In-Context Meta LoRA Generation

Yihua Shao[1,2] , **Minxi Yan**[3] , **Yang Liu**[4] , **Siyu Chen**[3] , **Wenjie Chen**[3] , **Xinwei Long**[2] , **Ziyang Yan**[5] , **Lei Li**[6] , **Chenyu Zhang**[5] , **Nicu Sebe**[5] , **Hao Tang**[2] , **Yan Wang**[3] , **Hao Zhao**[3] , **Mengzhu Wang**[1] and **Jingcai Guo**[1*]

[1]Department of COMP/LSGI, The Hong Kong Polytechnic University, Hong Kong SAR
[2]School of Computer Science, Peking University, China
[3]Institute for AI Industry Research (AIR), Tsinghua University, China
[4]Beijing Institute for General Artificial Intelligence, China
[5]Department of Information Engineering and Computer Science, University of Trento, Italy
[6]Department of Computer Science, University of Copenhagen, Denmark
yihuajerry@gmail.com, jc-jingcai.guo@polyu.edu.hk

Abstract

Low-rank Adaptation (LoRA) has demonstrated remarkable capabilities for task-specific fine-tuning. However, in scenarios involving multiple tasks, training a separate LoRA model for each task results in considerable inefficiency in terms of storage and inference. Moreover, existing parameter generation methods fail to capture the correlations among these tasks, making multitask LoRA parameter generation challenging. To address these limitations, we propose the In-Context Meta LoRA (ICM-LoRA), a novel approach that efficiently achieves task-specific customization of large language models (LLMs). Specifically, we use training data from all tasks to train a tailored generator, Conditional Variational Autoencoder (CVAE). CVAE takes task descriptions as inputs and produces task-aware LoRA weights as outputs. These LoRA weights are then merged with LLMs to create task-specialized models without the need for additional fine-tuning. Furthermore, we utilize in-context meta-learning for knowledge enhancement and task mapping to capture the relationship between tasks and parameter distributions. Consequently, our method achieves more accurate LoRA parameter generation for diverse tasks using CVAE. ICM-LoRA enables more accurate LoRA parameter reconstruction than current parameter reconstruction methods and is useful for implementing task-specific enhancements to LoRA parameters. Simultaneously, our method occupies 283MB, which is only 1% of the storage space required by the original LoRA. The code is available at https://github.com/YihuaJerry/ICM-LoRA
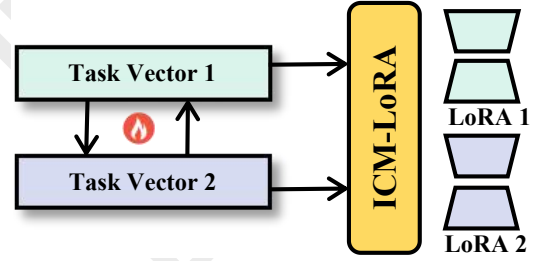
*Corresponding author: *Jingcai Guo*.

Figure 1: ICM-LoRA achieves accurate reconstruction of LoRA parameters through task vectors context modeling.

1 Introduction

Large-scale models (LLMs/MLMs) have become the cornerstone of modern AI applications [Xiao *et al.*, 2024; Dubey *et al.*, 2024; Shao *et al.*, 2024]. However, these models typically require substantial amounts of data for fine-tuning. We always fine-tune LLMs with Low-rank Adaptation (LoRA) [Hu *et al.*, 2021] using task-specific data. In scenarios with numerous subtasks [Erkoç *et al.*, 2023; Yan *et al.*, 2024], the current approach of training a separate LoRA for each subtask leads to inefficiency in storage and inference. For example, in multitask scenarios, the weights of LoRA can become prohibitively expensive to store, necessitating more efficient solutions. Although FM-Delta [Mizrahi *et al.*, 2017] employs a novel compression scheme that significantly reduces the need for storage by storing compressed fine-tuned models, it does not address the issue of capturing the correlations between sub-tasks.

Current parameter generation methods [Platanios *et al.*, 2018; Wortsman *et al.*, 2022; Jin *et al.*, 2024] can only generate LoRA parameters for a single task, and it is not possible to implement the simultaneous generation of LoRA weights required for different tasks using only one generator. Moreover, the current parameter generation training method lacks context modeling capability, making it difficult to implement multi-use multi-task enhancement of LoRA weights. This causes a significant storage burden when storing the LoRA weights and training data.

Therefore, we propose the **In-Context Meta LoRA (ICM-LoRA)** to implement the generation of different tasks LoRA with a self-designed generator, a Conditional Variational Autoencoder (CVAE). As shown in figure 1, ICM-LoRA utilizes task vectors for context modeling through in-context meta-learning, allowing the augmented CVAE to learn the parameter distribution features. By combining in-context learning and meta-learning and using task vectors as modeling labels, we achieve meta-enhancement for LoRA parameters. Furthermore, ICM-LoRA can eliminate the dependence on data and storage and only requires a generator to implement parameter generation.

We evaluate our method for both text and visual tasks on different models. For visual tasks, we select target detection tasks and use the COCO dataset [Lin *et al.*, 2014] to classify subclasses based on the detection task labels for the experiments. For language tasks, we choose The Pile [Gao *et al.*, 2020] as the training corpus and use five different subsets to simulate multi-class training tasks and validate the model on the validation set. The results indicate that CVAE can successfully generate different tasks' LoRA parameters. Compared with current methods, the generated LoRA parameters exhibit less accuracy loss. In addition, compared to original datasets and LoRA weights, our generator significantly reduce storage.

