LION: Implicit Vision Prompt Tuning

Anonymous submission

Abstract

Despite recent promising performances across a range of vision tasks, vision Transformers still have an issue of high computational costs. Recently, vision prompt learning has provided an economical solution to this problem without fine-tuning the whole large-scale model. However, the efficiency and effectiveness of existing models are still far from satisfactory due to the parameter cost of extensive prompt blocks and tricky prompt framework designs. In this paper, we propose a light-weight prompt framework named impLicit vIsion prOmpt tuNing (**LION**), which is motivated by deep implicit models with stable low memory costs for various complex tasks. In particular, we merely insect two equilibrium implicit layers in two ends of the pre-trained backbone with parameters frozen. Moreover, according to the lottery hypothesis, we further prune the parameters to relieve the computation burden in implicit layers. Various experiments have validated that our LION obtains promising performances on a wide range of datasets. Most importantly, LION reduces up to 11.5 % of training parameter numbers while obtaining higher performance than the state-of-the-art VPT, especially under challenging scenes. Furthermore, we find that our proposed LION has an excellent generalization performance, making it an easy way to boost transfer learning in the future.

Introduction

With the development of computer vision, models with more robust representations and larger sizes have been developed. Despite this, training these models with many parameters is becoming increasingly challenging.

One common approach to addressing this issue is pre-training on a large dataset, such as ImageNet (Deng et al. 2009), for general vision tasks and then fine-tuning the model on downstream tasks to improve performance. While this method has been widely used, several drawbacks should be considered. Firstly, fine-tuning requires a large amount of computational resources, especially for large models such as ViT-B (Dosovitskiy et al. 2020) (85.84M parameters) and Swin-B (Liu et al. 2021b) (86.87M parameters). Secondly, the model may become overfitted to the small target dataset and cannot be used for other tasks after fine-tuning. This phenomenon means separate sets of model parameters are needed for each task, leading to a high storage requirement.

In recent years, prompt-based learning has generated considerable interest in the natural language processing (NLP) community because of its fantastic performance on various downstream problems (Liu et al. 2021a; Li and Liang 2021; Lester, Al-Rfou, and Constant 2021). Prompt tuning aims to design a trainable lightweight block as a supplementary input, which can guide or direct the generation of powerful vision representations to achieve desirable performances, rather than fine-tuning pre-trained models to adapt to downstream tasks.

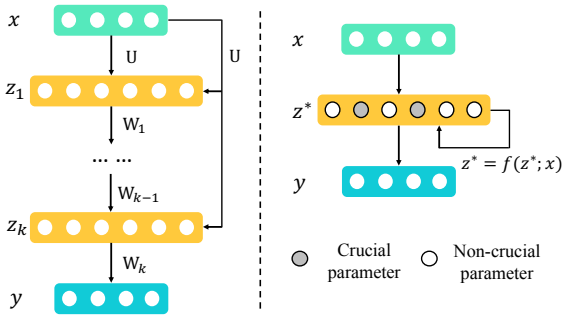

Figure 1: Demonstration of the implicit vision prompt layer. The left part shows the traditional construction of the prompt block by stacking MLPs. The right part is our LION with the implicit equilibrium layer and robust training for the prompt block.

To design a prompt framework that combines both a lightweight architecture and strong representation capability, we conducted a comprehensive study and analysis of the limitations of current vision prompt tuning methods. **First**, existing approaches insert trainable networks as prompt blocks between each layer of the network (Jia et al. 2022), assuming that the feature representations from different levels contribute to the network's generalization performance, especially for low- and mid-level representations. This, however, goes against the lightweight design philosophy of prompt tuning. The architecture design is also complex and heavily reliant on tuning skills, making applying to various vision backbone models with different architectures challenging. **Second**, finding the right prompt is a challenging task that often takes a significant amount of time. Even small changes in the activation input can significantly impact performance. This can be attributed to the depth of big model architectures, making the trainable parameters of shallow network layers more challenging to train and converge.

Based on the challenges above, we naturally raise a question, ***can we design a single-layer network as the prompt block with favorable convergence to iterate continuously?*** We hope it can achieve the effect of multi-layer network training and thus significantly reduce the training parameters. Therefore, we propose the impLicit vIsion prOmpt tuNing (**LION**), which is motivated by deep implicit models with stable low memory costs for various complex tasks. In particular, we merely insect two equilibrium implicit layers in two ends of the pre-trained backbone with parameters frozen. LION enables tuning various vision models, including convolutional neural networks (CNNs) and vision transformers.

Specifically, LION constructs a lightweight prompt framework to generate task-specific discriminative prompts for each downstream input image. LION can generate a compact and robust downstream model that adapts tuning demands across a wide range of vision tasks while only training lightweight additional parameters per downstream task, which is implemented by blending the input and the representations with the learned prompts. Besides, since the hyper-parameters, like the learning rate, can severely affect the robustness of training the vision prompts, we prune the parameters in these two layers according to the lottery hypothesis. Only the critical parameters are kept in training to avoid over-fitting. More surprising is that LION essentially compresses the parameters of the existing vision prompt network, which allows it to be generalized to any subsequent vision prompt tuning method with trainable parameters.

Our proposed LION can be used to tune CNN-based and Transformer-based vision models, surpassing fine-tuning for various recognition tasks of image classification. It can perform well under a variety of practical scenarios, including generic objects, class imbalance, and few-shot learning. Learning vision-specific information while maintaining the pretrained knowledge, our LION delivers an average improvement of up to 1.3% compared with VPT, but with much fewer trainable parameters. In summary, the main contributions of our work are three-fold:

- We propose LION, a significantly lightweight yet effective vision tuning method that leverages a single trainable implicit layer to adapt the pretrained model to downstream visual tasks.
- We construct a more robust optimization strategy by lottery hypothesis while proving the excellent convergence of our method through theoretical analysis and validation.
- Experimental results show that LION outperforms previous vision prompt tuning methods, evidencing its feasibility and usefulness for vision models, especially on few-shot and long-tail scenarios.

