THE COUPLING IS THE LEVER: CROSS-GROUP GEOMETRY FOR JOINT OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-task models contain heterogeneous parameter groups, including shared and task-specific components as well as experts and routers in mixture-of-experts architectures. Even when individual groups are well conditioned, optimization can remain slow or unstable because important directions may emerge only from their interaction. We show that these hidden joint modes arise from cross-group curvature that is missed when parameter groups are preconditioned independently. Near local minima, they create slow directions, while near saddles, individually stable groups can become unstable jointly. For a fixed multi-task objective, we characterize when accounting for this joint geometry improves local convergence and accelerates saddle escape. We then develop CoPEN (Coupling-aware Preconditioning for Efficient Network Optimization), which augments tensor-wise preconditioning with a low-dimensional model of cross-group curvature without forming the full Hessian. CoPEN applies to both shared–task-specific architectures and input-dependent expert–router interactions in mixture-of-experts models. Controlled and model-scale experiments show improved optimization progress and validation performance over the evaluated optimizer and multi-task baselines.

1 INTRODUCTION

Modern learning systems increasingly use a single model to support diverse tasks and behaviors within a shared parameterization (Vandenhende et al., 2022; Yang et al., 2025). Multi-task learning (Ruder, 2017) makes this structure explicit through shared and task-specific parameters, while foundation models can exhibit a more implicit form in which different queries induce different behaviors within the same model. Mixture-of-experts (MoE) models (Shazeer et al., 2017; Fedus et al., 2022; Riquelme et al., 2021) introduce an additional form of structure, where routing determines which experts participate for each input and therefore which parameter groups interact during optimization. Across these settings, parameter groups with different roles must adapt together, and updating one group can change the learning conditions faced by others. As a result, important optimization behavior can emerge only through interactions across parameter groups and remain hidden when each group is considered in isolation.

Two complementary lines of work address different aspects of this problem. Multi-task optimization methods directly target imbalance and interference among task learning signals by reweighting task losses or modifying how task gradients are combined (Chen et al., 2018; Yu et al., 2020; Liu et al., 2023; Qin et al., 2025; Wang et al., 2026). A separate line of general-purpose optimization research improves parameter-space conditioning independent of the multi-task setting. Adaptive optimizers rescale updates using gradient statistics, while structured optimizers additionally exploit matrix or tensor structure to transform updates across multiple coordinates (Kingma & Ba, 2015; Gupta et al., 2018; Vyas et al., 2025; Liu et al., 2025; Modoranu et al., 2026). In models that support heterogeneous tasks or input-dependent computation, these methods improve how the resulting learning signal is translated into parameter updates, but they do not explicitly account for task- or input-dependent interactions among parameter groups. In conventional multi-task models, such interactions arise between shared and task-specific components and vary across tasks, while in MoE models the participating experts and their interactions additionally depend on the input.

We study this behavior through cross-parameter coupling, where a change in one parameter group alters the optimization response of another. The strength and direction of this coupling depend

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.

on which groups participate in the prediction and how their roles interact, so they can vary across tasks and inputs. Such coupling can create joint directions that are not visible from any group in isolation. Near a local minimum, individually well-conditioned groups can still combine into a slowly converging direction because their interaction creates a poorly conditioned joint mode. Near a saddle, each group can have positive curvature on its own even though their interaction creates a negative-curvature joint direction, which we refer to as a coupled saddle. Our analysis characterizes these two regimes and establishes conditions under which accounting for cross-group curvature improves local convergence beyond independent group-wise preconditioning and accelerates escape along coupled unstable directions.

Based on this analysis, we develop *CoPEN* (Coupling-aware Preconditioning for Efficient Network Optimization), which augments tensor-wise preconditioning based on Shampoo statistics with a low-dimensional model of cross-group curvature. CoPEN coordinates selected parameter groups without forming the full Hessian, retaining scalable within-group preconditioning while additionally modeling cross-group interactions. The same construction applies to conventional shared–task-specific architectures and to input-dependent expert and router interactions in MoE models. Controlled toy experiments show faster convergence and earlier escape from coupled saddles when cross-group curvature is retained. Multi-task ViT and MoE experiments further show improved optimization progress and validation performance over the evaluated optimizer and multi-task baselines.

Our main contributions are as follows:

- **Joint optimization modes beyond within-group preconditioning.** We show that cross-group curvature can create slow or unstable joint directions even when individual parameter groups are well conditioned, and characterize these effects near local minima and coupled saddles.
- **Scalable coupling-aware preconditioning.** We develop CoPEN, which augments tensor-wise preconditioning with a low-dimensional model of cross-group curvature, capturing interactions omitted by independent group-wise preconditioning without forming the full Hessian.
- **Generality across parameter structures.** The same formulation applies to explicit shared–task-specific coupling and to input-dependent interactions among experts and routers in MoE models. Controlled and model-scale experiments validate the predicted optimization behavior and demonstrate improved optimization progress and validation performance in the evaluated settings.

2 Related work

Multi-task learning and optimization. Multi-task learning uses a shared model to learn multiple prediction problems while allowing specialized components when needed (Ruder, 2017; Vandenhende et al., 2022). A central optimization challenge is that tasks can differ in learning speed, scale, and preferred update direction. Loss-balancing methods such as GradNorm (Chen et al., 2018) and FAMO (Liu et al., 2023) adapt task contributions during training. MGDA (Sener & Koltun, 2018) and Nash-MTL (Navon et al., 2022) formulate the joint update through multi-objective optimization or bargaining, while PCGrad (Yu et al., 2020) and CAGrad (Liu et al., 2021a) modify task gradients to reduce harmful interference. More recently, NTKMTL (Qin et al., 2025) uses neural tangent kernel information to balance task learning dynamics, and SVFair (Wang et al., 2026) uses Shapley values to guide fair gradient aggregation. Tuned loss weighting remains an important reference for evaluating these methods (Kurin et al., 2022; Lin et al., 2022). These approaches primarily address how learning signals from different tasks should be coordinated.