In summary, the contributions of our approach can be summarized as follows:

1) We propose a novel framework, In-Context Meta LoRA (ICM-LoRA), which uses a self-designed parameter generator, a Conditional Variational Autoencoder (CVAE), to generate LoRA weights, addressing the inefficiency of training separate LoRA models for multiple sub-tasks.

2) We employ in-context meta-learning for knowledge enhancement and task matching, which enables the generator to better learn the correspondence between tasks and model parameter distributions.

3) Compared to existing methods, CVAE can generate task-specific LoRA parameters that are the same as or even better than the original LoRA. In addition, our ICM-LoRA could cost only 1% storage compared with the original datasets.

2 Related Works

2.1 Parameters Generation

The core of parameter generation is to help the model generate a distribution similar to of the that original model. As one of the pioneers, [Platanios *et al.*, 2018] introduced a contextual parameter generator (CPG) addressing the challenge of training separate models for each language pair in neural machine translation (NMT). Some methods like stochastic neural networks [Sompolinsky *et al.*, 1988; Bottou and others, 1991; Wong, 1991; Schmidt *et al.*, 1992; Murata *et al.*, 1994; Graves, 2011] and Bayesian neural networks [Neal, 2012; Kingma, 2013; Rezende *et al.*, 2014; Kingma *et al.*, 2015; Blundell *et al.*, 2015; Gal and Ghahramani, 2016] improved the robustness and generalisation of the model through the

prior probability distribution of the parameters, but these methods performed poorly in large-scale or complex scenarios. HyperNetworks [Ha *et al.*, 2016] generate the parameters of large networks through small networks. With the development of diffusion, methods such as G.pt [Peebles *et al.*, 2022] and P-diff families [Wang *et al.*, 2024; Zhao *et al.*, 2021] began to use diffusion to generate normal scale parameters; however, they are limited in generating parameters that are too large or too small. Furthermore, COND P-DIFF [Jin *et al.*, 2024] first applies parameter generation to generate LoRA parameters, but it only generates LoRA models for coarse-grained tasks, and its parameters for generating LoRA for fine-grained tasks do not perform well. Therefore, we design a fine-grained task Lora generator that uses in-context learning (Sec. 2.2) to enhance the ability of context understanding of a generator model such as diffusion.

2.2 In-Context Learning

In-Context Learning (ICL) has emerged as a powerful paradigm in machine learning. As a pioneering work, [Brown *et al.*, 2020] reveals for the first time the learning ability of large language models in the presence of a small number of examples. Building upon this, for training LLMs, MetaICL [Min *et al.*, 2021] integrated tasks into the ICL format and enabled models to achieve performance similar to direct fine-tuning. Lamda [Thoppilan *et al.*, 2022] emphasizes instruction tuning for models to better understand task descriptions instead of just examples. Self-instruct [Wang *et al.*, 2022b] enables LLMs to generate instructions for task alignment to explore enhancing ICL, while [Wei *et al.*, 2022] introduced Chain-of-Thoughts (CoT) as an intermediate step between input and output to boost LLM reasoning in ICL. Task separation ICL, such as Self-Ask [Press *et al.*, 2022] and ICAP [Chi and Wylie, 2014], has explored multi-stage ICL, where tasks are broken down into simpler sub-tasks, each with its own set of demonstrations, and LLMs can process them individually. SuperICL [Xu *et al.*, 2023] utilizes smaller models as plugins to effectively execute tasks within the LLM framework, demonstrating the potential of hybrid model approaches in ICL. In scenario understanding, In-Context LoRA (IC-LoRA) [Huang *et al.*, 2024] and [Hendel *et al.*, 2023] respectively apply ICL to image generation by diffusion and context classification by LLM. In this paper, we apply ICL to the generator to help it better understand the context information in Lora.

2.3 Dataset Condensation

Dataset Condensation (DC) aims to create a compact and representative subset of the original training data. As a foundational work, [Zhao *et al.*, 2020] first tried to compress the data by matching the gradients of the synthetic dataset with the original to ensure that the condensed dataset retains the essential characteristics for effective model training. [Zhao and Bilen, 2021] further achieves more efficient data enhancement to synthesise more informative synthetic images by using differentiable twin enhancement (DSA). Building on this, [Zhao and Bilen, 2023] also explores DC with Distribution Matching, optimizing synthetic data to match the original distribution in embedding spaces via maximum mean
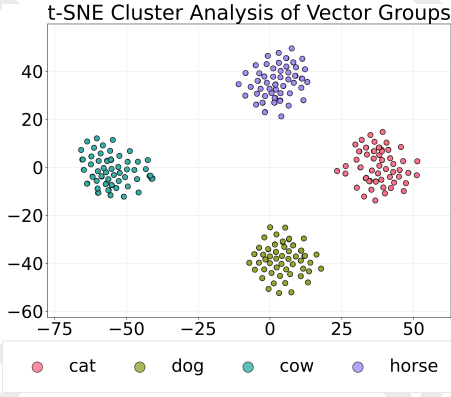
Figure 2: **Task Hidden Space Distribution.** Hidden states of tasks LoRA parameters have clustering phenomena.

discrepancy (MMD). [Wei *et al.*, 2024] compresses data by matching latent space quantiles and minimizing distribution fit statistics and [Lee *et al.*, 2022] further advances the field by modifying the loss function to capture class differences with a bi-level warm-up strategy for stable optimization. [He *et al.*, 2024] integrates multiple dataset compression processes to get datasets of various sizes and introduces adaptive subset loss to reduce subset degradation. [Wang *et al.*, 2022a] proposes a new method for dataset compression by aligning features, while [Liu *et al.*, 2024] introduces a dual-domain matching method for dataset condensation in time series classification, which further extends the applicability of DC to different data types. In our work, we discard the original dataset and deposit the different task information of the dataset into the generator model for data information aggregation and compression.