Related Work

Vision Fine-tuning

With the development of deep neural networks, more and more works fine-tune the pre-trained models as the backbone for downstream tasks (Zaken, Goldberg, and Ravfogel 2022; Croce and Hein 2022; Wortsman et al. 2022), which are trained on large-scale image datasets for common tasks like image classification. Fine-tuning is highly flexible: it can be applied to new input domains or tasks with different output semantics.

Recent work has focused on developing fine-tuning methods that allow for the adaptation of the entire network (Houlsby et al. 2019; Pfeiffer et al. 2020a,b), rather than just a subset of layers. These methods often use regularization techniques, such as early stopping, to prevent overfitting and ensure the stability of the optimization process. Another line of related work has focused on developing fine-tuning methods that can be used in a transfer learning setting (Gusak et al. 2022; Sung, Cho, and Bansal 2022; Lin, Madotto, and Fung 2020; Houlsby et al. 2019), where the pre-trained model is adapted to a new task in a different domain. This has proven to be a challenging problem, as the pre-trained model may need to be better suited to the new task due to domain shift.

Prompt-based Learning

Prompt-based learning (Liu et al. 2021a) is a technique that utilizes task-specific descriptions to enhance the understanding of downstream tasks by pre-trained models. This approach was popularized by the GPT series (Brown et al. 2020; Radford et al. 2018, 2019) in the field of NLP and specifically, GPT-3 by treating each downstream task as a masked language modeling problem, where the model generates a textual response in the prompt. This has led to many studies focused on developing effective prompt strategies for extracting knowledge from pre-trained language models. Similarly, recent vision-language models (Lester, Al-Rfou, and Constant 2021; Gao, Fisch, and Chen 2020; Li and Liang 2021; Schick and Schütze 2020; Talmor et al. 2020) have achieved impressive performance on various vision tasks without the need for fine-tuning. However, these prompt-based tuning methods are not suitable for pre-trained vision models. Our work aims to bridge this gap by developing a parameter-efficient prompt tuning approach specifically for vision models, to adapt frozen pre-trained vision models to downstream tasks across a broad distribution.

Deep Implicit Models

In recent years, there has been a growing interest in using implicit layers in deep learning. Researchers have explored different approaches to utilizing numerical analysis methods to replace the representation mechanism in existing deep networks. Some notable examples include SparseMAP (Niculae et al. 2018), OptNet (Amos and Kolter 2017), and SATNet (Wang et al. 2019). These approaches have shown promising results in improving the efficiency and performance of deep learning models. One particular type of implicit model that has gained attention is Deep Equilibrium Models (DEQ) (Bai, Kolter, and Koltun 2019). DEQ is an implicit model with infinite depth, yet it is interesting as a single-layer network because it allows for analytical backpropagation through the equilibrium point. Regardless of the depth of the network, training and predicting with DEQ only require constant memory. Moreover, DEQ has achieved comparable performance with efficient memory cost, as illustrated in (Xie et al. 2021). Another advantage of DEQ is its interpretability. The use of

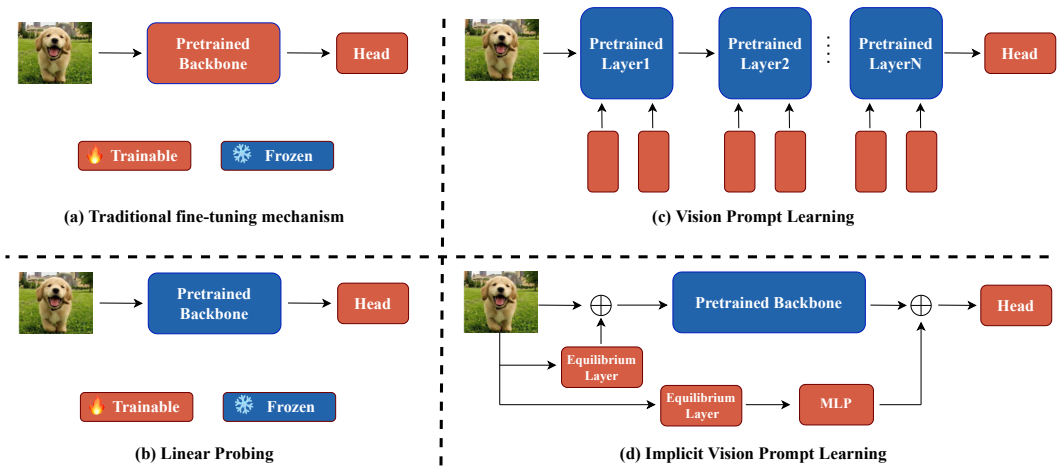

Figure 2: Structures of the fine-tuning and our LION. We add two implicit layers, which are only injected in front of the input and behind the output of the pre-trained backbone respectively, as the vision prompts to enrich the vision input and representation.

implicit models can make it difficult to interpret the behavior of the network, but DEQ provides a transparent mechanism for understanding the network's inner workings. These advantages make DEQ a suitable candidate for constructing lightweight vision prompt layers.