Other methods incorporate task structure more directly into parameter updates. Selective task-group updates (Jeong & Yoon, 2025) use task affinity to organize sequential updates, while Rep-MTL (Wang et al., 2025) regulates task contributions at the representation level. MTAdam (Malkiel & Wolf, 2021) and AdaTask (Yang et al., 2023) maintain task-dependent adaptive statistics, and MAdam (Liu et al., 2026) adapts update geometry according to task preferences. These methods connect task-level heterogeneity with how parameters are updated. Our focus is different. For a fixed training objective, we study optimization structure that arises from the interaction among parameter groups themselves, including shared–task-specific interactions and input-dependent expert–router interactions. CoPEN therefore complements methods that determine how task signals are weighted, combined, or assigned to updates.

Preconditioning and nonconvex optimization. A separate line of general-purpose optimization research addresses poor conditioning in parameter space. Adam (Kingma & Ba, 2015) and AdamW (Loshchilov & Hutter, 2019) use coordinatewise gradient statistics to rescale updates. Structured optimizers capture richer dependencies across parameters. K-FAC (Martens & Grosse, 2015) uses structured approximations to curvature, Shampoo (Gupta et al., 2018) and SOAP (Vyas et al., 2025) exploit matrix or tensor structure, and Muon (Jordan, 2024; Liu et al., 2025) transforms momentum matrices through approximate orthogonalization. NorMuon (Li et al., 2025) adds neuronwise normalization, while Trion (Modoranu et al., 2026) uses dynamically selected low-rank subspaces to construct structured updates. Curvature-aware methods such as AdaHessian (Yao et al., 2021) and Sophia (Liu et al., 2024) instead use diagonal curvature information to scale updates. These methods provide different trade-offs between optimization geometry and computational cost.

Structured preconditioning need not be restricted to independent parameter blocks. K-FAC, for example, admits approximations containing inter-layer information, and Abreu et al. (2026) show in transformer experiments that accurate layerwise curvature can recover much of the iteration-efficiency benefit of full Gauss–Newton updates. Our question is therefore not whether cross-block information can be useful in general, but which interactions become important when parameter groups participate differently across tasks or inputs. CoPEN targets this structure explicitly by retaining scalable tensor-wise conditioning while modeling selected cross-group interactions in a low-dimensional joint curvature space.

Nonconvex optimization raises a related question near saddle points. Saddle-free Newton (Dauphin et al., 2014) uses Hessian curvature to reshape updates around negative-curvature directions. Gradient methods can also avoid strict saddles under suitable conditions without explicitly estimating curvature (Lee et al., 2016; 2019), while perturbed gradient descent (Jin et al., 2017) provides finite-time escape guarantees. A distinct line of work targets neighborhood robustness through sharpness-aware optimization, including SAM (Foret et al., 2021), its multi-task extension SAMO (Ban et al., 2025), and the curvature-based SASSHA (Shin et al., 2025). In contrast, CoPEN focuses on saddle structure induced specifically by cross-group curvature, where a joint negative-curvature direction can emerge even when individual parameter groups remain positively curved.

3 Problem Definition

3.1 Task-dependent landscapes and parameter coupling

We first formalize cross-parameter coupling in the conventional multi-task setting, where task structure and parameter sharing are explicit. Consider the joint optimization of $K \geq 2$ tasks. Let θ collect the model parameters, and let $f_{\theta,k}$ denote its prediction for task k. Each task has a data distribution $\mathcal{D}_k$ and a loss function ℓ_k, defining the risk

$$\mathcal{R}_k(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_k}\big[\ell_k(f_{\theta,k}(x), y)\big]. \tag{1}$$

Each task therefore induces its own gradient and curvature over the common parameter space. We analyze a fixed positive scalarization

$$L(\theta) = \sum_{k=1}^{K} w_k \mathcal{R}_k(\theta), \qquad w_k > 0, \qquad \sum_{k=1}^{K} w_k = 1. \tag{2}$$

The weights specify the task trade-off and remain fixed during the convergence analysis.

Let $\theta_s \in \mathbb{R}^{d_s}$ denote parameters shared by all tasks and $\theta_k \in \mathbb{R}^{d_k}$ task-specific parameters used only by task k, such as a task-specific head or adapter. Absorbing the fixed weights into $L_k = w_k \mathcal{R}_k$, we obtain

$$L(\theta) = \sum_{k=1}^{K} L_k(\theta_s, \theta_k), \qquad \theta = (\theta_s; \theta_{1:K}), \tag{3}$$

where $\theta_{1:K} = (\theta_1, \ldots, \theta_K)$ and $d_t = \sum_k d_k$. At an iterate where each task loss is twice continuously differentiable in a neighborhood, the Hessian $H = \nabla^2 L(\theta)$ has the form

$$H = \begin{pmatrix} H_{ss} & H_{st} \\ H_{ts} & H_{tt} \end{pmatrix}. \tag{4}$$

Here H_{ss} describes the curvature within the shared parameters, while $H_{tt} = \mathrm{diag}(H_{tt}^{(1)}, \ldots, H_{tt}^{(K)})$ contains the task-specific curvatures $H_{tt}^{(k)} = \nabla^2_{\theta_k,\theta_k} L_k$. The off-diagonal blocks capture cross-group curvature:

$$H_{st} = \big[H_{st}^{(1)}, \ldots, H_{st}^{(K)}\big], \qquad H_{st}^{(k)} = \nabla^2_{\theta_s,\theta_k} L_k, \qquad H_{ts} = H_{st}^{\top}. \tag{5}$$

We refer to the interaction represented by these cross-group curvature blocks as *shared–task-specific coupling*. To first order, a task-specific displacement $\Delta\theta_k$ changes task k's shared gradient by $H_{st}^{(k)}\Delta\theta_k$, while a shared displacement $\Delta\theta_s$ changes its task-specific gradient by $(H_{st}^{(k)})^{\top}\Delta\theta_s$. Distinct task-specific modules have zero direct cross-curvature blocks, but their updates can still interact indirectly through the shared parameters. This coupling is distinct from alignment among the current task gradients, and its optimization effect depends on the cross-group curvature relative to the curvature within each group.