3 Methodology

In this section, we present our approach in terms of an overview, task vector extraction, and parameter sampling and reconstruction for model customization.

3.1 Overview

As shown in Figure 2, we extracted the final hidden states of five different categories, dog, sofa, cat, bicycle and motorbike, at the last time step from the last layer of Florence-2's [Xiao *et al.*, 2024] decoder and visualized them with S-NE [Van der Maaten and Hinton, 2008]. The visualization demonstrates that hidden states from different categories form distinct clusters. For simplicity, we refer to the final hidden states at the last time step from the last layer of the decoders as 'task vectors' [Hendel *et al.*, 2023].

From the information mentioned above, we can draw two conclusions: First, task vectors from different categories are discriminative, as they form distinct clusters. Second, task vectors can represent the high-level features of different categories because the last layer and final time step typically capture high-level representations.

Therefore, task vectors satisfy the two key properties of condition vectors: discriminability and representativeness of the target condition. Based on this, we hypothesize that task

vectors can serve as condition vectors to effectively guide the generation process in the CVAE model.

Figure 3 provides an illustrative overview of the proposed method. Our method comprises three parts.

1) **Preparing LoRA parameter data and extracting task vectors.** We fine-tune a Large Language Model (LLM) or a Large Vision-Language Model (LVLM) on a specific task category and save checkpoints from the final stages of the training process. These checkpoints serve as training data for subsequent generative models. Next, we perform inference using the fine-tuned model on randomly selected samples from a specific task category, such as cat or dog detection. During inference, we extract the hidden states from the last layer at the final time step. These hidden states are then averaged to derive a task vector that represents a specific task category.

2) **Training the CVAE model.** The LoRA parameters extracted from the fine-tuned model checkpoints, along with the task vectors, are used as training data to train a Conditional Variational Autoencoder (CVAE). We use in-context meta-learning to implement CVAE for modeling relationships between multiple tasks and to enhance LoRA parameter distribution learning.

3) **Generating and applying LoRA parameters.** Using the trained CVAE, we sample from a Gaussian distribution to reconstruct the LoRA parameters for the target task category. The reconstructed LoRA parameters are then used to perform inference on the test set, enabling the model to generalize effectively to the specific task.

3.2 Task Vector Extraction

Considering the contextual capabilities of large-scale models and the evidence that in-context learning can generate task-specific representations [Hendel *et al.*, 2023], we fine-tune a pre-trained model using LoRA [Hu *et al.*, 2021] for a specific task category.

Due to the hidden state corresponding to the last token, we extracted the hidden state of each sample h_i^{last} with the last token generated by LLM. For a specific set of task samples $\{x_i\}_{i=1}^{N}$, LLM generates hidden states of the task from the last layer. These hidden states are then averaged to produce a compact *task vector*, which can be expressed as Eq. (1):

$$v_{\text{task}} = \frac{1}{N} \sum_{i=1}^{N} h_i^{\text{last}}, \tag{1}$$

where N is the number of task-specific samples, $v_{\text{task}} \in R^d$ represents the task vector for the given category, and d represents the dimensionality of the hidden state.

Since the last time steps of many natural language processing scenarios contain complete information about the input sequence, we choose the last token to generate the hidden state h_i^{last}. For the task of mapping an input sequence to a single vector, the hidden state of the last marker typically integrates information from all previous time steps, thus becoming a compact representation of the overall semantics of input x_i.
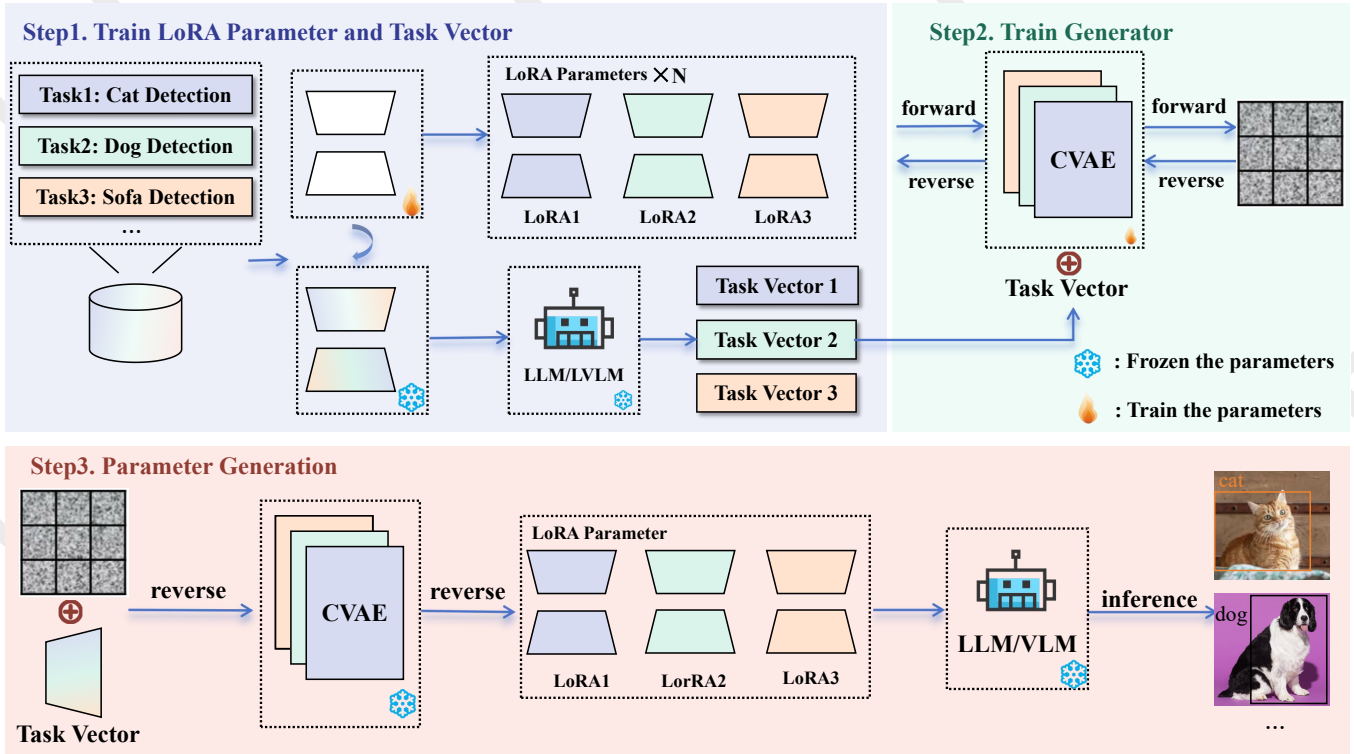
Figure 3: **Overall Framework of ICM-LoRA.** We train different task LoRAs based on task categories and generate their task vectors. Then, we train a self-designed CVAE utilizing these task data by in-context meta learning. Finally, we generate the task LoRA by training CVAE.