Methodology

Overall Architecture

In vision prompt tuning, the goal is to adapt a pre-trained vision model to downstream tasks without modifying its weights and achieve comparable results with the commonly used fine-tuning method. Mathematically, given a pre-trained large-scale vision model $\mathcal{G}$ with parameters Θ, it can be decomposed into two parts, the backbone $\mathcal{F}$ with frozen parameters Θ_f and the head layer $\mathcal{H}$ with trainable parameters Θ_h. Thus, the input image $x_i \in R^{H\times W\times 3}$ from the down-stream dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$. Our method, named LION, accomplishes the goal with an extremely lightweight prompt block $\mathcal{P}$ with a few trainable parameters Θ_p, which utilizes the implicit equilibrium layer for activation. We only insert two prompt blocks in front of the input and head layers, respectively. Suppose that the output of the backbone is $z_i = \mathcal{F}(x_i; \Theta_f)$, two prompt-based blending representations can be written as:

$$\tilde{x}_i = \alpha_1 x_i + \beta_1 \mathcal{P}_1(x_i; \Theta_{p1}), \tag{1}$$

$$\tilde{z}_i = \alpha_2 \mathcal{F}(\tilde{x}_i; \Theta_f) + \beta_2 MLP\left(\mathcal{P}_2(z_i; \Theta_{p2})\right). \tag{2}$$

Here, MLP represents the fully-connected layer for representation projection. α_i and β_i represent balance coefficients determined based on their importance in the attention mechanism (Mnih et al. 2014). To generate these balance coefficients, we initialize two adjustable parameters, g_{α_i} and g_{β_i}, subject to the following constraints: $\alpha_i + \beta_i = 1$ and $0 < \alpha_i, \beta_i < 1$.

$$\alpha_i = \frac{e^{g_{\alpha_i}}}{e^{g_{\alpha_i}} + e^{g_{\beta_i}}}, \beta_i = \frac{e^{g_{\beta_i}}}{e^{g_{\alpha_i}} + e^{g_{\beta_i}}} \tag{3}$$

In this way, the task-specific knowledge of the downstream data is distilled and effectively incorporated into the trainable parameter set: $\Theta_t = Concat(\Theta_{p1}||\Theta_{p2}||\Theta_h)$, activating the pre-trained model. The prompt-based blending representation promotes a balance between the original representations and the learned prompt, resulting in adaptive control. Our prompt blocks perform a generic architectural modification that enables the pre-trained vision model to be adapted to a downstream task with only a few additional parameters Θ_t.

Intuitively, LION is illustrated in Figure 2. The whole process can be listed as follows: **i)** Blend the downstream input image with the vision prompt using an adaptive coefficient by feeding the downstream input image into the equilibrium layer. **ii)** Feed the combined image to the frozen pre-trained model to get the feature representations. **iii)** Input the downstream image into the equilibrium layer to produce vision prompts, and map the prompts to the representation size to create a combination. **iv)** Add a fully-connected layer to the top layers of the pre-trained model for the final prediction.

Implicit Prompt Design

The Deep Equilibrium (DEQ) Model, as described in (Bai, Kolter, and Koltun 2019), employs a single layer that finds the fixed point of an iterative procedure. This layer, capable of expressing the entire deep network as an equilibrium computation, is just as powerful as multiple stacked explicit layers. Our proposed method, LION, leverages this capability by using a single DEQ layer to adapt a pre-trained vision model to downstream tasks through implicit layer training. The downstream inputs are equipped with a single implicit layer, implemented as a ResNet layer (He et al. 2016), which is trained to learn task-specific prompts while the pre-trained model remains frozen. Drawing inspiration from previous studies (Islam et al. 2021; Lin et al. 2017; Yosinski et al. 2014), we aim to improve the feature representation by utilizing high-frequency and low-frequency information as the vision prompt. To attain this goal while reducing the pa-

rameter storage burden, we propose using two-level prompt blocks, located prior to the input layer and the head layer. Our design of this lightweight architecture is supported by the demonstration of its convergence ability in Section .

Unlike a conventional network where the output is the activation from the L th layer, the output of an equilibrium layer is the equilibrium point itself. Therefore, the forward evaluation could be any procedure that solves for this equilibrium point. Conventional deep neural networks, if they converge to equilibrium, can be considered a form of fixed-point iterations for the forward process:

$$z^* = \mathcal{P}(z^*, x; \Theta_p) \tag{4}$$

Our goal will be to compute the vector-Jacobian product $\frac{\partial z^*(\cdot)}{\partial(\cdot)}^T y$ for some vector y, where $(\cdot)$ here is a stand-in for any quantity we want to differentiate the fixed point with respect to (i.e., the input x, or any parameters of the function $\mathcal{P}$, both of which of course will affect the final fixed point z^*). Since this vector-Jacobian product is the key aspect to integrating these DEQ layers within backpropagation, such a routine allows us to integrate the DEQ layer within standard automatic differentiation tools.

The derivation of the vector-Jacobian product largely mirrors that in previous chapters, but we include the full derivation again here for completeness. Differentiating both sides of the fixed point solution, we have:

$$\frac{\partial z^*(\cdot)}{\partial(\cdot)} = \left(I - \frac{\partial \mathcal{P}(z^*, x)}{\partial(z^*)}\right)^{-1} \frac{\partial \mathcal{P}(z^*, x)}{\partial(\cdot)} \tag{5}$$

In order to calculate the vector-Jacobian product, we need the following information:

$$\left(\frac{\partial z^*(\cdot)}{\partial(\cdot)}\right)^T y = \left(\frac{\partial \mathcal{P}(z^*, x)}{\partial(\cdot)}\right)^T \left(I - \frac{\partial \mathcal{P}(z^*, x)}{\partial(z^*)}\right)^{-T} y \tag{6}$$

The critical term of interest here is the solution in a linear system, and we can utilize the proxy $o = (I - \frac{\partial \mathcal{P}(z^*, x)}{\partial(z^*)})^{-T} y$ to solve the fixed point equation and compute the final Jacobian vector product.