3.2 Input-dependent coupling in mixture-of-experts models

In mixture-of-experts (MoE) models, parameter participation additionally depends on the input. A router combines expert computations for each query (Shazeer et al., 2017). For an input x from task k, let $h_\ell(x)$ be its representation entering routed layer ℓ. The mixture output is

$$u_\ell(x) = \sum_{e=1}^{E_\ell} \pi_{\ell,e}(h_\ell(x); \psi_\ell) F_{\ell,e}(h_\ell(x); \phi_{\ell,e}), \tag{6}$$

where $F_{\ell,e}$ is expert e, ψ_ℓ parameterizes the router, and the nonnegative gates sum to one. The task index k is distinct from the expert index e: an expert can serve several tasks, and inputs from the same task can take different routes.

Write $\theta = (\xi; \phi; \psi)$, where $\phi = \phi_{\ell,e}$, $\psi = \psi_\ell$, and ξ collects other trainable shared parameters. Each expert and router defines a parameter block, with any modules in ξ partitioned explicitly as well. For blocks b, c, define $H_{bc} = \nabla^2_{\theta^{(b)},\theta^{(c)}} L$. In particular,

$$H^{\mathrm{EE}}_{(\ell,e),(m,e')} = \nabla^2_{\phi_{\ell,e},\phi_{m,e'}} L, \quad H^{\mathrm{ER}}_{(\ell,e),m} = \nabla^2_{\phi_{\ell,e},\psi_m} L, \quad H^{\mathrm{RR}}_{\ell,m} = \nabla^2_{\psi_\ell,\psi_m} L. \tag{7}$$

Expert–expert (EE) and router–router (RR) coupling refer to distinct blocks, while expert–router (ER) coupling links the two types. Unlike task-specific heads, different experts can contribute to the same prediction and therefore need not have zero mixed curvature. Router updates change how expert outputs contribute to the prediction, while expert updates change the loss sensitivity to routing.

Task distributions determine which queries contribute to the objective, while routing determines which experts participate for each query. The resulting cross-group curvature therefore depends not only on the architecture, but also on the input-dependent pattern of parameter participation. We analyze how these interactions affect convergence and saddle escape in Section 4.

4 How Parameter Coupling Affects Optimization

Cross-group curvature can create joint optimization directions that are not apparent within individual parameter groups. Near a local minimum, such directions can remain poorly conditioned even after the curvature within each group is accounted for. Near a saddle, each group can have positive curvature on its own while their interaction creates a negative-curvature joint direction. We analyze when these effects limit optimization and how accounting for cross-group curvature changes the update. For the fixed multi-task objective in Eq. (2), consider the preconditioned gradient update

$$\theta_{t+1} = \theta_t - \eta_t M_t^{-1} g_t, \qquad g_t = \nabla L(\theta_t). \tag{8}$$

Here $\eta_t > 0$ is the learning rate and M_t is a symmetric positive-definite (SPD) preconditioning matrix. This condition makes $-M_t^{-1} g_t$ a descent direction whenever $g_t \neq 0$. Keeping the task scalarization fixed isolates the effect of parameter-space coupling from changes in the task trade-off. We compare preconditioning that accounts only for curvature within parameter groups with preconditioning that also represents their cross-group curvature.

4.1 When does accounting for coupling improve convergence?

We first consider optimization near a local minimum and ask whether accounting for the curvature within each parameter group is sufficient. Consider a local minimum θ_* with positive-definite Hessian $A = \nabla^2 L(\theta_*)$. Block 1 denotes the shared parameters and block 2 stacks the task-specific parameters. In the notation of Eq. (4), $A_{11} = H_{ss}(\theta_*)$, $A_{22} = H_{tt}(\theta_*)$, and $A_{12} = H_{st}(\theta_*)$. To separate within-group curvature from cross-group interaction, define

$$D = \mathrm{diag}(A_{11}, A_{22}), \qquad C = A_{11}^{-1/2} A_{12} A_{22}^{-1/2}, \qquad D^{-1/2} A D^{-1/2} = \begin{pmatrix} I & C \\ C^\top & I \end{pmatrix}. \tag{9}$$

The block-diagonal preconditioner D accounts exactly for the curvature within the shared and task-specific groups while excluding their cross-group curvature. The matrix C therefore captures the interaction that remains after each group has been normalized by its own curvature. We define its *normalized coupling strength* as

$$\rho = \|C\|_2 , \tag{10}$$

where $\|\cdot\|_2$ is the spectral norm. Since $A \succ 0$, we have $0 \le \rho < 1$.

If $\sigma_i(C)$ is a singular value of C, the normalized Hessian in Eq. (9) has paired eigenvalues $1-\sigma_i(C)$ and $1+\sigma_i(C)$. The strongest interaction therefore produces joint curvatures $1-\rho$ and $1+\rho$. When ρ is close to one, the former becomes nearly flat even though the curvature within each group has already been accounted for. A block-diagonal update must use a single scalar learning rate that accommodates both directions, which limits progress along the nearly flat joint direction.

We next compare the block-diagonal preconditioner D with a preconditioner that also represents cross-group curvature. The normalized coupling strength ρ characterizes the interaction that remains after within-group preconditioning and, in this two-block setting, also determines the best local contraction achievable using only within-group curvature. Let $\widehat{C}$ denote an approximation of the cross-group curvature in the preconditioner. We use

$$M_* = D^{1/2} \begin{pmatrix} I & \widehat{C} \\ \widehat{C}^\top & I \end{pmatrix} D^{1/2}, \qquad \|\widehat{C}\|_2 < 1.$$

The within-group blocks remain unchanged from D, while $\widehat{C}$ adds the cross-group adjustment.