Furthermore, we extract the hidden states from the final layer because this layer typically represents the most abstract and task-specific feature space. As information flows through the network, the first few layers typically encode general linguistic or structural features, whereas the last layer captures high-level semantic features specific to the current task. This recursion allows the final layer to act as a task feature extractor, providing a representation that is well-suited for generating task vectors. The abstract nature of the last layer ensures that the generated task vector x_{task} can effectively capture the key features of the task.

3.3 Conditional Variational Autoencoder

Based on the Variational Autoencoder (VAE) [Kingma, 2013], we employ a Conditional Variational Autoencoder (CVAE) to model the distribution of LoRA parameters conditioned on task vectors. The CVAE consists of an encoder $q_\phi(z \mid l, v_{\text{task}})$, which maps the LoRA parameter l and task vector v_{task} to a latent representation z, and a decoder $q_\theta(z \mid l, v_{\text{task}})$, which reconstructs l from z and v_{task}. The encoder and decoder are conditioned on the task vector v_{task}, which provides additional information to guide the generation of LoRA parameters.

The encoder $q_\phi(z \mid l, v_{\text{task}})$ models the approximate posterior distribution over the latent variables z, given LoRA parameter l and task vector v_{task}. The encoder takes as input the concatenation of the LoRA parameter l and the task vector v_{task}, which is denoted as Eq. (2),

$$x = [l; v_{\text{task}}] \in R^{d_l + d_{\text{task}}}, \tag{2}$$

where d_l and v_{task} represent the dimensions of the LoRA parameter l and the task vector v_{task}, respectively.

The concatenated vector x passed through a neural network, which outputs the parameters of the approximate posterior distribution. This process can be expressed as Eq. (3),

$$q_\phi(z \mid l, v_{\text{task}}) = \mathcal{N}\left(z; \mu_\phi, \sigma_\phi^2(x)\right), \tag{3}$$

where $\mu_\phi(x)$ and $\sigma_\phi^2(x)$ are the mean and variance of the latent variable z, computed from the input x.

The latent variable z is then sampled from this distribution using the reparameterization trick, which can be represented as Eq. (4):

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \tag{4}$$

where $\epsilon \sim \mathcal{N}(0, I)$ is a noise term, and $\odot$ represents element-wise multiplication.

The decoder $p_\theta(l \mid z, v_{\text{task}})$ models the likelihood of the LoRA parameter l given the latent variable z and task vector v_{task}. The decoder receives z and v_{task} as input, which are concatenated as Eq. (5),

$$x' = [z; v_{\text{task}}] \in R^{d_z + d_{\text{task}}}, \tag{5}$$

where d_z is the dimensionality of the latent variable z.

The concatenated vector x' is then input into a neural network to output the parameters of the likelihood distribution for the LoRA parameter l, and the process can be represented as Eq. (6):

$$p_\theta(z, v_{\text{task}}) = \mathcal{N}\left(l, \hat{\mu}_\theta(x'), \hat{\sigma}_\theta^2(x')\right), \tag{6}$$

where $\hat{\mu}_\theta(x')$ and $\hat{\sigma}_\theta^2(x')$ are the mean and variance of the predicted LoRA parameter l, computed from the input x'.

The decoder aims to minimize the reconstruction error, ensuring that the generated LoRA parameters match the true parameters as closely as possible. The latent space $z \in R^k$ is assumed to follow a Gaussian prior distribution which could be represented as Eq. (7):

$$p(z) = \mathcal{N}(z; 0, I), \tag{7}$$

where I is the identity matrix.