Robust Training

We utilize the robust training mechanism for trainable parameters Θ_t motivated by (Frankle and Carbin 2019) to overcome the instability during prompt tuning. The Lottery Ticket Hypothesis in it posits that only a subset of a model's parameters is crucial for achieving good generalization, the rest are likely to overfit. To address this, we employ a criterion to separate crucial and non-crucial parameters and optimize them differently. Crucial parameters are updated with stochastic gradient descent, while non-crucial parameters are constrained to reduce their ability to overfit.

Recent pruning methods (Frankle and Carbin 2019; Yeom et al. 2021; Xia et al. 2021) suggest that crucial parameters should have substantial magnitudes, as they play a key role in network propagation. Furthermore, early optimization shows that parameters with large gradients tend to contribute to generalized patterns (Molchanov et al. 2019,?), which are crucial for learning from clean samples. Therefore, the parameters' values and gradients should be considered when determining their importance. To capture this, we use the product of their values and gradients as the criterion for determining criticality. In mathematical terms, the criticality of parameter θ_t from $\Theta_t = \{\theta_t^i\}_{i=1}^M$ is represented as:

$$z(\theta_t) = \left| \frac{\partial \mathcal{L}}{\partial \theta_t} \cdot \theta_t \right|, \tag{7}$$

where $\mathcal{L}$ represents the loss function.

The equation shows that when the gradient or value of a parameter is close to zero, its criticality is low, making it a non-crucial parameter that is prone to overfit. On the other hand, when the value of $z(\theta_t)$ is immense, θ_t is considered a crucial parameter for learning basic and generalized patterns. To control the number of crucial parameters, we introduce a threshold τ, such that crucial parameters are selected and represented as:

$$\Theta_t^c = \{\theta_t | z(\theta_t) \geq \tau\}, \tag{8}$$

$$\Theta_t^n = \{\theta_t | z(\theta_t) < \tau\}. \tag{9}$$

Unlike pruning methods, we do not eliminate the non-crucial parameters Θ_t^n. Instead, we adopt a distinct optimization strategy. Here, the crucial parameters are updated in the usual way, while the non-crucial parameters are restricted to converge to zero for better generalization. Mathematically, the update rule for $\theta_t \in \Theta_t^c$ is represented as:

$$\theta_t \leftarrow \theta_t - \eta \frac{\partial \mathcal{L}}{\partial \theta_t}, \tag{10}$$

The symbol η denotes the learning rate in the above equation. For the non-crucial parameters, we shrink them utilizing strict regularization instead of minimizing the loss. In other words, the update rule for $\theta_t \in \Theta_t^n$ is:

$$\theta_t \leftarrow \theta_t - \eta sign(\theta_t). \tag{11}$$

Optimization

For our training process, we keep the pre-trained parameters Θ_f intact and only modify a limited set of parameters Θ_t. This selective update of parameters makes our LION modular and efficient - it allows us to utilize an existing pre-trained vision model without having to modify or re-train it. Instead, we add a small number of additional parameters specific to each task, which can be formulated as:

$$\theta_t^* = \underset{\theta_t}{argmin} \frac{1}{|\mathcal{D}|} \sum_{i=1}^{N} \ell\left(\mathcal{H}(\tilde{z}_i), y_i\right) \tag{12}$$

With the pre-trained model frozen, we minimize the prediction error using cross-entropy (CE) loss. The ability of our LION to adapt a pre-trained vision model to a wide range of tasks while maintaining a high level of accuracy makes it a desirable solution for deployment in cloud services. The potential benefits of reduced computational and storage overhead, as well as the ability to offer real-time adaptation to new tasks, make our method an ideal choice for cloud service providers seeking to improve their offerings.

Theoretical Analysis

Theoretical setup. Our proposed LION aims to provide prompts to the vision input $x \in R^d$ and the representation $z \in R^h$ derived by the backbone. The whole network maps x to the label $y \in R$ with the loss $l(y, \hat{y})$ like cross-entropy, etc. We consider the network to be $f_\theta(x) = v^{\mathrm{T}}\sigma(Wx)$, where $v \in R^k$, $W \in R^{k\times d}$, and σ is an element-wise activation function like ReLU. Assume that the random variables $x, y \sim P$ and the population loss can be denoted as $\mathcal{L}(\theta) = E\left[l(f_\theta(x), y)\right]$.

Note that we have only two prompt blocks in different positions and architectures. We show that we can directly add the vision prompts to the first layer of the pre-trained model, while it cannot be implemented on the last layer. We should utilize another MLP block to ensure the optimal solution. Here we assume the σ to be ReLU: $\sigma(x_i) = max(x_i, 0)$. Given that the model is pre-trained with parameters $\hat{\theta} : (\hat{v}_{pre}, \hat{W}_{pre})$ which reach the optimal solution $\mathcal{L}(\hat{v}, \hat{W}) = 0$. With the single-layer DEQ as the vision prompts network, we can derive the vision prompts with the suppose that: $x_{pro} = Ax$ and $z_{pro} = Bz$ for some invertible matrix A, B, where the corresponding label is unchanged: $y_{pro} = y$.

