NORMALIZE THE UPDATE, NOT THE WEIGHTS: EFFICIENT LOSS WEIGHTING FOR MULTI-OBJECTIVE PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern deep models are often pretrained with multi-objective losses whose relative weights act as hyperparameters. Selecting these weights through random search or Bayesian optimization is computationally expensive because it requires many independent training runs. We introduce GraP, an online bilevel method that adapts pretraining loss weights to improve downstream performance through gradient alignment. A central challenge is scale ambiguity: uniformly rescaling the weights changes the magnitude of the composite update and therefore the effective learning rate. Rather than normalizing the weights themselves, GraP constrains the composite update norm. We show theoretically that this normalization preserves the optimal alignment direction and demonstrate its empirical benefits. To enable efficient optimization, GraP computes per-loss gradients in the shared representation space and reuses their weighted combination, reducing the number of full-backbone backward passes from K to one per iteration. Across event-sequence modeling tasks, GraP improves on adaptive-weighting baselines and matches or improves on carefully tuned hyperparameters, while requiring only a single training run with about 34% additional training-time overhead. In self-supervised computer vision, GraP remains competitive with standard tuning strategies.

1 INTRODUCTION

Modern deep learning systems are commonly pretrained on large-scale unlabeled data and then adapted to downstream tasks (Devlin et al., 2019; Grill et al., 2020; Babaev et al., 2022). In many pretraining pipelines, the objective is a weighted combination of multiple loss terms (Padhi et al., 2021; Estepa et al., 2023; Karpukhin & Savchenko, 2025). The choice of these weights can substantially affect representation quality and downstream performance, yet they are often selected manually or through black-box hyperparameter optimization.

Standard tuning strategies, including grid search, random search, and Bayesian optimization, typically require many independent training runs (Snoek et al., 2012). Their cost becomes substantial when pretraining is expensive, or the objective contains many loss terms. Uniform weighting avoids this search but does not account for differences in the scales, optimization dynamics, or downstream relevance of individual objectives. Gradient-based hyperparameter optimization and adaptive auxiliary-task weighting offer more efficient alternatives by using downstream or primary-task information to update loss weights online (Franceschi et al., 2018; Du et al., 2018; Lin et al., 2019). However, applying these ideas to composite pretraining raises two practical questions: how to resolve the scale ambiguity between loss weights and the magnitude of the backbone update, and how to compute the weight updates without performing a full backward pass for every loss term.

To address these questions, we introduce Gradient-aligned Pretraining (GraP), an online bilevel method specialized to composite pretraining. GraP adapts the loss weights by aligning the composite pretraining gradient with a downstream gradient. Rather than normalizing the weights directly, it normalizes the resulting composite update, separating the relative contributions of the losses from the scale of the backbone update. To reduce computational cost, GraP computes per-loss alignment

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.

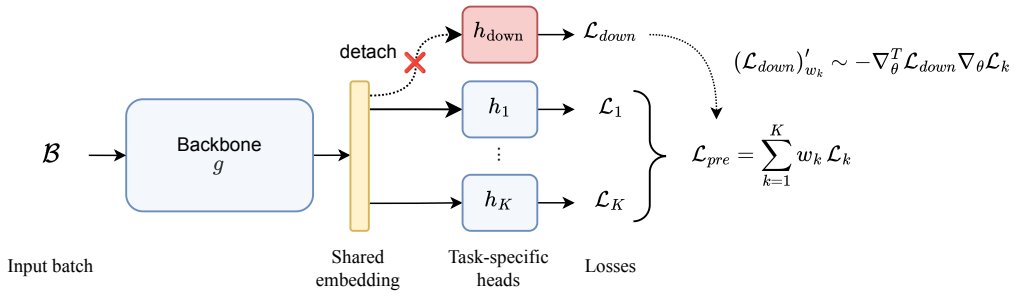


Figure 1: Overview of the proposed GraP framework. A shared backbone produces embeddings used by multiple pretraining heads and a detached downstream head. Loss weights are updated by aligning the composite pretraining gradient with the downstream gradient in the shared representation space.

signals in the shared representation space and reuses their weighted combination for model training. This reduces the number of full-backbone backward passes from K to one per iteration.

The main contributions of this paper are as follows:

1. We analyze the scale ambiguity of loss weighting, show that conventional weight normalization can alter the selected composite-gradient direction, and propose normalizing the composite update instead.
2. We develop an efficient shared-representation implementation that reduces the number of full-backbone backward passes from K to one per iteration. We theoretically relate the resulting representation-space approximation to its parameter-space counterpart.
3. We evaluate GraP on event-sequence modeling and self-supervised computer vision. GraP performs comparably to or better than search-tuned and adaptive-weighting baselines on event-sequence tasks, remains competitive on vision benchmarks, and requires only a single training run with moderate overhead. The learned weights can also highlight objectives with limited empirical contribution.

2 Problem Formulation

This paper studies the problem of tuning weights in composite pretraining objectives to improve downstream task performance. Let $\mathcal{D} = \{x_i\}_{i=1}^N$ be a pretraining dataset, where a subset $\mathcal{D}_s = \{(x_i, y_i)\}$ contains downstream labels, while the remaining examples are unlabeled. Let $\mathcal{B}$ denote a minibatch sampled from $\mathcal{D}$, and let $\mathcal{B}_s \subseteq \mathcal{B}$ denote its labeled subset. We consider a model consisting of a shared backbone $z = g(\mathcal{B}, \theta)$, which produces embeddings, and task-specific heads, as illustrated in Fig. 1. Each pretraining objective $\mathcal{L}_k$, $k = 1, \ldots, K$, is applied through a corresponding head $h_k(z, \theta_k)$, while the downstream task uses a separate head $h_{\text{down}}(z, \theta_{\text{down}})$ with loss $\mathcal{L}_{\text{down}}$.

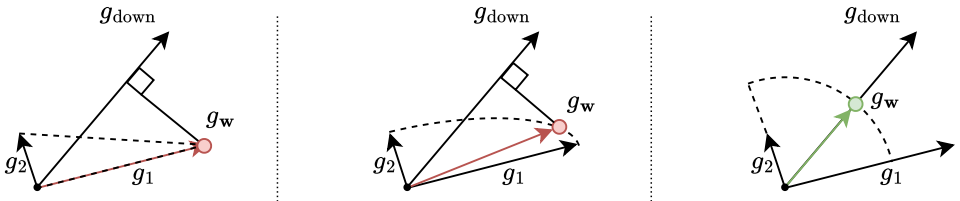
The model is trained using the composite pretraining objective

$$\mathcal{L}_{\text{pre}}(\mathcal{B}, \mathbf{w}; \theta, \theta_{1:K}) = \sum_{k=1}^{K} w_k \, \mathcal{L}_k\big(h_k(g(\mathcal{B}, \theta), \theta_k)\big), \tag{1}$$

where $\{\mathcal{L}_k\}_{k=1}^K$ are task-specific loss terms and $\mathbf{w} = (w_1, \ldots, w_K)$ are their weights. Similar to previous works, we restrict the weights to be non-negative, as negative values would reverse the optimization direction of individual losses and may lead to unstable training (Sener & Koltun, 2018).

Our goal is to select $\mathbf{w}$ such that the representation learned during pretraining performs well on the downstream task. For a labeled minibatch $\mathcal{B}_s = (X, Y)$, downstream performance is measured by

$$\mathcal{L}_{\text{down}}(\mathcal{B}_s; \theta, \theta_{\text{down}}) = \mathcal{L}_{\text{down}}\big(h_{\text{down}}(g(X, \theta), \theta_{\text{down}}), Y\big). \tag{2}$$



(a) Weights sum to 1 (b) Weights L2-norm is 1 (c) Gradient L2-norm is 1

Figure 2: Comparison of normalization strategies. The objective is to maximize the alignment between the composite gradient $g_{\mathbf{w}} = w_1 g_1 + w_2 g_2$ and the downstream gradient g_{down}. Colored circles indicate the optimal solutions under different normalization schemes. Weight-sum and weight-norm normalizations distort the optimal composite-gradient direction, while composite-gradient normalization preserves the optimal alignment direction.

We formalize loss-weight selection as a bilevel optimization problem. Omitting dataset expectations for simplicity, the outer problem is

$$\mathbf{w}^* \in \underset{\mathbf{w}\in\mathbb{R}_+^K, \mathbf{w}\neq\mathbf{0}}{\arg\min}\ \mathcal{L}_{\text{down}}\big(\mathcal{B}_s; \theta^*(\mathbf{w}), \theta^*_{\text{down}}(\mathbf{w})\big). \tag{3}$$

The pretrained backbone and pretraining heads are determined by the inner problem

$$\big(\theta^*(\mathbf{w}), \theta^*_{1:K}(\mathbf{w})\big) \in \underset{\theta,\theta_{1:K}}{\arg\min}\, \mathcal{L}_{\text{pre}}(\mathcal{B}, \mathbf{w}; \theta, \theta_{1:K}), \tag{4}$$

while the downstream head is fitted to the resulting representation:

$$\theta^*_{\text{down}}(\mathbf{w}) \in \underset{\theta_{\text{down}}}{\arg\min}\, \mathcal{L}_{\text{down}}\big(\mathcal{B}_s; \theta^*(\mathbf{w}), \theta_{\text{down}}\big). \tag{5}$$

Thus, downstream labels guide the selection of the pretraining loss weights but do not directly optimize the backbone. In practice, we approximate the inner solution using truncated stochastic gradient updates and adapt the weights online, as described in the following section.

3 Method

We first derive an online update for the pretraining loss weights. We then address the scale ambiguity of this update through composite-gradient normalization and introduce a shared-representation approximation that avoids computing a full backbone gradient for every loss.

3.1 Online loss-weight optimization

We derive an SGD-based update for the loss weights $\mathbf{w}$ under the bilevel formulation of Sec. 2. We approximate the inner optimization by a single SGD step, corresponding to the simplest form of truncated backpropagation (Shaban et al., 2019). In App. A, we show that, for sufficiently small learning rates, longer optimization rollouts yield an analogous weight-update rule.