The difference $A - M_*$ therefore captures the cross-group curvature that is not represented by the preconditioner. We measure this remaining discrepancy after preconditioning by

$$\varepsilon = \|M_*^{-1/2}(A - M_*)M_*^{-1/2}\|_2. \tag{11}$$

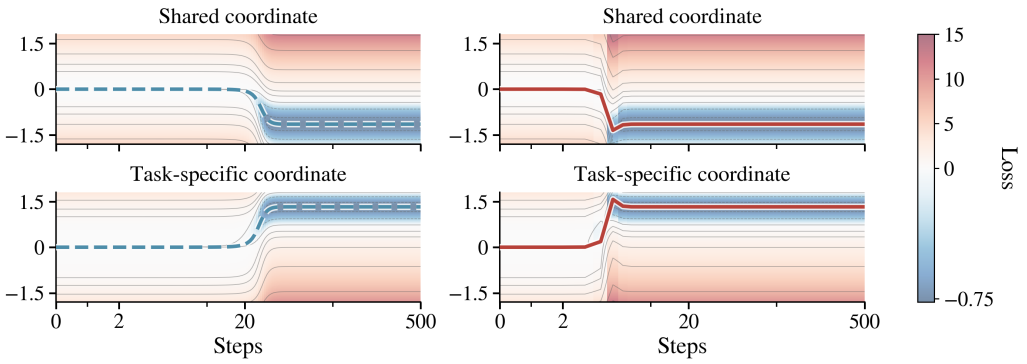
To compare the two updates, we measure local contraction in the Hessian-induced norm $\|v\|_A = (v^\top A v)^{1/2}$. The corresponding contraction factor gives the largest fraction of local error that can remain after one linearized update, so smaller values indicate faster local convergence. The following result compares the factor ε for the coupling-aware preconditioner with the factor ρ achievable using only within-group curvature.

Theorem 1 (Cross-group curvature and local contraction)**.** *At a local minimum θ_* with $A \succ 0$, let D, C, ρ, and M_* be defined as above. If $\|C - \widehat{C}\|_2 < \rho(1 - \|\widehat{C}\|_2)$, then*

$$\|I - M_*^{-1}A\|_A = \varepsilon < \rho = \inf_{\eta>0} \|I - \eta D^{-1}A\|_A. \tag{12}$$

Furthermore, if L is C^2 near θ_* and $M(\theta)$ is a continuous SPD preconditioner with $M(\theta_*) = M_*$, then every $\varepsilon < q < \rho$ admits a neighborhood in which the unit-step iteration satisfies $\|\theta_t - \theta_*\|_A \le q^t \|\theta_0 - \theta_*\|_A$ for all sufficiently close initializations.

The theorem shows that once within-group curvature is fixed, the remaining local convergence behavior is governed by how well the cross-group interaction is represented. When the approximation error is small enough, the coupling-aware preconditioner achieves a smaller worst-case local contraction factor than any constant-step update based only on within-group curvature. For a C^2 loss, the same ordering also extends locally to the nonlinear update. When the cross-group adjustment satisfies the condition in Theorem 1, starting sufficiently close to θ_* yields geometric decay of the parameter error with a contraction factor smaller than the block-diagonal factor ρ. The formal conditions, proof, and extension to general MoE parameter partitions are given in Appendix A.2.



(a) Block-diag. (b) CoPEN

Figure 1: **Coupled-saddle toy. (a)** Block-diag. and **(b)** CoPEN start from the same perturbation near the saddle. Block-diag. is the matched control with cross-group terms removed. The vertical axes show the shared parameter coordinate (top) and task-specific parameter coordinate (bottom), with trajectories plotted over 500 updates. CoPEN leaves the saddle neighborhood earlier, while both methods approach the same minimum. Background colors show one-dimensional loss slices along the plotted coordinate, with the other coordinate fixed at the current iterate.

4.2 When does accounting for coupling accelerate saddle escape?

The same normalized coupling geometry leads to a different regime near a saddle. In Section 4.1, $\rho < 1$ and the strongest joint direction has positive normalized curvature $1 - \rho$. We now consider a stationary point $\theta_\dagger$ with Hessian $H = \nabla^2 L(\theta_\dagger)$ whose shared block H_{11} and stacked-task-specific block H_{22} are positive definite, while the full Hessian is indefinite. Defining D, C, and ρ as in Eqs. (9) and (10) using the Hessian blocks at $\theta_\dagger$, the conditions $H_{11}, H_{22} \succ 0$ and H indefinite imply $\rho > 1$. The same joint direction then has curvature $1 - \rho < 0$. Thus, each parameter group has positive curvature on its own, while cross-group curvature creates a jointly unstable direction.

A block-diagonal update can move away from this saddle when the iterate has a nonzero component along the unstable direction. Its scalar learning rate must also remain stable along the positive-curvature directions, which limits how quickly that component can grow. To study whether cross-group curvature can increase this growth, consider

$$M_\kappa = D^{1/2} \begin{pmatrix} I & \kappa C/\rho \\ \kappa C^\top/\rho & I \end{pmatrix} D^{1/2}, \qquad 0 \le \kappa < 1. \tag{13}$$

Here κ controls the strength of the cross-group adjustment, with $\kappa = 0$ recovering preconditioning based only on within-group curvature. We choose $\eta_\kappa = c_0(1+\kappa)/(1+\rho)$, where $0 < c_0 < 2$, so that the positive-curvature modes remain stable under the same relative step-size choice.

