguishes between RCA approaches based on whether their manipulation configurations λ are fixed by manual design or optimizable through learning. This reflects a core decision between leveraging human expertise versus automated discovery to determine effective manipulation strategies.

Fixed Manipulation: The Power of Human Design. Often, fixed manipulations refer to manually designed, non-learnable manipulations, which leverage human intuition and domain knowledge to craft effective prompts (Brown et al., 2020) and instructions (Kojima et al., 2022; Wei et al., 2022b). This approach dominates PI methods, beginning with GPT-3’s pioneering demonstrations of in-context learning (Brown et al., 2020), sparking extensive and rapidly evolving subsequent research into instruction design strategies (Kojima et al., 2022; Lu et al., 2022; Mishra et al., 2021; Wei et al., 2022b; Zhou et al., 2023). Examples include structured reasoning frameworks like chain-of-thought prompting (Wei et al., 2022b); task decomposition strategies (Kojima et al., 2022); cross-task instruction transfer (Mishra et al., 2021) and structured decomposition of reasoning process (Lu et al., 2022).

In the visual domain, fixed manipulations take the form of “hard” visual prompts, such as points, boxes, or masks. Methods like SAM (Kirillov et al., 2023), SEEM (Zou et al., 2023b), and Painter (Wang et al., 2023b) illustrate how simple visual cues effectively guide complex visual processing tasks (like segmentation processes) without parameter updates. Multimodal systems like CLIP (Radford et al., 2021) show similar sensitivity to fixed textual prompts—CLIP models could be directed through a textual instruction (e.g., “a photo of a [CLASS]”)—a capability enhanced by techniques like zero-shot prompt engineering (Allingham et al., 2023). Grounding-focused methods like Koh et al. (2023) and Liu et al. (2024b) leverage text instructions to direct visual attention to specific regions.

Fixed manipulations offer several benefits: they are immediately deployable without training, applicable to black-box models accessible only through APIs, and typically more interpretable than learned alternatives. Yet, they typically require more careful domain expertise and/or human intuition, which may not always discover optimal adaptation strategies for complex tasks.

Learnable Manipulation: Optimization-driven Adaptation. Learnable manipulation-based RCA approaches, alternatively, treat reprogrammability as an optimization problem, using gradient descent to

discover effective manipulation parameters λ , with a training set \mathbb{D}^T from the target task, such that

$$\arg \min_{\lambda} \mathbb{E}_{(\mathbf{x}^T, \mathbf{y}^T) \sim \mathbb{D}^T} [\mathcal{L}(O_{\omega} I_{\lambda, \tau, \ell}(f(I_{\lambda, \tau, \ell}(\mathbf{x}^T, c))), \mathbf{y}^T)]. \quad (2)$$

Here, $\mathcal{L}(\cdot, \cdot)$ is a task-appropriate loss function (e.g., cross-entropy for classification).

This approach characterizes most MR methods. Beginning with [Elsayed et al. \(2019\)](#), who introduced adversarial reprogramming to repurpose an ImageNet-pretrained image classifier for grid-counting, the field has since expanded to include diverse applications. Studies have explored cross-domain adaptation ([Tsai et al., 2020](#)), continuous vector injection into embedding spaces ([Hambardzumyan et al., 2021](#); [Neekhara et al., 2019](#)), and reprogramming LLMs for protein sequence infilling ([Melnyk et al., 2023](#)).

PT methods similarly leverage learnable manipulations. [Lester et al. \(2021\)](#); [Li and Liang \(2021\)](#) showed that optimizing a small set of continuous embedding vectors could match fine-tuning performance in LLMs. For vision-language models (VLMs), CoOp ([Zhou et al., 2022b](#)) demonstrated that hard-crafted prompts, e.g., “a photo of a [CLASS]” can be replaced by learnable continuous vectors for more expressive representations. CoCoOp ([Zhou et al., 2022a](#)) extended this idea to novel classes through image-conditioned prompting. To further facilitate cross-modal alignment, MaPLe ([Khattak et al., 2023](#)) simultaneously tuned textual and visual modalities, while PLOT ([Chen et al., 2022](#)) minimized distributional discrepancy between modalities that regularizes prompt optimization.

Learnable manipulations typically achieve higher performance than their fixed counterparts, but require access to model gradients, task-specific training data, and computational resources for optimization.

Remark 1. Both fixed and learnable manipulations can be further guided by contextual information $c \in \mathcal{C}$, which can take various forms. For example, we can let c be *sample-specific information* that dynamically adjusts adaptation based on input-specific characteristics ([Cai et al., 2024b](#); [Qin and Eisner, 2021](#); [Shin et al., 2020](#)) or *domain-specific information* ([Cai et al., 2025a,b](#); [Dai et al., 2023](#); [Gao et al., 2021](#); [Han et al., 2022](#)). In practice, the research trajectory often progresses from fixed manipulations for exploration, progresses to learnable manipulations for performance optimization. The optimal choice depends on a trade-off between performance, resource constraints, and model access. Interestingly, with the scaling of model size, fixed manipulations are gaining renewed importance due to the emergent capabilities of large models, as exemplified by in-context learning in LLMs like GPT-4 ([Achiam et al., 2023](#)) and DeepSeek ([Liu et al., 2024a](#)), as well as zero-shot vision task capabilities in SAM ([Kirillov et al., 2023](#)).

3.2. Manipulation Location: Where to Modify

The second dimension categorizes RCA methods based on the location within the model architecture where the manipulation occurs. This choice influences the level of model access required, the expressivity of the manipulation, and its computational cost. We identify three primary locations: input, embedding, and hidden spaces.

Raw Input Space \mathcal{X}^S Manipulations. This strategy modifies raw target inputs before they enter the model, preserving the original computational flow while redirecting model behavior toward the target task. This strategy presents the broadest applicability across model architectures and access scenarios.

Raw input manipulations have been extensively employed in MR ([Elsayed et al., 2019](#); [Englert and Lazic, 2022](#); [Neekhara et al., 2019](#); [Tsai et al., 2020](#)) for unimodal models and multimodal VLMs ([Bahng et al., 2022](#); [Wu et al., 2024a](#)). Similarly, PI methods rely heavily on input space manipulations, as seen in in-context learning demonstrations ([Brown et al., 2020](#)), chain-of-thought (CoT) reasoning examples ([Wei et al., 2022b](#)), and the multimodal interleaving of visual and textual data in models like Flamingo ([Alayrac et al., 2022](#)). [Tsimpoukelli et al. \(2021\)](#) demonstrated visual reasoning abilities by providing pre-trained

models with appropriate multi-modal examples in the input context. The adaptation context c , in this case, can be exemplified by SMM (Cai et al., 2024b), which specifies manipulation placements using a sample-wise mask generator.

Since input-level manipulations are universally applicable across diverse models, they are suitable for black-box scenarios where only input-output or API access is available. They also preserve the original model’s computational pipeline, avoiding potential destabilization caused by architectural modifications. However, directly manipulating discrete input modalities (e.g., text) with continuous data (e.g., learnable tokens) can be difficult. Additionally, input-level manipulations may struggle to overcome strong biases encoded within the model’s internal representations (Zhao et al., 2021), motivating research into complementary approaches operating at deeper model interfaces.