Given a minibatch $\mathcal{B}$ and its labeled subset $\mathcal{B}_s = (X, Y)$, let

$$g_k = \nabla_\theta \mathcal{L}_k\big(h_k(g(\mathcal{B}, \theta), \theta_k)\big), \qquad k = 1, \ldots, K, \tag{6}$$

denote the backbone gradient of the k-th pretraining loss. The composite pretraining gradient is

$$g_{\text{pre}}(\mathbf{w}) = \sum_{k=1}^{K} w_k g_k = G^\top \mathbf{w}, \tag{7}$$

where $G \in \mathbb{R}^{K\times p}$ contains $g_k^\top$ as its k-th row and $p = \dim(\theta)$. An SGD step with learning rate η gives

$$\theta' = \theta - \eta G^\top \mathbf{w}. \tag{8}$$

During this local update, we treat the downstream-head parameters as fixed. Since θ' depends on $\mathbf{w}$, differentiating the downstream loss through Eq. 8 yields

$$\frac{\partial \mathcal{L}_{\text{down}}}{\partial w_k} = g'^{\top}_{\text{down}} \frac{\partial \theta'}{\partial w_k} = -\eta g'^{\top}_{\text{down}} g_k, \tag{9}$$

where

$$g'_{\text{down}} = \nabla_{\theta'} \mathcal{L}_{\text{down}}\big(\mathcal{B}_s; \theta', \theta_{\text{down}}\big). \tag{10}$$

Thus, the pretraining loss receives a larger weight when its backbone gradient aligns with the downstream gradient after an SGD update.

Computing g'_{down} requires evaluating the model at θ'. Instead, we use the online first-order approximation

$$g'_{\text{down}} \approx g_{\text{down}} = \nabla_{\theta} \mathcal{L}_{\text{down}}\big(\mathcal{B}_s; \theta, \theta_{\text{down}}\big). \tag{11}$$

Under standard smoothness assumptions, $g'_{\text{down}} = g_{\text{down}} + O(\eta)$, so the resulting error in Eq. 9 is of order $O(\eta^2)$. Treating the gradients as fixed within one iteration, the corresponding local alignment objective is

$$\max_{\mathbf{w} \in \mathbb{R}^K_+, \mathbf{w} \neq 0} \quad \mathbf{w}^{\top} G g_{\text{down}}. \tag{12}$$

A direct implementation requires computing each g_k separately and therefore performs K full backward passes through the backbone. In Sec. 3.3, we replace these parameter-space gradients with substantially cheaper gradients at the shared representation.

3.2 Composite-gradient normalization

Without an additional constraint, the alignment objective in Eq. 12 is unbounded whenever a positively aligned direction exists.

Common approaches constrain the weights themselves, for example by requiring $\sum_k w_k = 1$ or $\|\mathbf{w}\|_2 = 1$. Such constraints control the scale of $\mathbf{w}$ but not the norm of $G^{\top}\mathbf{w}$, which also depends on the norms and correlations of the individual gradients. Consequently, maximizing the unnormalized alignment under a weight-space constraint may favor large-magnitude gradients rather than the best-aligned composite-gradient direction, as illustrated in Fig. 2.

We instead constrain the norm of the composite gradient:

$$\left\|G^{\top}\mathbf{w}\right\|_2 = \left\|\sum_{k=1}^{K} w_k g_k\right\|_2 = 1. \tag{13}$$

Equivalently, for nonzero composite gradients, we optimize

$$\max_{\mathbf{w} \in \mathbb{R}^K_+, G^{\top}\mathbf{w} \neq 0} \quad \frac{\mathbf{w}^{\top} G g_{\text{down}}}{\|G^{\top}\mathbf{w}\|_2}. \tag{14}$$

This objective is invariant to a common positive rescaling of $\mathbf{w}$ and therefore separates the relative loss contributions from the magnitude of the resulting update.

By the Cauchy–Schwarz inequality, Eq. 14 is upper-bounded by $\|g_{\text{down}}\|_2$. If there exists a nonnegative weight vector satisfying $G^{\top}\mathbf{w} = \alpha g_{\text{down}}$ for some $\alpha > 0$, the bound is attained by an aligned composite gradient. More generally, the objective selects the attainable composite-gradient direction with the largest downstream alignment. A formal statement is given in App. B.

3.3 Shared-representation approximation

The normalized objective can also be interpreted as distance minimization:

$$\left\| \frac{G^{\top}\mathbf{w}}{\|G^{\top}\mathbf{w}\|_2} - g_{\text{down}} \right\|_2^2 = 1 + \|g_{\text{down}}\|_2^2 - 2\frac{\mathbf{w}^{\top} G g_{\text{down}}}{\|G^{\top}\mathbf{w}\|_2}. \tag{15}$$

Since g_{down} is independent of $\mathbf{w}$ within an iteration, maximizing Eq. 14 is equivalent to minimizing this distance.

Algorithm 1 GraP for Online Pretraining Loss Weighting

initialize $w_k \leftarrow 1, k = 1, \dots, K$
for each minibatch $\mathcal{B}$ with labeled subset $\mathcal{B}_s$ **do**
  $z \leftarrow g(\mathcal{B}, \theta), \quad z_s \leftarrow z|_{\mathcal{B}_s}$
  $o_k \leftarrow h_k(z, \theta_k), \quad o_{\text{down}} \leftarrow h_{\text{down}}(z_s, \theta_{\text{down}})$
  $\tilde{g}_k \leftarrow \nabla_z \mathcal{L}_k(o_k), \quad \tilde{g}_{\text{down}} \leftarrow \nabla_{z_s} \mathcal{L}_{\text{down}}(o_{\text{down}}, Y)$
  $\bar{\mathbf{w}} \leftarrow \mathbf{w} / \left(\|\tilde{G}^\top \mathbf{w}\|_2 + \epsilon\right)$
  $\mathbf{w}^+ \leftarrow \left[\mathbf{w} + \eta \nabla_{\mathbf{w}} \left(\bar{\mathbf{w}}^\top \tilde{G} \tilde{g}_{\text{down}}\right)\right]_+$
  $\theta \leftarrow \theta - \eta \nabla_\theta \langle z, \bar{\mathbf{w}}^\top \tilde{G} \rangle$
  update each θ_k using $\nabla_{\theta_k} \mathcal{L}_k(o_k)$
  update θ_{down} using $\nabla_{\theta_{\text{down}}} \mathcal{L}_{\text{down}}(o_{\text{down}}, Y)$
  $\mathbf{w} \leftarrow \mathbf{w}^+$
end for

Computing Eq. 14 directly requires all K parameter-space gradients. Following prior methods that compare task gradients at a shared representation (Sener & Koltun, 2018), we instead differentiate each loss only with respect to the backbone embedding $z = g(\mathcal{B}, \theta)$:

$$\tilde{g}_k = \nabla_z \mathcal{L}_k\big(h_k(z)\big), \quad \tilde{g}_{\text{down}} = \nabla_{z_s} \mathcal{L}_{\text{down}}\big(h_{\text{down}}(z_s), Y\big), \tag{16}$$

where z_s denotes the embeddings corresponding to $\mathcal{B}_s$. We view $\tilde{g}_{\text{down}}$ as a gradient with respect to the full minibatch embedding, with zero entries for examples outside $\mathcal{B}_s$. Let $\tilde{G} \in \mathbb{R}^{K \times m}$ contain $\tilde{g}_k^\top$ as its k-th row. GraP optimizes the representation-space approximation

$$\max_{\mathbf{w} \in \mathbb{R}_+^K, \tilde{G}^\top \mathbf{w} \neq 0} \quad \frac{\mathbf{w}^\top \tilde{G} \tilde{g}_{\text{down}}}{\|\tilde{G}^\top \mathbf{w}\|_2}. \tag{17}$$

This replacement is an approximation because the backbone Jacobian can change gradient norms and angles. Let $J_\theta = \partial z / \partial \theta$. By the chain rule,

$$G^\top \mathbf{w} = J_\theta^\top \tilde{G}^\top \mathbf{w}, \qquad g_{\text{down}} = J_\theta^\top \tilde{g}_{\text{down}}. \tag{18}$$

Consequently, the unnormalized mismatch satisfies

$$\left\|G^\top \mathbf{w} - g_{\text{down}}\right\|_2 \leq \|J_\theta^\top\|_2 \left\|\tilde{G}^\top \mathbf{w} - \tilde{g}_{\text{down}}\right\|_2. \tag{19}$$

Thus, representation-space mismatch controls an upper bound on the corresponding parameter-space mismatch up to a Jacobian-dependent factor. Further analysis and assumptions, including approximation of the normalized objective from Eq. 14, are provided in App. C.

The computational advantage of the proposed objective is that the K gradients $\tilde{g}_k$ require backward operations only through the task-specific heads. Their weighted combination is then propagated through the backbone once. The method therefore reduces the number of full-backbone backward passes from K to one per iteration. The head-level computation and storage of the representation gradients remain $O(K)$, so the benefit is greatest when the heads are small relative to the shared backbone.

3.4 Final algorithm

Algorithm 1 summarizes the resulting procedure. The pretraining heads are updated using their unweighted losses, preventing their learning rates from being coupled to the current values of $\mathbf{w}$ and allowing them to continue training when some weights become zero. The downstream gradient is stopped at the shared representation: it updates the loss weights and the downstream head, but is not directly propagated into the backbone. Downstream supervision therefore influences the backbone only through the learned pretraining-loss weights.

Overall, GraP performs one backbone forward pass and one backbone backward pass per iteration, together with K inexpensive head-to-representation gradient computations.