Theorem 2 (Cross-group curvature and unstable-mode amplification)**.** *Let L be C^2 near a stationary point $\theta_\dagger$ with $H_{11}, H_{22} \succ 0$ and normalized coupling strength $\rho > 1$. Using M_κ from Eq. (13) with $\eta_\kappa = c_0(1+\kappa)/(1+\rho)$ for $0 < c_0 < 2$, every positive-curvature mode contracts under the linearized update, while perturbations along the most unstable joint direction are amplified by the per-step factor*

$$a_\kappa = 1 + c_0 \frac{\rho - 1}{\rho + 1} \frac{1 + \kappa}{1 - \kappa}. \tag{14}$$

This amplification factor $a_\kappa > 1$ increases strictly with κ.

The theorem shows that increasing the cross-group adjustment strengthens local growth along the coupled unstable direction without sacrificing stability along positive-curvature modes. This is a local statement around the saddle. At exact stationarity the gradient is zero, so escape requires a nonzero perturbation. Under local regularity and initialization conditions, increased amplification also yields a finite exit-time bound. The proof and local escape analysis are given in Appendix A.3.

We illustrate this behavior with the coupled-saddle toy in Fig. 1. Starting from the same perturbation, CoPEN leaves the saddle neighborhood earlier than Block-diag., while both methods approach the

Algorithm 1 CoPEN update

Require: parameters θ_t, minibatch loss f_t, scope S and its groups, cached Q, K, refresh interval T_{refresh}, damping λ_c, mixing weight γ

Compute the minibatch gradient g_t
Update tensor-wise statistics and compute the tensor-wise update b_t
if the cache is empty or T_{refresh} steps have elapsed **then**
    Build group-supported Q from b_t, g_t, and the displacement since the previous refresh
    Compute $K = Q^\top \widetilde{G}_S Q$ using streamed curvature products
    Cache Q and K
end if
Compute $\delta_{t,S}$ using Eq. (16)
Set δ_t to $\delta_{t,S}$ on S and b_t elsewhere
return δ_t

same minimum. The practical CoPEN construction is introduced in Section 5, and the experimental details are given in Appendix D.

5 Coupling-Aware Preconditioning

The analysis in Section 4 motivates preconditioning that accounts for both parameter tensors and their interactions. CoPEN operates at two levels. Within each trainable tensor, it uses preconditioning based on Shampoo statistics. Across selected structural groups, each of which may contain multiple tensors, it applies a low-dimensional cross-group adjustment based on projected curvature.

Within-tensor preconditioning. For each parameter tensor, we maintain axis-wise gradient second-moment matrices following Shampoo (Gupta et al., 2018), and use their eigenvectors to define a coordinate basis. Following SOAP (Vyas et al., 2025), we apply adaptive moment normalization in this basis and map the resulting direction back to the original parameter coordinates. Collecting these directions across all parameter tensors gives the tensor-wise update b_t. This stage applies tensor-wise preconditioning, while the cross-group adjustment below models interactions among larger structural groups using projected curvature.

Estimating cross-group interactions. The coupling adjustment operates on larger structural groups rather than individual tensors. We choose a scope S of parameters to couple and partition it into groups indexed by i, such as transformer blocks and task heads in ViT, or routed layers in MoE. Each group can contain multiple parameter tensors. For group i, we use the entries of the tensor-wise update b_t and gradient g_t corresponding to that group, together with its parameter displacement since the previous refresh. We orthonormalize these nonzero directions to obtain $Q_{i,t}$. Stacking the group bases gives Q_t, whose columns span the low-dimensional subspace used for the coupling adjustment.

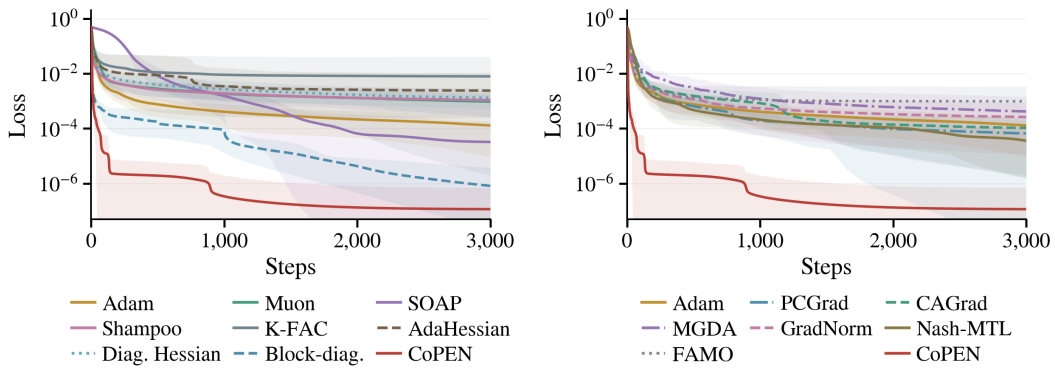
We approximate parameter-space curvature by pulling back a positive-semidefinite loss curvature in output space through the prediction Jacobian, $\widetilde{G}_S = J_S^\top W J_S$ (Schraudolph, 2002; Martens, 2020). Here J_S maps perturbations of the selected parameters to output changes, and $W \succeq 0$ measures loss curvature with respect to those outputs. If d_S is the number of selected parameters, m the output dimension, and r the number of basis directions, then $J_S \in \mathbb{R}^{m \times d_S}$, $Q_t \in \mathbb{R}^{d_S \times r}$, and $K_t \in \mathbb{R}^{r \times r}$. The projected curvature is

$$K_t = Q_t^\top \widetilde{G}_S Q_t, \qquad (K_t)_{ij} = (J_i Q_{i,t})^\top W (J_j Q_{j,t}). \tag{15}$$

The diagonal blocks of K_t describe curvature within the represented groups, while its cross-group blocks describe how directions from different groups jointly affect the model output. We compute K_t from the directional Jacobian products $J_S Q_t$ without forming the full Jacobian or Hessian.