Embedding Space \mathcal{E} Manipulations. Embedding space manipulations operate on the model’s first learned representation layer, modifying token or feature embeddings before they flow through deeper network layers. They leverage the model’s learned representations while largely preserving the original processing pipeline, but require at least partial white-box access.

These manipulations are frequently used in both MR and PT. In MR, learned vectors replace certain word embeddings to redirect model behavior (Hambardzumyan et al., 2021). PT approaches like (Lester et al., 2021; Li and Liang, 2021) or P-Tuning (Liu et al., 2022) prepended or modified trainable embeddings at strategic positions. For VLMs, methods like CoOp and CoCoOp learned context textual embeddings that guide visual processing. CLIP-Adapter (Gao et al., 2024) incorporated lightweight adaptation layers at the embedding level to improve few-shot performance. DPT (Zhang et al., 2022) remapped target tasks back to the source domain. Current trends include improving transferability through methods like multi-task prompt learning (Vu et al., 2021) and enhancing efficiency with approaches such as mixture-of-prompts (Qin and Eisner, 2021). Mathematically, these manipulations can be expressed as operations on the embedding space \mathcal{E} of the pre-trained model f , either through addition, concatenation, or parametric transformation of embedding vectors.

Embedding-level manipulations offer several advantages: they typically involve few parameters, work particularly well with Transformer architectures (Vaswani et al., 2017), and can leverage pre-trained token-level knowledge. However, they require direct access to the model’s internal states, precluding usage in black-box scenarios.

Hidden Representation Space \mathcal{H} Manipulations. The most invasive form of manipulation targets intermediate activations within the network, requiring full white-box access but potentially offering the most powerful adaptation effects.

While MR rarely operates directly in hidden space, PT methods increasingly leverage these deeper modifications. P-Tuning v2 (Liu et al., 2021b) inserted learnable components between transformer layers, showing that deeper placement of prompts leads to more effective reprogramming. Multi-layer prompting like UniPELT (He et al., 2022) and UniAdapter (Lu et al., 2023) combined manipulations at strategic network depths for enhanced adaptation. Zhang et al. (2024c) explored the equivalence between visual prompts applied to internal activations and layer normalization.

This approach has proven particularly valuable for cross-modal applications. For example, VLMs may use task-specific visual adapters that transform features at multiple network depths (Sung et al., 2022). ImageBind (Girdhar et al., 2023) and T-Few (Sanh et al., 2022) employed cross-modal alignment layers to transform hidden representations for instruction-based control across multiple input types.

Formally, these manipulations act on the hidden representations $\mathbf{h}_i \in \mathcal{H}_i \subset \mathcal{H}$ at various depths in the

network, modifying the information flow through deeper layers.

Remark 2. The research trend shows increasing interest in targeting deeper model interfaces, particularly for complex cross-domain and cross-modal tasks. However, input-level manipulations remain essential for their universal applicability across diverse model architectures and access scenarios.

3.3. Manipulation Operator: How to Transform

The third dimension examines how RCA approaches integrate manipulations with existing model representations. This is orthogonal to both location and format, as each operator type can be implemented across different interfaces with either fixed or learnable λ .

Additive Operators ($\tau = \text{add}$). These operators superimpose patterns onto existing representations through $I_{\lambda, \text{add}, \ell}(\mathbf{x}^T) = \mathbf{x}^T + \lambda_c$, where λ represents manipulation parameters added directly to the input or intermediate representations at location ℓ . Additive operators form the basis for many MR approaches (Bahng et al., 2022; Elsayed et al., 2019; Neekhara et al., 2022). They are less common in PI, except for visual instructions—bounding boxes (Chen et al., 2023b; Huang et al., 2024; Jiang et al., 2024; Lin et al., 2024; Ma et al., 2024), markers (Nasiriany et al., 2024; Shtedritski et al., 2023; Yang et al., 2023), and pixel-level instructions (Bar et al., 2022; Liu et al., 2023b; Zhang et al., 2024a)—include additive visual cues to highlight regions of interest without fundamentally altering the input structure, when combining instructions with input data for new tasks (Kirillov et al., 2023).

The mathematical simplicity of additive operators makes them particularly amenable to gradient-based optimization and provides a direct pathway for backpropagation when learning λ .

Concatenative Operators ($\tau = \text{concat}$). These operators extend inputs or representations by appending, prepending, or strategically inserting new elements. In MR, concatenative operators appear in approaches expanding input dimensionality (Wu et al., 2024a; Zhang et al., 2024b), such as by padding learnable patterns around target images (Wu et al., 2024a; Zhang et al., 2024b) or pretending learnable tokens to text input embeddings (Hambardzumyan et al., 2021). Concatenation dominates PT, from regular token prepending (Lester et al., 2021; Li and Liang, 2021) to dynamic insertion (Qin and Eisner, 2021). P-Tuning v2 (Liu et al., 2021b) concatenated prompts at multiple layers. Structured Prompting (Hao et al., 2022) incorporated domain knowledge into concatenated prompts. Concatenative operators also form the foundation of ICL (Brown et al., 2020; Wei et al., 2022b) in PI, particularly in CoT prompting (Wei et al., 2022b), where reasoning steps are appended to demonstrations. Moreover, the sensitivity of ICL to the order of concatenated examples has been analyzed (Min et al., 2022; Zhang et al., 2023b).

The rise of Transformer-based architectures has particularly amplified the utility of concatenative operators, as these models naturally process sequences of arbitrary length through self-attention mechanisms.

Parametric Operators ($\tau = \text{param}$). These operators apply complex operations like projections or conditional normalization to inputs or intermediate representations: $I_{\lambda, \text{param}, \ell}(\mathbf{x}^T) = \lambda_c(\mathbf{x}^T)$, where $\lambda(\cdot)$ denotes the transformation function. In MR, parametric operators are used in approaches that apply non-linear transformations of input features for cross-domain adaptation (Tsai et al., 2020). For PT, parametric operators are strongly associated with adapter-based approaches (He et al., 2022; Houlsby et al., 2019; Wang et al., 2021). Within PI, parametric operators appear in retrieval-augmented (RAG) approaches (Borgeaud et al., 2022; Dong et al., 2023a; Lewis et al., 2020; Trivedi et al., 2022), which transform inputs by incorporating retrieved documents.

Remark 3. The historical trajectory shows additive operators dominating early work due to their connection to adversarial examples and gradient-based optimization. Concatenative operators gained prominence with the rise of Transformer-based architectures, while parametric operators have recently gained traction as researchers seek more expressive adaptation mechanisms for complex tasks.

3.4. Output Alignment Requirements

The last dimension categorizes RCA approaches based on how they map model outputs to align with target task requirements, reflecting the mechanisms $O_\omega : \mathcal{Y}^S \rightarrow \mathcal{Y}^T$ used to interpret pre-trained model’s native outputs for the target domain.

Identity Mapping. The simplest form of alignment occurs when model outputs directly correspond to target task outputs without additional processing. This applies when source and target output spaces are naturally compatible. Identity mapping is common in PI and PT for unimodal generative tasks, where the model’s output text directly serves as the task output. As models increase in capability, identity mapping becomes viable for increasingly complex tasks, where visual instruction following through direct generation is demonstrated without post-processing (Liu et al., 2023a, 2021c).