Method	Churn	AgePred	AlfaBattle	MIMIC-III	Taobao	AVG
Supervised RNN	79.43± 0.64	61.48± 0.29	78.87± 0.18	91.56± 0.24	69.57± 0.20	76.18± 0.12
RNN Equal Weights	83.79± 0.79	63.76± 0.41	78.57± 0.00	91.25± 0.05	70.20± 0.49	77.51± 0.20
RNN GradNorm	83.41± 0.41	**64.80± 0.32**	79.75± 0.28	90.72± 0.11	70.12± 0.31	77.76± 0.14
RNN DWA	83.74± 0.69	64.58± 0.18	78.49± 0.03	91.19± 0.07	70.27± 0.26	77.66± 0.15
RNN MGDA	77.79± 0.43	63.47± 0.27	78.66± 0.00	88.84± 0.23	70.42± 0.15	75.84± 0.11
RNN PCGrad	83.77± 0.67	64.10± 0.29	78.55± 0.09	91.20± 0.10	70.23± 0.28	77.57± 0.16
RNN GraP (Our)	**84.00± 0.13**	64.04± 0.05	**79.86± 0.08**	**91.63± 0.10**	**70.70± 0.12**	**78.05± 0.05**
Transformer Equal Weights	**83.50± 0.95**	61.27± 0.60	75.01± 0.05	89.37± 0.05	67.83± 0.55	75.39± 0.25
Transformer GradNorm	82.92± 0.65	62.31± 0.36	**78.56± 0.06**	87.12± 1.72	**69.80± 0.26**	76.14± 0.38
Transformer DWA	83.42± 0.58	58.82± 0.25	75.85± 0.21	89.01± 0.20	67.87± 0.50	74.99± 0.17
Transformer MGDA	82.24± 0.39	60.38± 0.22	77.03± 0.03	83.58± 0.35	69.72± 0.33	74.59± 0.13
Transformer PCGrad	82.87± 0.73	58.57± 0.32	75.79± 0.05	88.95± 0.15	68.05± 0.44	74.85± 0.18
Transformer GraP (Our)	83.46± 1.18	**63.43± 0.23**	78.02± 0.10	**91.02± 0.11**	68.23± 0.31	**76.83± 0.25**
HT-Transformer Equal Weights	83.96± 0.65	54.76± 1.15	76.65± 0.22	87.39± 0.59	68.62± 0.19	74.28± 0.30
HT-Transformer GradNorm	84.45± 0.36	62.24± 0.45	**79.55± 0.01**	84.48± 0.70	69.83± 0.00	76.11± 0.18
HT-Transformer DWA	84.11± 0.48	59.06± 0.78	76.84± 0.22	81.32± 0.21	68.28± 0.58	73.92± 0.23
HT-Transformer MGDA	**84.56± 0.40**	60.17± 0.13	78.78± 0.24	83.71± 0.58	**70.07± 0.00**	75.46± 0.15
HT-Transformer PCGrad	84.22± 0.39	58.93± 0.37	76.52± 0.07	82.65± 2.51	68.71± 0.48	74.21± 0.52
HT-Transformer GraP (Our)	83.44± 0.36	**62.92± 0.16**	79.23± 0.20	**91.01± 0.11**	69.23± 0.00	**77.17± 0.09**

Table 1: Event-sequence results. We report the mean and standard deviation across 4 seeds. The best method for each dataset-backbone combination is shown in bold.

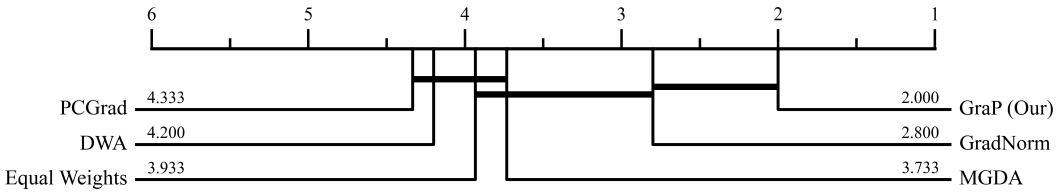


Figure 3: Critical difference-style diagram with paired Student's t-tests and Holm correction.

4 EXPERIMENTS

We evaluate GraP in two domains: event-sequence modeling and computer vision. We compare it with equal weighting and established methods for combining multiple training objectives: GradNorm (Chen et al., 2018), Dynamic Weight Averaging (DWA) (Liu et al., 2019), MGDA (Sener & Koltun, 2018), and PCGrad (Yu et al., 2020). When available, we also compare against weights obtained through hyperparameter search. Our experimental design follows the standard training pipelines of the corresponding benchmarks. The multi-task learning baselines combine the same K pretraining objectives as GraP but do not use the downstream loss or labels in their backbone updates. This preserves their original task-agnostic formulation and separates loss weighting from auxiliary supervised training. Additional experimental details and dataset statistics are provided in App. D, while ablation studies are presented in App. F. A comparison with OL-AUX in the reinforcement learning domain is provided in App. G.

Rationale for baseline selection. The selected baselines represent complementary strategies for combining multiple objectives within a single training run. DWA adapts weights according to relative loss reduction; GradNorm balances gradient magnitudes while accounting for training rates; MGDA selects the minimum-norm convex combination of task gradients, targeting Pareto stationarity; and PCGrad projects conflicting gradients to reduce interference. Together, they cover loss-dynamics weighting, gradient balancing, Pareto-oriented aggregation, and conflict mitigation. Related bilevel methods, such as AuxiLearn (Navon et al., 2021), use downstream performance to optimize auxiliary objectives but require second-order derivative operations and iterative inverse-Hessian–vector product approximation, increasing computational and implementation complexity. GraP instead uses only first-order gradients from standard backward passes. OL-AUX (Lin et al., 2019) is closer to our formulation, but lacks GraP's update normalization and shared representation

approximation. The variant of OL-AUX for pretraining is evaluated in the ablation study (Sec. F). Unlike the main baselines, GraP combines pretraining objectives through explicit differentiation, efficiently resolves the normalization ambiguity, and avoids multiple backward passes through the backbone.

4.1 EVENT SEQUENCES

Experiment design. We consider five real-world datasets of event sequences: *Churn*, *AgePred*, *AlfaBattle*, *MIMIC-III*, and *Taobao*. We follow the standard preprocessing, pretraining, and evaluation protocol for event sequence modeling introduced in prior work (Karpukhin & Savchenko, 2025). These datasets cover a range of downstream tasks, including churn prediction, age group classification, credit default prediction, mortality prediction, and user activity prediction, as outlined in App. D.

Each dataset consists of event sequences with heterogeneous features, where each event contains multiple fields (typically 3–18). We formulate next-token prediction (NTP) as a composite objective combining cross-entropy losses for categorical features and mean absolute error for continuous values. This naturally yields many loss components, making the setting suitable for evaluating loss-weight tuning methods. We additionally include a contrastive loss (Babaev et al., 2022), resulting in the following pretraining objective:

$$\mathcal{L}_{\text{pre}}^{\text{seq}} = \sum_{i=1}^{F} w_i \mathcal{L}_{\text{NTP}_i} + w_c \mathcal{L}_{\text{Contrastive}}, \tag{20}$$

where F denotes the number of data fields. During pretraining, we also tune a fully connected downstream classification head using cross-entropy loss. This head is used for weight tuning. Final model quality is evaluated by training a downstream classifier on top of frozen embeddings, using gradient boosting models as in prior work (Sakhno et al., 2025).

Results. Tab. 1 reports results on five event-sequence datasets. The supervised RNN generally underperforms unsupervised pretraining with equal weights, illustrating the benefit of pretraining when labeled data are limited. The proposed GraP method achieves the highest average performance and consistently improves upon the equal-weight baseline. Across the evaluated backbones, GraP obtains the best results on AlfaBattle, MIMIC-III, and Taobao, while MGDA performs best on Churn and GradNorm (Chen et al., 2018) performs best on AgePred, making it the strongest competing adaptive-weighting baseline overall. In Fig. 3, we additionally present a variant of the Critical Difference diagram (Demšar, 2006), where pairwise significance is assessed using paired Student's t-tests over matched seed-level evaluations, followed by Holm correction for multiple comparisons. According to this diagram, GraP achieves the best average rank and is significantly better than most competing methods. The difference between GraP and GradNorm does not remain significant after Holm correction; however, the uncorrected paired t-test indicates a significant difference in favor of GraP ($p \approx 0.0078$). The complete trajectories of the learned weights are provided in App. H.

Comparison with Optuna. In our standard setting, we combine NTP and contrastive objectives. Recent work provides Optuna-tuned weights for the NTP objective only (Karpukhin & Savchenko, 2025). In Tab. 2, we compare GraP with weights obtained via Optuna's Bayesian optimization (Akiba et al., 2019). The results show that our method achieves performance on par with weights tuned using Bayesian search.

Method	Churn	AgePred	AlfaBattle	MIMIC-III	Taobao	AVG
RNN Optuna	82.09± 0.92	**63.85± 0.41**	80.64± 0.13	**91.29± 0.20**	**69.26± 0.19**	**77.43**
RNN GraP (Our)	**82.21± 0.55**	63.83± 0.63	**80.72± 0.03**	90.86± 0.21	69.19± 0.51	77.36
Transformer Optuna	83.27± 0.62	63.93± 0.58	78.44± 0.14	**90.79± 0.07**	68.94± 0.28	77.07
Transformer GraP (Our)	**83.83± 0.90**	**64.18± 0.35**	**78.62± 0.06**	90.68± 0.38	**69.70± 0.28**	**77.40**
HT-Transformer Optuna	84.57± 0.21	62.21± 0.61	**81.09± 0.06**	**91.76± 0.07**	70.25± 0.04	77.97
HT-Transformer GraP (Our)	**85.07± 0.39**	**62.31± 0.76**	80.93± 0.13	91.69± 0.09	**70.78± 0.20**	**78.16**

Table 2: Event-sequence results without contrastive loss. We report the mean and standard deviation across 4 seeds. The best method for each dataset-backbone combination is shown in bold. "Optuna" denotes weights obtained via Bayesian optimization.

Method	CIFAR-10	CIFAR-100	ImageNet-100	AVG
All4One (default)	93.08± 0.12	71.48± 0.16	85.92± 0.03	**83.49**
All4One Equal Weights	92.70± 0.16	71.45± 0.14	**86.08± 0.10**	83.41
All4One W/O Attn-NNCLR	93.11± 0.10	71.32± 0.47	85.65± 0.06	83.36
All4One GraP	**93.16± 0.09**	**71.51± 0.35**	85.81± 0.08	**83.49**

Table 3: Image classification results. The mean and standard deviation across 4 seeds are reported. The best method for each dataset is shown in bold.