Combining tensor-wise and cross-group preconditioning. Let $b_{t,S}$ denote the part of the tensor-wise update corresponding to the selected scope S. Projecting $b_{t,S}$ onto Q_t gives the reduced coefficients $Q_t^\top b_{t,S}$, which we precondition using K_t. The final CoPEN update on this scope, $\delta_{t,S}$, maps the resulting adjustment back to parameter space:

$$\delta_{t,S} = b_{t,S} + \gamma Q_t \big[\lambda_c (K_t + \lambda_c I)^{-1} - I\big] Q_t^\top b_{t,S}. \tag{16}$$



(a) Standard optimizer comparison (b) Multi-task optimization comparison

Figure 2: **Shared-MLP toy.** Training loss on a nonlinear shared-MLP toy, comparing CoPEN with **(a)** standard optimizers and **(b)** multi-task optimization methods. The multi-task baselines in (b) use Adam as their base optimizer. Curves show the mean over six paired runs, with all 3,000 updates shown. Shading indicates the pointwise minimum–maximum range.

The damping $\lambda_c > 0$ stabilizes the reduced preconditioner, and $\gamma \in [0, 1]$ controls the contribution of the coupling adjustment. Setting $\gamma = 0$ recovers the tensor-wise update. Parameters outside S retain their tensor-wise updates.

The bases Q_t and projected curvature K_t are refreshed periodically and reused between refreshes, while b_t is recomputed at every step. This keeps the cross-group adjustment low dimensional and avoids repeated full curvature estimation. The main full-model experiments refresh Q_t and K_t every 25 updates. Additional implementation details are provided in Appendix C.

6 Experiments

Controlled experiments. The shared-MLP toy in Fig. 2 isolates shared–head coupling. Its matched Block-diag. control uses the same within-group construction and differs only by removing cross-group entries. CoPEN reaches the control's 3,000-update loss in 892 updates, a 3.36× speedup, showing that the gain is not explained by within-group preconditioning alone. This is consistent with Theorem 1, which identifies cross-group curvature as a remaining convergence bottleneck. In the coupled-saddle toy in Fig. 1, CoPEN and Block-diag. start from the same perturbation and approach the same minimum, but CoPEN leaves the saddle neighborhood earlier, consistent with the unstable-mode amplification in Theorem 2. Additional routed-MoE results in Appendix D show the same pattern for expert–router coupling, with CoPEN converging faster while changing how optimization motion is distributed across experts and routers.

Model-scale setup. We evaluate CoPEN on ten dense-prediction tasks from the Taskonomy tiny split (Zamir et al., 2018) using pretrained ViT-B/16 (Dosovitskiy et al., 2021) and V-MoE-B/16 (Riquelme et al., 2021). All methods update all backbone and task-head parameters with the same preprocessing, batch size, initialization, and training stream. We compare against nine general-purpose optimizers and eight multi-task optimization methods. Each method is tuned over 24 hyperparameter configurations under the same search budget, and the configuration selected by periodic-validation loss is used for the comparison. The validation results in Table 1 are averaged over three seeds and evaluated on the full validation split. We report both mean validation loss and Δ_m, the mean relative task-loss reduction against separately trained single-task models, where positive values indicate average improvement over single-task training. Experimental protocols, baseline configurations, and CoPEN implementation details are provided in Appendices C and E.

Model-scale results. The model-scale results preserve the pattern observed in the controlled experiments: explicitly modeling cross-group interactions improves optimization even when the coupling is represented only through a low-dimensional approximation. The improvement appears in both the shared-backbone ViT and the input-dependent MoE, indicating that the benefit is not specific to a single parameter partition or interaction pattern. CoPEN improves both aggregate validation

Table 1: **Model-scale comparison on Taskonomy.** Δ_m is the mean relative task-loss reduction against single-task models. The lowest validation loss and highest Δ_m are shown in bold.

	ViT			MoE		
Method	Val. loss ↓	$\Delta_m(\%)$ ↑	s/step	Val. loss ↓	$\Delta_m(\%)$ ↑	s/step
Optimizers						
AdamW (Loshchilov & Hutter, 2019)	0.14442	+5.31	0.155	0.14606	+1.59	0.248
Shampoo (Gupta et al., 2018)	0.15813	-8.30	0.310	0.15078	-2.83	2.180
K-FAC† (Martens & Grosse, 2015)	0.25997	-116.68	0.922	0.34727	-458.76	2.430
AdaHessian (Yao et al., 2021)	0.16624	-18.68	0.564	0.15735	-11.69	0.909
Muon (Jordan, 2024)	0.14543	+4.20	0.258	0.14562	+1.32	0.473
SOAP (Vyas et al., 2025)	0.14579	+3.77	0.288	0.14624	+0.92	0.462
SASSHA (Shin et al., 2025)	0.19799	-71.87	0.324	0.20053	-80.90	0.521
NorMuon (Li et al., 2025)	0.14494	+4.56	0.263	0.14563	+1.40	0.485
Trion (Modoranu et al., 2026)	0.14506	+4.53	0.289	0.14518	+1.99	0.524
Multi-task optimization (MTO)						
MGDA (Sener & Koltun, 2018)	0.16532	-18.04	0.836	0.16886	-28.45	1.461
GradNorm (Chen et al., 2018)	0.14933	-0.74	0.277	0.14823	-2.53	0.490
PCGrad (Yu et al., 2020)	0.14495	+4.59	0.867	0.14646	+0.39	1.532
CAGrad (Liu et al., 2021a)	0.14859	+0.57	0.835	0.14975	-3.46	1.479
Nash-MTL (Navon et al., 2022)	0.14637	+2.68	0.833	0.14631	+0.71	1.457
FAMO (Liu et al., 2023)	0.15073	-0.79	0.194	0.15042	-2.98	0.318
NTKMTL-SR (Qin et al., 2025)	0.14596	+3.63	0.457	0.14662	+0.45	0.538
SVFair (Wang et al., 2026)	0.14782	+1.28	0.864	0.14830	-1.78	1.495
CoPEN (Ours)	**0.14306**	**+6.82**	0.327	**0.14344**	**+4.34**	0.672

Table 2: **Time to validation-loss targets.** Targets are measured on the fixed validation subset used for periodic evaluation. Each entry reports the updates ($\times 10^3$) required to reach each target loss and the corresponding training time relative to AdamW.

