A LIGHT BILEVEL REFINEMENT ALIGNS SELF-SUPERVISED REPRESENTATIONS FOR STRONGER TASK-SPECIFIC LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-supervised pretraining learns representations that are broadly transferable across downstream tasks, yet direct fine-tuning can be suboptimal due to misalignment between self-supervised and downstream task objectives, potentially degrading pretrained features beneficial to the downstream task. The BiSSL framework addressed this by introducing a transitional training stage formulated as a bilevel optimization problem, in which the downstream task objective guides the self-supervised learning process in refining pretrained representations to better facilitate subsequent fine-tuning. However, BiSSL relies on conventional bilevel optimization solving techniques whose costly implicit hypergradient approximations render the method increasingly impractical for contemporary model architectures. To make it efficient and scalable, we introduce *BiSSLight*, which combines M-FAC-based implicit gradient approximation with parameter-efficient fine-tuning via LoRA, enabling efficient application at larger scales that were previously impractical. Evaluation across multiple downstream tasks and contemporary model architectures shows that BiSSLight consistently improves downstream performance, with gains becoming more pronounced as model size increases despite stronger baselines. The method is highly computationally efficient, reducing computation time by more than a factor of ten compared to its predecessor on a ViT-H backbone.

1 INTRODUCTION

Self-supervised learning (SSL) has become a dominant paradigm for learning transferable representations from large-scale unlabeled data (Chen et al., 2020; Grill et al., 2020; He et al., 2022; Assran et al., 2023; Uelwer et al., 2025). SSL enables models to acquire general-purpose representations that can subsequently be adapted to downstream tasks through supervised fine-tuning, substantially alleviating the need for task-specific labeled data. However, the pretext task objectives and corresponding unlabeled data distributions encountered during SSL often differ considerately from those of the downstream task. Such misalignment can lead to inefficient adaptation when fine-tuned directly from the pretrained initialization, potentially causing degradation or overwriting of pretrained features that are beneficial to the downstream task (Kumar et al., 2022; Qu & Xie, 2024; Shen et al., 2022; Liao et al., 2025; Wang et al., 2024).

The BiSSL framework (Zakarias et al., 2026) addresses this issue through the introduction of a transitional training stage that is applied to self-supervised pretrained backbones before downstream fine-tuning. This stage couples the self-supervised pretext and downstream objectives through a bilevel optimization problem (Zhang et al., 2024), allowing the downstream objective to guide the self-supervised learning toward refining the pretrained representations in a manner that better supports subsequent fine-tuning. As a result, BiSSL produces a backbone initialization that is better aligned with the downstream task, leading to consistently improved accuracy when fine-tuned across varying combinations of downstream and pretext tasks. However, these gains come at a substantial computational cost, largely due to the need to approximate implicit hypergradients during bilevel optimization, whose computational overhead grows unfavorably with scale. Consequently, BiSSL is poorly suited to the architectural scales commonly used in contemporary deep learning setups.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

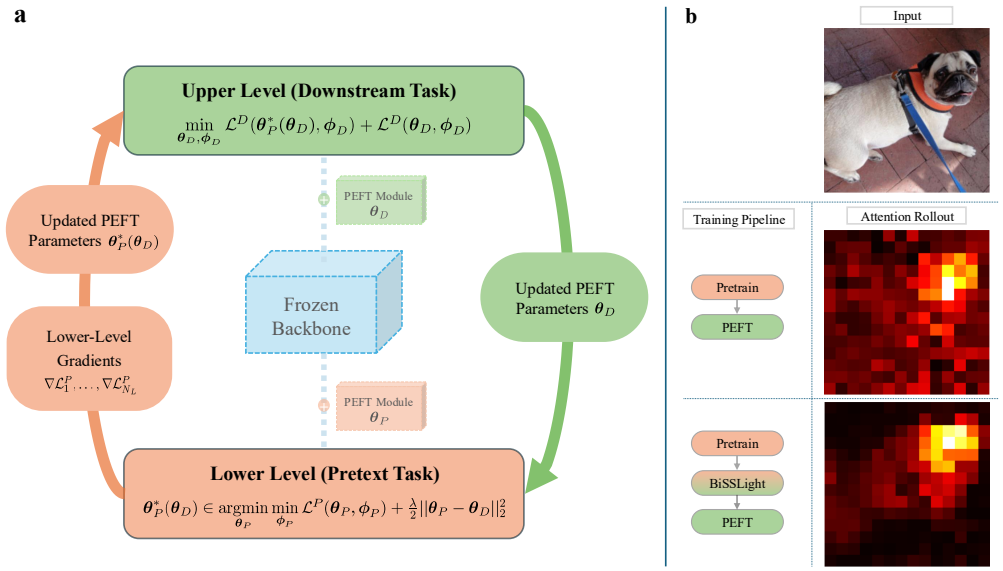

Figure 1: **a)** Overview of the BiSSLight training setup. Two distinct PEFT modules are attached to a shared frozen pretrained backbone and optimized alternately within the lower and upper-levels of the bilevel problem, respectively (see Section 3.3 for details). After training, the lower-level PEFT parameters $\boldsymbol{\theta}_P^*(\boldsymbol{\theta}_D)$ are transferred to initialize downstream fine-tuning. **b)** Attention Rollout example comparing directly fine-tuning the pretrained model with fine-tuning the BiSSLight-trained model, suggesting that attention patterns are more concentrated on regions associated with the task-relevant object when applying BiSSLight. Further details and examples are provided in Section 4.4.

Meanwhile, modern downstream adaptation practice has increasingly shifted toward parameter-efficient fine-tuning (PEFT) methods (Jia et al., 2022; Chen et al., 2022; Xin et al., 2024) such as Low-Rank Adaptation (LoRA) (Hu et al., 2022), reflecting a broader emphasis on scalability and computational efficiency when adapting large pretrained models. While BiSSL can in principle be combined with PEFT, its heavy bilevel optimization machinery largely negates the efficiency benefits that motivate PEFT in the first place, reducing its practical appeal.

In this work, we introduce *BiSSLight*, an efficient and scalable variant of BiSSL that preserves its core objective of resolving misalignment between the pretext and downstream tasks. BiSSLight achieves this by employing the Matrix-Free Approximate Curvature (M-FAC) algorithm (Frantar et al., 2021), which we use for the first time in a bilevel optimization setting to approximate the upper-level implicit hypergradients without introducing additional hyperparameters. When combined with LoRA, this enables a training procedure that is substantially faster and more memory efficient than its predecessor. Importantly, BiSSLight retains the general applicability of the original framework, introducing no restrictions to the choice of pretext task, downstream task, or model architecture. Furthermore, it remains a drop-in intermediate training stage that can be applied to off-the-shelf pretrained backbones without requiring any modifications to either the pretraining or fine-tuning stages, offering straightforward integration into existing self-supervised transfer workflows. An overview of the method is shown in Figure 1.

We benchmark BiSSLight across a range of downstream tasks and backbone sizes. BiSSLight consistently improves downstream accuracy over directly fine-tuning pretrained models, with gains becoming more pronounced as model scale increases despite increasingly strong baselines. Importantly, as illustrated in Figure 2, these improvements are achieved with modest computational overhead, requiring only five GPU-hours to apply BiSSLight to a ViT-L/16 backbone. Additional experiments demonstrate robustness across a wide range of LoRA ranks and provide qualitative indications of more focused attention on task-relevant regions. Taken together, these results establish BiSSLight as an effective approach for improving downstream transfer in contemporary settings.

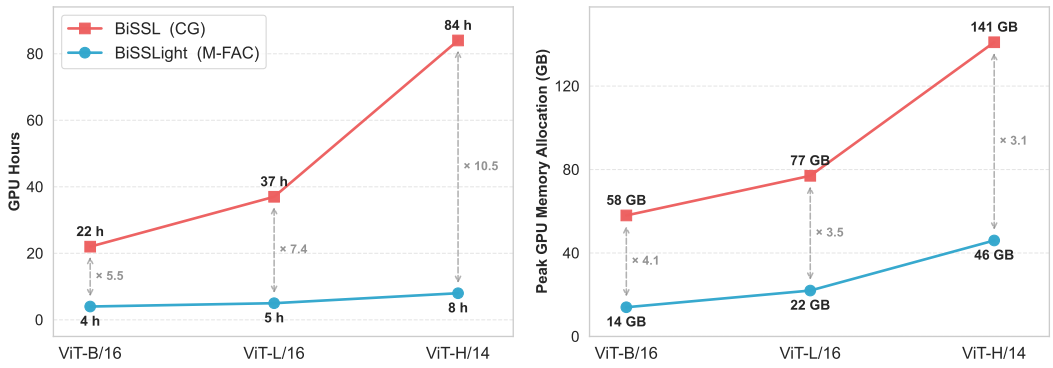

Figure 2: Computational scaling of BiSSL and BiSSLight with increasing backbone model size. Left: total computation time in GPU-hours. Right: peak GPU memory allocation. BiSSLight replaces the CG-based hypergradient approximation employed by BiSSL with M-FAC, which drastically reduces computational overhead. Notably, the relative runtime advantage becomes increasingly pronounced as model scale grows, and peak memory usage remains consistently several-fold lower.

