Model Reprogramming: Resource-Efficient Cross-Domain Machine Learning

Pin-Yu Chen

IBM Research
pin-yu.chen@ibm.com

Abstract

In data-rich domains such as vision, language, and speech, deep learning prevails to deliver high-performance task-specific models and can even learn general task-agnostic representations for efficient finetuning to downstream tasks. However, deep learning in resource-limited domains still faces multiple challenges including (i) limited data, (ii) constrained model development cost, and (iii) lack of adequate pre-trained models for effective finetuning. This paper provides an overview of *model reprogramming* to bridge this gap. Model reprogramming enables resource-efficient cross-domain machine learning by repurposing and reusing a well-developed pre-trained model from a source domain to solve tasks in a target domain *without* model finetuning, where the source and target domains can be vastly different. In many applications, model reprogramming outperforms transfer learning and training from scratch. This paper elucidates the methodology of model reprogramming, summarizes existing use cases, provides a theoretical explanation of the success of model reprogramming, and concludes with a discussion on open-ended research questions and opportunities.

1 Introduction

Designing and developing a top-notch deep learning model is a time-consuming and costly process. It is no secret that employing a high-capacity neural network model consisting of a tremendous number of trainable parameters, together with a proper selection of the network architecture and hyperparameter optimization, can lead to state-of-the-art machine learning performance when trained on a massive amount of data. Take the Generative Pre-trained Transformer 3 (GPT-3) (Brown et al. 2020) as an example, which is one of the largest language models ever trained to date. GPT-3 has 175 billion parameters and is trained on a dataset consisting of 499 Billion tokens. The estimated training cost is about 4.6 Million US dollars even with the lowest priced GPU cloud on the market in 2020[1]. Such a large-scale language model is shown to be effective when applied to several downstream language-related tasks in *the same domain (source)*. However, having invested so much to obtain a top-notch model, one interesting question to ask is: Can we reuse this valuable asset for machine learning in *another domain (target)*, especially in the resource-limited setting when at least one of the following scenarios is concerned: (i) lack of high-quality pre-trained models in the target domain for finetuning, (ii) scarcity of the available data in the target domain, and (iii) constraints on the model development and training cost (e.g. limited memory or training epochs).

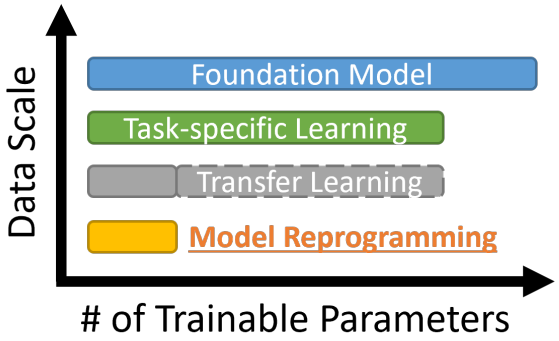

Figure 1: Visual illustration of data scale (bottom to top: small to large) and number (#) of trainable parameters (left to right: small to large) in different machine learning paradigms. We note that the visualization does not reflect the actual relative differences due to excessively varying orders. A foundation model like GPT-3 has 175 billion trainable parameters and 499 billion tokens as training data. The trainable parameters in model reprogramming can be as few as the size of the data input (e.g., the number of image pixels can be in the order of thousands or fewer), and model reprogramming is particularly suited to small-scale data regime. In model reprogramming, the visualization does not take into account the pre-trained source model because it is kept intact and unchanged. The dashed box in transfer learning means variations in the number of model parameters used for fine-tuning, ranging from only training the last dense layer (linear head) to fine-tuning all parameters. The number of training epochs may also vary for each paradigm.

To address these challenges, this paper provides an overview of the *model reprogramming* framework towards resource-efficient cross-domain machine learning. The general rationale behind model reprogramming lies in repurpos-

[1]See https://lambdalabs.com/blog/demystifying-gpt-3

Symbol	Meaning
$\mathcal{S}$ / $\mathcal{T}$	source/target domain
$\mathcal{X}_\mathcal{S}$ / $\mathcal{X}_\mathcal{T}$	the space of source/target data samples
$\mathcal{Y}_\mathcal{S}$ / $\mathcal{Y}_\mathcal{T}$	the space of source/target data labels
$\mathcal{D}_\mathcal{S} \subseteq \mathcal{X}_\mathcal{S} \times \mathcal{Y}_\mathcal{S}$ / $\mathcal{D}_\mathcal{T} \subseteq \mathcal{X}_\mathcal{T} \times \mathcal{Y}_\mathcal{T}$	source/target data distribution
$(x, y) \sim \mathcal{D}$	data sample x and one-hot coded label y drawn from $\mathcal{D}$
$K_\mathcal{S}/K_\mathcal{T}$	number of source/target labels
$f_\mathcal{S} : \mathbb{R}^d \mapsto [0, 1]^K$	pre-trained K-way source classification model
$\eta : \mathbb{R}^K \mapsto [0, 1]^K$	softmax function in neural network, and $\sum_{k=1}^K [\eta(\cdot)]_k = 1$
$z(\cdot) \in \mathbb{R}^K$	logit (pre-softmax) representation, and $f(x) = \eta(z(x))$
$\ell(x, y) \triangleq \|f(x) - y\|_2$	risk function of (x, y) based on classifier f
$\mathbb{E}_\mathcal{D}[\ell(x, y)] \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(x, y)]$	population risk based on classifier f
δ	additive input transformation on target data
θ / ω	parameters of input transformation / output mapping layers
$M \in \{0, 1\}^d$	binary mask indicating which input dimension is trainable
$h : \mathcal{Y}_\mathcal{S} \mapsto \mathcal{Y}_\mathcal{T}$	source-target label mapping function