	ViT		MoE	
Method	Loss ≤ 0.150	Loss ≤ 0.140	Loss ≤ 0.150	Loss ≤ 0.140
AdamW	6.144 / 1.00	31.744 / 1.00	6.656 / 1.00	21.504 / 1.00
Muon	2.560 / 0.70	14.336 / 0.75	2.048 / **0.59**	14.336 / 1.27
SOAP	2.560 / 0.78	21.504 / 1.26	2.560 / 0.72	16.896 / 1.46
CoPEN (Ours)	**1.536 / 0.53**	**8.192 / 0.55**	**1.536** / 0.63	**6.656 / 0.84**

performance and average task-wise performance relative to separately trained single-task models, indicating that the gain is not obtained only by redistributing error across tasks. Unlike the MTO baselines, which modify task weighting or gradient aggregation, CoPEN leaves the task objective fixed and instead conditions the resulting gradient across parameter groups, supporting cross-group parameter coupling as a complementary optimization axis to task-level coordination. The time-to-target results in Table 2 further show that CoPEN reaches the targets in about 3.2–4.3× fewer updates than AdamW across both architectures. Despite the additional per-step cost of coupling estimation, this corresponds to about 16–47% lower training time. Appendices F and F.3 provide additional training curves and computational-cost results, including peak GPU memory.

7 Conclusion

We presented CoPEN, which uses cross-parameter coupling to coordinate multi-task optimization. Our analysis establishes when retaining mixed curvature improves convergence beyond block-diagonal preconditioning and accelerates saddle escape while preserving positive-curvature stability. CoPEN combines tensor-wise preconditioning from Shampoo statistics with a low-dimensional joint-curvature model without forming the full Hessian, covering shared–task-specific networks and input-dependent interactions in MoE. Toy studies and ViT and MoE evaluations demonstrate improved optimization progress and validation performance. These results highlight cross-group geometry as a complement to within-group preconditioning and task-gradient coordination.

AI USE STATEMENT

In this work, we used generative AI tools to assist with experimental design, code implementation and debugging, literature review, and manuscript editing for clarity and readability. We did not use generative AI tools to generate the datasets used in our experiments.

ETHICS STATEMENT

We do not collect new human-subject data or release a new dataset. The proposed method concerns general-purpose optimization rather than an application-specific deployment setting, and we do not identify additional ethical concerns specific to its use in the experiments considered here.

REPRODUCIBILITY STATEMENT

Appendix A provides the assumptions, proofs, and supporting theoretical results. Appendix C gives the implementation details of CoPEN, and Appendix D gives the settings and results of the controlled toy experiments. Appendix E describes the datasets, preprocessing, model architectures, and training and tuning protocols. Additional model-scale results, constant-learning-rate experiments, and computational-cost analysis are provided in Appendix F. We plan to publicly release the code and experiment configurations.

REFERENCES

Natalie Abreu, Nikhil Vyas, Sham Kakade, and Depen Morwani. The potential of second-order optimization for LLMs: A study with full Gauss–Newton. In *International Conference on Learning Representations*, pp. 123816–123833, 2026. URL https://arxiv.org/abs/2510.09378.

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998. doi: 10.1162/089976698300017746.

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning, 2020. URL https://arxiv.org/abs/2002.09018.

Hao Ban, Gokul Ram Subramani, and Kaiyi Ji. SAMO: A lightweight sharpness-aware approach for multi-task optimization with joint global-local perturbation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 785–795, 2025. URL https://arxiv.org/abs/2507.07883.

R. H. Byrd, S. L. Hansen, Jorge Nocedal, and Y. Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016. doi: 10.1137/140954362. URL https://arxiv.org/abs/1401.7020.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning (ICML)*, 2018. arXiv:1711.02257.

Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2010.06808.

Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. arXiv:1406.2572.

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. URL https://arxiv.org/abs/2010.11929.

Runa Eschenhagen, Aaron Defazio, Tsung-Hsien Lee, Richard E. Turner, and Hao-Jun Michael Shi. Purifying Shampoo: Investigating Shampoo's heuristics by decomposing its preconditioner. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2506.03595.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. arXiv:2101.03961.

Vladimir Feinberg, Xinyi Chen, Y. Jennifer Sun, Rohan Anil, and Elad Hazan. Sketchy: Memory-efficient adaptive regularization with frequent directions. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2302.03764.

Heshan Fernando, Han Shen, Miao Liu, Subhajit Chaudhury, Keerthiram Murugesan, and Tianyi Chen. Mitigating gradient bias in multi-objective learning: A provably convergent approach. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.12624.

Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. arXiv:2109.04617.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.01412.

Kenji Fukumizu and Shun-ichi Amari. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural Networks*, 13(3):317–327, 2000.

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker-factored eigenbasis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL https://arxiv.org/abs/1806.03884.

Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-Newton methods for training deep neural networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/192fc044e74dffea144f9ac5dc9f3395-Abstract.html.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning (ICML)*, pp. 1842–1850, 2018. URL https://proceedings.mlr.press/v80/gupta18a.html.

Paul W. Holland and Roy E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics—Theory and Methods*, 6(9):813–827, 1977. doi: 10.1080/03610927708827533. URL https://www.tandfonline.com/doi/abs/10.1080/03610927708827533.

Majid Jahani, MohammadReza Nazari, Rachael Tappenden, Albert Berahas, and Martin Takac. SONIA: A symmetric blockwise truncated optimization algorithm. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 487–495, 2021. URL https://proceedings.mlr.press/v130/jahani21a.html.

Adrián Javaloy and Isabel Valera. RotoGrad: Gradient homogenization in multitask learning. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=T8wHz4rnuGL.