The objective is to maximize the evidence lower bound (ELBO), which consists of two terms: the reconstruction term and the regularization term. The ELBO could be expressed as Eq. (8),

$$\mathcal{L} = -E_{q_\phi(z|l,v_{\text{task}})} [\log p_\theta(l \mid z, v_{\text{task}})] + \\ KL(q_\phi(z \mid l, v_{\text{task}}) \parallel p(z)), \tag{8}$$

where $KL(\cdot \parallel \cdot)$ represents the Kullback-Leibler divergence. The first term encourages the decoder to reconstruct accurate LoRA parameters, while the second term regularizes the latent space to match the Gaussian prior.

By conditioning both the encoder and decoder on the task vector v_{task}, the model learns to generate LoRA parameters that are specific to the given task, leading to task-aware representations in the latent space.

For CVAE to generate task LoRA parameters, we utilize CLIP's [Radford *et al.*, 2021] text encoder to output the task vector v_{task}. A sample z is drawn from the prior distribution $p(z)$, and the decoder generates the corresponding LoRA parameter:

$$l_{\text{generated}} = p_\theta(l \mid z, v_{\text{task}}). \tag{9}$$

4 Experiments

In this section, we evaluated several tasks on LLMs and MLMs. We evaluated the task performance of the LoRA [Hu *et al.*, 2021] generated by the current LoRA parameters generation methods. This demonstrates the effectiveness and reasonableness of our approach.

4.1 Experiment Setting

Baselines. We chose the original model, LoRA, LoRA generated by Model Soup [Wortsman *et al.*, 2022] and COND P-DIFF [Jin *et al.*, 2024] as baseline to test the advantages of our method on different tasks.

Datasets. For the computer vision task, we select the most representative target detection task for to experiment. We choose the COCO [Lin *et al.*, 2014] dataset and divide it into different subclasses based on the detection task labels. For the language modelling task, we employ The Pile [Gao *et al.*, 2020] as the training corpus. To simulate the multi-category training tasks, we pick five various subsets from the Pile corpus and validate our method on the test sets.

Data Preparation. The model fine-tuning process also produces a series of LoRA matrices $\{L_t\}_{t=1}^{T}$ with different ranks r, where T represents the number of fine-tuning steps. Each matrix $L_t \in \mathbf{R}^{m \times n}$ is flattened into a one-dimensional vector $l_t \in \mathbf{R}^{m \times n}$ to facilitate alignment with the task vector v_{task}.

These flattened LoRA parameters, along with the corresponding task vectors, form the training dataset $\{(v_{\text{task}}, l_t)\}$ for the self-designed CVAE.

Training Strategies. The CVAE model employs a 12-layer 1D CNN architecture for both the encoder and decoder. The loss function for the CVAE combines the Kullback-Leibler divergence (KLD) and reconstruction loss, with the KLD weight set to 0.005. The loss function could be expressed as Eq. (8) We fine-tuned the model on a specific task using LoRA (Low-Rank Adaptation) for a total of 150 epochs, saving the LoRA parameters from the final 50 epochs. The task vector is extracted from the last token of the last layer in the CLIP [Radford *et al.*, 2021]. Subsequently, the CVAE model is trained for 2,000 epochs to ensure robust learning of the latent space. All experiments were conducted on a single NVIDIA A800 GPU, with each experiment taking approximately 3 hours to complete.

4.2 Main Results

We conduct experiments on computer vision tasks and natural language processing tasks respectively, and demonstrate that our approach generalizes across models and can be adapted to tasks of multiple modalities.

Object Detection. As shown in Table 1, we selected several subsets of the more conventional tasks and fine-tuned on Florence-2 [Xiao *et al.*, 2024]. The LoRA parameters generated by ICM-LoRA in the subset of expert tasks in the COCO dataset have the smallest difference in effect from the original LoRA parameters, and even the LoRA parameters generated by ICM-LoRA outperform the original LoRA in some of the tasks. This suggests that our method generates more LoRA parameters than the other methods. complete. By adding in-context learning, ICM-LoRA's understanding of task scenarios is enhanced compared to COND P-DIFF, implementing the effect of outperforming the original LoRA on some tasks.

As shown in Table 3, our method significantly achieves task-specific dataset compression using much less storage than the original LoRA weights with the original dataset. The compression of the visual dataset is achieved through the parameter generation method, which significantly reduces the storage cost. This shows that our approach not only generates task-corresponding LoRAs more accurately but also enables task-based data compression.

As shown in Figure 4, both Model Soup and COND P-DIFF were poorly labeled in the detection, and COND P-DIFF even had a false detection in the complex environment. This indicates that the reconstructed LoRA cannot be successfully adapted to LVLM.

Language Modeling. When it comes to the language modelling task, we set the LoRA rank $r = 2$. We fine-tuned the Llama-3-8B [Dubey *et al.*, 2024] model across different tasks and proposed the result of five subsets from the Pile corpus, including ArXiv, Books, Ubuntu, Wikipedia, and Gutenberg. As shown in Table 2, compared to other methods, ICM-LoRA achieves the lowest perplexity and bits-per-character. This outperformance clearly shows its superiority in these specific tasks of language modelling.