Table 1: Mathematical notation

ing and reusing a well-developed pre-trained model from a source domain to solve new tasks in a target domain *without model finetuning* (i.e., model parameters are *frozen*). Specifically, model reprogramming introduces an input transformation layer and an output mapping layer to the pre-trained source model to empower cross-domain machine learning. Model reprogramming is favorable to the resource-limited setting because it (i) enables the reuse of pre-trained models from data-rich and well-studied domains (e.g., vision, language, and speech); (ii) attains data efficiency by only training the added input transformation and output mapping layers; and (iii) reuses available models and spares model development from scratch. The underlying working mechanism of model reprogramming in terms of the *how* and the *why* will be explained in detail throughout this paper.

As a visual illustration, Figure 1 compares the data scale and the number of trainable parameters for different machine learning paradigms, including model reprogramming, transfer learning, task-specific learning (training from scratch), and foundation model (pre-training and fine-tuning). Data scale refers to the training data size. The number of trainable parameters relates to the model development cost, and models with more training parameters usually have higher training complexity. We elucidate each paradigm as follows.

- **Model reprogramming** only requires training the inserted input transformation and output mapping layers while keeping the source pre-trained model intact (see Figure 2). Notably, the number of trainable parameters does not include the source pre-trained model because its parameters are unchanged during model reprogramming. Therefore, it is particularly applicable to the small-data regime and features relatively low training complexity.
- **Transfer learning** is a common practice for in-domain knowledge transfer (e.g., pre-trained on an English-based task and finetuned for other English-related natural language processing tasks). The general principle is that some features learned from a source domain can be useful for machine learning in a target domain via model finetuning. Transfer learning starts from a pre-trained source model and then finetunes a subset of the model parameters using the target-domain data. The number of trainable parameters can vary based on the size of the selected subset for finetuning. If all model parameters are used for finetuning, the number of trainable parameters is the same as that of training from scratch, though the training epochs of transfer learning may be fewer. If only the last dense layer (i.e., a linear head) in the neural network is randomly initialized and made trainable, then the number of trainable parameters can be comparable to that of model reprogramming. The dashed box in Figure 1 indicates the variation in the size of trainable parameters for transfer learning. One notable limitation of transfer learning is that in some target domains, there may lack of adequate pre-trained models from similar domains for effective finetuning.
- **Task-specific learning** refers to training the parameters of a machine learning model by minimizing a task-specific loss (e.g., cross entropy classification loss on a given task). The model parameters are often randomly initialized and trained from scratch. In the small-scale data regime, training from scratch with large models usually yields unsatisfactory performance even when data augmentation is used.
- **Foundation model** (Bommasani et al. 2021) features task-agnostic pre-training (often on a large-scale dataset) and efficient finetuning to downstream tasks. It is becoming a new trend in machine learning research due to its capability to learn general and discriminative representations of the considered data modality. The training of foundation model often follows the methodology of self-supervised learning, such as contrastive learning with self-generated positive and negative pairs, or masked token prediction. See (Jaiswal et al. 2021) for more details. The current practice of training foundation models still requires a massive amount of data for pre-training and a gigantic model for learning general-purpose representations,

such as the GPT-3 model (Brown et al. 2020).

It is worth noting that the origin of model reprogramming can be traced back to the adversarial reprogramming method proposed in (Elsayed, Goodfellow, and Sohl-Dickstein 2019). The authors in (Elsayed, Goodfellow, and Sohl-Dickstein 2019) demonstrate that ImageNet-1K image classifiers can be reprogrammed for classifying CIFAR-10 and MNIST images, as well as counting squares in an image, with mediocre accuracy. It is originally cast as an adversarial machine learning technique due to the implication that an attacker can leverage model reprogramming to repurpose the function of a pre-trained model without notice of the model provider, therefore causing ethical concerns or negative impacts. Beyond the adversarial purpose, the subsequent works such as (Tsai, Chen, and Ho 2020; Vinod, Chen, and Das 2023; Yang, Tsai, and Chen 2021) show that model reprogramming can be used as a resource-efficient cross-domain machine learning tool in a variety of problem domains and data modalities.

The remainder of this paper is organized as follows. Section 2 introduces the general framework of model reprogramming and summarizes existing works and use cases. Section 3 provides theoretical explanations on the success of model reprogramming. Finally, Section 4 discusses some important open-ended research questions and opportunities for model reprogramming. For clarity, Table 1 presents the main mathematical notations used in this paper.

2 Model Reprogramming Framework

In this section, we start by introducing a generic framework and algorithmic procedure for model reprogramming (Section 2.1). Then, we highlight some use cases of model reprogramming in current studies (Section 2.2).

2.1 Methodology of Model Reprogramming

Figure 1 shows a generic framework of model reprogramming and some examples of cross-domain reprogramming in the literature. In general, two new modules, an *input transformation layer* and an *output mapping layer*, are added to a frozen pre-trained source model to enable reprogramming. We will elucidate how these two layers are realized and implemented in the following paragraphs.