4.2 Computer vision

Experiment design. We evaluate our method in the computer vision domain using the `solo-learn` (Da Costa et al., 2022), a standardized framework for self-supervised representation learning. We consider experiments on *CIFAR-10*, *CIFAR-100* (Krizhevsky et al., 2009), and *ImageNet-100*, a subset of ImageNet (Krizhevsky et al., 2012). We apply our approach to *All4One* (Estepa et al., 2023), a recent self-supervised method that combines multiple objectives inspired by Barlow Twins (Zbontar et al., 2021), BYOL (Grill et al., 2020), and NNCLR (Dwibedi et al., 2021). We evaluate model quality following the benchmark protocol by online training a linear classifier on top of detached representations and reporting validation top-1 accuracy.

Results. Tab. 3 summarizes the results. The default and equal-weight configurations perform similarly across the three datasets, suggesting that All4One is relatively insensitive to moderate changes in its loss weights. Accordingly, GraP performs comparably to the tuned baseline on average, with slightly better results on the CIFAR datasets and slightly worse performance on ImageNet-100.

Interestingly, the weights learned by GraP reveal consistent suppression of the Attn-NNCLR objective, whose weight approaches zero during training (App. H). To investigate this behavior, we remove Attn-NNCLR from the pretraining objective (*All4One W/O Attn-NNCLR* in Tab. 3) and observe performance close to that of the default model. This experiment provides evidence that Attn-NNCLR makes a limited contribution in the evaluated setting and, more broadly, demonstrates how the weights learned by GraP can expose low-impact components of a composite pretraining objective.

4.3 Computational Cost and Memory Usage

Computation speed. Tab. 4 compares the computational cost of a single training run, GraP, and classical hyperparameter optimization methods for RNN-based models on event-sequence datasets. Across datasets, our method introduces a moderate overhead compared to standard training, with an average slowdown of 44% in throughput and 34% in epoch time. The exact overhead depends on the dataset and model configuration (e.g., 79% overhead on AlfaBattle due to its larger number of loss terms), but remains within a constant factor (Sec. 3.3). In contrast, Bayesian optimization and TPE typically require dozens of independent training runs (Snoek et al., 2012; Bergstra et al., 2013), resulting in orders-of-magnitude higher computational cost. Under a conservative estimate of 10 runs per parameter, the effective overhead ranges from 4000% to 19000%. Thus, the proposed method achieves competitive performance while requiring only a single training run with moderate additional computation.

Metric	Method	Churn	AgePred	AlfaBattle	MIMIC-III	Taobao	AVG
RPS (records per sec.)	Baseline	2062	110	152	68	94	
	GraP	1266	82	86	54	79	
	Overhead	*63%*	*34%*	*79%*	*26%*	*19%*	*44%*
Epoch time (s)	Baseline	1.64	35.7	552.2	58.7	31.5	
	GraP	2.16	46.6	946.8	70.2	36.9	
	Overhead	*32%*	*31%*	*71%*	*20%*	*17%*	*34%*
Estimated Bayesian / TPE cost		*7000%*	*4000%*	*19000%*	*4000%*	*4000%*	*7600%*

Table 4: RNN training and hyperparameter tuning speed comparison.

Memory usage. Our approach does not require multiple model replicas or optimization states across runs. The additional memory overhead comes from storing per-loss gradients at the embedding level,

which is typically small compared to the backbone size in large transformer models. However, for smaller sequential models, the relative overhead can be more noticeable. For example, memory usage increases by a factor of 2.8 on AgePred and 3.7 on AlfaBattle compared to standard training.

5 Related Work

Classical hyperparameter optimization. Classical HPO treats training as a black-box procedure and searches over configurations using grid search, random search, Bayesian optimization, or related model-based methods (Snoek et al., 2012; Bergstra et al., 2013). While effective, these approaches typically require many independent training runs, making them expensive for large-scale pretraining with multiple loss terms.

Differentiable hyperparameter optimization. Gradient-based HPO differentiates validation performance with respect to hyperparameters such as learning rates, initialization parameters, and regularization strengths (Maclaurin et al., 2015; Fu et al., 2016; Franceschi et al., 2018). Subsequent work reduced the computational cost using truncated backpropagation or implicit differentiation (Luketina et al., 2016; Lorraine et al., 2020). In particular, Luketina et al. (2016) proposed efficient online updates of continuous hyperparameters based on their effect on validation performance. We adapt this perspective to multi-objective pretraining, addressing the resulting gradient-norm ambiguity and exploiting shared representations to avoid redundant backward passes through the backbone.

Multi-task and multi-objective learning. Learning with multiple objectives is central to multi-task learning, where task gradients may conflict and simple scalarization can lead to suboptimal trade-offs (Sener & Koltun, 2018; Quinton & Rey, 2024). Methods such as DWA, GradNorm, MGDA, and PCGrad adapt loss weights or manipulate task gradients using loss dynamics, gradient magnitudes, Pareto-oriented aggregation, or gradient-conflict mitigation (Liu et al., 2019; Chen et al., 2018; Sener & Koltun, 2018; Yu et al., 2020). Other works study uncertainty-based and scalable loss weighting (Kendall et al., 2018; Li et al., 2024; Gregoire et al., 2024; Xiao et al., 2025), as well as Pareto-stationary gradient-combination methods (Senushkin et al., 2023; Quinton & Rey, 2024). These approaches generally target settings in which all tasks are simultaneously of interest, whereas our goal is to tune pretraining loss weights for one designated downstream task.

Auxiliary learning. Auxiliary-learning methods explicitly use a main task to determine the contribution of auxiliary objectives. Gradient-similarity methods select or weight auxiliary gradients according to their alignment with the main task (Du et al., 2018). Particularly related to GraP, OL-AUX (Lin et al., 2019) learns auxiliary-task weights online from their effect on the main-task objective, using a first-order approximation to avoid differentiating through the full optimization trajectory. AuxiLearn (Navon et al., 2021) instead formulates auxiliary learning as a bilevel problem and uses implicit differentiation to learn how auxiliary losses are combined. GraP builds on the first-order perspective, addressing the normalization ambiguity and introducing shared-representation gradients and gradient reuse to reduce the computational cost of loss-weight optimization.

6 Conclusion

In this paper, we studied loss-weight selection for composite pretraining objectives using a bilevel formulation that targets downstream performance. We introduced GraP, a gradient-based method that aligns the composite pretraining update with a downstream gradient, normalizes the resulting update rather than the weights, and reduces computational cost through shared-representation gradients and gradient reuse.

Empirically, GraP improves average performance over adaptive-weighting baselines and matches or improves upon carefully tuned hyperparameters across event-sequence modeling tasks, while requiring only a single training run with moderate overhead. In self-supervised vision, where objectives are typically better balanced and involve fewer components, the approach remains competitive with standard tuning strategies.

Overall, the results suggest that gradient-based loss-weight optimization is an efficient alternative to classical multi-task learning and search-based hyperparameter optimization methods, particularly in settings with many interacting loss terms or limited computational budgets.

REPRODUCIBILITY STATEMENT

We provide the source code for the event-sequence and computer vision experiments as supplementary material. The code includes all necessary configuration files with the exact hyperparameters used to reproduce the reported results.

AI USE STATEMENT

Large language models were used for language editing and text polishing, including improving clarity, grammar, and presentation. AI-assisted coding tools were also used for code editing and debugging. The authors reviewed and tested all AI-assisted code. The research ideas, methodology, experimental design, analysis, and conclusions were developed by the authors. The authors take responsibility for the final content of the paper and the correctness of the accompanying code.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Dmitrii Babaev, Nikita Ovsov, Ivan Kireev, Maria Ivanova, Gleb Gusev, Ivan Nazarov, and Alexander Tuzhilin. Coles: Contrastive learning for event sequences with self-supervision. In *Proceedings of the 2022 International Conference on Management of Data*, pp. 1190–1199, 2022.

James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pp. 115–123. PMLR, 2013.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the International Conference on Machine Learning*, 2018.

Victor Guilherme Turrisi Da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. solo-learn: A library of self-supervised methods for visual representation learning. *Journal of Machine Learning Research*, 23(56):1–6, 2022.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Mehrdad Farajtabar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9588–9597, 2021.

Imanol G Estepa, Ignacio Sarasúa, Bhalaji Nagarajan, and Petia Radeva. All4one: Symbiotic neighbour contrastive learning via self-attention and redundancy reduction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 16243–16253, 2023.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, 2018.

Jie Fu, Hongyin Luo, Jiashi Feng, Kian Hsiang Low, and Tat-Seng Chua. Drmad: Distilling reverse-mode automatic differentiation for optimizing hyperparameters of deep neural networks. *arXiv preprint arXiv:1601.00917*, 2016.

Emilie Gregoire, Muhammad Hafeez Chaudhary, and Sam Verboven. Sample-level weighting for multi-task learning with auxiliary tasks: E. grégoire et al. *Applied Intelligence*, 54(4):3482–3501, 2024.

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.

Ivan Karpukhin and Andrey Savchenko. Ht-transformer: Event sequences classification by accumulating prefix information with history tokens. *arXiv preprint arXiv:2508.01474*, 2025.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *The Twelfth International Conference on Learning Representations*, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Yuanze Li, Chun-Mei Feng, Qilong Wang, Guanglei Yang, and Wangmeng Zuo. Unprejudiced training auxiliary tasks makes primary better: A multitask learning perspective. *IEEE Transactions on Neural Networks and Learning Systems*, 36(7):12091–12105, 2024.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations*, 2016.

Xingyu Lin, Harjatin Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1871–1880, 2019.

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.

Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International conference on machine learning*, pp. 2952–2960. PMLR, 2016.

Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, 2015.

Aviv Navon, Idan Achituve, Haggai Maron, Gal Chechik, and Ethan Fetaya. Auxiliary learning by implicit differentiation. In *International Conference on Learning Representations*, 2021.

Inkit Padhi, Yair Schiff, Igor Melnyk, Mattia Rigotti, Youssef Mroueh, Pierre Dognin, Jerret Ross, Ravi Nair, and Erik Altman. Tabular transformers for modeling multivariate time series. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3565–3569. IEEE, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Pierre Quinton and Valérian Rey. Jacobian descent for multi-objective optimization. *arXiv preprint arXiv:2406.16232*, 2024.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL https://jmlr.org/papers/v22/20-1364.html.