Method	Dog		Bicycle		Cat		Sofa		Motorbike	
	MAP50	MAP75	MAP50	MAP75	MAP50	MAP75	MAP50	MAP75	MAP50	MAP75
Original LoRA	0.96	0.89	0.90	0.81	0.94	0.90	0.95	0.86	0.82	0.78
Original Model	0.92	0.87	0.89	0.80	0.93	0.89	0.00	0.00	0.00	0.00
Model Soup	0.93	0.87	0.90	0.78	0.93	0.88	0.81	0.74	0.81	0.72
COND P-DIFF	0.94	0.87	0.90	0.77	0.93	0.89	0.84	0.78	0.80	0.74
ICM-LoRA	**0.96**	**0.89**	**0.90**	**0.81**	**0.95**	**0.91**	**0.95**	**0.86**	**0.83**	**0.78**

Table 1: **Parameter Reconstruction Results for LoRA Rank $r = 2$ in Object Detection Task.** ICM-LoRA generates LoRA weights are closest to the original LoRA, and even better than the original LoRA in some tasks.

Method	ArXiv		Books		Ubuntu		Wikipedia		Gutenberg	
	PPL	BPC	PPL	BPC	PPL	BPC	PPL	BPC	PPL	BPC
Original LoRA	6.75	0.41	7.07	0.48	9.66	0.58	5.54	0.43	8.60	0.62
Original Model	7.76	0.44	7.67	0.51	10.00	0.59	6.07	0.46	8.75	0.63
Model Soup	7.00	0.43	7.08	0.48	9.67	0.59	5.56	0.45	8.61	0.63
COND P-DIFF	6.73	0.41	7.10	0.49	9.67	0.58	5.55	0.44	8.60	0.62
ICM-LoRA	**6.74**	**0.40**	**7.07**	**0.48**	**9.65**	**0.58**	**5.54**	**0.43**	**8.59**	**0.61**

Table 2: **Parameter Reconstruction Results for LoRA Rank $r = 2$ in Language Modeling.** Compared with baseline methods, ICM-LoRA generates LoRA parameters with a lower PPL ($\downarrow$) and BPC ($\downarrow$) on different subsets, equaling or even surpassing the original LoRA.

Method	Rank			
	$r = 1$	$r = 2$	$r = 4$	$r = 8$
Original COCO	25G			
Original LoRA	2.1G	4.3G	8.5G	16.9G
Model Soup	423MB	453MB	478MB	504MB
COND P-DIFF	314MB	314MB	318MB	326MB
ICM-LoRA	**283MB**	**283MB**	**283MB**	**283MB**

Table 3: **Storage Memory Required for Different Methods.** In the vision task, ICM-LoRA and COND P-DIFF take up less storage overhead compared to the original dataset with the original LoRA parameter weights.

In some subtasks, ICM-LoRA achieved the same PPL and BPC as the original LoRA and even achieve lower PPL and BPC than the original LoRA. This indicates that even in language tasks, ICM-LoRA can reconstruct the LoRA parameters and even reconstruct LoRA with more reasonable parameter distribution in some specific subtasks.

Since Llama-3 [Dubey *et al.*, 2024] and Florence-2 [Xiao *et al.*, 2024] have different parameter distributions and model constructions, the LoRA construction has a different parameter distribution. We argue that ICM-LoRA can adapt both multi-modal and language tasks, and can adapt different models with diverse parameter distributions. Therefore, we conclude that ICM-LoRA is highly effective in enhancing the performance of language modeling for parameter generation.

4.3 Ablation Studies

In this section, we will first discuss the effect of different ranks and parametric quantities of LoRA on the generation of task LoRA. Then we discuss the effect of different number of convolutional layers n on model performance during sampling. Finally, we discuss the impact on different task vector
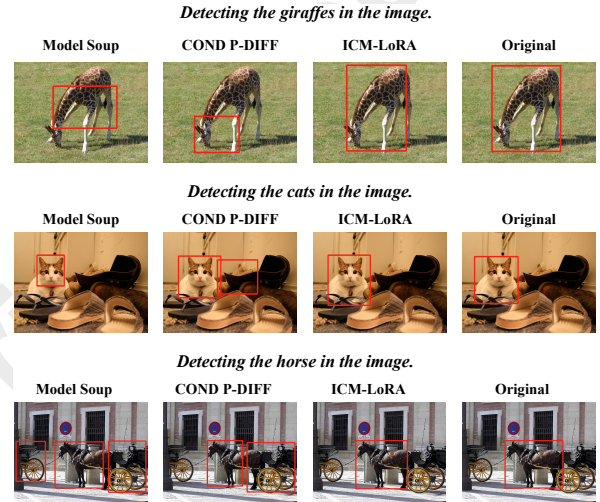

Figure 4: **Visual Comparison of Different Methods for Generating LoRA.** The LoRA generated by ICM-LoRA is most similar to the original LoRA effect.

generated by CLIP's vision encoder and text encoder.
Impact of LoRA Rank r and Parameter Number P. We trained the LoRA parameters with rank r of 1,2,4,8 respectively and trained CVAE using these LoRA parameters. The task LoRA parameters were generated based on the task vectors and evaluated on the COCO dataset. Meanwhile, we tested several other methods and discussed the effect of LoRA parameter size on the generated LoRA parameters. We selected dogs and cats as examples and reported their **MAP50**.