For ease of illustration, we focus on the setting of reprogramming a pre-trained classifier $f_{\mathcal{S}}(\cdot)$ from a source domain to solve a classification task in a target domain. The notation of the model parameters associated with $f_{\mathcal{S}}(\cdot)$ is omitted because these parameters are fixed and unchanged during reprogramming. Without loss of generality, we assume $f_{\mathcal{S}}(\cdot)$ takes a vectorized input of dimension $d_{\mathcal{S}}$, which includes continuous data domains (e.g., image pixels, audio signals, numeric tabular features, etc) and discrete tokenized data domains (e.g., words, image patches, etc). The output of $f_{\mathcal{S}}(\cdot) \in [0,1]^{K_{\mathcal{S}}}$ is a $K_{\mathcal{S}}$-dimensional vector of class prediction scores over $K_{\mathcal{S}}$ source classes.

We also note that the framework of model reprogramming makes no constraints on the source and target domains (e.g., domain similarity, knowledge transfer, etc), as long as their data formats are consistent. For instance, (Neekhara et al. 2022) shows an example of cross-domain reprogramming on an image classifier for sentence sentiment classification by mapping word tokens to image patches. Given consistent data formats, the only assumptions imposed by reprogramming are that (i) the target data dimension is no greater than the source data dimension (i.e., $d_{\mathcal{T}} \leq d_{\mathcal{S}}$); and (ii) the number of target class labels is no greater than that of source class labels (i.e., $K_{\mathcal{T}} \leq K_{\mathcal{S}}$). These two assumptions are based on the rationale that model reprogramming is applicable to the setting of reprogramming a pre-trained model from a more complex source domain to a simpler target domain with a lower input dimension and a smaller number of class labels, but the reverse setting may not be applicable.

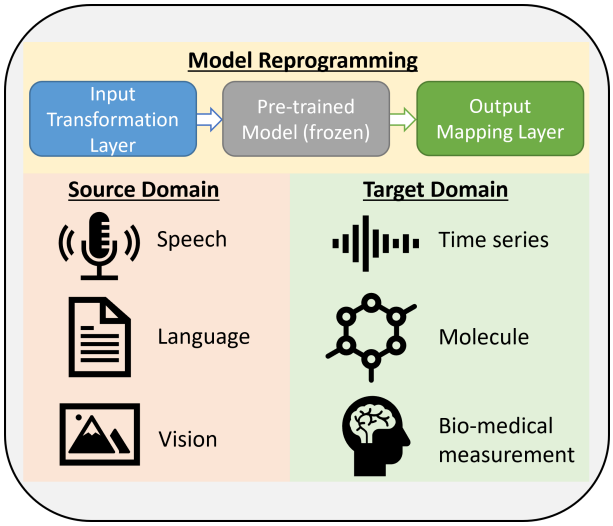

Figure 2: Illustration of the model reprogramming framework (top) and some examples of cross-domain machine learning via model reprogramming (bottom). Model reprogramming enables cross-domain machine learning by adding two modules, an input transformation layer (blue box) and an output mapping layer (green box), to a pre-trained model selected from a source domain. When reprogrammed to solve target-domain tasks, the pre-trained source model is frozen and its model parameters are unchanged. Examples of cross-domain machine learning include reprogramming speech models for time-series (Yang, Tsai, and Chen 2021), language models for molecules (Vinod, Chen, and Das 2023), and general imaging models for bio-medical measurements (Tsai, Chen, and Ho 2020).

Input transformation layer. Given a target-domain data sample $x_{\mathcal{T}} \in \mathbb{R}^{d_{\mathcal{T}}}$, the input transformation layer transforms $x_{\mathcal{T}}$ into another data sample $\widetilde{x}_{\mathcal{T}} \in \mathbb{R}^{d_{\mathcal{S}}}$, which fits the input dimension of the pre-trained source model $f_{\mathcal{S}}$. The input transformation function is parameterized with a set of trainable parameters θ, which is formally defined as $\widetilde{x}_{\mathcal{T}} = \mathsf{Input\text{-}Transform}(x_{\mathcal{T}}|\theta)$. For continuous data, the transformation function can be as simple as a universal trainable additive input perturbation (i.e., a bias term) δ with a binary mask $M \in \{0,1\}^{d_{\mathcal{S}}}$ indicating which dimension is

trainable. Putting in mathematical expressions, we have

$$\widetilde{x}_{\mathcal{T}} = \mathsf{Zero\text{-}Padding}(x_{\mathcal{T}}) + \underbrace{M \odot \delta}_{:=\theta}, \tag{1}$$

where the operation $\mathsf{Zero\text{-}Padding}(\cdot) \in \mathbb{R}^{d_{\mathcal{S}}}$ means augmenting the input with extra dimensions of zero values, and the notation $\odot$ denotes the Hadamard (element-wise) vector product. The number of ones in M indicates the location and the effective number of trainable parameters. For instance, in (Elsayed, Goodfellow, and Sohl-Dickstein 2019; Tsai, Chen, and Ho 2020), a target sample $x_{\mathcal{T}}$ is placed at the center of its zero-padded version $\widetilde{x}_{\mathcal{T}}$, where the location of $x_{\mathcal{T}}$ is indicated by the masked index set $\{i : M_i = 0\}$ and hence by (1) the masked set is not trainable. The remaining zero-padded dimensions, indicated by $\{i : M_i = 1\}$, are made trainable. We also note that if the data input range of $f_{\mathcal{S}}$ is bounded, for example within $[-1, 1]^{d_{\mathcal{S}}}$, one can apply the change-of-variable technique to satisfy the constraint, such as making $\theta = \tanh(M \odot W)$, where $W \in \mathbb{R}^{d_{\mathcal{S}}}$ denotes unconstrained optimization variables and $\tanh$ denotes the hyperbolic tangent function. The input transformation function also allows placing multiple replicates of a target sample in the transformed data sample, such as in (Yang, Tsai, and Chen 2021).