Artem Sakhno, Ivan Kireev, Dmitrii Babaev, Maxim Savchenko, Gleb Gusev, and Andrey Savchenko. Pytorch-lifestream: Learning embeddings on discrete event sequences. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 11104–11108, 2025.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Dmitry Senushkin, Nikolay Patakin, Arseny Kuznetsov, and Anton Konushin. Independent component alignment for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20083–20093, 2023.

Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. In *The 22nd international conference on artificial intelligence and statistics*, pp. 1723–1732. PMLR, 2019.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Peiyao Xiao, Chaosheng Dong, Shaofeng Zou, and Kaiyi Ji. Ldc-mtl: Balancing multi-task learning through scalable loss discrepancy control. *arXiv preprint arXiv:2502.08585*, 2025.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pp. 12310–12320. PMLR, 2021.

A MULTI-STEP SGD ROLLOUT

In Sec. 3.1, we derived the loss-weight update using a single-step SGD approximation. We now extend this analysis to a multi-step rollout and show that, to first order, the required hypergradient depends on the accumulated pretraining gradients along the trajectory.

Setup. Recall the composite pretraining loss

$$L_{\mathrm{pre}}(\theta, \mathbf{w}) = \sum_{k=1}^{K} w_k L_k(\theta),$$

and consider an SGD trajectory of $n+1$ steps:

$$\theta_{t+1} = \theta_t - \eta \sum_{k=1}^{K} w_k \nabla_\theta L_k(\theta_t), \qquad t = 0, \dots, n.$$

The weights $\mathbf{w}$ are held fixed during the rollout. We are interested in the hypergradient of the downstream loss $L_{\mathrm{down}}(\theta_{n+1})$ with respect to w_i.

Theorem A.1 (First-order multi-step hypergradient)**.** *Assume that each L_k is twice continuously differentiable and that L_{down} is continuously differentiable. Assume further that, along the rollout, there exist constants $G, H, D < \infty$ such that*

$$\|\nabla_\theta L_i(\theta_t)\|_2 \le G, \qquad \left\| \sum_{k=1}^{K} w_k \nabla_\theta^2 L_k(\theta_t) \right\|_{\mathrm{op}} \le H,$$

and

$$\|\nabla_\theta L_{\mathrm{down}}(\theta_{n+1})\|_2 \le D.$$

For a fixed rollout length n and sufficiently small step size η, the hypergradient satisfies

$$\frac{\partial L_{\mathrm{down}}(\theta_{n+1})}{\partial w_i} = -\eta \nabla_\theta L_{\mathrm{down}}(\theta_{n+1})^\top \sum_{t=0}^{n} \nabla_\theta L_i(\theta_t) + O(n^2\eta^2).$$

Proof. Let

$$\Omega_i^t := \frac{\partial \theta_t}{\partial w_i}$$

denote the sensitivity of the parameters to the weight w_i. Since θ_0 does not depend on w_i, we have $\Omega_i^0 = 0$.

Differentiating the SGD update yields

$$\Omega_i^{t+1} = \Omega_i^t - \eta \nabla_\theta L_i(\theta_t) - \eta \sum_{k=1}^{K} w_k \nabla_\theta^2 L_k(\theta_t) \Omega_i^t.$$

Using the assumptions of the theorem,

$$\|\Omega_i^{t+1}\|_2 \le (1 + \eta H)\|\Omega_i^t\|_2 + \eta G.$$

Unrolling this inequality from $\Omega_i^0 = 0$ gives

$$\|\Omega_i^t\|_2 \le \eta G \sum_{s=0}^{t-1} (1 + \eta H)^s = O(t\eta).$$

Thus, for fixed n,

$$\|\Omega_i^t\|_2 = O(n\eta).$$

Substituting this bound into the Hessian-dependent term gives

$$\left\| \eta \sum_{k=1}^{K} w_k \nabla_\theta^2 L_k(\theta_t) \Omega_i^t \right\|_2 \le \eta H \|\Omega_i^t\|_2 = O(n\eta^2).$$

Therefore, the recursion becomes

$$\Omega_i^{t+1} = \Omega_i^t - \eta \nabla_\theta L_i(\theta_t) + O(n\eta^2).$$

Unrolling from $t = 0$ to n gives

$$\Omega_i^{n+1} = -\eta \sum_{t=0}^{n} \nabla_\theta L_i(\theta_t) + O(n^2\eta^2).$$

Finally, applying the chain rule,

$$\frac{\partial L_{\text{down}}(\theta_{n+1})}{\partial w_i} = \nabla_\theta L_{\text{down}}(\theta_{n+1})^\top \Omega_i^{n+1}.$$

Since the downstream gradient is bounded, multiplying the remainder by this gradient preserves its order, which yields the desired result. □

Discussion. Theorem A.1 shows that, up to first order in η, the hypergradient is determined by the alignment between the downstream gradient at the end of the rollout and the *accumulated* gradients of each pretraining loss along the trajectory.

Importantly, all steps contribute at order $O(\eta)$, while the deviation from this accumulation arises from Hessian-dependent transport terms, which are of higher order $O(n^2\eta^2)$. Thus, the one-step approximation used in Sec. 3.1 corresponds to the special case $n = 0$.

In practice, the accumulated gradients can be approximated using an exponential moving average:

$$m_i^{t+1} = \beta m_i^t + (1 - \beta)\nabla_\theta L_i(\theta_t),$$

and the alignment score becomes

$$\nabla_\theta L_{\text{down}}(\theta_t)^\top m_i^t.$$

In preliminary experiments, we did not observe improvements from using exponential moving average gradients. This is likely because the weight-vector trajectory under SGD is already smooth, with the degree of smoothing effectively controlled by the learning rate. Moreover, maintaining EMA gradients introduces additional memory overhead and is not compatible with the proposed upper-bound approximation.

B Composite Gradient Normalization

The unnormalized alignment objective is positively homogeneous in the loss weights: scaling $\mathbf{w}$ by a positive constant scales both the objective value and the magnitude of the composite gradient without changing the relative contributions of the losses. This creates an ambiguity in the update scale. To remove this ambiguity, we normalize the weighted pretraining gradient by its norm, which yields the following objective:

$$\underset{\mathbf{w}\in\mathbb{R}^K,\, w_k\geq 0}{\arg\max} \frac{\mathbf{w}^\top G g_{\text{down}}}{\|\mathbf{w}^\top G\|}. \tag{21}$$

Because the objective is invariant under positive rescaling of $\mathbf{w}$, we may restrict the feasible set to $\|\mathbf{w}\|_2 = 1$. Moreover, division by the constant $\|g_{\text{down}}\|_2$ does not change the optimizer. The resulting objective is therefore equivalent to maximizing the cosine similarity between the composite and downstream gradients:

$$\underset{\mathbf{w}\in\mathbb{R}^K,\, w_k\geq 0,\, \|\mathbf{w}\|=1}{\arg\max} \cos\left(\mathbf{w}^\top G,\, g_{\text{down}}\right). \tag{22}$$

The following proposition shows that the optimum corresponds to directional alignment between the composite pretraining gradient and the downstream gradient.

Proposition 1. *Assume that there exists* $\mathbf{w} \in \mathbb{R}_+^K$ *such that*

$$\mathbf{w}^\top G = \alpha g_{\mathrm{down}}$$

for some $\alpha > 0$*. Then the objective*

$$\max_{\substack{\mathbf{w}\in\mathbb{R}_+^K \\ \mathbf{w}^\top G \neq 0}} \frac{\mathbf{w}^\top G g_{\mathrm{down}}}{\|\mathbf{w}^\top G\|_2}$$

is maximized when the composite pretraining gradient is aligned with g_{down}.

Proof. By the Cauchy–Bunyakovsky–Schwarz inequality,

$$\frac{\mathbf{w}^\top G g_{\mathrm{down}}}{\|\mathbf{w}^\top G\|} \leq \|g_{\mathrm{down}}\|, \tag{23}$$

with equality if and only if

$$\mathbf{w}^\top G = \alpha g_{\mathrm{down}}$$

for some $\alpha > 0$. □

Thus, in the ideal case, the normalized objective recovers a composite pretraining gradient whose direction coincides with the downstream gradient. This motivates composite-gradient normalization when updating the loss weights $\mathbf{w}$.

C Embedding-Space Approximation

In Sec. 3.3, we replace parameter-space gradient comparisons with comparisons in the shared representation space. This section explains the resulting approximation. We first express the normalized alignment objective as a distance minimization problem and then show that the shared-space distance controls the corresponding mismatch after propagating the normalized shared gradient through the backbone.

Setup. We consider a fixed training iteration and treat the minibatch, model parameters, and loss gradients as fixed while updating the loss weights. Let

$$z = g(\mathcal{B}, \theta)$$

denote the flattened shared representation of the minibatch, and assume that $g(\mathcal{B}, \cdot)$ is differentiable at the current parameters θ. Let

$$J_\theta = \frac{\partial z}{\partial \theta}$$

be the corresponding backbone Jacobian.