As shown in Table 4 and Figure 5, as the rank r of LoRA gradually increases, the number of LoRA parameters also increases. The other methods gradually decrease the effect on LoRA reconstruction as the number of LoRA parameters increases, which indicates that these methods cannot adapt to

Rank	P	Model Soup		COND P-DIFF		ICM-LoRA	
		Dog	Cat	Dog	Cat	Dog	Cat
r = 1	241241	0.93	0.93	0.94	0.93	0.95	0.97
r = 2	482482	0.93	0.93	0.94	0.93	0.96	0.95
r = 4	964964	0.93	0.92	0.93	0.93	0.95	0.96
r = 8	1929928	0.91	0.91	0.90	0.90	0.95	0.96

Table 4: **Impact of LoRA Rank and Parameter Number.** Our method is more robust in higher rank and with more parameters.

rank=1

Model Soup   COND P-DIFF   ICM-LoRA   Original

rank=2

Model Soup   COND P-DIFF   ICM-LoRA   Original

rank=4

Model Soup   COND P-DIFF   ICM-LoRA   Original

rank=8

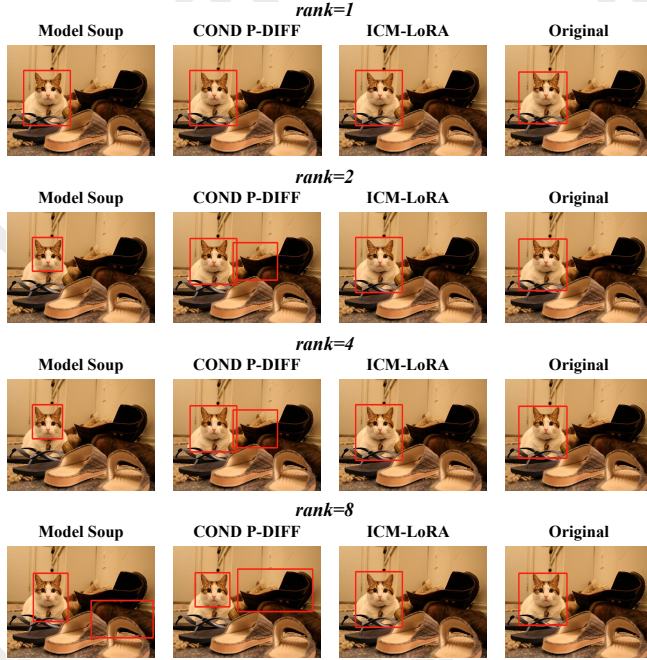Model Soup   COND P-DIFF   ICM-LoRA   Original

Figure 5: **Visualization of LoRA Rank Impacts.** For the task *"Detecting cats in the image".*, ICM-LoRA is less affected by LoRA rank compared to other methods.

the reconstruction of LoRA with a large number of parameters. The detection effect of our method is almost the same as the original LoRA with the increase of LoRA parameters, which indicates that our method is more robust and can adapt to the reconstruction of LoRA weights with different numbers of parameters.

Impact of Convolutional Layers Number. We evaluate the effect of sampling convolutional layers with different LoRA ranks and number of layers on the model's generation of LoRA weights for the task *"Detecting Cats in Pictures"*.

As shown in Table 5, as the LoRA rank and the number of parameters rise, the effect of sampling progressively decreases the fewer the convolutional layers. This indicates that the deeper the network is, the better the model samples the parameters. However, when the network convolution layer is too deep, it causes the model to fail to learn the parameter distribution characteristics of the fewer parameter LoRA. Therefore, we choose to use 12 convolutional layers to sample the LoRA parameters in our experiments.

Impact on Text and Vision Task Vector. We generate text task vectors using *"Detect the cat in the picture."* and *"Detect the dog in the picture."* by text encoder. Then generate vision

	$r=1$	$r=2$	$r=4$	$r=8$
$n=10$	0.93	0.93	0.91	0.89
$n=11$	0.93	0.93	0.92	0.91
$n=12$	0.93	0.93	0.93	0.93
$n=13$	0.93	0.93	0.93	0.93
$n=14$	0.93	0.93	0.93	0.93
$n=20$	0.90	0.91	0.92	0.93

Table 5: **Impact of Sampling Convolutional Layers.** LoRAs with larger ranks require more convolutional layers to be sampled, but too many convolutional layers can lead to poor model sampling.

Vision	Text	Dog	Cat
✓	×	0.96	0.95
×	✓	0.96	0.95
✓	✓	0.96	0.95

Table 6: **Impact of Text and Vision Task Vector.** The visual task vectors and the text task vectors reconstructed by LoRA have essentially the same effect in both tasks.

task vectors using cat and dog images. Finally we evaluate the reconstructed LoRA parameters on a subset of cats and dog in COCO and report **MAP50**.

As it shown in Table 6, task vectors instructed by vision and task input has equal impact in generate the task vector, So we consider that task vectors of different modalities have equivalent effects on parameter generation. And the simultaneous use of multimodal task vectors is not possible for the process of enhancing parameter generation.

5 Conclusion

In this paper, we propose ICM-LoRA, a novel framework that uses a self-designed parameters generator, Conditional Variational Autoencoder (CVAE), which could generate LoRA parameters to implement model customization. ICM-LoRA achieves task vectors and LoRA parameter context modeling by combining in-context learning and meta-learning, which allowing CVAE to learn LoRA parameter distributions more accurately.

Our method achieves accurate task instruct LoRA parameter generation with only CVAE, eliminating the need for additional training data and storage. The experimental results on both language modeling and object detection tasks have further validated the effectiveness of our approach could apply to different models with different tasks. ICM-LoRA can also reduce storage costs and improve computational efficiency. Overall, ICM-LoRA represents a significant advancement in parameter generation and large-scale model customization.

Acknowledgements

References

[Blundell *et al.*, 2015] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

[Bottou and others, 1991] Léon Bottou et al. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[Chi and Wylie, 2014] Michelene TH Chi and Ruth Wylie. The icap framework: Linking cognitive engagement to active learning outcomes. *Educational psychologist*, 49(4):219–243, 2014.