2 Background

2.1 From Self-Supervised Pretraining To Downstream Fine-Tuning

We first consider the standard pipeline of self-supervised pretraining followed by fine-tuning on a downstream task (Uelwer et al., 2025). During self-supervised pretraining, we train a composite model of the form $h^P_{\boldsymbol{\phi}_P} \circ f_{\boldsymbol{\theta}}$ to solve a pretext task on an unlabeled dataset $\mathcal{D}^P = \{\mathbf{x}_i\}_{i=1}^{N_P}$, $\mathbf{x} \in \mathcal{X}$, where $f_{\boldsymbol{\theta}} : \mathcal{X} \mapsto \mathbb{R}^N$ is a feature-extracting backbone parameterized by $\boldsymbol{\theta}$, $h^P_{\boldsymbol{\phi}_P} : \mathbb{R}^N \mapsto \mathcal{Z}$ is a task-specific head with parameters $\boldsymbol{\phi}_P$ (e.g., the decoder in a masked autoencoder (He et al., 2022)) and $\mathcal{Z}$ denotes the output space associated with the pretext task. Given the pretext task objective $\mathcal{L}^P(\boldsymbol{\theta}, \boldsymbol{\phi}_P; \mathcal{D}^P)$, self-supervised pretraining boils down to solving the optimization problem

$$\min_{\boldsymbol{\theta}, \boldsymbol{\phi}_P} \mathcal{L}^P(\boldsymbol{\theta}, \boldsymbol{\phi}_P; \mathcal{D}^P). \tag{1}$$

After pretraining, the learned backbone parameters $\boldsymbol{\theta}^*$ are transferred to initialize training a downstream model in which the pretext task head is discarded and replaced by a downstream task-specific head. Given the labeled dataset $\mathcal{D}^D = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{N_D}$ where $\mathbf{x} \in \mathcal{X}$, $\mathbf{y} \in \mathcal{Y}$, the model takes the form $h^D_{\boldsymbol{\phi}_D} \circ f_{\boldsymbol{\theta}}$, where $h^D_{\boldsymbol{\phi}_D} : \mathbb{R}^N \mapsto \mathcal{Y}$ is the downstream task head (e.g., a linear layer) with parameters $\boldsymbol{\phi}_D$. Given the downstream objective $\mathcal{L}^D(\boldsymbol{\theta}, \boldsymbol{\phi}_D; \mathcal{D}^D)$, fine-tuning corresponds to solving

$$\min_{\boldsymbol{\theta}, \boldsymbol{\phi}_D} \mathcal{L}^D(\boldsymbol{\theta}, \boldsymbol{\phi}_D; \mathcal{D}^D) \quad \text{s.t.} \quad \boldsymbol{\theta} \in \mathcal{N}(\boldsymbol{\theta}^*), \tag{2}$$

where the notation reflects that fine-tuning is initialized from the pretrained parameter solution $\boldsymbol{\theta}^*$, with the fine-tuned backbone remaining in some neighborhood of the pretrained solution $\mathcal{N}(\boldsymbol{\theta}^*)$. For notational simplicity, we from now on omit the dataset specification from the training objective notation, e.g., $\mathcal{L}^P(\boldsymbol{\theta}, \boldsymbol{\phi}_P) := \mathcal{L}^P(\boldsymbol{\theta}, \boldsymbol{\phi}_P; \mathcal{D}^P)$.

2.1.1 The Intermediate BiSSL Training Stage

Although fine-tuning adapts the pretrained representations to better comply with the downstream task, this adaptation may be inefficient when the representations induced by the pretext task are not fully aligned with the downstream task. To address this issue, BiSSL (Zakarias et al., 2026) introduces an intermediate alignment stage that is applied after pretraining and before standard downstream fine-tuning. This stage explicitly incorporates downstream supervision to guide the self-supervised learning process, steering the backbone toward downstream-aligned representations while preserving useful pretrained features. This provides a more favorable initialization, reducing the extent to which the subsequent fine-tuning procedure must reshape the learned representations

and ultimately improving downstream performance. BiSSL formulates this transitional training stage as the following bilevel optimization problem:

$$\min_{\boldsymbol{\theta}_D, \boldsymbol{\phi}_D} \quad \mathcal{L}^D(\boldsymbol{\theta}_P^*(\boldsymbol{\theta}_D), \boldsymbol{\phi}_D) + \mathcal{L}^D(\boldsymbol{\theta}_D, \boldsymbol{\phi}_D) \tag{3}$$

$$\text{s.t.} \quad \boldsymbol{\theta}_P^*(\boldsymbol{\theta}_D) \in \operatorname*{argmin}_{\boldsymbol{\theta}_P} \min_{\boldsymbol{\phi}_P} \mathcal{L}^P(\boldsymbol{\theta}_P, \boldsymbol{\phi}_P) + \frac{\lambda}{2}\|\boldsymbol{\theta}_P - \boldsymbol{\theta}_D\|_2^2, \tag{4}$$

Here, $\boldsymbol{\theta}_P$ and $\boldsymbol{\theta}_D$ denote two distinct backbone parameterizations, both initialized from the pretrained solution $\boldsymbol{\theta}^*$. The regularization term in the lower-level objective in equation 4 enforces proximity between the two backbones, constraining the lower-level optimization to remain close to the upper-level backbone while refining the learned representations. The hyperparameter $\lambda > 0$ controls the strength of this coupling. The upper-level objective in equation 3 depends explicitly on the lower-level solution $\boldsymbol{\theta}_P^*(\boldsymbol{\theta}_D)$, which enables it to adjust its own backbone parameters $\boldsymbol{\theta}_D$ in a way that guides the lower-level optimization to benefit the downstream task. After solving the bilevel problem, the adapted backbone parameters $\boldsymbol{\theta}_P^*(\boldsymbol{\theta}_D)$ are used to initialize standard downstream fine-tuning as in equation 2, replacing the original pretrained backbone $\boldsymbol{\theta}^*$. Unlike many common bilevel optimization setups, which often require considerable effort to design, implement, and tune optimization procedures for one or both levels (Zhang et al., 2024), BiSSL is comparatively straightforward to apply, as it directly reuses established pretext and downstream optimization procedures for the respective lower and upper-levels.

The Upper-Level Derivative The practical BiSSL training algorithm repeatedly alternates between N_L lower-level and N_U upper-level optimization steps for a total of T iterations. While the lower-level gradient is comparatively straightforward to compute, the first term of the upper-level derivative with respect to the backbone parameters $\boldsymbol{\theta}_D$ is more involved. Using the derivations provided in Zakarias et al. (2026), this derivative can be expressed as

$$\frac{\mathrm{d}\,\mathcal{L}^D(\boldsymbol{\theta}_P^*(\boldsymbol{\theta}_D), \boldsymbol{\phi}_D)}{\mathrm{d}\,\boldsymbol{\theta}_D} = \lambda\Big[\nabla^2_{\boldsymbol{\theta}}\mathcal{L}^P(\boldsymbol{\theta}, \boldsymbol{\phi}_P)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_P^*(\boldsymbol{\theta}_D)} + \lambda I\Big]^{-1} \nabla_\theta \mathcal{L}^D(\boldsymbol{\theta}, \boldsymbol{\phi}_D)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_P^*(\boldsymbol{\theta}_D)}, \tag{5}$$

where I is the identity matrix. A key practical challenge in evaluating this derivative lies in approximating the inverse of a high-dimensional Hessian matrix multiplied to a vector. The original BiSSL method adopts the conjugate gradient (CG) method (Nazareth, 2009; Shewchuk, 1994) to approximate this product, which is a canonical approach in bilevel optimization settings of this structure for computational tractability (Zhang et al., 2024). However, CG still requires calculating multiple sequential Hessian-vector products for each upper-level gradient estimate, which incurs substantial memory consumption and runtime overhead. As the training scale grows, this overhead becomes increasingly restrictive, making the original method progressively less suitable at larger scales.

2.2 The M-FAC Algorithm

The Fisher information matrix provides a surrogate for the Hessian under the assumption that a model's output distribution matches its training data distribution (Ly et al., 2017; Singh & Alistarh, 2020). Under this equivalence, we can reasonably approximate the Hessian from a loss function $\mathcal{L}$ via the empirical Fisher information matrix, defined as $\hat{F} = \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \nabla_\theta \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}) \nabla_\theta \mathcal{L}(\boldsymbol{\theta}; \mathbf{x})^\top$, where $\mathcal{D}$ denotes a dataset and $\boldsymbol{\theta}$ the trainable model parameters. Since $\hat{F}$ is a sum of rank-one matrices, we can compute its inverse-vector products efficiently using the Woodbury-Sherman-Morrison formula (Woodbury, 1950). Given a collection of m gradients $\nabla\mathcal{L}_1, \ldots, \nabla\mathcal{L}_m$, the Matrix-Free Approximate Curvature (M-FAC) algorithm (Frantar et al., 2021) provides an efficient recursive equation of inverse empirical Fisher-vector products based on this formula, given by

$$\hat{F}_i^{-1}\mathbf{q} = \hat{F}_{i-1}^{-1}\mathbf{q} - \frac{\hat{F}_{i-1}^{-1}\nabla\mathcal{L}_i(\hat{F}_{i-1}^{-1}\nabla\mathcal{L}_i)^\top}{m + \nabla\mathcal{L}_i^\top \hat{F}_{i-1}^{-1}\nabla\mathcal{L}_i}\mathbf{q} = \gamma^{-1}\mathbf{q} - \sum_{j=1}^{i} \frac{\mathbf{v}_j^\top \mathbf{q}}{m + \nabla\mathcal{L}_j^\top \mathbf{v}_j}\mathbf{v}_j, \tag{6}$$

for $i = 1, \ldots, m$, where $\mathbf{v}_i = \hat{F}_{i-1}^{-1}\nabla\mathcal{L}_i$ and $\mathbf{q}$ is an arbitrary query vector with initialization $\hat{F}_0^{-1}\mathbf{q} = \gamma^{-1}\mathbf{q}$ for a small $\gamma > 0$. The vectors $\mathbf{v}_i$ are computed recursively using the expression above, and the final quantity $\hat{F}_m^{-1}\mathbf{q}$ serves as an approximation of $H^{-1}\mathbf{q}$. Notably, this recursion

effectively approximates the inverse of $\hat{F}+\gamma I$, rather than $\hat{F}$, to ensure invertibility. This distinction plays an important role in the application of M-FAC within BiSSL, as discussed in Section 3.1.

A key practical advantage of this formulation is that all vectors $\mathbf{v}_i$ and associated scalar denominators $m + \nabla\mathcal{L}_i\top\mathbf{v}_i$ depend only on the stored gradients and can therefore be computed once, independently of the query vector $\mathbf{q}$. Once these quantities are available, applying the inverse Fisher approximation to a new query vector requires only a small number of inner products and vector additions, as evident in equation 6, making each evaluation of $\hat{F}_m^{-1}\mathbf{q}$ computationally inexpensive.

Related Work The related work discussion is deferred to Appendix A.

3 BiSSLight: A Scalable Rewiring of BiSSL

We streamline BiSSL into ***BiSSLight*** by integrating M-FAC for scalable and computationally efficient inverse Hessian-vector product approximation and LoRA for reducing memory usage.

3.1 Approximating Inverse Hessian-Vector Products

The upper-level derivative in equation 5 involves an inverse Hessian-vector product (IHVP), which constitutes the main computational bottleneck. To mitigate this, we aim to replace the conjugate gradient (CG)-based approximation with the M-FAC algorithm, covered in Section 2.2. The Hessian appearing in the upper-level gradient of equation 5 is (in shorthand) $H = \nabla^2_{\boldsymbol{\theta}_P}\mathcal{L}^P + \lambda I$. Recalling that M-FAC approximates $(H + \gamma I)^{-1}$, we can exploit the structure of H by applying M-FAC only to the pretext task Hessian term $\nabla^2_{\boldsymbol{\theta}_P}\mathcal{L}^P$ while setting $\gamma = \lambda$. The regularization term λI is thereby incorporated implicitly, which conveniently yields an approximation of exactly H^{-1} in this setting. Since BiSSL is initialized from a pretrained model that performs the pretext task well, the empirical Fisher is expected to capture relevant local curvature information of the pretext task objective, making it a useful surrogate for the Hessian and thus rendering M-FAC well suited here.