Proposition 1. *There exists the vision prompt $x_{pro} = Ax$ for invertible A and $y_{pro} = y$ that can minimize the population loss: $min_W\mathcal{L}(\hat{v}, W) = 0$. However, the vision prompt $z_{pro} = Bz$ may not be sufficient: there exists such B such that the population loss is non-zero for any choice of the parameter v: $min_v\mathcal{L}(v, \hat{W}) > 0$.*

Proof. Let $\hat{B}, \hat{v}$ can reach the optimal solutions so that $y = \hat{v}\sigma(\hat{W}x)$ for all x, y. Let $W = \hat{W}A^{-1}$, we have for all x_{pro}

$$\hat{v}\sigma(Wx_{pro}) = \hat{v}\sigma(\hat{W}A^{-1}Ax) = \hat{v}\sigma(\hat{W}x) = y \qquad (13)$$

Therefore, the parameters $\hat{v}, W$ achieves $\mathcal{L}(\hat{v}, W) = 0$.

Following is a counterexample showing that last-layer prompts are impossible. Since σ is the element-wise ReLU function, $\hat{W}z$ has only positive entries for all z. Let $B = -I$, which is an invertible diagonal matrix full of -1. Then for any v, we have $v\sigma(\hat{W}z_{pro}) = v\sigma(-\hat{W}x) = 0$, so the expected loss is positive. Therefore, $min_v\mathcal{L}(v, \hat{W}) > 0$. □

Complexity Analysis

We also compare our complexity with several baselines to demonstrate our superiority. For example, in the case of the ViT model, there are N^2 visual tokens as the input, each with a dimension of d. We first examine MLP-based methods, such as Adapter and Bias, which require $2d\tilde{d} \cdot L$ additional trainable parameters in L layers for the projection from dimension d to $\tilde{d}$. Next, for VPT, it requires nd additional parameters in each layer due to the insertion of n prompts, which needs nLd trainable parameters in total. Our LION only add two prompt blocks with $m\tilde{d}$ parameters ($m \ll d$). In practice, our proposed LION has demonstrated significant advantages over Adapter and VPT, even with only slightly fewer parameters during the training stage in the below. This is due to the unique way in which LION utilizes the input data, which allows it to make more efficient use of the available parameters and achieve better results. Additionally, LION has a more flexible architecture that allows it to adapt to different types of input data, making it more versatile and applicable to a wide range of tasks.

Experiment

Experimental Setting

Dataset. CIFAR10 (Krizhevsky, Hinton et al. 2009) is a dataset of 60,000 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. **CIFAR100** (Krizhevsky, Hinton et al. 2009) is a dataset of the same size as CIFAR10, but it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. **ImageNet100** (Deng et al. 2009) is a subset of the ImageNet dataset containing 100 classes of natural images. Each class has between 500 and 1000 images for training and 50 to 100 for testing. **Flower** (Nilsback and Zisserman 2008) is a dataset of images of flowers from 5 different species. It contains 4242 images, with 80-90 images per class. **Stanford Dogs** (Khosla et al. 2011) is a dataset of images of 120 breeds of dogs, with a total of 20,580 images. **Stanford Cars** (Gebru et al. 2017) is a dataset of cars, with a total of 16,185 images of 196 classes of cars. The dataset is organized by make, model, and year. **Clothing** (Tanaka et al. 2018) is a dataset containing images of various clothing types.

Baselines. We compare LION to several commonly used protocols, including: 1) Retraining trains the entire vision model from scratch; 2) Head Fine-tuning fine-tunes the last layers of a pre-trained model while freezing the remaining layers and retraining the head classifier; 3) Fine-tuning adjusts the weights of a pre-trained model and retrains the head classifier; 4) Adapter (Houlsby et al. 2019) adds a new adapter structure to the transformer and updates only its parameters; 5) Bias (Zaken, Ravfogel, and Goldberg 2021) updates only the bias terms of the parameters; 6) VPT (Jia et al. 2022) fine-tunes the model by incorporating prompts as input tokens.

Implementation Details. We use the model pre-trained on ImageNet as the initialization for the following tuning for a fair comparison. Additionally, we extend our method to include CNN-based (ResNet-50, ResNet-101 (He et al. 2016)) and Transformer-based (ViT (Dosovitskiy et al. 2020), Swin Transformer (Liu et al. 2022)) backbones. The implementation of baselines for additional backbones involves leveraging the core idea presented in the original paper, and adapting it to suit the specific capabilities of each new backbone architecture. In experiments on the datasets above, we utilize the Adam optimizer with a momentum of 0.9, batch size of 64, and learning rate of 1e-5. The whole experiments are implemented on the NVIDIA V100 GPU with PyTorch.

Performances

Image Classification. Quantitative results can be seen in Table 1 and Table 2. It can be observed that the proposed LION performs the best overall on all six tasks when using all four models as the backbones. The best performance for

Table 1: The performance of LION and existing tuning baselines on six classification tasks using CNN-based pre-trained models. Params represents the maximum number of parameters that can be trained. The unit of measurement for Params is M.