For each pretraining objective, define its gradient with respect to the shared representation as

$$\tilde{g}_k = \nabla_z \mathcal{L}_k(h_k(z)), \qquad k = 1, \ldots, K. \tag{24}$$

The downstream loss may be evaluated only on the labeled subset of the minibatch. We view its gradient as a vector in the full representation space by inserting zeros at positions corresponding to unlabeled examples:

$$\tilde{g}_{\mathrm{down}} = \nabla_z \mathcal{L}_{\mathrm{down}}(h_{\mathrm{down}}(z)). \tag{25}$$

Let $\tilde{G}$ contain $\tilde{g}_k^\top$ as its k-th row, and define the composite shared-space gradient

$$\tilde{g}_{\mathrm{pre}}(\mathbf{w}) = \tilde{G}^\top \mathbf{w} = \sum_{k=1}^{K} w_k \tilde{g}_k. \tag{26}$$

By the chain rule, the corresponding backbone gradients are

$$g_{\mathrm{pre}}(\mathbf{w}) = G^\top \mathbf{w} = J_\theta^\top \tilde{g}_{\mathrm{pre}}(\mathbf{w}), \qquad g_{\mathrm{down}} = J_\theta^\top \tilde{g}_{\mathrm{down}}. \tag{27}$$

Distance interpretation of normalized alignment. Recall the normalized parameter-space objective introduced in Sec. 3.2:

$$\arg\max_{\mathbf{w}\in\mathbb{R}_+^K,\, g_{\mathrm{pre}}(\mathbf{w})\neq 0} \frac{\langle g_{\mathrm{pre}}(\mathbf{w}), g_{\mathrm{down}}\rangle}{\|g_{\mathrm{pre}}(\mathbf{w})\|_2}. \tag{28}$$

For every feasible $\mathbf{w}$,

$$\left\| \frac{g_{\mathrm{pre}}(\mathbf{w})}{\|g_{\mathrm{pre}}(\mathbf{w})\|_2} - g_{\mathrm{down}} \right\|_2^2 = 1 + \|g_{\mathrm{down}}\|_2^2 - 2\frac{\langle g_{\mathrm{pre}}(\mathbf{w}), g_{\mathrm{down}}\rangle}{\|g_{\mathrm{pre}}(\mathbf{w})\|_2}. \tag{29}$$

Because the first two terms do not depend on $\mathbf{w}$, maximizing Eq. 28 is exactly equivalent to minimizing the distance in Eq. 29.

Computing this objective directly requires the individual parameter-space gradients of all K pretraining losses. Following prior work that compares task gradients at a shared representation (Sener & Koltun, 2018; Chen et al., 2018; Senushkin et al., 2023), GraP instead optimizes

$$\arg\max_{\mathbf{w}\in\mathbb{R}_+^K,\, \tilde{g}_{\mathrm{pre}}(\mathbf{w})\neq 0} \frac{\langle \tilde{g}_{\mathrm{pre}}(\mathbf{w}), \tilde{g}_{\mathrm{down}}\rangle}{\|\tilde{g}_{\mathrm{pre}}(\mathbf{w})\|_2}. \tag{30}$$

The same identity gives

$$\left\| \frac{\tilde{g}_{\mathrm{pre}}(\mathbf{w})}{\|\tilde{g}_{\mathrm{pre}}(\mathbf{w})\|_2} - \tilde{g}_{\mathrm{down}} \right\|_2^2 = 1 + \|\tilde{g}_{\mathrm{down}}\|_2^2 - 2\frac{\langle \tilde{g}_{\mathrm{pre}}(\mathbf{w}), \tilde{g}_{\mathrm{down}}\rangle}{\|\tilde{g}_{\mathrm{pre}}(\mathbf{w})\|_2}. \tag{31}$$

Therefore, the GraP objective exactly minimizes the distance between the embedding-normalized composite gradient and the downstream gradient in the shared representation space.

Error bound for shared-space normalization. GraP first normalizes the composite gradient in the shared space and then propagates it through the backbone. The following result shows that the shared-space distance in Eq. 31 controls the mismatch between the corresponding vectors after this propagation.

Theorem C.1 (Shared-space normalization error bound)**.** *For any* $\mathbf{w}$ *such that* $\tilde{g}_{\mathrm{pre}}(\mathbf{w}) \neq 0$*,*

$$\left\| J_\theta^\top \frac{\tilde{g}_{\mathrm{pre}}(\mathbf{w})}{\|\tilde{g}_{\mathrm{pre}}(\mathbf{w})\|_2} - g_{\mathrm{down}} \right\|_2 \leq \left\| J_\theta^\top \right\|_{\mathrm{op}} \left\| \frac{\tilde{g}_{\mathrm{pre}}(\mathbf{w})}{\|\tilde{g}_{\mathrm{pre}}(\mathbf{w})\|_2} - \tilde{g}_{\mathrm{down}} \right\|_2, \tag{32}$$

where

$$\left\| J_\theta^\top \right\|_{\mathrm{op}} = \sup_{\|v\|_2=1} \left\| J_\theta^\top v \right\|_2$$

is the induced Euclidean operator norm of the Jacobian transpose.

Proof. Using $g_{\mathrm{down}} = J_\theta^\top \tilde{g}_{\mathrm{down}}$ from Eq. 27, we obtain

$$\begin{aligned} &\left\| J_\theta^\top \frac{\tilde{g}_{\mathrm{pre}}(\mathbf{w})}{\|\tilde{g}_{\mathrm{pre}}(\mathbf{w})\|_2} - g_{\mathrm{down}} \right\|_2 \\ &= \left\| J_\theta^\top \left(\frac{\tilde{g}_{\mathrm{pre}}(\mathbf{w})}{\|\tilde{g}_{\mathrm{pre}}(\mathbf{w})\|_2} - \tilde{g}_{\mathrm{down}} \right) \right\|_2 \\ &\leq \left\| J_\theta^\top \right\|_{\mathrm{op}} \left\| \frac{\tilde{g}_{\mathrm{pre}}(\mathbf{w})}{\|\tilde{g}_{\mathrm{pre}}(\mathbf{w})\|_2} - \tilde{g}_{\mathrm{down}} \right\|_2, \end{aligned} \tag{33}$$

where the inequality follows directly from the definition of the induced operator norm. □

Theorem C.1 shows that a small mismatch under the GraP shared-space objective implies a small mismatch after the embedding-normalized composite gradient and the downstream gradient are mapped through the backbone Jacobian, up to a Jacobian-dependent factor. Since the Jacobian is fixed with respect to $\mathbf{w}$ within one weight-update iteration, GraP minimizes an efficiently computable upper bound on this propagated mismatch.

This is a one-sided guarantee. In general, minimizing the shared-space distance need not yield exactly the same weights as minimizing the corresponding parameter-space distance, because the Jacobian may stretch different shared-space directions by different amounts. Nevertheless, the result directly justifies the shared-space approximation without requiring the Jacobian to preserve angles or normalized directions.

D Training Details

Here we summarize the main hyperparameters and training details. Additional information is available in the configuration files provided with the source code.

Event Sequences. Tab. 5 presents an overview of the datasets.

Dataset	# Seq.	Fields / # Losses	Mean length	Downstream		
				Target	Classes	Metric
Churn	10217	6	99.3	Churn	2	ROC AUC
AgePred	50000	3	875	Age group	4	Accuracy
AlfaBattle	1466527	18	234	Default	2	ROC AUC
MIMIC-III	52103	3	407	Mortality	2	ROC AUC
Taobao	9904	3	527	Activity	2	ROC AUC

Table 5: Event-sequence dataset statistics

We follow the training setup from the HT-Transformer paper (Karpukhin & Savchenko, 2025). We train all models using the Adam optimizer (Kingma & Ba, 2015) with a fixed learning rate of 0.001. Depending on the dataset, training runs for up to 30–120 epochs with early stopping based on validation performance. We run experiments on NVIDIA A100 GPUs: two GPUs for all datasets except AlfaBattle, which uses four GPUs. Dataset-specific hyperparameters are listed in Tab. 6.

Dataset	Epochs	Hidden layer		Transf.	Positional encoding	
		RNN	Transf.	layers	m	M
Churn	120	256	256	4	4.2	6500
AgePred	45	1536	512	8	0.003	8
AlfaBattle	60	1024	512	10	0.2	500
MIMIC-III	30	1280	512	8	1	4000
Taobao	30	1280	512	8	0.008	30

Table 6: Dataset-specific parameters.

Computer vision. For the computer vision experiments, we use the solo-learn benchmark (Da Costa et al., 2022). We train all models with the LARS optimizer (You et al., 2017). We train for 1000 epochs on CIFAR-10 and CIFAR-100, and for 400 epochs on ImageNet-100. We train on a single A100 GPU for the CIFAR datasets and on two A100 GPUs for ImageNet-100.

E Additional Computer Vision Results

The complete set of computer vision experiments is reported in Tab. 7. Overall, the methods differ only slightly in average performance. We therefore conclude that loss balancing has a limited impact on the All4One method from the *solo-learn* benchmark. Notably, removing the Attn-NNCLR loss, which GraP assigns a low weight to (App. H), has little effect on performance across the CIFAR datasets.

Method	CIFAR-10	CIFAR-100	ImageNet-100	AVG
All4One (default)	93.08± 0.12	71.48± 0.16	85.92± 0.03	83.49
All4One W/O Attn-NNCLR	93.11± 0.10	71.32± 0.47	85.65± 0.06	83.36
All4One Equal Weights	92.70± 0.16	71.45± 0.14	86.08± 0.10	83.41
All4One GradNorm	93.01± 0.04	**71.81± 0.23**	85.81± 0.07	**83.54**
All4One DWA	92.72± 0.11	71.44± 0.34	**86.31± 0.05**	83.49
All4One MGDA	92.81± 0.11	69.98± 0.19	83.79± 0.03	82.19
All4One PCGrad	92.69± 0.13	71.61± 0.24	86.27± 0.07	83.53
All4One GraP	**93.16± 0.09**	71.51± 0.35	85.81± 0.08	83.49

Table 7: Image classification results. The mean and standard deviation across 4 seeds are reported. The best method for each dataset is shown in bold.

F Ablation studies

On potential label leakage. Our method updates the loss weights using labeled data, raising the question of whether this introduces label leakage and effectively makes backbone training partially supervised. We address this concern in several ways. First, the weights influence the backbone only indirectly through the composite gradient. The backbone and weight updates are computed simultaneously, while the weights evolve gradually, typically following smooth and monotonic trajectories (see App. H). This limits rapid batch-specific adaptation and the amount of label-specific information that can be transferred through the weights. Second, we evaluate how the amount of labeled data used for weight tuning affects performance on the Churn and MIMIC-III datasets. Tab. 8 shows that the method is robust to the fraction of labeled samples, suggesting that labels primarily provide a weak guidance signal for weight selection rather than directly supervising representation learning.

Labeled data	Churn	MIMIC-III
5%	83.61± 0.48	91.36± 0.02
10%	83.32± 0.12	91.74± 0.04
20%	84.80± 0.15	91.58± 0.14
50%	84.00± 0.70	91.58± 0.06
100%	84.45± 0.55	91.50± 0.05

Table 8: GraP classification results on Churn and MIMIC-III for different fractions of labeled data.