For discrete data, the input transformation function can be a set of inserted trainable tokens at the data input in (Hambardzumyan, Khachatrian, and May 2021), or a trainable token embedding mapping function $\mathcal{V}_{\mathcal{T}} \approx \mathcal{V}_{\mathcal{S}}\theta$ via dictionary learning in (Vinod, Chen, and Das 2023), where the rows of $\mathcal{V}_{\mathcal{S}}/\mathcal{V}_{\mathcal{T}}$ are the embeddings of the source/target tokens.

Output mapping layer. There are two major approaches to realizing the output mapping layer. The first approach is *source-target label mapping*, by specifying a many-to-one surjective mapping $h : \mathcal{Y}_{\mathcal{S}} \mapsto \mathcal{Y}_{\mathcal{T}}$ from source class labels to target class labels. Take the example of reprogramming an ImageNet pre-trained source model for autism spectrum disorder (ASD) classification based on brain-region correlation graphs, the source label subset {Tench,Goldfish,Hammerhead} can be assigned to the target label {ASD}, and another non-overlapping source label subset can be assigned to the other target label {non-ASD}. The prediction probability of a target class $t \in \mathcal{Y}_{\mathcal{T}}$ on an input-transformed data sample $\widetilde{x}_{\mathcal{T}}$ in (1) is the average prediction probability of the source labels specified by h and $f_{\mathcal{S}}$. Mathematically, let $\mathcal{B} \subset \mathcal{Y}_{\mathcal{S}}$ denote the subset of source labels mapping to the target label $t \in \mathcal{Y}_{\mathcal{T}}$. Then, the class prediction of t is the aggregated prediction based on $f_{\mathcal{S}}$ over the assigned source labels, which is defined as

$$\mathsf{Prob}(t|f_{\mathcal{S}}(\widetilde{x}_{\mathcal{T}})) = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} \mathsf{Prob}(s|f_{\mathcal{S}}(\widetilde{x}_{\mathcal{T}})) \tag{2}$$

where $|\mathcal{B}|$ denotes the number of labels in $\mathcal{B}$. In (Tsai, Chen, and Ho 2020), the authors show that frequency-based greedy label mapping based on original responses before training can improve reprogramming performance when compared with random label mapping. (Chen et al. 2023b) further shows that iterative greedy mapping gives better results. Moreover, assigning more (but non-overlapping) source labels to a target label can also improve the final performance. Instead of using pre-specified label mapping, one can also learn a label mapping function h by treating it as an optimal label assignment problem.

The second approach is *adding a trainable dense layer* (linear head) with a set of trainable parameters ω between the source model's output of dimension $K_{\mathcal{S}}$ (or the model's penultimate output) and the target model's output of dimension $K_{\mathcal{T}}$, such as in (Kloberdanz, Tian, and Le 2021; Hambardzumyan, Khachatrian, and May 2021; Arif, Gittens, and Chen 2023).

Model training and evaluation. After introducing the two aforementioned modules, input transformation and output mapping layers, to a pre-trained source model $f_{\mathcal{S}}$ (see Figure 2), the target-domain training set $\{x_{\mathcal{T}}^{(i)}, y_{\mathcal{T}}^{(i)}\}_{i=1}^{n}$ is used to evaluate the associated task loss $\mathsf{Loss}(\widehat{y}_{\mathcal{T}}, y_{\mathcal{T}}|\theta, \omega)$ (e.g., the cross entropy loss), where $\widehat{y}_{\mathcal{T}} = \mathsf{Output\text{-}Mapping}(f_{\mathcal{S}}(\mathsf{Input\text{-}Transform}(x_{\mathcal{T}}|\theta))|\omega)$ is the reprogrammed model prediction on the target-domain data sample $x_{\mathcal{T}}$, and $y_{\mathcal{T}}$ is the groundtruth target class label. The reprogramming parameters θ and ω (ω can be omitted if a source-target label mapping h is used instead) are trained and updated based on $\mathsf{Loss}(\widehat{y}_{\mathcal{T}}, y_{\mathcal{T}}|\theta, \omega)$ and $\{x_{\mathcal{T}}^{(i)}, y_{\mathcal{T}}^{(i)}\}_{i=1}^{n}$ in an end-to-end manner using an optimization algorithm such as a gradient-based method. In the restricted model access setting when the pre-trained source model $f_{\mathcal{S}}$ is a black-box function that only provides model outputs at queried data inputs and back-propagation through $f_{\mathcal{S}}$ is infeasible, such as a machine learning based application programming interface (API) or proprietary software, black-box model reprogramming can be realized by using gradient-free methods (e.g., zeroth-order optimization (Liu et al. 2020)) for training (Tsai, Chen, and Ho 2020). Finally, the optimized parameters θ^* and ω^* are used together with the pre-trained source model $f_{\mathcal{S}}$ for evaluation.

Algorithmic procedure. Below we describe the generic algorithmic procedure for model reprogramming.