Backbone	Method	CIFAR10	CIFAR100	ImageNet100	Flower	Dogs	Cars	Clothing	Params
ResNet-50	Retraining	0.8281	0.5178	0.7088	0.8649	0.7942	0.6898	0.7649	23.529
	Head-tuning	0.7627	0.4738	0.6167	0.8531	0.7713	0.6257	0.7324	0.277
	Fine-tuning	0.8564	0.5271	0.7194	0.8942	0.8126	0.7548	0.7852	23.529
	Adapter	0.8375	0.6021	0.7185	0.8702	0.8279	0.6816	0.7633	0.673
	Bias	0.8319	0.5965	0.7003	0.8694	**0.8316**	0.7371	0.7803	0.494
	VPT	0.8547	**0.6289**	0.7257	0.8876	0.8147	0.7459	0.7870	0.812
	LION	**0.8628**	0.5484	**0.7372**	**0.8976**	0.8269	**0.7642**	**0.7892**	0.097
ResNet-101	Retraining	0.8357	0.5365	0.7275	0.8691	0.8015	0.7014	0.7892	43.713
	Head-tuning	0.7689	0.4941	0.6287	0.8654	0.7786	0.6386	0.7641	0.461
	Fine-tuning	0.8745	0.5487	0.7469	0.8989	0.8168	0.7715	0.8119	43.713
	Adapter	0.8456	0.6233	0.7382	0.8745	0.8325	0.6895	0.7893	0.684
	Bias	0.8517	0.6148	0.7249	0.8721	**0.8421**	0.7598	0.7894	0.517
	VPT	0.8723	**0.6319**	0.7470	0.8898	0.8198	0.7581	0.8092	0.838
	LION	**0.8830**	0.5898	**0.7492**	**0.9024**	0.8311	**0.7762**	**0.8165**	0.097

Table 2: The performance of LION and existing tuning baselines on six classification tasks using Transformer-based pre-trained models. Params represents the maximum number of parameters that can be trained. The unit of measurement for Params is M.

Backbone	Method	CIFAR10	CIFAR100	ImageNet100	Flower	Dogs	Cars	Clothing	Params
ViT-B	Retraining	0.8761	0.5592	0.7277	0.8735	0.8145	0.7165	0.7959	85.721
	Head-tuning	0.7914	0.5124	0.6431	0.8617	0.8019	0.6522	0.7671	0.187
	Fine-tuning	0.9035	0.6499	0.7544	0.9003	0.8298	0.7934	0.8356	85.721
	Adapter	0.8612	0.6319	0.7411	0.8777	0.8317	0.6934	0.8229	0.372
	Bias	0.8898	0.6109	0.7326	0.8709	0.8348	0.7295	0.8210	0.215
	VPT	0.9049	**0.6689**	0.7596	0.9013	**0.8367**	0.7682	0.8378	0.523
	LION	**0.9077**	0.6541	**0.7612**	**0.9054**	0.8361	**0.7991**	**0.8397**	0.124
Swin-B	Retraining	0.8896	0.5730	0.7316	0.8794	0.8357	0.7233	0.8112	86.954
	Head-tuning	0.7991	0.5265	0.6558	0.8775	0.8150	0.6614	0.7739	0.295
	Fine-tuning	0.9166	0.6631	0.7710	0.9056	0.8359	0.8016	0.8398	86.954
	Adapter	0.8795	0.6511	0.7498	0.8812	0.8341	0.6952	0.8277	0.331
	Bias	0.8971	0.6118	0.7401	0.8749	0.8442	0.7567	0.8261	0.287
	VPT	0.9132	**0.6816**	**0.7781**	0.9026	0.8393	0.7982	**0.8434**	0.686
	LION	**0.9189**	0.6705	0.7769	**0.9061**	**0.8455**	**0.8027**	0.8431	0.242

each task is highlighted in bold. For example, on CIFAR100, LION achieves an accuracy of 54.84 % when using ResNet-50, and 58.98 % when using ResNet-101, the highest among all the methods. For the transformer-based (i.e., ViT-B and Swin-B) methods, LION achieves accuracy improvement of 1.36 %, 0.68 %, 0.51 %, 3.31 %, 0.38 % and 0.19 % over other tuning methods, respectively on CIFAR10, CIFAR100, ImageNet100, Flower, Dogs, and Cars datasets. Regarding the maximum number of trainable parameters, the proposed method has the lowest value among all the methods, with only 0.097 M trainable parameters. It is also worth noting that the performance of the other methods varies depending on the task and the backbone model used. For example, while VPT performs well on CIFAR100, it is not as effective on other tasks. On the other hand, the adapter method performs relatively well on the Flower and Dogs tasks, but not as well on the other tasks. In summary, the proposed LION is the most effective tuning method across all tasks and backbone models, with the advantage of having the lowest number of trainable parameters.

Long-tail Class Distribution. We perform experiments on benchmark datasets that have a long-tail class distribution, such as CIFAR10-LongTail and CIFAR100-LongTail. The results of the imbalance ratio 50 and 100 are shown in Table 3. Our proposed LION algorithm outperforms the best baseline, VPT, under all the settings. Specifically, we observe a gain of approximately 4% in validation accuracy, while reducing the trainable parameters by 88%. This trend holds across other settings as well. Additionally, when compared with VPT on long-tailed CIFAR-10 with an imbalanced ratio of 100 under ViT-B, LION achieves superior validation accuracy using only 4.2x fewer trainable parameters. When evaluated under Swin-B, LION outperforms VPT on long-tailed CIFAR-10 with imbalanced ratios of 50 and 100, while also reducing the trainable parameters by 64.7%.

Few-shot Learning. We perform experiments on benchmark datasets, such as Pets (Parkhi et al. 2012), Food-10 (Bossard, Guillaumin, and Van Gool 2014), Cars (Gebru et al. 2017), and Flower (Nilsback and Zisserman 2008) with eight examples per class, which are widely used for evaluating few-shot learning algorithms. Our experimental results, shown in Table 3, demonstrate that LION achieves state-of-the-art results on average, while using the fewest trainable parameters. It is worth noting that although VPT outperforms our method on some datasets for the image classification task, we still achieve the best performance on all datasets in this scenario, which highlights our method's advantage. These observations confirm the capability and efficiency of our method in the low-data regime and further verify the effectiveness of the lightweight implicit vision prompt design.

Table 3: Results of extensive experiments on the long-tailed, and few-shot datasets to validate the ability against class imbalance and sample scarcity. IR represents the imbalance ratio and the unit of measurement for Params is M.