Finally, the supervised fine-tuning (SFT) results in Tab. 9 demonstrate that supervision still provides additional information not captured during pretraining. In this sense, the proposed method is comparable to standard hyperparameter optimization in terms of label supervision.

Method	Churn	AgePred	AlfaBattle	MIMIC-III	Taobao	AVG
Supervised	79.43± 0.64	61.48± 0.29	78.87± 0.18	91.56± 0.24	69.57± 0.20	76.18
Auxiliary	84.14± 0.46	64.28± 0.24	78.81± 0.07	91.26± 0.15	70.49± 0.33	77.80
Equal Weights	83.79± 0.79	63.76± 0.41	78.57± 0.00	91.25± 0.05	70.20± 0.49	77.51
Equal Weights + SFT	83.44± 0.46	64.32± 0.70	79.60± 0.04	**92.23± 0.18**	70.40± 0.24	78.00
GraP No norm.	**84.26± 0.55**	63.86± 0.41	78.39± 0.05	91.27± 0.11	69.65± 0.14	77.48
GraP Sum norm.	83.80± 0.58	63.43± 0.61	79.12± 0.12	91.25± 0.20	**70.78± 0.32**	77.67
GraP	83.54± 0.71	64.31± 0.08	79.82± 0.07	91.44± 0.05	70.56± 0.14	77.94
GraP + SFT	83.09± 0.81	**64.78± 0.55**	**80.66± 0.19**	92.05± 0.06	70.40± 0.30	**78.20**

Table 9: Comparison with auxiliary training and different weight normalization strategies.

Weights normalization. In Sec. 3.2, we introduced a normalization strategy based on the norm of the weighted composite gradient. Tab. 9 compares the proposed approach with standard weight-sum normalization (*Sum norm.*) (Sener & Koltun, 2018), and a variant with unconstrained weights (*No norm.*). The proposed normalization improves average performance across datasets, with the largest gains observed in settings with relatively low variance across runs.

Auxiliary training. In this experiment, we compare the standard pipeline of unsupervised pretraining followed by supervised fine-tuning (SFT) with auxiliary training, in which the supervised and

pretraining objectives are optimized jointly. Tab. 9 shows that auxiliary training improves upon the equal-weight pretraining baseline. However, pretraining followed by SFT achieves higher average performance, possibly because separating the two stages reduces overfitting to the limited labeled data. GraP slightly outperforms auxiliary training even without fine-tuning, while applying SFT yields the best overall performance.

G Reinforcement Learning Experiments

We additionally evaluate GraP in a reinforcement learning setting following the VisualFetchReach experimental design of OL-AUX (Lin et al., 2019). Our implementation follows the publicly available OL-AUX codebase,[1] but is reimplemented in PyTorch (Paszke et al., 2019) using Stable-Baselines3 (Raffin et al., 2021). We use DDPG (Lillicrap et al., 2016) with Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) and consider the five auxiliary tasks used by OL-AUX: autoencoding, egomotion prediction, forward dynamics, inverse dynamics, and optical-flow prediction. We compare three variants: standard DDPG, OL-AUX, and the proposed GraP method. All methods are evaluated using two random seeds, and the curves report the mean with standard deviation across seeds.

Fig. 4 shows the evaluation success rate on VisualFetchReach. Both auxiliary-task weighting approaches substantially accelerate learning compared with the DDPG baseline. GraP reaches a high success rate earlier in training and subsequently maintains performance close to 1.0; OL-AUX exhibits similar final performance but somewhat slower initial learning. In contrast, DDPG learns considerably more slowly over the same training budget.

To further examine the behavior of the adaptive weighting methods, Fig. 5 shows the evolution of the five auxiliary-task weights. OL-AUX assigns increasingly large weights to several auxiliary objectives, with substantial variation across seeds, whereas the weights produced by GraP remain small and comparatively consistent across runs. Despite this difference in weighting dynamics, GraP achieves competitive or faster learning in terms of downstream success rate. These results further indicate that GraP can be applied beyond supervised pretraining to adapt auxiliary objectives during reinforcement learning.

H Example Weights Trajectories

Below we present the plots of the weight trajectories during training:

- Churn: Fig. 6,
- AgePred: Fig. 7,
- AlfaBattle: Fig. 8,
- MIMIC-III: Fig. 9,
- Taobao: Fig. 10,
- CIFAR-10: Fig. 11,
- CIFAR-100: Fig. 12,
- ImageNet-100: Fig. 13.

[1] https://github.com/Xingyu-Lin/auxiliary-tasks-rl

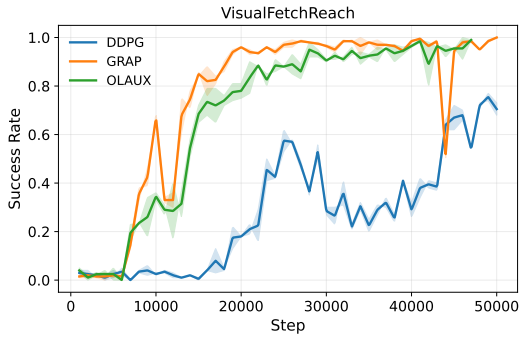


Figure 4: Evaluation success rate on VisualFetchReach. Curves show the mean over two seeds, with shaded regions indicating one standard deviation.

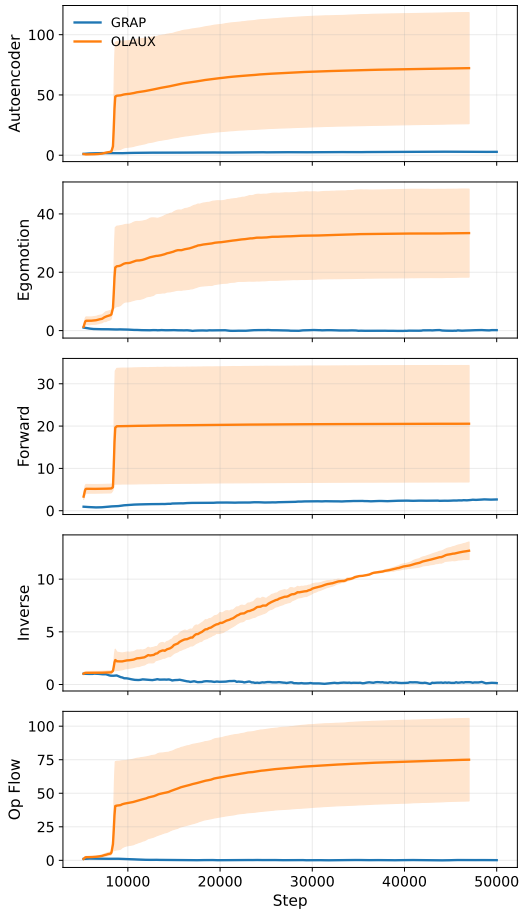


Figure 5: Evolution of the auxiliary-task weights on VisualFetchReach for GraP and OL-AUX. From top to bottom: autoencoder, egomotion, forward dynamics, inverse dynamics, and optical flow. Curves show the mean over two seeds, with shaded regions indicating one standard deviation.

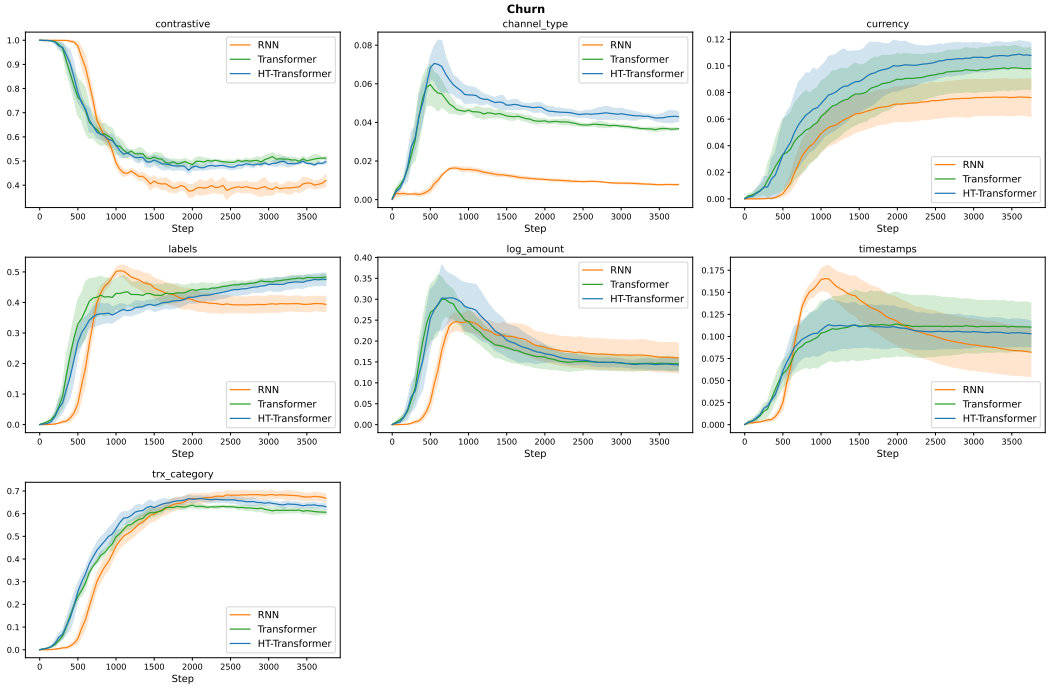


Figure 6: Weights' trajectories for the Churn dataset.