Recycling Lower-Level Gradients Unlike CG, M-FAC relies on a collection of gradients rather than repeated Hessian-vector products. This aligns naturally with the alternating BiSSL optimization procedure, as we can reuse the N_L pretext-task minibatch gradients computed prior to each lower-level update for the M-FAC approximation, thereby avoiding computing any additional gradients. Although these gradients are evaluated at slightly different parameter values, the regularization term in equation 4 keeps them close to the upper-level parameters, limiting drift during the N_L updates and making their reuse of a reasonable foundation for the M-FAC approximation.

Block-Wise Approximation Since the M-FAC approximation is constructed from N_L gradients, the contribution of the pretext task objective to the inverse Hessian approximation has rank at most N_L. In high-dimensional models, such a global low-rank approximation can be overly coarse. We therefore apply M-FAC in a block-wise manner, approximating inverse Hessian-vector products separately within each parameter block. This yields an N_L-rank approximation *within each block*, resembling a block-diagonal curvature approximation commonly used in large-scale second-order optimization (Singh & Alistarh, 2020; Wang et al., 2019; Gupta et al., 2018). Although cross-block interactions are ignored, the resulting approximation is distributed more evenly across the network.

We now derive the block-wise recursion used to compute the vectors $\mathbf{v}_i$ used in M-FAC based on equation 6. Let $(\mathbf{q})_{\langle k\rangle}$, $k = 1, \ldots, K$ denote a block partition of a vector $\mathbf{q}$. Given N_L stored lower-level pretext task minibatch gradients $\nabla\mathcal{L}_1^P, \ldots, \nabla\mathcal{L}_{N_L}^P$, the recursion for block k is

$$(\mathbf{v}_i)_{\langle k\rangle} = \left(\hat{F}_{i-1}^{-1}\nabla\mathcal{L}_i^P\right)_{\langle k\rangle} \approx \lambda^{-1}\left(\nabla\mathcal{L}_i^P\right)_{\langle k\rangle} - \sum_{j=1}^{i-1}(\mathbf{v}_j)_{\langle k\rangle}\frac{(\mathbf{v}_j)_{\langle k\rangle}^\top\left(\nabla\mathcal{L}_i^P\right)_{\langle k\rangle}}{N_L + \underbrace{(\mathbf{v}_j)_{\langle k\rangle}^\top\left(\nabla\mathcal{L}_j^P\right)_{\langle k\rangle}}_{(d_j)_{\langle k\rangle}}}, \tag{7}$$

for $i = 1, \ldots, N_L$ where we define $(d_i)_{\langle k\rangle} := (\mathbf{v}_i)_{\langle k\rangle}^\top\left(\nabla\mathcal{L}_i^P\right)_{\langle k\rangle}$ and also store these values in practice. Aside from storing $K \cdot N_L$ scalar denominators instead of N_L, a negligible overhead, this

block-wise formulation incurs no additional storage overhead compared to a global approximation. The computational overhead is likewise negligible and can be parallelized across blocks due to the absence of cross-block dependencies, although we do not exploit this parallelism in our implementation. The recursion can either be executed online during lower-level optimization as new minibatch gradients become available, or after completion of the lower-level stage using the stored gradients.

3.1.1 The Updated Upper-Level Gradient Approximation

After computing $(\mathbf{v}_1)_{\langle k \rangle}, \ldots, (\mathbf{v}_{N_L})_{\langle k \rangle}$ and $(d_1)_{\langle k \rangle}, \ldots, (d_{N_L})_{\langle k \rangle}$ via equation 7, the resulting approximation to the k'th block of the first term of the upper-level derivative in equation 5 is

$$\left(\frac{\mathrm{d}\,\mathcal{L}^D(\boldsymbol{\theta}_P^*(\boldsymbol{\theta}_D), \boldsymbol{\phi}_D)}{\mathrm{d}\,\boldsymbol{\theta}_D}\right)_{\langle k \rangle} = \lambda\left(\left[\nabla^2_{\boldsymbol{\theta}}\mathcal{L}^P(\boldsymbol{\theta}, \boldsymbol{\phi}_P)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_P^*(\boldsymbol{\theta}_D)} + \lambda I\right]^{-1}\nabla\mathcal{L}^D\right)_{\langle k \rangle} \tag{8}$$

$$\approx \lambda\left(\hat{F}^{-1}_{N_L}\nabla\mathcal{L}^D\right)_{\langle k \rangle} \tag{9}$$

$$\approx \left(\nabla\mathcal{L}^D\right)_{\langle k \rangle} - \lambda\sum_{j=1}^{N_L}(\mathbf{v}_j)_{\langle k \rangle}\frac{(\mathbf{v}_j)^\top_{\langle k \rangle}\left(\nabla\mathcal{L}^D\right)_{\langle k \rangle}}{N_L + (d_j)_{\langle k \rangle}}, \tag{10}$$

for each block $k = 1, \ldots, K$ where $\nabla\mathcal{L}^D := \nabla_{\boldsymbol{\theta}}\mathcal{L}^D(\boldsymbol{\theta}, \boldsymbol{\phi}_D)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_P^*(\boldsymbol{\theta})}$. Beyond selecting the block partition, for which a layer-wise partitioning is often a sensible default, the approximation introduces no new hyperparameters requiring tuning, making it straightforward to adopt in place of the former CG-based approach. Since the computations in both equation 7 and equation 10 involve only a small number of elementary vector operations, the associated M-FAC quantities and the inverse Hessian-vector product approximation can typically be computed on the CPU with negligible performance impact, thereby providing an additional opportunity to reduce GPU memory consumption.

3.2 Parameter-Efficient BiSSL with LoRA

To further reduce the memory footprint, we freeze the pretrained backbone and apply LoRA modules with identical shapes at both the upper and lower-levels. Freezing the backbone eliminates the need to maintain separate full-scale models for each level, enabling BiSSL to operate on a single shared backbone with two lightweight sets of LoRA modules and thereby achieve a memory footprint comparable to that of standard PEFT. Furthermore, as LoRA restricts the lower-level optimization to a relatively low-dimensional parameter subspace, the M-FAC vectors in equation 7 are stored and manipulated only within this low-dimensional LoRA parameter space. This substantially reduces the memory and computational cost of the approximation, keeping the memory requirements of the approximation manageable even at large model scales.

From an optimization perspective, this modification does not alter the structure of the bilevel problem, as it is applied exactly as before, but with optimization carried out over the LoRA parameters instead. Concretely, we simply interpret $\boldsymbol{\theta}_D$ and $\boldsymbol{\theta}_P$ in equation 3 and equation 4 as the LoRA parameters rather than the full backbone weights, and afterwards transfer the fitted lower-level LoRA module $\boldsymbol{\theta}_P^*(\boldsymbol{\theta})$ as the initialization for subsequent standard PEFT.

3.3 Revised Training Algorithm

The BiSSLight training procedure follows the same alternating bilevel structure as the original method, repeatedly alternating between lower-level and upper-level optimization phases. Before alternating back to the upper-level, the stored pretext task gradients are reused to compute the quantities required by the M-FAC recursion in equation 6, which are then used for the upper-level gradient approximation as in equation 10. Figure 1 provides a conceptual overview of the training procedure, while additional algorithmic details and pseudocode are provided in Appendix B.

4 Experiments and Results

This section benchmarks the impact of BiSSLight on modern high-capacity ViT backbone architectures (Dosovitskiy et al., 2021). The main experiments rely on the Masked Autoencoder (MAE)

Table 1: Comparison of classification accuracies obtained via directly applying PEFT on ViT-B/16, ViT-L/16 and ViT-H/14 backbones pretrained with the MAE pretext task and by applying BiSSLight prior to fine-tuning. Bold font indicates a statistically significant improvement in accuracy obtained with BiSSLight, occurring in the vast majority of the evaluated settings.

	Pets	DTD	VOC07	Flowers	CUB200	Aircrafts
ViT-B/16:						
PEFT	91.7±0.3	69.9±0.5	80.9±0.3	92.8±0.5	79.8±0.5	70.6±0.6
BiSSLight+PEFT	**92.1±0.2**	70.3±0.4	**81.7±0.2**	**93.5±0.3**	**81.2±0.3**	**72.7±0.3**
ViT-L/16:						
PEFT	94.4±0.3	74.1±0.6	85.5±0.2	94.6±0.4	84.4±0.4	79.1±0.4
BiSSLight+PEFT	**94.7±0.2**	**74.9±0.4**	**85.9±0.1**	**95.8±0.2**	**85.6±0.3**	**82.2±0.3**
ViT-H/14:						
PEFT	95.2±0.3	75.8±0.5	86.5±0.2	96.0±0.3	85.9±0.4	80.4±1.0
BiSSLight+PEFT	95.5±0.2	**76.7±0.2**	**87.3±0.1**	**97.2±0.1**	**87.2±0.2**	**87.1±0.3**

pretext task (He et al., 2022) and use the corresponding official ViT-B/16, ViT-L/16 and ViT-H/14 checkpoints, pretrained on ImageNet-1K (Deng et al., 2009) at a resolution of 224×224 pixels. MAE pretraining has demonstrated particularly strong performance when fine-tuned across a range of downstream tasks (Bolya et al., 2023; Chen et al., 2022; Zhou et al., 2023; Xu et al., 2022b; Liu et al., 2023; Khanna et al., 2025), making MAE-pretrained models a challenging and practically relevant setting for evaluating BiSSLight. Whereas the evaluation of the original BiSSL work was limited to ResNet-50 sized backbones (He et al., 2016) using input resolutions of only 96×96 pixels, BiSSLight makes the framework practical under the model and input scales commonly adopted in modern computer vision research (He et al., 2022; Oquab et al., 2024; He et al., 2023). Unless otherwise specified, we apply LoRA with rank $r = 64$ to the query and value projection matrices in every attention layer, while keeping all pretrained backbone parameters frozen. Complete implementation and evaluation details are provided in Appendix C.

4.1 Downstream Task Performance

We evaluate downstream performance on six image classification benchmarks spanning texture recognition (DTD) (Cimpoi et al., 2014), multi-label object classification (VOC07) (Everingham et al., 2010), and fine-grained visual classification, including CUB200 (Wah et al., 2011), Pets (Parkhi et al., 2012), Flowers (Nilsback & Zisserman, 2008), and Aircrafts (Maji et al., 2013).

Table 1 reports classification accuracies obtained by direct fine-tuning of MAE-pretrained models and by applying BiSSLight prior to fine-tuning. Across the considered datasets, BiSSLight consistently improves the average accuracy while reducing performance variance across repeated runs. Moreover, in most cases, the observed accuracy gains are statistically significant.