Backbone	Method	CIFAR10-LongTail		CIFAR100-LongTail		Fine-Grained Few-Shot				Params
		IR100	IR50	IR100	IR50	Pets	Food-101	Cars	Flower	
ResNet-50	Head-tuning	0.7136	0.7358	0.3569	0.3891	0.6881	0.6013	0.3846	0.7218	0.277
	Fine-tuning	0.7638	0.7962	0.4157	0.4468	0.7340	0.6531	0.4184	0.7735	23.529
	VPT	0.7721	0.7956	0.4192	0.4510	0.7385	0.6522	0.4189	0.7790	0.812
	LION	**0.7831**	**0.8114**	**0.4328**	**0.4699**	**0.7412**	**0.6584**	**0.4203**	**0.7814**	0.097
ResNet-101	Head-tuning	0.7268	0.7496	0.3751	0.4036	0.7027	0.6159	0.4008	0.7542	0.461
	Fine-tuning	0.7754	0.8083	0.4309	0.4681	0.7593	0.6694	0.4325	0.8058	43.713
	VPT	0.7881	0.8068	0.4356	0.4710	0.7603	0.6715	0.4351	0.8134	0.838
	LION	**0.8021**	**0.8294**	**0.4662**	**0.4982**	**0.7635**	**0.6783**	**0.4470**	**0.8175**	0.097
ViT-B	Head-tuning	0.6849	0.7304	0.3871	0.4315	0.7245	0.6447	0.4139	0.7691	0.187
	Fine-tuning	0.7692	0.7834	0.4778	0.5243	0.7719	0.6834	0.4498	0.8217	85.721
	VPT	0.7631	0.7806	0.4785	0.5254	0.7731	0.6894	0.4524	0.8412	0.523
	LION	**0.7829**	**0.8008**	**0.5024**	**0.5517**	**0.7822**	**0.6911**	**0.4588**	**0.8507**	0.124
Swin-B	Head-tuning	0.6947	0.7485	0.4118	0.4568	0.7324	0.6529	0.4309	0.7814	0.295
	Fine-tuning	0.7852	0.7995	0.4826	0.5491	0.7786	0.6764	0.4625	0.8386	86.954
	VPT	0.7725	0.8011	0.4910	0.5482	0.7797	**0.6979**	0.4692	0.8497	0.686
	LION	**0.8006**	**0.8231**	**0.5276**	**0.5708**	**0.7901**	0.6976	**0.4773**	**0.8526**	0.242

Table 4: Ablation study on important inner modules under two different backbones.

Backbone	Method	CIFAR10	CIFAR100	ImageNet100
ResNet-50	LION-P1	0.8501	0.5349	0.7146
	LION-P2	0.8239	0.5136	0.6989
	LION-R	0.8516	0.5356	0.7197
	LION	0.8628	0.5484	0.7372
ViT-B	LION-P1	0.8876	0.6413	0.7492
	LION-P2	0.8527	0.6304	0.7186
	LION-R	0.8896	0.6309	0.7437
	LION	0.9077	0.6541	0.7612

Ablation Study

With the ImageNet pre-trained ResNet-50 and Vit-B, we perform extensive ablation studies to analyze the developed LION systematically. We introduce three model variants as follows: (1) **LION-P1** removes the equilibrium layer before the pre-trained backbone to activate low-level features, i.e., $\mathcal{P}_1$ in the Eq. 1. (2) **LION-P2** removes the equilibrium layer behind the pre-trained backbone to activate high-level features, i.e., $\mathcal{P}_2$ in the Eq. 2. (3) **LION-R** removes the robust training mechanism with the standard optimization for all the parameters, i.e., optimization in Eq. 8 and Eq. 9. The results of these model variants are summarized in Table 4. We have the following observations. First, our LION outperforms LION-P1 and LION-P2, which indicates that implicit vision prompt blocks work for vision semantic information activation. Second, LION-P2 obtains much worse than LION, showing that the high-level activation is vital for the vision prompt tuning. Third, the robust training mechanism allows better network optimization, as shown by LION-R's lower performance of 2% compared to LION, validating the superiority of robust training for better optimization.

Sensitivity Analysis

In Figure 3, we investigate the sensitivity of two hyper-parameters: τ in the robust training mechanism, which separates the crucial and non-crucial parameters in Eq. 8 and Eq. 9. Moreover, we also investigate the number of the layer in DEQ by stacking the single layer attracted by its striking performances. We first vary τ in $\{0.2, 0.4, 0.6, 0.8\}$ with other

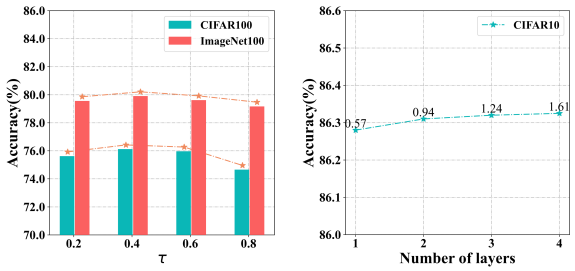

Figure 3: Sensitivity analysis on two hyper-parameters.

parameters fixed. As τ rises, the performance first increases and then decreases a little. The potential reason is that too large τ could filter the essential parameters for optimization. We can observe that the performance of ours is not sensitive to τ in the range of $[0.4, 0.6]$, and we can set it to any value in that interval. Further, we changed the number of layers from 1 to 4 with other parameters fixed. Obviously, our method can achieve a slight performance gain with the layer ranging from 1 to 4 but at the cost of several times the computation time. Therefore, τ and the layer number are set to 0.4 and 1 as default, respectively.