1. Initialization: Load pre-trained source model $f_{\mathcal{S}}(\cdot)$ and target domain training set $\{x_{\mathcal{T}}^{(i)}, y_{\mathcal{T}}^{(i)}\}_{i=1}^{n}$; randomly initialize θ and ω
2. Input transformation: Obtain transformed input data $\widetilde{x}_{\mathcal{T}} = \mathsf{Input\text{-}Transform}(x_{\mathcal{T}}|\theta)$, where θ is the set of trainable parameters for input transformation
3. Output mapping: Obtain the prediction on the target task via $\widehat{y}_{\mathcal{T}} = \mathsf{Output\text{-}Mapping}(f_{\mathcal{S}}(\widetilde{x}_{\mathcal{T}})|\omega)$, where ω is the set of trainable parameters for output mapping[2]
4. Model training: Optimize θ and ω by evaluating a task-specific loss $\mathsf{Loss}(\widehat{y}_{\mathcal{T}}, y_{\mathcal{T}}|\theta, \omega)$ on $\{x_{\mathcal{T}}^{(i)}, y_{\mathcal{T}}^{(i)}\}_{i=1}^{n}$
5. Outcome: Reprogrammed model from $f_{\mathcal{S}}(\cdot)$ with optimized trainable parameters θ^* and ω^* such that $\widehat{y}_{\mathcal{T}} = \mathsf{Output\text{-}Mapping}(f_{\mathcal{S}}(\mathsf{Input\text{-}Transform}(x_{\mathcal{T}}|\theta^*))|\omega^*)$

[2] In output mapping, trainable parameters may not be necessary if one uses a specified source-target label mapping function.

Reference	Source domain	Source model	Target domain	Highlights
(Elsayed, Goodfellow, and Sohl-Dickstein 2019)	General image	ImageNet	CIFAR-10/MNIST/counting	first work; mediocre accuracy
(Neekhara et al. 2019)	Text	LSTM/CNN	Character/Word level tasks	context-based vocabulary mapping
(Tsai, Chen, and Ho 2020)	General image	ImageNet/API	Bio-medical measurement/image	black-box reprogramming; new SOTA
(Vinod, Chen, and Das 2023)	Text	BERT	Biochemical sequence	vocabulary embedding mapping
(Kloberdanz, Tian, and Le 2021)	General image	ImageNet	Caltech 101/256 (reduced)	trainable input & output layers
(Lee, Suh, and Ramchandran 2020; Dinh et al. 2022)	Image/Spectrogram	GAN	Image/Spectrogram	reprogram GAN to conditional GAN
(Randazzo et al. 2021)	MNIST/lizard pattern	Neural CA	MNIST/Lizard pattern	stable out-of-training configurations
(Hambardzumyan, Khachatrian, and May 2021)	Text	BERT & variants	GLUE/SuperGLUE	trainable tokens and data efficiency
(Yang, Tsai, and Chen 2021)	Speech	Attention-RNN	Univariate time series	new/same SOTA on 19/30 datasets
(Yen et al. 2023)	Speech	Attention-RNN	Low-resource speech	new SOTA; reprogramming+finetuning
(Chen, Fan, and Ye 2021)	General image	ImageNet	Financial transaction	overlay image and transaction feature
(Neekhara et al. 2022)	General image	ViT/ImageNet	Sequence	text sentences and DNA sequences
(Jing et al. 2023)	Graph	GNN	Graph-based tasks	3D object recognition & action recognition
(Melnyk et al. 2023)	Text	BERT	Protein sequence	antibody sequence infilling with diversity

Table 2: Summary of model reprogramming use cases. LSTM means long short-term memory, CNN/RNN means convolutional/recurrent neural network, API means application programming interface, and SOTA means state of the art. BERT stands for bidirectional encoder representations from transformers. GLUE stands for the general language understanding evaluation benchmark. GAN stands for generative adversarial network. CA stands for cellular automata. ViT stands for vision transformer. GNN stands for graph neural network. The table is actively updated at https://github.com/IBM/model-reprogramming.

2.2 Model Reprogramming Use Cases

Model reprogramming has shown success and improved performance for resource-efficient cross-domain machine learning on a wide range of data domains, pre-trained source models, and machine learning tasks. Table 2 summarizes some studies on model reprogramming. Without loss of generality, in what follows we highlight two representative use cases for each data format (continuous or discrete) featuring improved task performance and resource efficiency.

Continuous Data Domain

- *Black-box adversarial reprogramming (BAR)* (Tsai, Chen, and Ho 2020): BAR extends the original adversarial reprogramming framework (Elsayed, Goodfellow, and Sohl-Dickstein 2019) to enable reprogramming black-box models and demonstrates both data and cost efficiency in low-resource bio-medical applications. For example, the authors reprogram ImageNet pre-trained deep neural network classifiers to classify autism spectrum disorder (ASD) based on brain-region correlation graphs and report new state-of-the-art accuracy on this challenging data-limited task. Moreover, they also demonstrate that different commercial prediction APIs can be reprogrammed at an affordable cost to solve different tasks without knowing the details of the underlying machine learning model. With the cost of about 20 US dollars, two Clarifai.com APIs (Not Safe For Work and Moderation) and a traffic sign classification model trained by the Microsoft Custom Vision API were reprogrammed for several bio-medical classification tasks with good accuracy.
- *Voice2Series (V2S)* (Yang, Tsai, and Chen 2021): V2S reprograms a speech model (acoustic signal as input and speech command prediction as output) for univariate time-series classification. Time series data include but are not limited to medical diagnosis (e.g., physiological signals such as electrocardiogram (ECG)), finance/weather forecasting, and industrial measurements (e.g., sensors and Internet of Things (IoT)). In general, time-series data are small-scale because they are not as abundant and easily accessible as speech data. Evaluated on UCR time series classification datasets (Dau et al. 2019), V2S outperforms or ties with the best baseline on 19 out of 30 datasets.