Structured Alignment: Rule-based post-processing. Rule-based alignment applies deterministic procedures to extract relevant information from model outputs, enabling structured prediction without additional training. Still, it depends on human-designed interpretation rules. MR rarely employs complex rule-based alignment, since the output spaces between source and target tasks are often significant. This strategy is central to many PI methods, with CoT (Wei et al., 2022b) and ScratchPad (Nye et al., 2021) applying structured parsing to extract reasoning paths from generated text. ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023) interpreted structured outputs like action sequences with pattern-based extraction. Multimodal instruction models like GPT-4V (Achiam et al., 2023) define O_ω as a pre-defined function that implements parsing, extraction, or validation rules designed to interpret model outputs in task-specific contexts.

Statistical Alignment: Non-parametric Label Mapping. Statistical alignment establishes correspondences between source and target domains based on statistical relationships, without introducing additional trainable parameters. This approach is predominantly used in the MR literature for classification tasks. Techniques include random label mapping (Elsayed et al., 2019), frequent label mapping (Tsai et al., 2020), iterative maximization of mutual information between source predictions and target categories (Chen et al., 2023a), and establishing full probabilistic relationships between source and target labels (BLM (Cai et al., 2024a)), based on the co-occurrence statistics of model predictions (in \mathcal{Y}^S) and ground-truth labels (in \mathcal{Y}^T).

Linear Alignment: Parametric Transformation. Linear alignment introduces parametric functions, typically neural networks, to transform source model outputs to target requirements, offering greater flexibility through the optimizable mapping matrix but requiring additional parameters and training data. In MR, this corresponds to replacing statistical alignment with a linear probe (Neekhara et al., 2022). In contrast, learnable alignment is more common in PT (Gu et al., 2022; Li and Liang, 2021), where a learnable classification head (e.g., verbalizer) was employed atop prompted representations.

Remark 4. The choice of alignment strategies reflects the fundamental properties of different adaptation paradigms: statistical and learnable alignments predominate in MR due to the typical mismatch between

source and target tasks, while identity and rule-based alignments are more common in PT and PI due to the inherent flexibility of generative models. Nowadays, as pre-trained models scale in size and capability, we observe a trend toward simpler alignment strategies, with larger models (especially large-language models) capable of producing outputs that directly address target tasks without complex post-processing, when target tasks also fall under the same modality.

4. Emergent Insights from the Reprogrammability Perspective

Our framework (Sec. 2) and taxonomy (Sec. 3) do more than just catalogue existing methods. By showing how this perspective illuminates some interesting phenomena, e.g., in-context learning and chain-of-thought reasoning, as well as the trade-offs inherent in different adaptation choices, we can gain insights into how these techniques relate to fundamental properties of the pre-trained model itself.

4.1. Understanding In-Context Learning (ICL) and Chain-of-Thought (CoT)

ICL represents one of the most intriguing capabilities of LLMs—the ability to adapt to new tasks through demonstrations without parameter updates—which we can analyze from the viewpoint of our reprogrammability framework, wherein ICL can be characterized as: A non-learnable manipulation ($\lambda \in \Lambda_{\text{fixed}}$ determined by demonstration design); Applied exclusively at the input interface ($\ell = \mathcal{X}^S$); Using concatenation ($\tau = \text{concat}$) to merge demonstrations with queries; and Employing minimal output alignment (typically identity or simple rule-based extraction). With this formulation, we can reconceptualize ICL as a specific instance of neural network reprogrammability beyond simply a mysterious emergent property, where

$$f'(\mathbf{x}^T) = O_{\text{identity}} \circ f \circ I_{\lambda_{\text{fixed}}, \text{concat}, \mathcal{X}^S}(\mathbf{x}^T, c_{\text{demonstrations}}).$$

From this perspective, ICL’s effectiveness stems from the highly developed interface sensitivity of the pre-trained LLM. This sensitivity, likely increases along with the scale of pre-training data and model capacity, allows strategic input manipulations (the demonstrations) to dynamically reconfigure or steer the model’s internal computational pathways. A possible reason is the attention mechanisms within Transformer-based architectures, as they allow the model to dynamically weigh the importance of different parts of the input context (including demonstrations) when processing the query. The demonstrations $c_{\text{demonstrations}}$ “prime” the attention patterns and subsequent computations to align with the exemplified task structure.

We argue that this interpretation is well-supported by several findings. [Min et al. \(2022\)](#) demonstrated that the ICL performance depends critically on aspects like demonstration ordering and input distribution, consistent with the view that concatenative input manipulations create different computational pathways by altering the context seen by the attention layers. Its effectiveness also scales predictably with model size ([Brown et al., 2020](#)), suggesting that larger models develop more sophisticated interface sensitivity that allows them to better utilize contextual cues. Recent research suggests that LLMs can learn to implement simple learning algorithms or recognize patterns within their forward pass. For instance, the emergence of induction heads ([Olsson et al., 2022](#))—attention heads that can complete sequences by copying patterns from the context—provides a concrete mechanism for how models might extract an implicit task-specific transformation function T_{task} from demonstrations $c_{\text{demonstrations}}$ and apply it to new inputs $f'(\mathbf{x}^T) \approx T_{\text{task}}(\mathbf{x}^T)$ ([Garg et al., 2022](#)). Thus, ICL can be viewed as the model performing a form of rapid, implicit meta-learning or Bayesian inference, where demonstrations act as task-specific “evidence” to condition its behavior ([Xie et al., 2022](#)). This way the design of $c_{\text{demonstrations}}$ is therefore important, as it directly defines the “program” the LLM executes.

CoT reasoning ([Wei et al., 2022b](#)) extends this framework by incorporating not just input-output examples, but also structured reasoning steps and intermediate patterns that guide the model through a sequence of

computational states. Within our framework, CoT represents a more specialized and explicit form of input manipulation, where $c_{\text{reasoning}}$ now includes explicit reasoning chains:

$$f'(\mathbf{x}^T) = O_{\text{extract}} \circ f \circ I_{\lambda_{\text{fixed}}, \text{concat}, \mathcal{X}^S}(\mathbf{x}^T, c_{\text{reasoning}})$$

where $c_{\text{reasoning}} = [c_{\text{demonstration}}; c_{\text{reason}}]$ contains demonstration complete with explicit intermediate reasoning steps, and O_{extract} is a potentially more complex output parser that identifies the final answer from the generated reasoning chain. This “reprograms” the model not only to map inputs to outputs, but to follow a particular algorithmic process. The reasoning steps in $c_{\text{reasoning}}$ serve as waypoints or a scaffold for the model’s autoregressive generation process, compelling it to materialize intermediate thoughts that were perhaps only implicitly navigated in standard ICL.

In summary, the reprogrammability perspective shows why CoT can dramatically improve performance on complex reasoning tasks: it explicitly guides the model’s computational trajectory through intermediate states that decompose the problem into manageable steps. By providing examples of how to reason, CoT leverages the LLM’s ability to pattern-match and continue sequences, effectively “reprogramming” it to activate or chain together internal circuits responsible for more elementary reasoning capabilities learned during pre-training (e.g., on text containing explanations, arguments, or narratives). The very act of verbalizing these intermediate steps, prompted by the CoT examples, can serve as a guidance constraint, ensuring the model follows a more structured and traceable thought process before arriving at an answer. This makes the implicit task transformation T_{task} of ICL more explicit and robust, effectively constraining the model to follow sample-specific reasoning patterns.