Conclusion

In conclusion, this paper proposes an efficient vision model named LION that addresses the heavy computational costs associated with ViT. By drawing inspiration from deep implicit models with stable memory costs, LION only requires two equilibrium implicit layers in two ends of the pre-trained main backbone with parameters in the backbone frozen. Additionally, pruning the parameters in these two layers according to the lottery hypothesis reduces the number of training parameters. LION can obtain higher performance with a smaller parameter size than the state-of-the-art baseline VPT, especially under challenging scenes. Our experiments demonstrate that LION has a good generalization performance, making it an easy way to boost applications in the future. Overall, LION provides an economical solution for vision tasks and is promising for a wide range of datasets.

References

Amos, B.; and Kolter, J. Z. 2017. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, 136–145. PMLR.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2019. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32.

Bossard, L.; Guillaumin, M.; and Van Gool, L. 2014. Food-101–mining discriminative components with random forests. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, 446–461. Springer.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Croce, F.; and Hein, M. 2022. Adversarial Robustness against Multiple and Single l_p-Threat Models via Quick Fine-Tuning of Robust Classifiers. In *International Conference on Machine Learning*, 4436–4454. PMLR.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Frankle, J.; and Carbin, M. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Gao, T.; Fisch, A.; and Chen, D. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Gebru, T.; Krause, J.; Wang, Y.; Chen, D.; Deng, J.; and Fei-Fei, L. 2017. Fine-grained car detection for visual census estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Gusak, J.; Cherniuk, D.; Shilova, A.; Katrutsa, A.; Bershatsky, D.; Zhao, X.; Eyraud-Dubois, L.; Shliazhko, O.; Dimitrov, D.; Oseledets, I.; and Beaumont, O. 2022. Survey on Efficient Training of Large Neural Networks. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 5494–5501. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2790–2799. PMLR.

Islam, A.; Chen, C.-F. R.; Panda, R.; Karlinsky, L.; Radke, R.; and Feris, R. 2021. A broad study on the transferability of visual representations with contrastive learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 8845–8855.

Jia, M.; Tang, L.; Chen, B.-C.; Cardie, C.; Belongie, S.; Hariharan, B.; and Lim, S.-N. 2022. Visual prompt tuning. *arXiv preprint arXiv:2203.12119*.

Khosla, A.; Jayadevaprakash, N.; Yao, B.; and Li, F.-F. 2011. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR workshop on fine-grained visual categorization (FGVC)*. Citeseer.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. *arXiv preprint*.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 4582–4597.

Lin, G.; Milan, A.; Shen, C.; and Reid, I. 2017. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1925–1934.

Lin, Z.; Madotto, A.; and Fung, P. 2020. Exploring Versatile Generative Language Model Via Parameter-Efficient Transfer Learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 441–459.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021b. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012–10022.

Liu, Z.; Ning, J.; Cao, Y.; Wei, Y.; Zhang, Z.; Lin, S.; and Hu, H. 2022. Video swin transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3202–3211.

Mnih, V.; Heess, N.; Graves, A.; et al. 2014. Recurrent models of visual attention. *Advances in neural information processing systems*, 27.

Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; and Kautz, J. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Niculae, V.; Martins, A.; Blondel, M.; and Cardie, C. 2018. Sparsemap: Differentiable sparse structured inference. In *International Conference on Machine Learning*, 3799–3808. PMLR.

Nilsback, M.-E.; and Zisserman, A. 2008. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 722–729. IEEE.

Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; and Jawahar, C. 2012. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, 3498–3505. IEEE.

Pfeiffer, J.; Kamath, A.; Rücklé, A.; Cho, K.; and Gurevych, I. 2020a. AdapterFusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*.

Pfeiffer, J.; Rücklé, A.; Poth, C.; Kamath, A.; Vulić, I.; Ruder, S.; Cho, K.; and Gurevych, I. 2020b. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*.

Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al. 2018. Improving language understanding by generative pre-training. *arXiv preprint*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Schick, T.; and Schütze, H. 2020. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

Sung, Y.-L.; Cho, J.; and Bansal, M. 2022. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5227–5237.

Talmor, A.; Elazar, Y.; Goldberg, Y.; and Berant, J. 2020. oLMpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8: 743–758.

Tanaka, D.; Ikami, D.; Yamasaki, T.; and Aizawa, K. 2018. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5552–5560.

Wang, P.-W.; Donti, P.; Wilder, B.; and Kolter, Z. 2019. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, 6545–6554. PMLR.

Wortsman, M.; Ilharco, G.; Gadre, S. Y.; Roelofs, R.; Gontijo-Lopes, R.; Morcos, A. S.; Namkoong, H.; Farhadi, A.; Carmon, Y.; Kornblith, S.; et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, 23965–23998. PMLR.

Xia, X.; Liu, T.; Han, B.; Gong, C.; Wang, N.; Ge, Z.; and Chang, Y. 2021. Robust early-learning: Hindering the memorization of noisy labels. In *International Conference on Learning Representations*.

Xie, X.; Wang, Q.; Ling, Z.; Li, X.; Wang, Y.; Liu, G.; and Lin, Z. 2021. Optimization induced equilibrium networks. *arXiv preprint arXiv:2105.13228*.

Yeom, S.-K.; Seegerer, P.; Lapuschkin, S.; Binder, A.; Wiedemann, S.; Müller, K.-R.; and Samek, W. 2021. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115: 107899.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27.

Zaken, E. B.; Goldberg, Y.; and Ravfogel, S. 2022. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 1–9.

Zaken, E. B.; Ravfogel, S.; and Goldberg, Y. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.