Discrete Data Domain

- *Representation reprogramming via dictionary learning (R2DL)* (Vinod, Chen, and Das 2023): Let $\mathcal{V}_\mathcal{S} \in \mathbb{R}^{N_\mathcal{S} \times d}$ and $\mathcal{V}_\mathcal{T} \in \mathbb{R}^{N_\mathcal{T} \times d}$ denote the vocabulary matrix of the source-domain and target-domain tokens, respectively, where the rows of $\mathcal{V}_\mathcal{S}$ and $\mathcal{V}_\mathcal{T}$ represent their token embedding vectors with the same dimension d while their token numbers $N_\mathcal{S}$ and $N_\mathcal{T}$ can be different. R2DL transforms the embedding in $\mathcal{V}_\mathcal{S}$ obtained from a pre-trained language model to represent the embedding in $\mathcal{V}_\mathcal{T}$ by finding the parameters $\theta \in \mathbb{R}^{d \times d}$ such that $\mathcal{V}_\mathcal{T} \approx \mathcal{V}_\mathcal{S}\theta$, where a dictionary learning algorithm such as the K-SVD solver (Aharon, Elad, and Bruckstein 2006) is used to obtain a column-wise sparse solution θ. Then θ is further updated with the output mapping layer and a task-specific loss. In the reduced-data setting, the authors show improved performance over training from scratch when reprogramming pre-trained English language models (e.g., BERT and LSTM) for biochemical sequence classification including toxicity and antimicrobial peptide prediction. The same idea has been extended to protein sequence generation tasks (Melnyk et al. 2023).
- *Word-level adversarial reprogramming (WARP)* (Hambardzumyan, Khachatrian, and May 2021): WARP inserts trainable prompt tokens to a pre-trained masked language model and uses a trainable dense layer for the output mapping. On the GLUE benchmark, WARP attains a comparable performance to modern language models while having a much smaller number of trainable parameters. The authors also demonstrate the data efficiency of WARP in the few-shot setting.

3 Theoretical Characterization of Model Reprogramming

This section summarizes the theoretical characterization and interpretation of the working mechanism of model reprogramming based on current studies.

3.1 Error Analysis in Representation Alignment

Using the notation in Table 1, let the source model $f_{\mathcal{S}}$ be a pre-trained K-way neural network classifier $f_{\mathcal{S}}(\cdot) = \eta(z_{\mathcal{S}}(\cdot))$ with a softmax layer $\eta(\cdot)$ as the model output. Let $\ell(x, y) \triangleq \|f(x) - y\|_2$ denote the root mean squared error and let $\mathbb{E}_{\mathcal{D}}[\ell(x, y)] \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(x, y)]$ denote its population risk on $(x, y) \sim \mathcal{D}$. Under some mild assumptions such as one-to-one label mapping, (Yang, Tsai, and Chen 2021) proves that the risk of the target task (target risk) via model reprogramming is upper bounded by the summation of two terms, the risk of the source model on the source task (source risk) and the representation alignment error measuring the distributional difference between the latent representations of the source and the reprogrammed target data based on the same source model $f_{\mathcal{S}}$. The theorem is stated as follows.

Theorem 1: Let θ^* be the learned input transformation parameters from (1) and assume one-to-one label mapping. The population risk for the target task via reprogramming a K-way source neural network classifier $f_{\mathcal{S}}(\cdot) = \eta(z_{\mathcal{S}}(\cdot))$, denoted by $\mathbb{E}_{\mathcal{D}_{\mathcal{T}}}[\ell_{\mathcal{T}}(\widetilde{x}_t(\theta^*), y_t)]$, is upper bounded by

$$\underbrace{\mathbb{E}_{\mathcal{D}_{\mathcal{T}}}[\ell_{\mathcal{T}}(\widetilde{x}_t(\theta^*), y_t)]}_{\text{target risk}} \leq \underbrace{\epsilon_{\mathcal{S}}}_{\text{source risk}} + 2\sqrt{K} \cdot \underbrace{\mathcal{W}_1(\mu(z_{\mathcal{S}}(\widetilde{x}_t(\theta^*)), \mu(z_{\mathcal{S}}(x_s)))_{x_t\sim\mathcal{D}_{\mathcal{T}},\, x_s\sim\mathcal{D}_{\mathcal{S}}}}_{\text{representation alignment loss via reprogramming}},$$

where $\widetilde{x}_t(\theta^*)$ is an input-transformed target data sample defined in (1) and $\mathcal{W}_1(\mu_a, \mu_b)$ denotes the Wasserstein-1 distance between two probability distributions μ_a and μ_b.

The theorem provides several insights into characterizing the success of model reprogramming, suggesting that the target risk via reprogramming can be improved (lower is better) when (i) a better source model (in terms of lower source risk) is used, and (ii) the source and reprogrammed target data representations based on $f_{\mathcal{S}}$ are better aligned (in terms of smaller Wasserstein distance). This analysis also explains that cross-domain model reprogramming is feasible as long as the target representations can be aligned with the source representations after reprogramming. In other words, model reprogramming can be interpreted as reusing a pre-trained source model as an efficient feature extractor to learn representation alignment. Some numerical evidence is given in (Yang, Tsai, and Chen 2021) to show that the Wasserstein distance indeed decreases during model reprogramming.