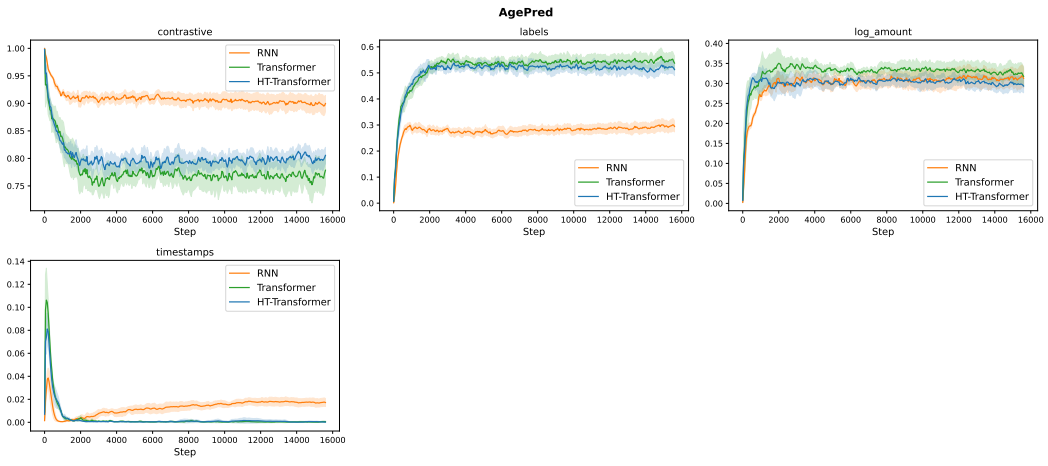


Figure 7: Weights' trajectories for the AgePred dataset.

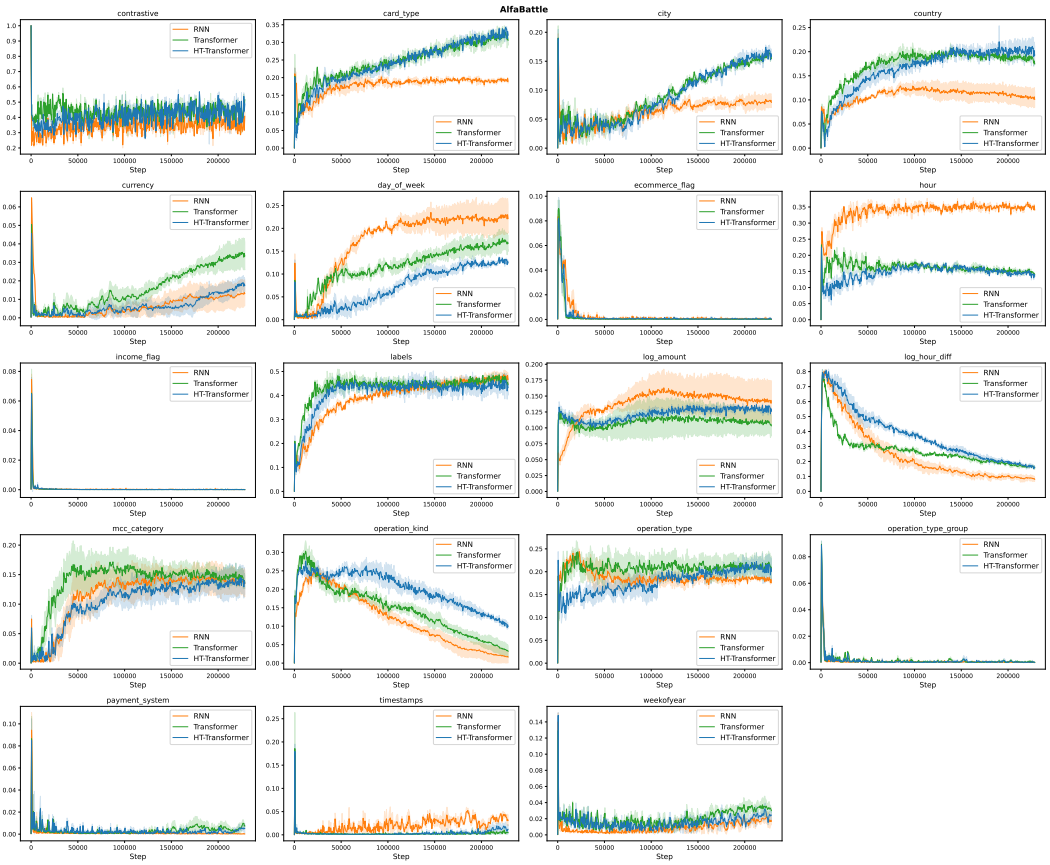


Figure 8: Weights' trajectories for the AlfaBattle dataset.

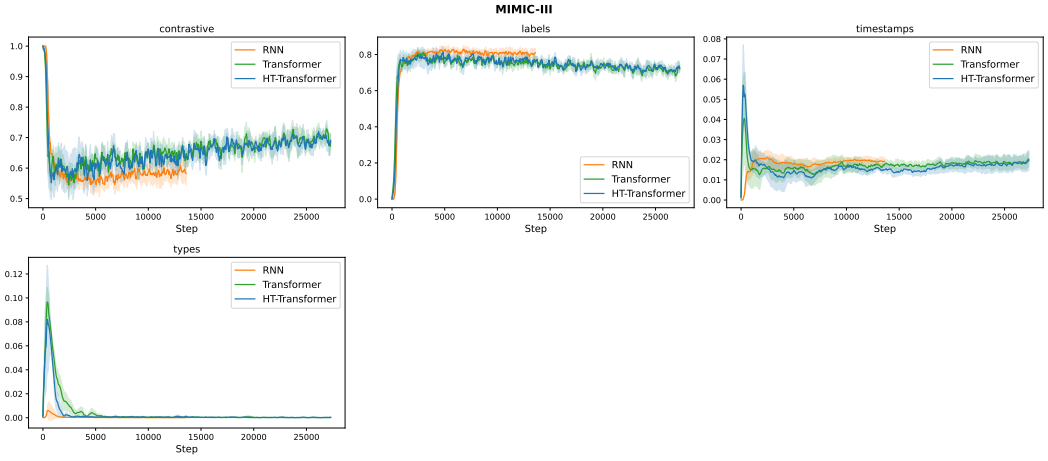


Figure 9: Weights' trajectories for the MIMIC-III dataset.

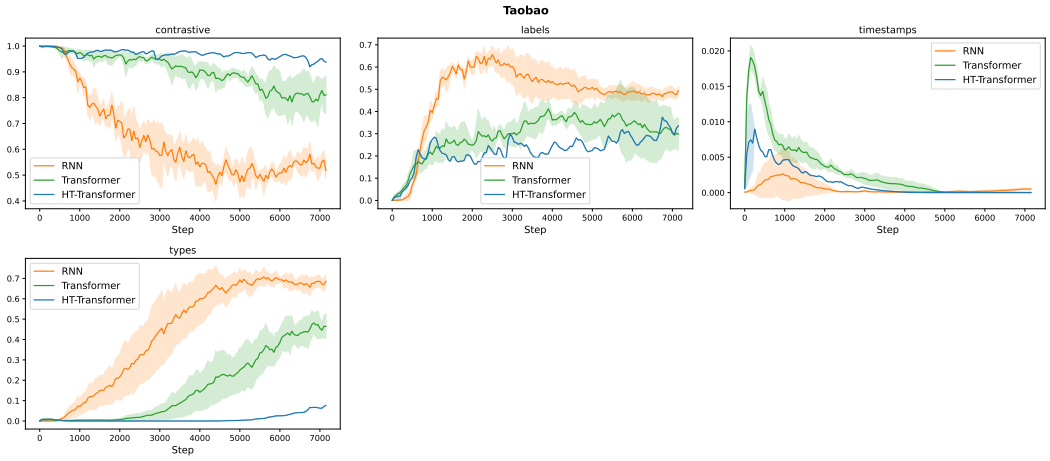


Figure 10: Weights' trajectories for the Taobao dataset.

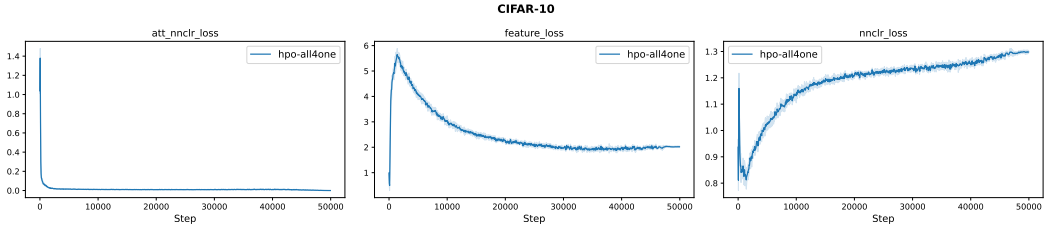


Figure 11: Weights' trajectories for the CIFAR-10 dataset.

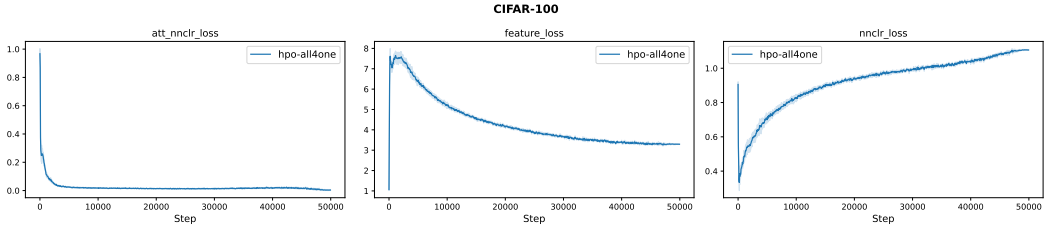


Figure 12: Weights' trajectories for the CIFAR-100 dataset.

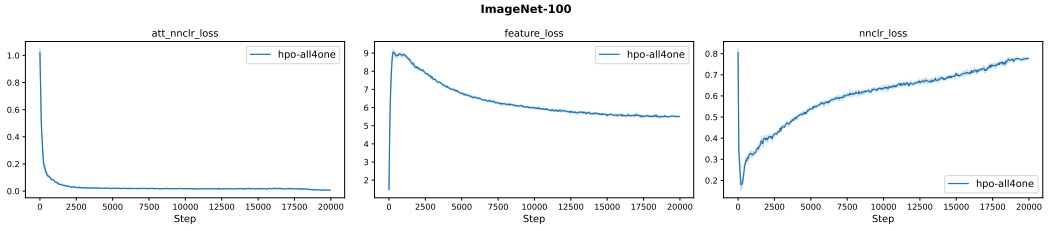


Figure 13: Weights' trajectories for the ImageNet-100 dataset.