4.2. Implications of Taxonomic Position

Our framework also suggests interesting trade-offs for design choices.

Efficiency-Effectiveness Trade-Offs. Different positions in our taxonomy present distinct trade-offs between computational requirements and task performance. We observe several patterns. First, input space manipulations generally require fewer computational resources (i.e., $\ll \mathcal{O}(|\mathbf{x}^T|)$ parameters) than manipulations performed across multiple hidden layers (i.e., $\mathcal{O}(\sum_{i=1}^L d_{\mathbf{h}_i})$ parameters required). This difference is amplified in large-scale models with many high-dimensional hidden states. Second, fixed configurations eliminate the need for training but may sacrifice performance compared to learnable manipulations. Lastly, the computational cost of different operators varies significantly. Additive manipulations maintain representation dimensionality, concatenative operators increase it (potentially requiring more computation in subsequent layers), and transformative operators add explicit computation steps. In practice, the impact of these differences depends on model architecture and the specific implementation.

These trade-offs create distinct adaptation profiles for different resource contexts since learnable RCA methods typically require hundreds to thousands of examples for effective optimization, while well-designed fixed manipulations can adapt with fewer examples. When labeled data is scarce, fixed manipulations could be more advantageous despite potentially lower ceiling performance. When computational resources are limited but data is abundant, learnable manipulations often provide the best balance of efficiency and effectiveness.

Generalization Properties. We also summarize a few findings regarding the generalization of reprogrammed models. (1) *Sample-specific Adaptation.* Cai et al. (2024b) observed that manipulations conditioned on input characteristics $I_{\lambda, \tau, \ell}(\mathbf{x}^T, c(\mathbf{x}^T))$ tend to generalize more effectively than static manipulations. This suggests dynamic adaptation better captures the underlying task structure rather than memorizing

specific data patterns. (2) *Representation Hierarchy Effects*. The manipulation location influences adaptation capabilities. As layer depth increases, manipulations operate on more abstract representations (i.e., deeper layers) that generalize better across similar tasks but lose modality-specific details. This explains why manipulations targeting deeper network layers (particularly within transformer architectures (He et al., 2022; Liu et al., 2021b)) often achieve strong cross-domain transfer. (3) *Scale-dependent Transfer*. The generalization gap between input-level reprogramming and fine-tuning follows a power law relationship with model size (Bahng et al., 2022). That is, input-level reprogramming can underperform fine-tuning with smaller models but achieves competitive results as model size increases. For models below a certain size threshold, fine-tuning consistently outperforms reprogramming. Above this threshold, reprogramming approaches competitive or superior performance with significantly fewer parameters (Wu et al., 2024a), e.g., from ResNet (He et al., 2016) to CLIP (Radford et al., 2021). Structured task decomposition improves generalization performance on new task-domain combinations (Andreas, 2022). Recently, Geng and Chen (2024) showed that reprogramming is more robust than conventional fine-tuning with out-of-distribution data (Han et al., 2025; Jiang et al., 2023; Zheng et al., 2023).

This suggests that adaptation strategies should be tailored to both the available pre-trained model and the expected transfer scenario. For large-scale deployment across related tasks, deeper manipulations with sample-specific components offer the strongest performance, whilst for targeted adaptation to a specific task with a smaller model, fine-tuning may remain competitive despite its parameter inefficiency.

Model Scale Dependencies. Large language models exhibit emergent reprogrammability in their ability to follow instructions and adapt to new tasks through few-shot demonstrations (Wei et al., 2022a), which can be formalized as a phase transition in the model’s interface sensitivity:

$$\text{Interface Sensitivity}(f, \lambda) \approx \begin{cases} \epsilon & \text{if } |f| < S_{\text{threshold}} \\ \beta \cdot \log(|f|/S_{\text{threshold}}) & \text{otherwise,} \end{cases}$$

where $|M|$ represents model size, $S_{\text{threshold}}$ is the emergence threshold, and β is a scaling factor. This behavior dictates distinctly different adaptation strategies for models above and below certain scale thresholds, as a sublinear relationship between model size and optimal manipulation volume is observed (Lester et al., 2021). Specifically, the proportion of parameters needed for effective adaptation decreases as models grow larger, following an empirically observed relationship, such that Optimal Prompt Size \propto Model Size $^\alpha$ with $\alpha < 1$. This sublinear scaling (Lester et al., 2021) suggests that larger models develop more parameter-efficient representational capabilities.

In addition, as model scale increases, the complexity of required output alignment operations decreases (Chowdhery et al., 2023). For large-scale deployment across related tasks, deeper manipulations with sample-specific components offer the strongest performance, whilst for targeted adaptation to a specific task with a smaller model, fully fine-tuning may remain competitive despite its parameter inefficiency.

5. Challenges and Future Studies

Despite the rapid progress and exciting potential, neural network reprogrammability faces open challenges, ranging from theoretical underpinnings to practical deployment. We outline key issues that future studies should grapple with to unlock the full, responsible potential of this paradigm.

5.1. Towards a Theory of Reprogrammability

One of the foremost open questions is *why* and *when* reprogrammability is effective. While early theories are emerging (Chung et al., 2025; Englert and Lazic, 2022; Petrov et al., 2023; Xie et al., 2022), such as interpreting

ICL as a form of implicit Bayesian inference system (Xie et al., 2022) or identifying that pre-trained models learn latent simulators (Hao et al., 2023), we lack a comprehensive framework predicting a given model’s reprogrammability for a given new task. Particularly, what intrinsic properties of a pre-trained model—its architecture, the diversity and scale of its training data, the nature of its learned feature representations (e.g., disentanglement, sparsity), or even its degree of overparameterization—render it amenable to being repurposed for a specific new task via input or context manipulations? For instance, how do models internally generalize effectively from a few prompt examples, provided that LLMs are known to be able to exhibit ICL abilities? Current vary, from prompts enabling implicit meta-learning by identifying shared latent structures, to prompts activating specific pre-learned “skills” or sub-networks, akin to how attention mechanisms route information. Is reprogramming essentially unlocking dormant “circuits” within the network, perhaps related to the lottery ticket hypothesis, which posits that sparse, trainable subnetworks exist for many tasks (Frankle and Carbin, 2018)? Or does high-dimensional overparameterization create such a rich functional landscape that solutions to many tasks are “accidentally” embedded and thus discoverable via prompting? Crucially, what are the theoretical limits of task complexity that can be handled by reprogrammability without modification to model weights? Closing this theoretical gap requires delving into concrete questions:

- **Feature Re-utilization vs. New Computation:** To what extent does reprogramming rely on mapping new task inputs to existing feature extractors versus coercing the model into performing novel computations it was not explicitly trained for?
- **Role of Pre-training Objectives:** How do different self-supervised or supervised pre-training objectives (e.g., masked language modeling (Devlin et al., 2019), contrastive learning (Radford et al., 2021), next-token prediction (Radford et al., 2019)) influence the pre-trained model’s reprogrammability for diverse downstream tasks?
- **Theoretical Limits:** What is the upper bound on task complexity or information novelty that a model can adapt to via reprogramming alone, without any weight changes? Can a model be reprogrammed to solve a task whose intrinsic complexity (e.g., related to its Kolmogorov complexity or statistical learning theoretic measures like VC dimension, if the reprogrammed system is viewed as a new learner) exceeds what can be “described” by the prompt within the model’s existing functional landscape?
- **Predictive Metrics for Reprogrammability:** Can we develop quantifiable metrics, derived from a model’s parameters or activation dynamics (e.g., by analyzing its sensitivity to different types of input perturbations), to predict its reprogramming aptitude for a given class of tasks?