A notable trend is that the benefits of BiSSLight increase with model scale. This is particularly evident on the aircrafts dataset, where the absolute gains grow from $+2.1$ percentage points for ViT-B to $+3.1$ for ViT-L and $+6.7$ for ViT-H, despite steadily improving baseline performance as model size increases. Similar behavior is observed on Oxford Flowers, where the gains increase from $+0.8$ percentage points for ViT-B to $+1.2$ for both ViT-L and ViT-H. Overall, these results suggest that the gains from BiSSLight are not exhausted by larger backbone capacities and may even be amplified when applied to stronger pretrained representations.

Different Pretext Task The original BiSSL framework was shown to provide downstream improvements across various pretext tasks. To verify that this property is preserved at larger model scales, we additionally evaluate BiSSLight using a ViT-H/14 backbone pretrained with the I-JEPA (Assran et al., 2023) pretext task. Experimental details and results are provided in Appendix D.1, showing that BiSSLight continues to provide consistent downstream performance gains.

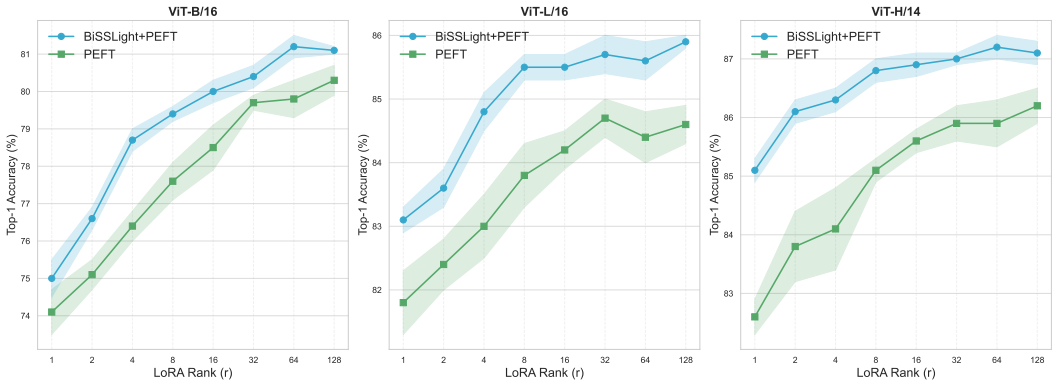

Figure 3: Test accuracies on the CUB200 dataset for varying LoRA ranks. Applying BiSSLight before PEFT consistently improves accuracy over directly applying PEFT across all evaluated ranks and backbone sizes, with gains generally becoming more pronounced at larger model scales.

4.2 Performance Across LoRA Ranks

To assess robustness across adaptation budgets, we vary the LoRA rank from $r = 1$ to $r = 128$ while keeping all other experimental settings identical to those described in Appendix C. We evaluate on the CUB200 dataset, which previously exhibited intermediate gains for BiSSLight compared to other benchmarks, making it a representative mid-range case. Figure 3 shows that BiSSLight consistently improves downstream performance across all examined ranks and backbone scales. The relative improvements obtained by BiSSLight generally become larger as backbone scale increases, consistent with the trend observed in Table 1. Moreover, BiSSLight often matches or exceeds the performance of substantially higher-rank baseline models while using far fewer trainable parameters. On ViT-H for example, BiSSLight with only $r = 2$ achieves performance comparable to the baseline with $r = 128$.

4.3 Comparing BiSSLight with BiSSL

To isolate the effect of the revised hypergradient approximation, BiSSL and BiSSLight are evaluated using identical backbone architectures, training schedules, hyperparameters, and LoRA configurations, as detailed in Appendix C. Consequently, the methods differ only in the hypergradient approximation procedure. Details of the comparison setup and the conjugate-gradient-based approximation used by BiSSL are provided in Appendix D.2.

Compute and Memory Costs Figure 2 compares the computational cost of BiSSL and BiSSLight for increasing backbone sizes. BiSSLight substantially reduces both runtime and memory requirements. The runtime advantage becomes increasingly pronounced as model scale grows, improving from approximately $5\times$ for ViT-B/16 to more than $10\times$ for ViT-H/14. Peak memory consumption is reduced by roughly $3-4\times$ across all evaluated architectures. Further analysis in Appendix D.2.2 shows that the reported memory values are conservative upper-bound estimates, and that BiSSLight operates with a memory footprint broadly comparable to standard PEFT.

Accuracy and Representation Similarity Appendix D.2.3 compares BiSSLight and BiSSL in terms of both downstream classification accuracy and representational similarity. BiSSLight achieves nearly identical downstream performance to the original BiSSL framework despite its substantially lower computational cost. The representational similarity analysis using Linear Centered Kernel Alignment (Linear CKA) (Kornblith et al., 2019) and Representational Alignment Analysis (RSA) (Kriegeskorte et al., 2008) shows that BiSSLight learns highly similar representations to BiSSL. Moreover, both methods exhibit nearly identical representational similarity patterns relative to the pretrained backbone across datasets, further supporting the conclusion that they induce comparable representational changes during training. Taken together, these findings suggest that the efficiency gains have limited impact on the resulting representations and downstream performance.

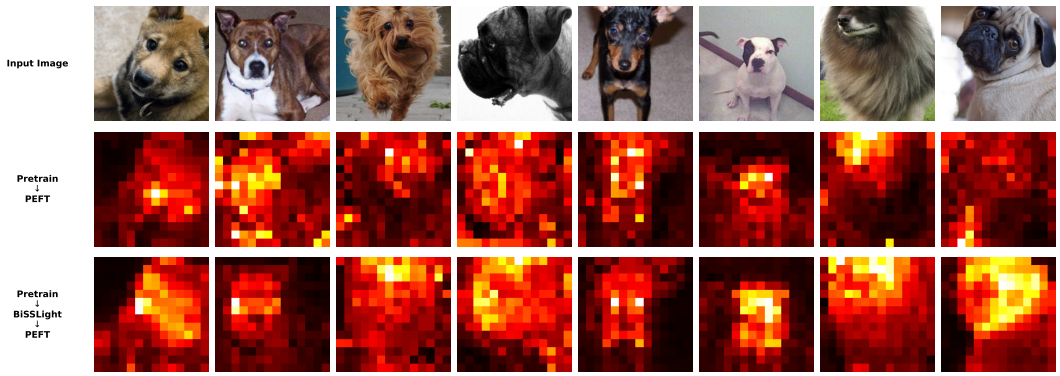

Figure 4: Attention rollout examples from the Pets validation set. The middle row corresponds to models directly fine-tuned from the MAE-pretrained checkpoint, while the bottom row shows models fine-tuned after the BiSSLight stage. BiSSLight produces attention patterns that are more concentrated on regions associated with the target object and less dispersed across the image.

4.4 Attention Rollout Visualization

To qualitatively assess how BiSSLight influences the learned representations, we visualize attention-score maps, which are commonly used to study learned attention patterns in ViTs (Caron et al., 2021; Oquab et al., 2024; Siméoni et al., 2026; Darcet et al., 2024; Khanna et al., 2025). To obtain a compact model-wide view, we apply attention rollout (Abnar & Zuidema, 2020), which aggregates attention across all transformer layers into a single relevance map, providing a holistic summary of attention propagation while remaining closely tied to the model's learned attention patterns. The visualizations are generated using a ViT-L/16 backbone, and results are shown for a randomly selected model acquired from one of the fine-tuning runs executed in conjunction with evaluating test accuracy (Described in Section C.2.1). Results for representative examples from the Pets validation set are shown in Figure 4, and Appendix D.3 provides additional examples from the Flowers and CUB200 datasets together with further discussion. These datasets were selected because they contain visually well-defined foreground objects and classes whose distinguishing characteristics are readily interpretable by non-expert observers, making qualitative differences in attention allocation more accessible to evaluate. The BiSSLight-trained models frequently produce attention maps that appear more concentrated on semantically relevant image regions. In multiple cases, attention is distributed across a larger portion of the class-relevant target object and exhibits stronger contrast between object regions and surrounding background content, suggesting improved alignment between the learned representations and task-relevant visual features.

5 Conclusion

We presented BiSSLight, a scalable variant of BiSSL that enables efficient alignment of self-supervised representations with downstream objectives prior to fine-tuning. By combining M-FAC-based implicit gradient approximation with parameter-efficient fine-tuning (PEFT), BiSSLight significantly reduces computational and memory overhead while preserving the generality of the original framework, remaining agnostic to the choice of model architecture, pretext task, downstream task, and PEFT method. We showed that BiSSLight remains effective across a range of model sizes, consistently improving downstream performance across various tasks, with gains becoming more pronounced as models grow. Beyond the observed accuracy improvements, attention rollout visualizations are consistent with this alignment perspective, suggesting that BiSSLight tends to allocate attention more strongly to task-relevant image regions. Overall, BiSSLight extends the applicability of BiSSL to contemporary adaptation settings and provides a practical mechanism for improving the transition from self-supervised pretraining to downstream fine-tuning at scale. We hope this work underscores the importance of addressing pretraining-downstream misalignment prior to fine-tuning to fully leverage self-supervised pretrained models and encourages further exploration of algorithmic approaches to address this and related challenges.

AI USE STATEMENT

In this work, we used generative AI tools to suggest and edit code, provide feedback on the readability and clarity of portions of the manuscript, and assist in improving the presentation of figures by suggesting refinements to plotting code. AI-assisted writing support was limited to revising and refining manuscript text and occasionally drafting preliminary text based on author-provided notes. All such content was reviewed and substantially revised by the authors prior to inclusion in the final manuscript. We did not use generative AI tools to formulate the core methodology, design experiments, conduct the experimental evaluation, or determine the scientific conclusions of this work. The authors have reviewed all AI-assisted work, and take full responsibility for the final content of this work, including any material produced with the assistance of generative AI tools.

REPRODUCIBILITY STATEMENT

We make a concerted effort to ensure the reproducibility of all results presented in this work. The proposed method is described in detail in Section 3 of the main paper, including the bilevel formulation, the M-FAC-based hypergradient approximation, and the incorporation of LoRA into the framework. Appendix B provides detailed pseudocode for the training algorithm, while Appendix C documents all implementation details required to reproduce the reported results. Additionally, we will publicly release the codebase together with the trained model checkpoints used in this work.[1]

REFERENCES

Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4190–4197, 2020.

Samira Abnar, Lisa Beinborn, Rochelle Choenni, and Willem Zuidema. Blackbox meets blackbox: Representational similarity & stability analysis of neural language models and brains. In Tal Linzen, Grzegorz Chrupała, Yonatan Belinkov, and Dieuwke Hupkes (eds.), *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 191–203, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4820.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023.

Reza Akbarian Bafghi, Nidhin Harilal, Claire Monteleoni, and Maziar Raissi. Parameter efficient fine-tuning of self-supervised vits without catastrophic forgetting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3679–3684. IEEE, 2024.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in neural information processing systems*, 35:16664–16678, 2022.