[Dubey *et al.*, 2024] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[Erkoç *et al.*, 2023] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14300–14310, 2023.

[Gal and Ghahramani, 2016] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[Gao *et al.*, 2020] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[Graves, 2011] Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

[Ha *et al.*, 2016] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[He *et al.*, 2024] Yang He, Lingao Xiao, Joey Tianyi Zhou, and Ivor Tsang. Multisize dataset condensation. *arXiv preprint arXiv:2403.06075*, 2024.

[Hendel *et al.*, 2023] Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, 2023.

[Hu *et al.*, 2021] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[Huang *et al.*, 2024] Lianghua Huang, Wei Wang, Zhi-Fan Wu, Yupeng Shi, Huanzhang Dou, Chen Liang, Yutong Feng, Yu Liu, and Jingren Zhou. In-context lora for diffusion transformers. *arXiv preprint arXiv:2410.23775*, 2024.

[Jin *et al.*, 2024] Xiaolong Jin, Kai Wang, Dongwen Tang, Wangbo Zhao, Yukun Zhou, Junshu Tang, and Yang You. Conditional lora parameter generation. *arXiv preprint arXiv:2408.01415*, 2024.

[Kingma *et al.*, 2015] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.

[Kingma, 2013] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[Lee *et al.*, 2022] Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. Dataset condensation with contrastive signals. In *International Conference on Machine Learning*, pages 12352–12364. PMLR, 2022.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[Liu *et al.*, 2024] Zhanyu Liu, Ke Hao, Guanjie Zheng, and Yanwei Yu. Dataset condensation for time series classification via dual domain matching. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1980–1991, 2024.

[Min *et al.*, 2021] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*, 2021.

[Mizrahi *et al.*, 2017] Tal Mizrahi, Yoram Revah, Yehonathan Refael Kalim, Elad Kapuza, and Yuval Cassuto. Fm-delta: Fault management packet compression. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 596–599. IEEE, 2017.

[Murata *et al.*, 1994] Noboru Murata, Shuji Yoshizawa, and Shun-ichi Amari. Network information criterion-determining the number of hidden units for an artificial neural network model. *IEEE transactions on neural networks*, 5(6):865–872, 1994.

[Neal, 2012] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[Peebles *et al.*, 2022] William Peebles, Ilija Radosavovic, Tim Brooks, Alexei A Efros, and Jitendra Malik. Learning to learn with generative models of neural network checkpoints. *arXiv preprint arXiv:2209.12892*, 2022.

[Platanios *et al.*, 2018] Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. Contextual parameter generation for universal neural

machine translation. *arXiv preprint arXiv:1808.08493*, 2018.

[Press *et al.*, 2022] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

[Radford *et al.*, 2021] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[Rezende *et al.*, 2014] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[Schmidt *et al.*, 1992] Wouter F Schmidt, Martin A Kraaijveld, Robert PW Duin, et al. Feed forward neural networks with random weights. In *International conference on pattern recognition*, pages 1–1. IEEE Computer Society Press, 1992.

[Shao *et al.*, 2024] Yihua Shao, Siyu Liang, Xiaolin Lin, Zijian Ling, Zixian Zhu, Minxi Yan, Haiyang Liu, Siyu Chen, Ziyang Yan, Yilan Meng, et al. Gwq: Gradient-aware weight quantization for large language models. *arXiv preprint arXiv:2411.00850*, 2024.

[Sompolinsky *et al.*, 1988] Haim Sompolinsky, Andrea Crisanti, and Hans-Jurgen Sommers. Chaos in random neural networks. *Physical review letters*, 61(3):259, 1988.

[Thoppilan *et al.*, 2022] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[Van der Maaten and Hinton, 2008] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[Wang *et al.*, 2022a] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12196–12205, 2022.

[Wang *et al.*, 2022b] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[Wang *et al.*, 2024] Kai Wang, Zhaopan Xu, Yukun Zhou, Zelin Zang, Trevor Darrell, Zhuang Liu, and Yang You. Neural network diffusion. *arXiv preprint arXiv:2402.13144*, 2024.

[Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[Wei *et al.*, 2024] Wei Wei, Tom De Schepper, and Kevin Mets. Dataset condensation with latent quantile matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7703–7712, 2024.

[Wong, 1991] Eugene Wong. Stochastic neural networks. *Algorithmica*, 6(1):466–478, 1991.

[Wortsman *et al.*, 2022] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[Xiao *et al.*, 2024] Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4818–4829, 2024.

[Xu *et al.*, 2023] Canwen Xu, Yichong Xu, Shuohang Wang, Yang Liu, Chenguang Zhu, and Julian McAuley. Small models are valuable plug-ins for large language models. *arXiv preprint arXiv:2305.08848*, 2023.

[Yan *et al.*, 2024] Ziyang Yan, Lei Li, Yihua Shao, Siyu Chen, Wuzong Kai, Jenq-Neng Hwang, Hao Zhao, and Fabio Remondino. 3dsceneeditor: Controllable 3d scene editing with gaussian splatting. *arXiv preprint arXiv:2412.01583*, 2024.

[Zhao and Bilen, 2021] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pages 12674–12685. PMLR, 2021.

[Zhao and Bilen, 2023] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6514–6523, 2023.

[Zhao *et al.*, 2020] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020.

[Zhao *et al.*, 2021] QiHao Zhao, Wei Hu, Yangyu Huang, and Fan Zhang. P-diff+: Improving learning classifier with noisy labels by noisy negative learning loss. *Neural Networks*, 144:1–10, 2021.