Answering these questions could profoundly inform how we design and train future foundation models to be inherently more efficient and broadly reprogrammable.

5.2. Evaluation: Benchmarks and Baselines

Because reprogrammability enables unconventional use of models, evaluating their performance fairly is tricky. If we repurpose an image classifier for a sentiment analysis task via input transformations, what will be an appropriate baseline? Is it a model of similar size trained from scratch on the sentiment task, a fine-tuned version of the original image classifier (if feasible), or highly specialized architectures designed for sentiment analysis? The choice of baseline dramatically affects perceived efficacy.

A pressing need exists for standardized evaluation frameworks and dedicated benchmarks for RCA methods. These are essential for fair comparison and for understanding when RCA should be used rather than parameter-centric adaptations. Key challenges include:

- **Designing Benchmark Suites:** An urgent need exists for comprehensive benchmark suites specifically for reprogramming. These should encompass a diverse set of source models (varied architectures,

sizes, pre-training data); a wide array of target tasks, spanning different modalities (text, vision, multi-modal), varying levels of abstraction, and diverse degrees of similarity/dissimilarity to the source model’s original task(s); standardized conditions for evaluation: data limitations (few-shot, zero-shot reprogramming), signal-to-noise ratios, presence of distribution shifts between reprogrammable data and test data, and computational budgets for learning the reprogrammed solution.

- **Defining Appropriate Baselines:** Clear guidelines are necessary for choosing appropriate baselines when evaluating RCA methods. These guidelines should consider factors such as the number of trainable parameters (if any), computational cost (e.g., FLOPs for training and inference), and the type of access to the pre-trained model (e.g., white-box vs. black-box).
- **Setting up Evaluation Metrics:** Evaluations should extend beyond task-specific accuracy. Crucially, reasonable additional dimensions include but are not limited to robustness and stability, program efficiency, computational gains, and transferability of the RCA solution across different tasks.

Without rigorous evaluation frameworks, assessing true progress in RCA research will remain challenging, potentially leading to fragmented efforts and incomparable results.

5.3. Limitations of Existing Programs

While the concept of reprogrammability is appealing for its efficiency and potential, current techniques, particularly those involving “prompts” (whether hard textual prompts or learned soft prompts), still exhibit practical limitations:

- **Data Efficiency:** Although RCA avoids fine-tuning the entire model, learning effective “soft” prompts can still demand a considerable amount of target-task data, where the optimization of program parameters can be a challenging non-convex problem, requiring careful initialization and sufficient examples to converge to a good solution. Future works may draw inspiration from meta-learning methods (Du et al., 2023; Finn et al., 2017; Garnelo et al., 2018a,b; Liu et al., 2021a; Ye and Yao, 2022; Ye et al., 2023a,b) and domain generalization (Zhong et al., 2024), to reduce required sample complexity.
- **Task Complexity and Scope:** Another limitation is that current RCA methods excel at tasks that align well with the pre-trained model’s inherent capabilities, such as classification, stylistic generation, or question answering, where the underlying knowledge is already present. However, for tasks requiring fundamentally new reasoning chains, complex symbolic manipulation not seen during pre-training (e.g., advanced multi-step mathematics for an LLM not extensively trained on it), or the acquisition of entirely novel skills, merely prepending a prompt or transforming inputs may be insufficient. In this case, it is worthwhile to incorporate complex reasoning strategies, e.g., causal reasoning (Chi et al., 2024), into the reprogramming pipeline.
- **Program Transferability:** Moreover, a practical hurdle is that programs, especially learned soft prompts, are often highly specific to the particular model (and sometimes even the specific checkpoint) they were tuned for. A prompt optimized for one LLM architecture rarely transfers effectively to another, even one of similar size or family. When the source model is needed to be adapted for multiple distributionally similar target tasks, this “hypersensitivity” restricts the reusability of prompts and forces re-optimization for each new task. While research into universal prompts or techniques for efficient prompt adaptation is ongoing, it remains a challenging frontier. Hard-coded prompts, though potentially more transferable, can be brittle and overly sensitive to minor variations in phrasing.
- **Optimization Challenges:** MR and soft PT typically rely on gradient access to the model to optimize the prompt parameters. In many real-world scenarios where models are accessed via restricted APIs (black-box access), gradient-based optimization is impossible. This leaves downstream users with manual prompt engineering or less efficient derivative-free optimization methods, such as evolutionary algorithms-based ones (Jiang et al., 2016), for discovering effective prompts. Developing more powerful

and sample-efficient black-box optimization techniques for prompt discovery in high-dimensional spaces is an open problem.

Addressing these limitations is key to making RCA a more versatile and robust adaptation solution.

5.4. Ethical and Security Concerns

Still, neural network reprogrammability introduces a new angle of ethical and security concerns, spanning from misuse to questions of accountability. As models become more easily redirectable, their potential for unintended or malicious applications grows.

- **Evasion of Safeguards (i.e., prompt injection & jailbreaking):** If anyone can repurpose a model via prompts, models could be coaxed into behaviors their creators did not intend (Chen et al., 2025). For example, LLMs might be reprogrammed via a cleverly crafted prompt to produce disallowed content, e.g., hate speech, even if safeguards are in place—a form of prompt injection attack (Liu et al., 2024c; Zou et al., 2023a). On the other hand, the reprogrammed models do not possess built-in robustness against malicious manipulations, e.g., adversarial and backdoor attacks (Chen et al., 2025). Ensuring that models cannot be easily reprogrammed for harmful purposes via building bespoke defenders, e.g., adversarial detectors (Zhang et al., 2025) and purifiers (Sun et al., 2025), is an important yet unresolved challenge.
- **Accountability and Liability:** Reprogrammability blurs the lines of responsibility. If a foundation model developed by Company A is reprogrammed by User B using a prompt (perhaps designed or shared by Community C) to generate illegal or harmful content that affects User D, who bears the liability? Is it the original model developer, the prompt designer, or the entity that deployed the reprogrammed system? There is an urgent need for clear legal and ethical frameworks for auditing reprogrammed AI systems, attributing actions, and assigning responsibility.
- **Fairness and Bias Propagation:** Pre-trained models inevitably inherit biases from their vast training datasets. RCA can inadvertently amplify these biases or introduce new ones when the model is applied to a novel task or data distribution (Itzhak et al., 2024). For example, a model exhibiting certain gender biases in its original text generation task might manifest these, or even new and unexpected biases, when reprogrammed for a resume screening task in a different demographic context. Research is needed into fairness-aware reprogramming techniques, methods to audit reprogrammed models for bias, and understanding how biases transform across tasks and reprogramming methods.
- **Alignment Challenges:** As the reprogrammability increases, developing robust alignment and monitoring techniques becomes essential. There is however a fundamental tension accordingly: excessively rigid constraints or safeguards may stifle legitimate, beneficial adaptability and innovation. Conversely, insufficient guardrails can enable widespread misuse or unintended harmful consequences. The core challenge is achieving alignment not just for the base model, but for the reprogrammed system—ensuring it adheres to the intended goals of the new task while respecting broader safety and ethical principles.