[1] https://github.com/42X4X01XZ/ICLR27_20859

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International conference on learning representations*, volume 2024, pp. 2632–2652, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

A.L. Dontchev and R.T. Rockafellar. *Implicit Functions and Solution Mappings: A View from Variational Analysis*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2014. ISBN 9781493910373.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010.

Elias Frantar, Eldar Kurtic, and Dan Alistarh. M-fac: Efficient matrix-free approximations of second-order information. In *Neural Information Processing Systems*, 2021.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. Parameter-efficient model adaptation for vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 817–825, 2023.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

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. In *ICLR*, 2022.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European conference on computer vision*, pp. 709–727. Springer, 2022.

Samar Khanna, Medhanie Irgau, David B. Lobell, and Stefano Ermon. ExPLoRA: Parameter-efficient extended pre-training to adapt vision transformers under domain shifts. In *Forty-second International Conference on Machine Learning*, 2025.

Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of neural network models: A survey of functional and representational measures. 57(9), May 2025. ISSN 0360-0300. doi: 10.1145/3728458.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 09–15 Jun 2019.

Nikolaus Kriegeskorte, Marieke Mur, and Peter A. Bandettini. Representational similarity analysis – connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2:4, 2008. ISSN 1662-5137. doi: 10.3389/neuro.06.004.2008.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)*, pp. 4582–4597, 2021.

Xingjian Li, Haoyi Xiong, Haozhe An, Cheng-Zhong Xu, and Dejing Dou. Rifle: Backpropagation in depth for deep transfer learning through re-initializing the fully-connected layer. In *International Conference on Machine Learning*, pp. 6010–6019. PMLR, 2020.

Chonghua Liao, Ruobing Xie, Xingwu Sun, Haowen Sun, and Zhanhui Kang. Exploring forgetting in large language model pre-training. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2112–2127, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.105.

Disen Liao, Felix Dangel, and Yaoliang Yu. Efficient bilevel optimization with KFAC-based hypergradients. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026.

Qin Liu, Zhenlin Xu, Gedas Bertasius, and Marc Niethammer. Simpleclick: Interactive image segmentation with simple vision transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22233–22243. IEEE, 2023.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International conference on artificial intelligence and statistics*, pp. 1540–1552. PMLR, 2020.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Alexander Ly, Maarten Marsman, Josine Verhagen, Raoul PPP Grasman, and Eric-Jan Wagenmakers. A tutorial on fisher information. *Journal of Mathematical Psychology*, 80:40–55, 2017.

S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

J. L. Nazareth. Conjugate gradient method. *WIREs Computational Statistics*, 1(3):348–353, 2009. doi: https://doi.org/10.1002/wics.13.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Haolin Pan, Yong Guo, Qinyi Deng, Hao-Fan Yang, Yiqun Chen, and Jian Chen. Improving fine-tuning of self-supervised models with contrastive initialization. *Neural networks : the official journal of the International Neural Network Society*, 159:198–207, 2022.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Helen Qu and Sang Michael Xie. Connect later: Improving fine-tuning for robustness with targeted augmentations. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 41769–41786. PMLR, 21–27 Jul 2024.

Aravind Rajeswaran, Chelsea Finn, Sham M. Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Neural Information Processing Systems*, 2019.

Kendrick Shen, Robbie M Jones, Ananya Kumar, Sang Michael Xie, Jeff Z. Haochen, Tengyu Ma, and Percy Liang. Connect, not collapse: Explaining contrastive learning for unsupervised domain adaptation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 19847–19878. PMLR, 17–23 Jul 2022.

Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, 1994.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seung Eun Yi, Michael Ramamonjisoa, Francisco Massa, Daniel HAZIZA, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Herve Jegou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. Featured Certification.

Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18098–18109. Curran Associates, Inc., 2020.

Tobias Uelwer, Jan Robine, Stefan Wagner, Marc Höftmann, Eric Upschulte, Sebastian Konietzny, Maike Behrendt, and Stefan Harmeling. A survey on self-supervised methods for visual representation learning. *Machine Learning*, 114, 03 2025. doi: 10.1007/s10994-024-06708-7.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *International conference on machine learning*, pp. 6566–6575. PMLR, 2019.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Max A Woodbury. Inverting modified matrices. Statistical Research Group, Memo. Rep. 42, Princeton University, Princeton, N. J, 1950.

Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. Noisytune: A little noise can help you finetune pretrained language models better. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

Yi Xin, JianJiang Yang, Siqi Luo, Yuntao Du, Qi Qin, Kangrui Cen, Yangfan He, Bin Fu, Xiaokang Yang, Guangtao Zhai, Ming-Hsuan Yang, and Xiaohong Liu. Parameter-efficient fine-tuning for pre-trained vision models: A survey and benchmark. *International Journal of Computer Vision*, 134, 2024.

Chengming Xu, Siqian Yang, Yabiao Wang, Zhanxiong Wang, Yanwei Fu, and Xiangyang Xue. Exploring efficient few-shot adaptation for vision transformers. *Transactions on Machine Learning Research*, 2022a. ISSN 2835-8856.

Mingle Xu, Sook Yoon, Yongchae Jeong, and Dong Park. Transfer learning for versatile plant disease recognition with limited data. *Frontiers in Plant Science*, 13:1010981, 11 2022b. doi: 10.3389/fpls.2022.1010981.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10665–10673, 2021.

Gustav Wagner Zakarias, Lars Kai Hansen, and Zheng-Hua Tan. BiSSL: Enhancing the alignment between self-supervised pretraining and downstream fine-tuning via bilevel optimization. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. J2C Certification.

Yihua Zhang, Prashant Khanduri, Ioannis Tsaknakis, Yuguang Yao, Mingyi Hong, and Sijia Liu. An introduction to bilevel optimization: Foundations and applications in signal processing and machine learning. *IEEE Signal Processing Magazine*, 41(1):38–59, 2024. doi: 10.1109/MSP.2024.3358284.

Yukun Zhou, Mark A. Chia, Siegfried K. Wagner, Murat S. Ayhan, Dominic J. Williamson, Robbert R. Struyven, Timing Liu, Moucheng Xu, Mateo G. Lozano, Peter Woodward-Court, Yuka Kihara, Naomi Allen, John E.J. Gallacher, Thomas Littlejohns, Tariq Aslam, Paul Bishop, Graeme Black, Panagiotis Sergouniotis, Denize Atan, Andrew D. Dick, Cathy Williams, Sarah Barman, Jenny H. Barrett, Sarah Mackie, Tasanee Braithwaite, Roxana O. Carare, Sarah Ennis, Jane Gibson, Andrew J. Lotery, Jay Self, Usha Chakravarthy, Ruth E. Hogg, Euan Paterson, Jayne Woodside, Tunde Peto, Gareth Mckay, Bernadette Mcguinness, Paul J. Foster, Konstantinos Balaskas, Anthony P. Khawaja, Nikolas Pontikos, Jugnoo S. Rahi, Gerassimos Lascaratos, Praveen J. Patel, Michelle Chan, Sharon Y.L. Chua, Alexander Day, Parul Desai, James E. Morgan, and Irene Stratton. A foundation model for generalizable disease detection from retinal images. *Nature*, 622(7981):156–163, October 2023.

Nicolas Zucchet and João Sacramento. Beyond backpropagation: bilevel optimization through implicit differentiation and equilibrium propagation. *Neural Computation*, 34(12):2309–2346, 2022.

A Related Work

Parameter-Efficient Fine-Tuning Parameter-efficient fine-tuning (PEFT) methods adapts large pretrained models by augmenting them with a relatively small number of trainable parameters while leaving the pretrained parameters fixed or by fitting only a small subset of the pretrained parameters (Xin et al., 2024). Representative approaches include adapters, which insert trainable bottleneck modules throughout the network (Houlsby et al., 2019; Chen et al., 2022), prompt tuning, which prepends learnable prompts to the input sequence (Lester et al., 2021; Jia et al., 2022), prefix tuning, which injects trainable prefix vectors into the attention mechanism (Li & Liang, 2021; Xu et al., 2022a), BitFit, which fine-tunes only the bias-terms of the pretrained model (Ben Zaken et al., 2022) and low-rank adaptation (LoRA), which introduces trainable low-rank matrix pairs whose product is added to selected pretrained weight matrices (Hu et al., 2022; Bafghi et al., 2024). These methods reduce memory and computational cost compared to full fine-tuning and can also improve robustness to overfitting while maintaining strong downstream performance. Due to its simplicity and effectiveness, LoRA is widely used in practice and serves as the PEFT instantiation in this work, although BiSSLight can in principle be employed with other PEFT techniques as well.

Enhancing Downstream Adaptation of Pretrained Backbones Beyond BiSSL (Zakarias et al., 2026), which is detailed in Section 2.1.1, other works have similarly explored improving the downstream transferability of self-supervised representations prior to fine-tuning through an intermediate adaptation stage. ExPLoRA (Khanna et al., 2025) adapts pretrained Vision Transformers through LoRA-based continued pretraining on downstream-domain data. Unlike ExPLoRA, which leverages downstream-domain data during the transitional stage and subsequently merges the adapted parameters back into the backbone before training a new set of adaptation parameters for the downstream task, BiSSLight directly incorporates the downstream objective into the optimization process and transfers the learned adaptation parameters directly to downstream PEFT. Other approaches instead conduct heuristic adaptation procedures intended to improve downstream adaptation. These include, among others, COIN (Pan et al., 2022), which is tailored to contrastive pretraining through supervised contrastive learning on downstream data, NoisyTune (Wu et al., 2022), which perturbs the pretrained weights with backbone parameter-dependent noise, and RIFLE (Li et al., 2020), which repeatedly reinitializes the classification head. In contrast to the aforementioned approaches, BiSSLight do not assume that a particular adaptation procedure is generally beneficial across downstream tasks. Instead, the bilevel optimization problem allows the downstream task itself to guide the self-supervised learning process, resulting in an adaptation procedure tailored to the task at hand.