3.2 Other Interpretations

In addition to representation alignment, we summarize different interpretations and analyses of model reprogramming based on existing works. Using first-order approximation, (Zheng et al. 2023) shows that the optimal loss decrement in reprogramming is equivalent to the ℓ_1-norm of the average input gradient. Consequently, model reprogramming is more successful when input gradients of target data are more aligned, and when inputs have higher dimensionality. (Hambardzumyan, Khachatrian, and May 2021) motivates the data efficiency and fast adaptivity of model reprogramming in language models from the perspective of learning optimal trainable input prompts for efficient adaptation to downstream tasks. Notably, when the source and target domains are both in computer vision, model reprogramming essentially reduces to visual prompting (Bahng et al. 2022).

4 Open-ended Research Questions, Ongoing Efforts, and Opportunities

In this section, we discuss several open-ended research questions and opportunities for model reprogramming.

Model Reprogramming beyond Supervision. Most of the existing works on model reprogramming focus on the supervised setting, where all data samples in target domains are associated with data labels for training. With the advance of semi-supervised (a mixture of labeled and unlabeled data samples), unsupervised (unlabeled data samples only), and self-supervised (self-generated pseudo labels) machine learning methods, it is essential to understand how unlabeled data and supervision-free training (possibly with data augmentation) improve model reprogramming.

Reprogramming Foundation Models. The emergence of foundation models (Bommasani et al. 2021), including large language models and generative AI applications, has led to a critical paradigm shift in the trend of machine learning research from designing task-specific and modality-dependent deep learning models to developing task-agnostic and modality-independent general-purpose models. Foundation models feature supervision-free pre-training and efficient finetuning to different downstream tasks. However, not every domain and every researcher has the luxury of accessing foundation models due to resource limitations. Demonstrating model reprogramming can serve as an affordable solution to repurposing and reusing a foundation model with resource efficiency can further accelerate AI-assisted scientific discovery and democratize machine learning research, as explored in (Xu et al. 2023).

Reprogramming for Improved Model Properties. Ensuring and instilling trustworthiness in machine learning based technology is the new norm for building a healthy and sustainable ecosystem between technology, end users, and our society. Studying how model reprogramming can be used as an efficient calibration tool to improve different trustworthiness properties while having minimal impact on the original utility is an important research direction. The model properties include fairness (Zhang et al. 2022), robustness (Chen et al. 2023a), explainability, privacy (Li et al. 2023), uncertainly quantification (Tang, Chen, and Ho 2022), and energy efficiency (Sun et al. 2023), to name few.

Joint Model Reprogramming and Fine-tuning. Theoretically, joint model reprogramming and fine-tuning on the right subset of the pre-trained model's parameters should perform no worse than standalone model reprogramming or transfer learning. However, in the resource-limited setting such joint training on a large-scale pre-trained model is challenging because identifying which model parameters to finetune is highly non-trivial. Studying how and when joint model reprogramming and fine-tuning can outperform standalone model reprogramming and transfer learning can

lead to more resource-efficient machine learning. Notably, when applied to the task of low-resource speech commands recognition (Yen et al. 2023) and music genre classification (Hung et al. 2023), such joint training with careful fine-tuning is shown to deliver improved performance. Moreover, although originally not cast as a model reprogramming method, the proposal of universal compute engine (Lu et al. 2021) that reuses a pre-trained transformer for attaining non-trivial performance on a diverse set of tasks, by joint fine-tuning the input embedding, the layer-norm parameters, and training the output mapping layer, suggests great potential for resource-efficient machine learning.

Implications on Machine Learning Systems with Heterogeneous Computing. The advantage of "no model fine-tuning" and resource efficiency in model reprogramming can be favorable to machine learning systems featuring heterogeneous computation and memory constraints, such as edge computing, cloud computing, and federated learning. For energy and memory efficiency, the source model (which is usually large) can be intact and stored on the server side, while the input transformation and output mapping layers can be implemented at the edge device for reprogramming. During training, the communication cost between the server and an edge device is expected to be greatly reduced because such a paradigm does not require fine-tuning the model parameters at the server. It also spares the need for the edge device to download and store a local copy of the model from the server. (Arif, Gittens, and Chen 2023) improves the privacy-utility tradeoff in differentially private training with model reprogramming over standard transfer learning methods in both centralized and federated learning scenarios.

References

Aharon, M.; Elad, M.; and Bruckstein, A. 2006. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11): 4311–4322.

Arif, H.; Gittens, A.; and Chen, P.-Y. 2023. Reprogrammable-FL: Improving Utility-Privacy Tradeoff in Federated Learning via Model Reprogramming. In *First IEEE Conference on Secure and Trustworthy Machine Learning*.

Bahng, H.; Jahanian, A.; Sankaranarayanan, S.; and Isola, P. 2022. Visual prompting: Modifying pixel space to adapt pre-trained models. *arXiv preprint arXiv:2203.17274*.

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, 1877–1901.

Chen, A.; Lorenz, P.; Yao, Y.; Chen, P.-Y.; and Liu, S. 2023a. Visual prompting for adversarial robustness. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. IEEE.

Chen, A.; Yao, Y.; Chen, P.-Y.; Zhang, Y.; and Liu, S. 2023b. Understanding and Improving Visual Prompting: A Label-Mapping Perspective. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Chen, L.; Fan, Y.; and Ye, Y. 2021. Adversarial Reprogramming of Pretrained Neural Networks for Fraud Detection. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2935–2939.