Finding the optimal balance between safety and utility represents a central challenge in the responsible deployment of reprogrammable neural networks.

6. Conclusion

We presented neural network reprogrammability as a unifying theme connecting model reprogramming, prompt tuning, and prompt instruction—three paradigms enabling us to reuse pre-trained models for new

purposes with minimal changes. We reviewed how this idea developed from early adversarial manipulations into a powerful toolkit for efficient model adaptation, and we offered a unified taxonomy to classify and compare methods across modalities. Looking ahead, we anticipate a convergence of ideas: insights from prompting LLMs will inform cross-domain reprogramming in vision and other fields, and vice versa, leading to a more cohesive field of study rather than disparate threads. Ultimately, neural reprogrammability reflects a shift in mindset: instead of training a new model for each problem, the model itself becomes a platform, and solving a new problem is a matter of programming that platform with the right inputs.

References

- J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.
- Z. Allen-Zhu, Y. Li, and Y. Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *NeurIPS*, 2019.
- J. U. Allingham, J. Ren, M. W. Dusenberry, X. Gu, Y. Cui, D. Tran, J. Z. Liu, and B. Lakshminarayanan. A simple zero-shot prompt weighting technique to improve prompt ensembling in text-image models. In *ICML*, 2023.
- J. Andreas. Language models as agent models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022.
- H. Bahng, A. Jahanian, S. Sankaranarayanan, and P. Isola. Exploring visual prompts for adapting large-scale models. *arXiv*, 2022.
- A. Bar, Y. Gandelsman, T. Darrell, A. Globerson, and A. Efros. Visual prompting via image inpainting. In *NeurIPS*, 2022.
- S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J. Powell, L. Cross, J. Cassirer, A. Purser, I. Budd, J. Paganini, L. Aslanides, L. Sifre, T. Irina, C. Blundell, and J. Rae. Improving language models by retrieving from trillions of tokens. In *ICML*, 2022.
- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.
- C. Cai, Z. Ye, L. Feng, J. Qi, and F. Liu. Bayesian-guided label mapping for visual reprogramming. In *NeurIPS*, 2024a.
- C. Cai, Z. Ye, L. Feng, J. Qi, and F. Liu. Sample-specific masks for visual reprogramming-based prompting. In *ICML*, 2024b.
- C. Cai, Z. Ye, L. Feng, J. Qi, and F. Liu. Attribute-based visual reprogramming for vision-language models. In *ICLR*, 2025a.
- C. Cai, Z. Ye, L. Feng, J. Qi, and F. Liu. Understanding model reprogramming for clip via decoupling visual prompts. In *ICML*, 2025b.
- N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017a.

- N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. Ieee, 2017b.
- A. Chen, Y. Yao, P.-Y. Chen, Y. Zhang, and S. Liu. Understanding and improving visual prompting: A label-mapping perspective. In *CVPR*, 2023a.
- G. Chen, W. Yao, X. Song, X. Li, Y. Rao, and K. Zhang. Plot: Prompt learning with optimal transport for vision-language models. In *ICLR*, 2022.
- K. Chen, Z. Zhang, W. Zeng, R. Zhang, F. Zhu, and R. Zhao. Shikra: Unleashing multimodal llm’s referential dialogue magic. *arXiv preprint arXiv:2306.15195*, 2023b.
- P.-Y. Chen. Model reprogramming: Resource-efficient cross-domain machine learning. In *AAAI*, 2024.
- P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *AAAI*, 2018.
- Y. Chen, S. Shao, E. Huang, Y. Li, P.-Y. Chen, Z. Qin, and K. Ren. Refine: Inversion-free backdoor defense via model reprogramming. In *ICLR*, 2025.
- H. Chi, F. Liu, W. Yang, L. Lan, T. Liu, B. Han, W. Cheung, and J. Kwok. Tohan: A one-step approach towards few-shot hypothesis adaptation. *NeurIPS*, 2021.
- H. Chi, H. Li, W. Yang, F. Liu, L. Lan, X. Ren, T. Liu, and B. Han. Unveiling causal reasoning in large language models: Reality or mirage? *NeurIPS*, 2024.
- A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. Palm: scaling language modeling with pathways. *JMLR*, 2023.
- M.-Y. Chung, J. Fan, H. Ye, Q. Wang, W.-C. Shen, C.-M. Yu, P.-Y. Chen, and S.-Y. Kuo. Model reprogramming demystified: A neural tangent kernel perspective. *arXiv preprint arXiv:2506.0620*, 2025.
- W. Dai, J. Li, D. Li, A. M. H. Tiong, J. Zhao, W. Wang, B. Li, P. Fung, and S. Hoi. Instructblip: towards general-purpose vision-language models with instruction tuning. In *NeurIPS*, 2023.
- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.
- H. Dong, W. Xiong, D. Goyal, R. Pan, S. Diao, J. Zhang, K. Shum, and T. Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023a.
- R. Dong, F. Liu, H. Chi, T. Liu, M. Gong, G. Niu, M. Sugiyama, and B. Han. Diversity-enhancing generative network for few-shot hypothesis adaptation. In *ICML*, 2023b.

- A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.
- J. Du, Z. Ye, B. Guo, Z. Yu, and L. Yao. Idnp: Interest dynamics modeling using generative neural processes for sequential recommendation. In *WSDM*, 2023.
- G. F. Elsayed, I. Goodfellow, and J. Sohl-Dickstein. Adversarial reprogramming of neural networks. In *ICLR*, 2019.
- M. Englert and R. Lazic. Adversarial reprogramming revisited. In *NeurIPS*, 2022.
- T. Fang, Y. Zhang, Y. Yang, C. Wang, and L. Chen. Universal prompt tuning for graph neural networks. In *NeurIPS*, 2023.
- A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard. Robustness of classifiers: from adversarial to random noise. *NeurIPS*, 2016.
- C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.
- J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.
- P. Gao, S. Geng, R. Zhang, T. Ma, R. Fang, Y. Zhang, H. Li, and Y. Qiao. Clip-adapter: Better vision-language models with feature adapters. *IJCV*, 2024.
- T. Gao, A. Fisch, and D. Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021.
- S. Garg, D. Tsipras, P. S. Liang, and G. Valiant. What can transformers learn in-context? a case study of simple function classes. In *NeurIPS*, 2022.
- M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. A. Eslami. Conditional neural processes. In *International conference on machine learning*, pages 1704–1713. PMLR, 2018a.
- M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. Eslami, and Y. W. Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.
- A. Geng and P.-Y. Chen. Model reprogramming outperforms fine-tuning on out-of-distribution data in text-image encoders. In *SaTML*, 2024.
- J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.
- R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra. Imagebind: One embedding space to bind them all. In *CVPR*, 2023.
- I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- J. Gu, Z. Han, S. Chen, A. Beirami, B. He, G. Zhang, R. Liao, Y. Qin, V. Tresp, and P. Torr. A systematic survey of prompt engineering on vision-language foundation models. *arXiv preprint arXiv:2307.12980*, 2023.