Approximating Inverse Hessian Vector Products Among bilevel optimization methods that compute hypergradients via the implicit function theorem (Dontchev & Rockafellar, 2014) such as BiSSL, a central challenge is the efficient approximation of the resulting inverse-Hessian vector products (Zhang et al., 2024). The most established approaches employ iterative procedures such as conjugate gradient (CG) (Shewchuk, 1994; Rajeswaran et al., 2019; Zucchet & Sacramento, 2022) and truncated Neumann series (Lorraine et al., 2020), but their reliance on repeated Hessian-vector product computations still makes them increasingly expensive at scale. Simpler alternatives employ diagonal approximations to improve efficiency, at the cost of a coarser approximation of the Hessian (Yao et al., 2021). Between these extremes lies a family of methods that replace the Hessian with the empirical Fisher information matrix and exploit its structure to derive more efficient inverse-Hessian approximations (Singh & Alistarh, 2020). One such method is Kronecker-factored Approximate Curvature (K-FAC) (Wang et al., 2019), which exploits the layer-wise structure of deep neural networks to construct efficient approximations and has recently been explored for scalable hypergradient computation in bilevel optimization (Liao et al., 2026). However, K-FAC relies on architecture-specific structural assumptions and requires access to both activations and gradients, making the approximation more tightly coupled to the underlying model architecture and training setup. In contrast, Matrix-Free Approximate Curvature (M-FAC) (Frantar et al., 2021) constructs the approximation solely from gradients, making it largely architecture- and PEFT-method agnostic and thereby helping BiSSLight preserve the plug-and-play nature of the original BiSSL framework.

B BiSSLight Training Algorithm

Algorithm 1 provides the complete pseudocode description of the resulting BiSSLight training procedure. The algorithm repeatedly alternates between solving the lower-level and upper-level optimization problems. After completing the lower-level optimization phase, the stored pretext gradients are used to compute the vectors and associated inner products required by the M-FAC recursion in equation 10. These quantities are stored in a collection M and are subsequently reused when constructing the upper-level gradient approximation. An equivalent alternative is to compute the M-FAC quantities online during the lower-level optimization phase rather than after it has completed. Since both approaches perform the same operations, they have identical computational and memory complexity. We adopt the latter variant in the pseudocode, as it is also used in the provided PyTorch implementation.

To ensure a stable initialization of the bilevel optimization, we initialize the LoRA modules at both levels with identical parameter values, thereby avoiding an artificial mismatch between the upper- and lower-level parameterizations at the start of training. Throughout, we denote the pretrained (and frozen) backbone parameters by $\boldsymbol{\psi}$, and the LoRA parameters associated with the lower- and upper-level objectives by $\boldsymbol{\theta}_P$ and $\boldsymbol{\theta}_D$, respectively.

Algorithm 1 BiSSLight Training Algorithm

Input: Initialized backbone and head parameters $\boldsymbol{\psi}$, $\boldsymbol{\phi}_P$, $\boldsymbol{\phi}_D$. Lower and upper-level training objectives $\mathcal{L}^P$, $\mathcal{L}^D$ and optimizers opt_P, opt_D. Regularization Weight $\lambda \in \mathbb{R}_+$. Number of training stage alternations $T \in \mathbb{N}$ with lower and upper-level steps $N_L, N_U \in \mathbb{N}$. LoRA rank $r \in \mathbb{N}$. Block partition $1, \dots, K$.

Initialize rank-r LoRA parameters $\boldsymbol{\theta}_D$ on the backbone $\boldsymbol{\psi}$ and set $\boldsymbol{\theta}_P \leftarrow \boldsymbol{\theta}_D$.

for $t = 1, \dots, T$ **do**
  Initialize $G \leftarrow \emptyset$. ▷ Lower-level gradient memory buffer
  for $i = 1, \dots, N_L$ **do** ▷ Lower-level
    Compute $\nabla \mathcal{L}_i^P \leftarrow \nabla_{\boldsymbol{\theta}} \mathcal{L}^P(\boldsymbol{\theta}, \boldsymbol{\phi}_P)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_P}$ and append a copy to G.
    Compute $\mathbf{g}_{\boldsymbol{\theta}_P} \leftarrow \nabla \mathcal{L}_i^P + \lambda(\boldsymbol{\theta}_P - \boldsymbol{\theta}_D)$.
    Compute $\mathbf{g}_{\boldsymbol{\phi}_P} \leftarrow \nabla_{\boldsymbol{\phi}} \mathcal{L}^P(\boldsymbol{\theta}_P, \boldsymbol{\phi})|_{\boldsymbol{\phi}=\boldsymbol{\phi}_P}$.
    Update $\boldsymbol{\theta}_P \leftarrow \text{opt}_P(\boldsymbol{\theta}_P, \mathbf{g}_{\boldsymbol{\theta}_P})$ and $\boldsymbol{\phi}_P \leftarrow \text{opt}_P(\boldsymbol{\phi}_P, \mathbf{g}_{\boldsymbol{\phi}_P})$.

  Compute $M = \left[\left((\mathbf{v}_i)_{\langle k \rangle}, (d_i)_{\langle k \rangle}\right)\right]_{i=1,\dots,N_L,\ k=1,\dots,K}$ using G via equation 7.

  for $n = 1, \dots, N_U$ **do** ▷ Upper-level
    Compute $\nabla \mathcal{L}^D \leftarrow \nabla_{\boldsymbol{\theta}} \mathcal{L}^D(\boldsymbol{\theta}, \boldsymbol{\phi}_D)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_P}$.
    Compute $\mathbf{q}$ (assembled block-wise) using M and $\nabla \mathcal{L}^D$ via equation 10.
    Compute $\mathbf{g}_{\boldsymbol{\theta}_D} \leftarrow \mathbf{q} + \nabla_{\boldsymbol{\theta}} \mathcal{L}^D(\boldsymbol{\theta}, \boldsymbol{\phi}_D)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_D}$.
    Compute $\mathbf{g}_{\boldsymbol{\phi}_D} \leftarrow \nabla_{\boldsymbol{\phi}} \mathcal{L}^D(\boldsymbol{\theta}_P, \boldsymbol{\phi})|_{\boldsymbol{\phi}=\boldsymbol{\phi}_D} + \nabla_{\boldsymbol{\phi}} \mathcal{L}^D(\boldsymbol{\theta}_D, \boldsymbol{\phi})|_{\boldsymbol{\phi}=\boldsymbol{\phi}_D}$.
    Update $\boldsymbol{\theta}_D \leftarrow \text{opt}_D(\boldsymbol{\theta}_D, \mathbf{g}_{\boldsymbol{\theta}_D})$ and $\boldsymbol{\phi}_D \leftarrow \text{opt}_D(\boldsymbol{\phi}_D, \mathbf{g}_{\boldsymbol{\phi}_D})$.

Return: Fitted LoRA Parameters $\boldsymbol{\theta}_P$.

These modifications make BiSSLight practically compatible with modern large-scale training setups, with a memory footprint comparable to standard PEFT. The primary computational and memory cost now lies in the lower-level pretext optimization, as the upper-level gradient approximation contributes only minimal overhead. Because the lower-level optimization remains effective with relatively small batch sizes, the method can be deployed on hardware similar to that used for standard PEFT, as confirmed in the GPU memory requirement comparisons in Table 4 of Section D.2.2.

C Implementation Details of the Main Experimental Setup

C.1 Self-Supervised Pretraining

We exclusively use the official pretrained weights achieved via the Masked Autoencoder pretext task (He et al., 2022). This includes both the encoder backbone and decoder head models, trained on the ImageNet-1k dataset (Deng et al., 2009) devoid of labels. We consider the three official ViT-B/16, ViT-L/16, and ViT-H/14 checkpoints pretrained at a resolution of 224×224. Full details of the MAE pretraining procedure are available in the original work (He et al., 2022).

C.2 Downstream Adaptation

For downstream fine-tuning, we apply standard data augmentations consisting of a random resized crop to 224×224 with a minimum crop ratio of 0.2, followed by a random horizontal flip and normalization. We use the AdamW optimizer (Loshchilov & Hutter, 2019) with weight decay 0.05, $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and train for 200 epochs unless otherwise specified. Drop-path regularization (Huang et al., 2016; Dosovitskiy et al., 2021) with rate 0.1 is applied for ViT-B and rate 0.2 for ViT-L and ViT-H.

We apply LoRA with rank $r = 64$ to each respective query and value matrices in every attention layer, following common conventions (Hu et al., 2022; Bafghi et al., 2024; Khanna et al., 2025). A linear classification head is attached to the backbone output corresponding to the auxiliary class token, and only the LoRA parameters and this linear head are trained. We experimented with additionally tuning normalization layers, but observed no notable performance differences compared to freezing them. For simplicity and efficiency, we therefore keep these layers frozen. Similarly, we evaluated the use of dropout within LoRA modules, but found no measurable gains, and thus omit it, hypothesizing that the combination of drop-path regularization and weight decay provides sufficient regularization.

We adopt a cosine decaying learning rate scheduler with a linearly increasing warm-up (Loshchilov & Hutter, 2017; He et al., 2022) over the first 5 epochs. For models fine-tuned directly from pretrained backbones, we perform a grid search over the base learning rates $[10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}, 10^{-2}, 3 \cdot 10^{-2}]$. For BiSSLight-trained backbones, we found that smaller learning rates are more suitable, likely due to the improved downstream alignment. Accordingly, we search over $[10^{-5}, 3{\cdot}10^{-5}, 10^{-4}, 3{\cdot}10^{-4}, 10^{-3}, 3{\cdot}10^{-3}]$. Besides this difference, BiSSLight-trained backbones are fine-tuned using the same setup as that used for directly fine-tuning the pretrained backbones.

C.2.1 Evaluation

After selecting the best learning rate based on the considered optimal combination of validation accuracy and loss, we perform 8 independent training runs with different random seeds. During each run, we retain the model checkpoint corresponding to the highest validation accuracy observed throughout training, and we evaluate validation accuracy after each epoch. Final results are reported as the mean and standard deviation of test accuracy across the 8 models.

To assess whether BiSSLight leads to statistically significant improvements over standard fine-tuning, we perform two-sample permutation tests comparing the classification accuracies obtained from the 8 runs of each method, and consider p-values below 0.05 to imply a statistical significance between the average accuracies of the PEFT and BiSSLight+PEFT results. We refer to the original BiSSL work (Zakarias et al., 2026) for details, which uses the same class of test to asses statistical significance.

Downstream Datasets The main experiments evaluate six image classification benchmarks: Describable Textures Dataset (DTD) (Cimpoi et al., 2014), PASCAL VOC 2007 (VOC07) (Everingham et al., 2010), CUB200 (Wah et al., 2011), Oxford-IIIT Pet (Parkhi et al., 2012), Oxford Flowers 102 (Nilsback & Zisserman, 2008), and FGVC Aircraft (Maji et al., 2013). For datasets without official validation splits, we reuse the splits from Zakarias et al. (2026), which randomly allocate roughly 20% of the training partition for validation while maintaining the same class balance. We use top-1 classification accuracy as the primary downstream metric for all datasets except VOC07, for which we report the standard 11-point mAP.

C.3 BiSSLight

The experimental setup of BiSSLight does to a large extend follow the one of the original BiSSL work (Zakarias et al., 2026), but we still describe all necessary implementation details here for completeness. We organize the description by first a remark on the LoRA configuration, then specifying level-specific settings, finally followed by details governing the overall bilevel training procedure.