Dau, H. A.; Bagnall, A.; Kamgar, K.; Yeh, C.-C. M.; Zhu, Y.; Gharghabi, S.; Ratanamahatana, C. A.; and Keogh, E. 2019. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6): 1293–1305.

Dinh, T.; Seo, D.; Du, Z.; Shang, L.; and Lee, K. 2022. Improved Input Reprogramming for GAN Conditioning. *arXiv preprint arXiv:2201.02692*.

Elsayed, G. F.; Goodfellow, I.; and Sohl-Dickstein, J. 2019. Adversarial Reprogramming of Neural Networks. In *International Conference on Learning Representations*.

Hambardzumyan, K.; Khachatrian, H.; and May, J. 2021. WARP: Word-level Adversarial ReProgramming. 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)*, 4921–4933. Online: Association for Computational Linguistics.

Hung, Y.-N.; Yang, C.-H. H.; Chen, P.-Y.; and Lerch, A. 2023. Low-Resource Music Genre Classification with Cross-Modal Neural Model Reprogramming. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. IEEE.

Jaiswal, A.; Babu, A. R.; Zadeh, M. Z.; Banerjee, D.; and Makedon, F. 2021. A survey on contrastive self-supervised learning. *Technologies*, 9(1): 2.

Jing, Y.; Yuan, C.; Ju, L.; Yang, Y.; Wang, X.; and Tao, D. 2023. Deep Graph Reprogramming. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 24345–24354.

Kloberdanz, E.; Tian, J.; and Le, W. 2021. An Improved (Adversarial) Reprogramming Technique for Neural Networks. In *International Conference on Artificial Neural Networks*, 3–15. Springer.

Lee, K.; Suh, C.; and Ramchandran, K. 2020. Reprogramming GANs via input noise design. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 256–271. Springer.

Li, Y.; Tsai, Y.-L.; Ren, X.; Yu, C.-M.; and Chen, P.-Y. 2023. Exploring the Benefits of Visual Prompting in Differential Privacy. *International Conference on Computer Vision*.

Liu, S.; Chen, P.-Y.; Kailkhura, B.; Zhang, G.; Hero, A.; and Varshney, P. K. 2020. A Primer on Zeroth-Order Optimization in Signal Processing and Machine Learning. *IEEE Signal Processing Magazine*.

Lu, K.; Grover, A.; Abbeel, P.; and Mordatch, I. 2021. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 1.

Melnyk, I.; Chenthamarakshan, V.; Chen, P.-Y.; Das, P.; Dhurandhar, A.; Padhi, I.; and Das, D. 2023. Reprogramming Pretrained Language Models for Antibody Sequence Infilling. In *International Conference on Machine Learning*, 24398–24419. PMLR.

Neekhara, P.; Hussain, S.; Du, J.; Dubnov, S.; Koushanfar, F.; and McAuley, J. 2022. Cross-modal Adversarial Reprogramming. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2427–2435.

Neekhara, P.; Hussain, S.; Dubnov, S.; and Koushanfar, F. 2019. Adversarial Reprogramming of Text Classification Neural Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Randazzo, E.; Mordvintsev, A.; Niklasson, E.; and Levin, M. 2021. Adversarial Reprogramming of Neural Cellular Automata. *Distill*, 6(5): e00027–004.

Sun, H.-L.; Hsiung, L.; Chandramoorthy, N.; Chen, P.-Y.; and Ho, T.-Y. 2023. NeuralFuse: Learning to Improve the Accuracy of Access-Limited Neural Network Inference in Low-Voltage Regimes. *arXiv preprint arXiv:2306.16869*.

Tang, Y.-C.; Chen, P.-Y.; and Ho, T.-Y. 2022. Neural Clamping: Joint Input Perturbation and Temperature Scaling for Neural Network Calibration. *arXiv preprint arXiv:2209.11604*.

Tsai, Y.-Y.; Chen, P.-Y.; and Ho, T.-Y. 2020. Transfer learning without knowing: Reprogramming black-box machine learning models with scarce data and limited resources. In *International Conference on Machine Learning*, 9614–9624.

Vinod, R.; Chen, P.-Y.; and Das, P. 2023. Reprogramming Pretrained Language Models for Protein Sequence Representation Learning. *arXiv preprint arXiv:2301.02120*.

Xu, S.; Yao, J.; Luo, R.; Zhang, S.; Lian, Z.; Tan, M.; and Wang, Y. 2023. Towards Efficient Task-Driven Model Reprogramming with Foundation Models. *arXiv preprint arXiv:2304.02263*.

Yang, C.-H. H.; Tsai, Y.-Y.; and Chen, P.-Y. 2021. Voice2Series: Reprogramming Acoustic Models for Time Series Classification. In *International Conference on Machine Learning*.

Yen, H.; Ku, P.-J.; Yang, C.-H. H.; Hu, H.; Siniscalchi, S. M.; Chen, P.-Y.; and Tsao, Y. 2023. Neural model reprogramming with similarity based mapping for low-resource spoken command classification. *INTERSPEECH Conference*.

Zhang, G.; Zhang, Y.; Zhang, Y.; Fan, W.; Li, Q.; Liu, S.; and Chang, S. 2022. Fairness reprogramming. *Advances in Neural Information Processing Systems*, 35: 34347–34362.

Zheng, Y.; Feng, X.; Xia, Z.; Jiang, X.; Demontis, A.; Pintor, M.; Biggio, B.; and Roli, F. 2023. Why adversarial reprogramming works, when it fails, and how to tell the difference. *Information Sciences*, 632: 130–143.