- Y. Gu, X. Han, Z. Liu, and M. Huang. Ppt: Pre-trained prompt tuning for few-shot learning. In *Proceedings of ACL*, 2022.
- X. Guo, F. Lin, C. Yi, J. Song, D. Sun, L. Lin, Z. Zhong, Z. Wu, X. Wang, Y. Zhang, et al. Deep transfer learning enables lesion tracing of circulating tumor cells. *Nature Communications*, 2022.
- K. Hambardzumyan, H. Khachatrian, and J. May. Warp: Word-level adversarial reprogramming. In *ACL-IJCNLP*, 2021.
- B. Han, J. Yao, T. Liu, B. Li, S. Koyejo, F. Liu, et al. Trustworthy machine learning: From data to models. *ps*, 2025.
- X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun. Ptr: Prompt tuning with rules for text classification. *AI Open*, 2022.
- S. Hao, Y. Gu, H. Ma, J. J. Hong, Z. Wang, D. Z. Wang, and Z. Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.
- Y. Hao, Y. Sun, L. Dong, Z. Han, Y. Gu, and F. Wei. Structured prompting: Scaling in-context learning to 1,000 examples. *arXiv preprint arXiv:2212.06713*, 2022.
- J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig. Towards a unified view of parameter-efficient transfer learning. In *ICLR*, 2022.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- P. Helber, B. Bischke, A. Dengel, and D. Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.
- N. Houlsby, A. Giurghi, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, pages 2790–2799, 2019.
- S. Huang, H. Chang, Y. Liu, Y. Zhu, H. Dong, P. Gao, A. Boularias, and H. Li. A3vlm: Actionable articulation-aware vision language model. *arXiv preprint arXiv:2406.07549*, 2024.
- Y.-N. Hung, C.-H. H. Yang, P.-Y. Chen, and A. Lerch. Low-resource music genre classification with cross-modal neural model reprogramming. In *ICASSP*, 2023.
- A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. *NeurIPS*, 2019.
- I. Itzhak, G. Stanovsky, N. Rosenfeld, and Y. Belinkov. Instructed to bias: Instruction-tuned language models exhibit emergent cognitive bias. *Transactions of the Association for Computational Linguistics*, 2024.
- P. Jiang, F. Liu, J. Wang, and Y. Song. Cuckoo search-designated fractal interpolation functions with winner combination for estimating missing values in time series. *Applied Mathematical Modelling*, 2016.
- S. Jiang, Y. Zhang, C. Zhou, Y. Jin, Y. Feng, J. Wu, and Z. Liu. Joint visual and text prompting for improved object-centric perception with multimodal large language models. *arXiv preprint arXiv:2404.04514*, 2024.
- X. Jiang, F. Liu, Z. Fang, H. Chen, T. Liu, F. Zheng, and B. Han. Detecting out-of-distribution data through in-distribution class prior. In *ICML*, 2023.
- M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. In *ICLR*, 2024.

- Y. Jing, C. Yuan, L. Ju, Y. Yang, X. Wang, and D. Tao. Deep graph reprogramming. In *CVPR*, 2023.
- M. U. Khattak, H. Rasheed, M. Maaz, S. Khan, and F. S. Khan. Maple: Multi-modal prompt learning. In *CVPR*, 2023.
- A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *CVPR*, 2023.
- J. Y. Koh, D. Fried, and R. R. Salakhutdinov. Generating images with multimodal language models. *Advances in Neural Information Processing Systems*, 36:21487–21506, 2023.
- T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. In *NeurIPS*, 2022.
- B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021.
- P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*, 2020.
- X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL/IJCNLP*, pages 4582–4597, 2021.
- J. Liang, D. Hu, and J. Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020.
- W. Lin, X. Wei, R. An, P. Gao, B. Zou, Y. Luo, S. Huang, S. Zhang, and H. Li. Draw-and-understand: Leveraging visual prompts to enable mllms to comprehend what you want. *arXiv preprint arXiv:2403.20271*, 2024.
- A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.
- F. Liu, W. Xu, J. Lu, and D. J. Sutherland. Meta two-sample testing: Learning kernels for testing with limited data. In *NeurIPS*, 2021a.
- H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *NeurIPS*, 2023a.
- S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pages 38–55. Springer, 2024b.
- W. Liu, X. Shen, C.-M. Pun, and X. Cun. Explicit visual prompting for low-level structure segmentations. In *CVPR*, 2023b.
- X. Liu, K. Ji, Y. Fu, W. L. Tam, Z. Du, Z. Yang, and J. Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021b.
- X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang. Gpt understands, too. *arXiv:2103.10385*, 2021c.
- X. Liu, K. Ji, Y. Fu, W. Tam, Z. Du, Z. Yang, and J. Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *ACL, Volume 2: Short Papers*, pages 61–68, 2022.
- X. Liu, N. Xu, M. Chen, and C. Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *ICLR*, 2024c.

- M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. *NeurIPS*, 2016.
- M. Long, Z. Cao, J. Wang, and M. I. Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, 2018.
- D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647, 2005.
- H. Lu, Y. Huo, G. Yang, Z. Lu, W. Zhan, M. Tomizuka, and M. Ding. Uniadapter: Unified parameter-efficient transfer learning for cross-modal modeling. *arXiv preprint arXiv:2302.06605*, 2023.
- P. Lu, S. Mishra, T. Xia, L. Qiu, K.-W. Chang, S.-C. Zhu, O. Tafjord, P. Clark, and A. Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *NeurIPS*, 2022.
- Y. Lu and J. Lu. A universal approximation theorem of deep neural networks for expressing probability distributions. In *NeurIPS*, 2020.
- M. Luo, X. Xu, Y. Liu, P. Pasupat, and M. Kazemi. In-context learning with retrieved demonstrations for language models: A survey. *TMLR*, 2024.
- C. Ma, Y. Jiang, J. Wu, Z. Yuan, and X. Qi. Groma: Localized visual tokenization for grounding multimodal large language models. In *ECCV*, 2024.
- I. Melnyk, V. Chenthamarakshan, P.-Y. Chen, P. Das, A. Dhurandhar, I. Padhi, and D. Das. Reprogramming pretrained language models for antibody sequence infilling. In *ICML*, 2023.
- S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *EMNLP*, 2022.
- S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi. Cross-task generalization via natural language crowd-sourcing instructions. *arXiv preprint arXiv:2104.08773*, 2021.
- S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024.
- P. Neekhara, S. Hussain, S. Dubnov, and F. Koushanfar. Adversarial reprogramming of text classification neural networks. In *EMNLP/IJCNLP*, pages 5215–5224, 2019.
- P. Neekhara, S. Hussain, J. Du, S. Dubnov, F. Koushanfar, and J. McAuley. Cross-modal adversarial reprogramming. In *CVPR*, 2022.
- M. Nye, A. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz, M. Bosma, D. Luan, C. Sutton, and A. Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.
- C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.
- S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 2009.
- A. Petrov, P. H. Torr, and A. Bibi. When do prompting and prefix-tuning work? a theory of capabilities and limitations. In *NeurIPS*, 2023.
- G. Qin and J. Eisner. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*, 2021.