LoRA Configuration Consistent with the fine-tuning setup described in Section C.2, we use the same LoRA rank during BiSSLight training to ensure compatibility and allow direct transfer of learned LoRA parameters. Unless otherwise stated, we hence apply LoRA with rank $r = 64$ to the respective query and value matrices in every attention layer and freeze all pretrained weights. Two identical sets of LoRA modules are instantiated for the lower- and upper-level objectives, respectively. Both sets are initialized with identical parameter values, after which they are optimized independently.

Lower-Level (Pretext Task) For the lower-level pretext task, we employ the same image augmentation pipeline as in the original MAE pretraining setup, which also matches the pipeline used during the downstream training stage: a random resized crop to 224×224 with a minimum crop ratio of 0.2, followed by a random horizontal flip and normalization. We similarly also train on the ImageNet-1k (Deng et al., 2009) devoid of labels. Although self-supervised pretraining typically relies on large batch sizes, preliminary testing showed that BiSSLight does not require large lower-level batches in order to perform well. We therefore use a batch size of 256, which also helps to keep the memory footprint further manageable.

We use the normalized pixel reconstruction loss from the original MAE implementation as well as a masking ratio of 0.75. We apply a cosine decaying learning rate schedule with linear warm-up over $10 \cdot N_L$ steps. The base learning rate for the LoRA backbone modules is set to 0.01. The pretrained MAE decoder head is fully unfrozen during lower-level training. While LoRA could also be applied to the decoder in highly memory-constrained settings, we found this unnecessary for our setup, as the decoder is per design comparatively small ($\sim$ 27M parameters). Since the decoder is initialized from pretrained weights, we use a smaller learning rate of 10^{-4} for the decoder head parameters, which yields stabler training. We use the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay 0.05, following the original MAE configuration. Note that all lower-level settings are kept fixed across all downstream tasks.

Upper-Level (Downstream Task) The upper-level optimization configuration deliberately follows the downstream fine-tuning setup in Section C.2 to a large extend. We use the same image augmentation pipeline, drop-path regularization, optimizer and cosine learning rate scheduler. The learning rate warm-up is performed over $10 \cdot N_U$ steps and we use a batch size of 64. For each dataset, we took the learning rate considered optimal for the pretext-pretrained-fine-tuned ViT-L model, and then obtained candidate learning rates by scaling this value by $2/3$, $1/2$, $1/3$, and, when necessary, by further reducing the learning rate within the same order of magnitude (e.g., $3 \cdot 10^{-4}$, $2 \cdot 10^{-4}$, $1 \cdot 10^{-4}$). The largest learning rate yielding stable optimization during continued BiSSLight training was ultimately selected.

Following the original BiSSL protocol, we initialize the downstream classification head using a brief linear probing phase prior to BiSSLight training. This improves training stability and convergence. The linear probing is performed for 20 epochs using a fixed learning rate equal to that used for the upper-level BiSSLight optimization. For simplicity, we otherwise use exact same optimizer configuration as for the upper-level.

Overall Details We run BiSSLight as detailed in Algorithm 1 for a total of $T = 500$ alternations, with $N_L = 20$ lower-level and $N_U = 8$ upper-level optimization steps per alternation. For the lower-level optimization, this corresponds to just under two full pretext task epochs given the employed batch size. The regularization weight is fixed to $\lambda = 0.001$ across all experiments. These settings all follow the original BiSSL implementation. For the block-wise M-FAC approximation, we consider all parameters of each layer as a block, which in our case includes the collective LoRA modules assigned to the query and value matrices in each attention layer.

Table 2: Trainable LoRA parameter counts (in M) for the evaluated ViT backbones across different ranks. The default configuration used throughout the paper ($r = 64$) corresponds to 2.75%, 2.07%, and 1.66% of the parameters of the ViT-B/16, ViT-L/16, and ViT-H/14 backbones, respectively.

Parameters (M)	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 16$	$r = 32$	$\mathbf{r = 64}$	$r = 128$	Full Backbone
ViT-B/16	0.04	0.07	0.15	0.30	0.59	1.18	**2.36**	4.72	85.80
ViT-L/16	0.10	0.20	0.39	0.79	1.57	3.15	**6.29**	12.58	303.30
ViT-H/14	0.16	0.33	0.66	1.31	2.62	5.24	**10.49**	20.97	630.77

C.4 LoRA Parameter Counts

Table 2 reports the number of trainable LoRA parameters for each backbone and rank configuration. As expected, the parameter count scales linearly with the LoRA rank, while remaining a small fraction of the frozen backbone size even for the default $r = 64$ used throughout this work.

D Additional Results

D.1 I-JEPA Pretrained Backbone

D.1.1 Implementation Details

We largely follow the experimental setup described in Appendix C, reporting only the modifications conducted for the I-JEPA (Assran et al., 2023) setting below.

Self-Supervised Pretraining We use the official I-JEPA ViT-H/14 checkpoint at a resolution of 224×224, pretrained on ImageNet-1k (Deng et al., 2009). We refer to the original work for the complete pretraining configuration.

Downstream Adaptation Since the pretrained I-JEPA model does not include a designated CLS token, we instead pass the average-pooled patch tokens from the final transformer layer through a batch normalization layer, followed by a linear layer. We use a weight decay of 0.0005 and employ almost the same augmentation pipeline as of the main experiments in Section C.2, except that the minimum random crop ratio is increased from 0.2 to 0.3 to match the original I-JEPA setup.

BiSSLight The lower-level training uses the same image augmentation pipeline employed during training of the original I-JEPA checkpoint. We initialize the teacher EMA schedule at 0.9995 (instead of the original 0.996), reflecting that training starts from a well-pretrained initialization and therefore does not require the teacher to track the student as closely during the early stages of optimization. We further use a weight decay of 0.0005 and a LoRA learning rate of 0.005. For the upper level, we also use a weight decay of 0.0005 and set the minimum crop ratio to 0.3 to remain consistent with the downstream fine-tuning setup.

D.1.2 Classification Accuracies

Table 3 reports downstream classification accuracies obtained using I-JEPA-pretrained models. BiSSLight consistently improves performance with statistical significance across all six benchmarks. Baseline accuracies are generally lower than those obtained with similarly sized MAE-pretrained models (see Table 1), which may partly reflect that I-JEPA was primarily designed with linear probing in mind rather than fine-tuning, as well as the fact that the I-JEPA checkpoint is trained for fewer epochs. Nevertheless, these results show that the benefits of BiSSLight extend beyond MAE pretraining and suggest that BiSSlight retains the ability of the original BiSSL framework to effectively adapt across pretext tasks.

Table 3: Comparison of classification accuracies obtained via directly applying PEFT to a ViT-H/14 backbone pretrained with the I-JEPA pretext task and by applying BiSSLight prior to PEFT. Bold font indicates a statistically significant improvement in accuracy obtained with BiSSLight, occurring in all evaluated settings.

I-JEPA	Pets	DTD	VOC07	Flowers	CUB200	Aircrafts
PEFT	94.6±0.1	73.1±0.4	82.4±0.2	95.1±0.4	86.4±0.3	76.0±0.6
BiSSLight+PEFT	**94.7**±0.1	**74.0**±0.6	**84.0**±0.3	**96.1**±0.3	**86.8**±0.2	**77.7**±0.4

D.2 Comparing BiSSLight with BiSSL

D.2.1 BiSSL Implementation Details

For the experiments in Section 4.3 that compares BiSSLight with the original BiSSL framework (Zakarias et al., 2026), we recreate the original optimization procedure by replacing M-FAC hypergradient approximation (Frantar et al., 2021) with the conjugate-gradient (CG) (Nazareth, 2009; Shewchuk, 1994) approach used in BiSSL, while keeping all other experimental settings identical.

Following the original implementation, we perform five CG iterations per upper-level gradient estimate, corresponding to five Hessian-vector product evaluations per upper-level optimization step. We employ a damping value of 1 (i.e., solving the system using $\lambda + 1$), consistent with standard CG practice, which we found sufficient for stable training. As a practical note, fused attention kernels are disabled during Hessian-vector product computations, as PyTorch fused attention kernels do not currently support the required second-order derivative calculations.

Apart from these modifications, the experimental setup is identical to that described in Appendix C. We refer the reader to the original BiSSL work for further details regarding the CG-based optimization procedure.

D.2.2 Compute Time and Memory Benchmark Setup Details

BiSSL was executed on two NVIDIA A40 GPUs for ViT-B/16 and ViT-L/16, and four A40 GPUs for ViT-H/14 due to its substantially higher memory requirements. Although BiSSLight can be executed on a single A40 GPU for all evaluated backbones, we use the same GPU configurations as for BiSSL to ensure a fair comparison in Figure 2. Reported GPU-hours are computed as the total training runtime multiplied by the number of GPUs used.

Peak memory usage is measured on a per-epoch basis. Before each training epoch, the recorded CUDA memory statistics are reset, after which the peak memory allocation of each GPU is measured during the training phase of the epoch. The per-GPU peaks are summed and logged immediately after training each epoch, and the largest such value observed across the entire training run is reported. Since memory peaks on different GPUs may not occur simultaneously, the reported values should be regarded as conservative upper-bound estimates. However, the same measurement protocol is applied to both methods compared in Figure 2, making the reported values suitable for relative comparison.

The results are based on training on the CUB200 dataset, and since both methods execute a fixed number of optimization steps, runtime does not depend on the size of downstream dataset, making the reported numbers broadly representative. We remark that the reported memory and GPU-hour measurements should primarily be interpreted as relative comparisons rather than absolute reference values, as practical resource requirements depend on both the implementation and hardware.

Single- versus Multi-GPU Memory Usage To provide additional context for the conservative memory measurements reported throughout the paper, we additionally evaluate BiSSLight on a single NVIDIA A40 GPU for each backbone and report the results in Table 4. A corresponding comparison is not possible for BiSSL, as its memory requirements exceed the capacity of a single A40 GPU. The results show that the memory requirements reported throughout the paper are conservative, with peak memory usage often being lower when measured directly on a single GPU. This

Table 4: Peak GPU memory requirements (in GB) for BiSSL, BiSSLight, and standard PEFT across different backbone scales and number of GPUs utilized, illustrating the practical memory footprint of BiSSLight relative to both BiSSL and conventional PEFT.

	GPUs (ViT-B/L/H)	ViT-B/16	ViT-L/16	ViT-H/14
BiSSL	(2/2/4)	58 GB	77 GB	141 GB
BiSSLight	(2/2/4)	14 GB	22 GB	46 GB
BiSSLight	(1/1/1)	13 GB	19 GB	33 GB
PEFT, BS=256	(1/1/2)	14 GB	36 GB	80 GB
PEFT, BS=64	(1/1/1)	4 GB	10 GB	22 GB

Table 5: Comparison of classification accuracies obtained after applying either BiSSL or BiSSLight prior to fine-tuning using a ViT-B/16 backbone. BiSSLight generally reproduces the downstream performance gains of the original BiSSL framework across the evaluated datasets despite its substantially lower computational requirements.