- A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.
- V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, J. Chaffin, K. Liu, et al. Multitask prompted training enables zero-shot task generalization. In *ICLR*, 2022.
- S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, P. S. Dulepet, S. Vidyadhara, D. Ki, S. Agrawal, C. Pham, G. Kroiz, F. Li, H. Tao, A. Srivastava, H. D. Costa, S. Gupta, M. L. Rogers, I. Goncarenko, G. Sarli, I. Galynker, D. Peskoff, M. Carpuat, J. White, S. Anadkat, A. Hoyle, and P. Resnik. The prompt report: A systematic survey of prompting techniques. *arXiv preprint arXiv:2406.06608*, 2024.
- T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.
- N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.
- A. Shtedritski, C. Rupprecht, and A. Vedaldi. What does clip know about a red circle? visual prompt engineering for vlms. In *ICCV*, 2023.
- Y. Sun, J. Zhang, Z. Ye, C. Xiao, and F. Liu. Sample-specific noise injection for diffusion-based adversarial purification. In *ICML*, 2025.
- Y.-L. Sung, J. Cho, and M. Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *CVPR*, 2022.
- C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.
- H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*, 2022.
- Y.-Y. Tsai, P.-Y. Chen, and T.-Y. Ho. Transfer learning without knowing: Reprogramming black-box machine learning models with scarce data and limited resources. In *ICML*, 2020.
- H.-A. Tsao, L. Hsiung, P.-Y. Chen, S. Liu, and T.-Y. Ho. Autovp: An automated visual prompting framework and benchmark. In *ICLR*, 2024.
- M. Tsimpoukelli, J. L. Menick, S. Cabi, S. Eslami, O. Vinyals, and F. Hill. Multimodal few-shot learning with frozen language models. In *NeurIPS*, 2021.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
- R. Vinod, P.-Y. Chen, and P. Das. Reprogramming pretrained language models for protein sequence representation learning. *Digital Discovery*, 2025.
- T. Vu, B. Lester, N. Constant, R. Al-Rfou, and D. Cer. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*, 2021.

- D. Wang, M. Li, X. Liu, M. Xu, B. Chen, and H. Zhang. Tuning multi-mode token-level prompt alignment across modalities. In *NeurIPS*, 2023a.
- R. Wang, D. Tang, N. Duan, Z. Wei, X. Huang, J. Ji, G. Cao, D. Jiang, and M. Zhou. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021.
- X. Wang, W. Wang, Y. Cao, C. Shen, and T. Huang. Images speak in images: A generalist painter for in-context visual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6830–6839, 2023b.
- J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus. Emergent abilities of large language models. *TMLR*, 2022a.
- J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022b.
- M. Weiss, N. Rahaman, F. Locatello, C. Pal, Y. Bengio, B. Schölkopf, E. L. Li, and N. Ballas. Neural attentive circuits. *NeurIPS*, 2022.
- J. Wu, X. Li, C. Wei, H. Wang, A. Yuille, Y. Zhou, and C. Xie. Unleashing the power of visual prompting at the pixel level. *TMLR*, 2024a.
- J. Wu, Z. Zhang, Y. Xia, X. Li, Z. Xia, A. Chang, T. Yu, S. Kim, R. A. Rossi, R. Zhang, S. Mitra, D. N. Metaxas, L. Yao, J. Shang, and J. J. McAuley. Visual prompting in multimodal large language models: A survey. *arXiv preprint arXiv:2409.15310*, 2024b.
- S. M. Xie, A. Raghunathan, P. Liang, and T. Ma. An explanation of in-context learning as implicit bayesian inference. In *ICLR*, 2022.
- C.-H. H. Yang, Y.-Y. Tsai, and P.-Y. Chen. Voice2series: Reprogramming acoustic models for time series classification. In *ICML*, 2021.
- J. Yang, H. Zhang, F. Li, X. Zou, C. Li, and J. Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023.
- S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In *ICLR*, 2023.
- Z. Ye and L. Yao. Contrastive conditional neural processes. In *CVPR*, 2022.
- Z. Ye, J. Du, Y. Liu, Y. Zhang, and L. Yao. Np-ssl: A modular and extensible self-supervised learning library with neural processes. In *CIKM*, 2023a.
- Z. Ye, J. Du, and L. Yao. Adversarially contrastive estimation of conditional neural processes. *arXiv preprint arXiv:2303.13004*, 2023b.
- H. Yen, P.-J. Ku, C.-H. H. Yang, H. Hu, S. M. Siniscalchi, P.-Y. Chen, and Y. Tsao. Neural model reprogramming with similarity based mapping for low-resource spoken command recognition. In *INTERSPEECH*, 2023.
- C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

- C. Zhang, J. Cho, F. D. Puspitasari, S. Zheng, C. Li, Y. Qiao, T. Kang, X. Shan, C. Zhang, C. Qin, F. Rameau, L.-H. Lee, S.-H. Bae, and C. S. Hong. A survey on segment anything model (sam): Vision foundation model meets prompt engineering. *arXiv preprint arXiv:2306.06211*, 2023a.
- J. Zhang, B. I. Rubinstein, J. Zhang, and F. Liu. One stone, two birds: Enhancing adversarial defense through the lens of distributional discrepancy. In *ICML*, 2025.
- N. Zhang, L. Li, X. Chen, S. Deng, Z. Bi, C. Tan, F. Huang, and H. Chen. Differentiable prompt makes pre-trained language models better few-shot learners. In *ICLR*, 2022.
- T. Zhang, X. Li, H. Fei, H. Yuan, S. Wu, S. Ji, C. C. Loy, and S. Yan. Omg-llava: Bridging image-level, object-level, pixel-level reasoning and understanding. In *NeurIPS*, 2024a.
- Y. Zhang, K. Zhou, and Z. Liu. What makes good examples for visual in-context learning? In *NeurIPS*. Curran Associates, Inc., 2023b.
- Y. Zhang, Y. Dong, S. Zhang, T. Min, H. Su, and J. Zhu. Exploring the transferability of visual prompting for multimodal large language models. In *CVPR*, pages 26562–26572, 2024b.
- Y. Zhang, H. Li, Y. Yao, A. Chen, S. Zhang, P.-Y. Chen, M. Wang, and S. Liu. Visual prompting reimaged: The power of activation prompts. 2024c.
- W. X. Zhao, X. Liu, J. Zhang, Y. Zhang, K. Zhou, and J.-R. Wen. A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT. *arXiv preprint arXiv:2302.09419*, 2023.
- Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh. Calibrate before use: Improving few-shot performance of language models. In *ICML*, 2021.
- H. Zheng, Q. Wang, Z. Fang, X. Xia, F. Liu, T. Liu, and B. Han. Out-of-distribution detection learning with unreliable out-of-distribution sources. *NeurIPS*, 2023.
- Z. Zhong, J. Hou, Z. Yao, L. Dong, F. Liu, J. Yue, T. Wu, J. Zheng, G. Ouyang, C. Yang, et al. Domain generalization enables general cancer cell annotation in single-cell and spatial transcriptomics. *Nature Communications*, 2024.
- D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *ICLR*, 2023.
- K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Conditional prompt learning for vision-language models. In *CVPR*, 2022a.
- K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Learning to prompt for vision-language models. *IJCV*, 2022b.
- A. Zou, Z. Wang, J. Z. Kolter, and M. Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023a.
- X. Zou, J. Yang, H. Zhang, F. Li, L. Li, J. Wang, L. Wang, J. Gao, and Y. J. Lee. Segment everything everywhere all at once. In *NeurIPS*, 2023b.