	Pets	DTD	VOC07	Flowers	CUB200	Aircrafts
PEFT	$91.7_{\pm 0.3}$	$69.9_{\pm 0.5}$	$80.9_{\pm 0.3}$	$92.8_{\pm 0.5}$	$79.8_{\pm 0.5}$	$70.6_{\pm 0.6}$
BiSSLight+PEFT	$92.1_{\pm 0.2}$	$70.3_{\pm 0.4}$	$81.7_{\pm 0.2}$	$93.5_{\pm 0.3}$	$81.2_{\pm 0.3}$	$72.7_{\pm 0.3}$
BiSSL+PEFT	$92.0_{\pm 0.5}$	$70.8_{\pm 0.4}$	$81.9_{\pm 0.1}$	$93.4_{\pm 0.4}$	$81.2_{\pm 0.3}$	$72.7_{\pm 0.3}$

effect is particularly pronounced for ViT-H/14, where the measured peak allocation decreases from 46 GB to 33 GB. Part of this discrepancy stems from summing peak allocations across GPUs, while additional differences may arise from the inherent memory overheads associated with multi-GPU training.

Table 4 also reports the memory requirements of the subsequent parameter-efficient fine-tuning (PEFT) stage, with implementation details provided in Section C.2, measured on the same hardware. Under the batch-size configuration used throughout the paper, the PEFT stage requires more memory than BiSSLight. Reducing the PEFT batch size to match the BiSSLight upper-level optimization substantially lowers its memory footprint, indicating that the primary memory overhead of BiSSLight originates from the lower-level pretext optimization. Nevertheless, the resulting memory requirements remain broadly comparable, suggesting that BiSSLight can generally be deployed on hardware similar to that used for standard PEFT.

D.2.3 Similarity of Classification Accuracies and Representations

To verify that the efficiency improvements of BiSSLight do not come at the expense of downstream performance, Table 5 compares BiSSLight against the original BiSSL framework on a ViT-B/16 backbone. BiSSLight achieves performance highly comparable to BiSSL across all datasets, differing by less than a fraction of a percentage point. Given the substantially higher computational demands of BiSSL, we restrict this comparison to a backbone size, which we consider sufficient to verify that the proposed efficiency improvements preserve downstream performance relative to the original framework.

Representational Similarity To additionally assess how similar the models learned via BiSSL and BiSSLight are, we compare the representations produced by the same models used in the downstream classification experiments for Table 5, namely the pretrained backbone, the BiSSL-trained backbone, and the BiSSLight-trained backbone for each downstream dataset. For each comparison, we compute two representational similarity metrics: Linear Centered Kernel Alignment (Linear CKA) (Kornblith et al., 2019) and Representational Similarity Analysis (RSA) (Kriegeskorte et al., 2008; Abnar et al., 2019; Klabunde et al., 2025). Linear CKA quantifies the alignment between two

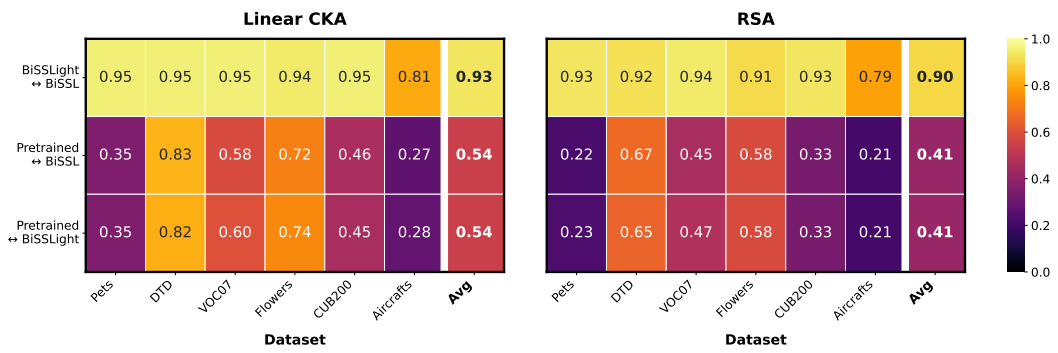

Figure 5: Dataset-specific representational similarity measured pair-wise using Linear CKA and RSA between combinations of the pretrained backbone, BiSSL-trained backbones and the BiSSLight-trained backbones. Each entry reports the representational similarity between two model variants evaluated on the indicated downstream dataset. Higher values indicate greater representational similarity. BiSSLight exhibits consistently high similarity to BiSSL across datasets while inducing comparable representational changes relative to the pretrained backbone.

feature spaces based on their centered covariance structure, while remaining invariant to orthogonal transformations and isotropic scaling. For RSA, we first construct representational dissimilarity matrices (RDMs) using pairwise cosine distances between samples, and subsequently compute the Spearman correlation between the upper-triangular entries of the resulting RDMs. RSA therefore measures the extent to which two representations preserve the same pairwise relationships between samples. Further details regarding both metrics can be found in the original works.

Figure 5 reports representational similarity scores between the pretrained backbone, BiSSL-trained models, and BiSSLight-trained models across all downstream datasets. Representations learned by BiSSL and BiSSLight exhibit consistently high similarity under both Linear CKA and RSA, with average scores of 0.93 and 0.90, respectively. While not equal to 1.0, these values indicate a very high degree of representational similarity, particularly given the stochastic nature of the training and differences in the implementations of the two methods that may lead to minor deviations. Additionally, both methods exhibit substantially lower similarity to the pretrained backbone across all datasets, indicating that the bilevel training stage induces non-trivial representational changes. Lastly, the similarity scores between the pretrained backbone and each of the two methods are remarkably consistent across datasets, with average Linear CKA and RSA scores of 0.54 and 0.41, respectively, in both cases. While this alone does not imply that the resulting representations are similar, when considered together with the high similarity between BiSSL and BiSSLight, it provides additional evidence that suggest both methods induce comparable representational changes.

Taken together, these findings indicate that replacing the CG-based hypergradient approximation with M-FAC has limited effect on the resulting learned representations. This is consistent with the comparable downstream performance observed in Table 5, and implies that BiSSLight induces representational changes highly similar to those of the original framework while substantially reducing computational cost.

D.3 Attention Rollout Analysis

To complement the analysis in Section 4.4, we include additional attention rollout maps for the Flowers and CUB200 datasets in Figures 6 and 7, respectively. Similar to the examples shown for Pets in Figure 4, the BiSSLight-trained models frequently produce attention maps that appear more concentrated on semantically relevant image regions. In multiple cases, the highlighted patches are distributed across a larger portion of the target object and exhibit stronger contrast between object regions and surrounding background content.

Despite the relatively modest accuracy improvements observed on Pets in Table 1, the qualitative differences are often particularly pronounced for this dataset. One possible explanation is that Pets

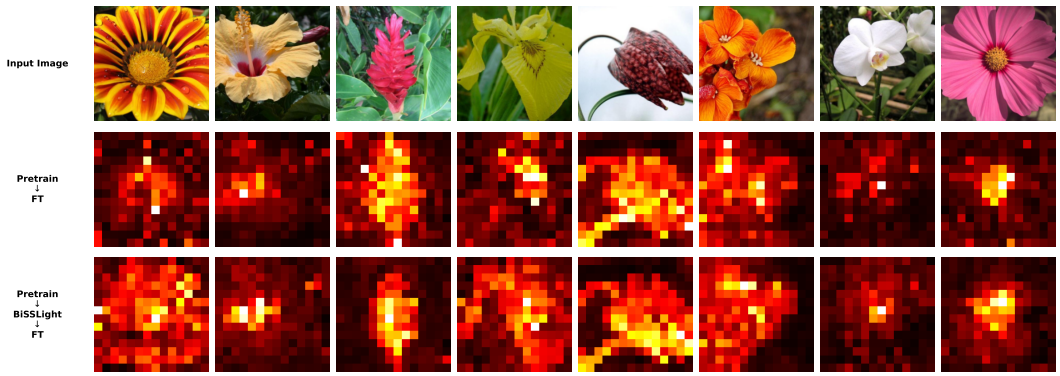

Figure 6: Attention rollout examples from the Flowers validation set. The middle row corresponds to models directly fine-tuned from the MAE-pretrained checkpoint, while the bottom row shows models fine-tuned after the BiSSLight stage. BiSSLight produces attention patterns that are more concentrated on regions associated with the target object and less dispersed across the image.

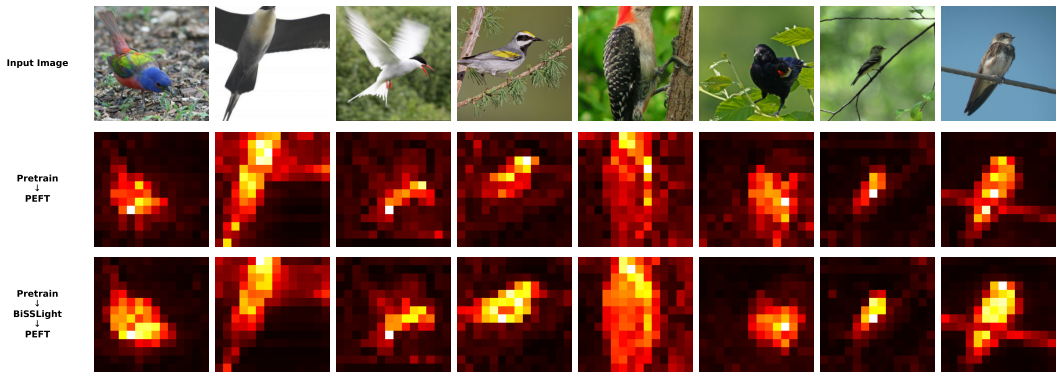

Figure 7: Attention rollout examples from the CUB200 validation set. The middle row corresponds to models directly fine-tuned from the MAE-pretrained checkpoint, while the bottom row shows models fine-tuned after the BiSSLight stage. BiSSLight produces attention patterns that are more concentrated on regions associated with the target object and less dispersed across the image.

is already well aligned with the ImageNet pretraining distribution, allowing the baseline models to achieve strong performance using only a subset of the task-relevant visual information. Consequently, the downstream classification task may be relatively insensitive to further improvements in attention allocation, even when the resulting representations exhibit more object-focused behavior.

It should be noted that attention rollout provides only a highly condensed and approximate view of the model's internal attention patterns. Consequently, we use these visualizations primarily to assess whether systematic differences emerge between the attention maps produced by the two training procedures rather than to explain individual model predictions. Under this interpretation, the observed differences appear reasonably consistent across datasets.