Wooseong Jeong and Kuk-Jin Yoon. Quantifying task priority for multi-task optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 363–372, 2024. doi: 10.1109/CVPR52733.2024.00042. URL https://openaccess.thecvf.com/content/CVPR2024/html/Jeong_Quantifying_Task_Priority_for_Multi-Task_Optimization_CVPR_2024_paper.html.

Wooseong Jeong and Kuk-Jin Yoon. Selective task group updates for multi-task optimization. In *International Conference on Learning Representations*, pp. 95046–95079, 2025. URL https://arxiv.org/abs/2502.11986.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1724–1732. PMLR, 2017. URL https://proceedings.mlr.press/v70/jin17a.html.

Keller Jordan. Muon: An optimizer for hidden layers in neural networks. Technical blog, 2024. URL https://kellerjordan.github.io/posts/muon/.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. arXiv:1705.07115.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

Abdoulaye Koroko, Ani Anciaux-Sedrakian, Ibtihel Ben Gharbia, Valérie Garès, Mounir Haddou, and Quang Huy Tran. Analysis and comparison of two-level KFAC methods for training deep neural networks. *Optimization Methods and Software*, 40(1):96–128, 2025. doi: 10.1080/10556788.2024.2380684. URL https://arxiv.org/abs/2303.18083.

Frederik Kunstner, Lukas Balles, and Philipp Hennig. Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1905.12558.

Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and M. Pawan Kumar. In defense of the unitary scalarization for deep multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2201.04122.

Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pp. 1246–1257. PMLR, 2016. URL https://proceedings.mlr.press/v49/lee16.html.

Jason D. Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. First-order methods almost always avoid strict saddle points. *Mathematical Programming*, 176(1–2):311–337, 2019. doi: 10.1007/s10107-019-01374-3. URL https://doi.org/10.1007/s10107-019-01374-3.

Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. NorMuon: Making Muon more efficient and scalable, 2025. URL https://arxiv.org/abs/2510.05491.

Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor W. Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research (TMLR)*, 2022. arXiv:2111.10603.

Wu Lin, Felix Dangel, Runa Eschenhagen, Juhan Bae, Richard E. Turner, and Alireza Makhzani. Can we remove the square-root in adaptive gradient methods? a second-order perspective. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 29949–29973, 2024. URL https://proceedings.mlr.press/v235/lin24e.html.

Wu Lin, Scott C. Lowe, Felix Dangel, Runa Eschenhagen, Zikun Xu, and Roger Grosse. Understanding and improving Shampoo and SOAP via Kullback–Leibler minimization. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/95a704bd2fdf8ef8242b4adcc7ce3c93-Abstract-Conference.html.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021a. arXiv:2110.14048.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. FAMO: Fast adaptive multitask optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2306.03792.

Fengbei Liu, Rachit Saluja, Sunwoo Kwak, Ruibo Wang, Ruining Deng, Heejong Kim, Johannes C. Paetzold, and Mert R. Sabuncu. MAdam: Metric-aware multi-objective Adam, 2026. URL https://arxiv.org/abs/2606.03904.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *International Conference on Learning Representations*, pp. 1621–1650, 2024. URL https://arxiv.org/abs/2305.14342.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for LLM training, 2025. URL https://arxiv.org/abs/2502.16982.

Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *International Conference on Learning Representations (ICLR)*, 2021b.

Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-end multi-task learning with attention. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. arXiv:1803.10704.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1711.05101.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018. doi: 10.1145/3219819.3220007. URL https://doi.org/10.1145/3219819.3220007.

Debabrata Mahapatra and Vaibhav Rajan. Multi-task learning with user preferences: Gradient descent with controlled ascent in Pareto optimization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6597–6607, 2020. URL https://proceedings.mlr.press/v119/mahapatra20a.html.

Itzik Malkiel and Lior Wolf. MTAdam: Automatic balancing of multiple training loss terms. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10713–10729, 2021. doi: 10.18653/v1/2021.emnlp-main.837. URL https://aclanthology.org/2021.emnlp-main.837/.

James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning*, 2010. URL https://icml.cc/Conferences/2010/papers/458.pdf.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. URL http://jmlr.org/papers/v21/17-678.html.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning (ICML)*, pp. 2408–2417, 2015. URL https://proceedings.mlr.press/v37/martens15.html.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. URL https://arxiv.org/abs/1604.03539.

Ionut-Vlad Modoranu, Mher Safaryan, Erik Schultheis, Maksim Riabinin, Artem Chumachenko, and Dan Alistarh. Trion: FFT-based dynamic subspace selection for low-rank adaptive optimization of LLMs. In *International Conference on Learning Representations*, pp. 93071–93092, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/964484b27b2d9bf5b2bf571676242597-Abstract-Conference.html.

Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. Learning the pareto front with hypernetworks. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=NjF772F4ZZR.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In *International Conference on Machine Learning (ICML)*, 2022. arXiv:2202.01017.

Barak A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994. doi: 10.1162/neco.1994.6.1.147. URL https://doi.org/10.1162/neco.1994.6.1.147.

Xiaohan Qin, Xiaoxing Wang, Ning Liao, and Junchi Yan. NTKMTL: Mitigating task imbalance in multi-task learning from neural tangent kernel perspective. In *Advances in Neural Information Processing Systems*, volume 38, pp. 48299–48327, 2025. doi: 10.52202/085713-1612. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4522de4178bddb36b49aa26efad537cf-Abstract-Conference.html.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://papers.neurips.cc/paper_files/paper/2021/hash/48237d9f2dea8c74c2a72126cf63d933-Abstract.html.

Sebastian Ruder. An overview of multi-task learning in deep neural networks, 2017. URL https://arxiv.org/abs/1706.05098.

Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002. doi: 10.1162/08997660260028683. URL https://nic.schraudolph.org/pubs/Schraudolph02.pdf.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. arXiv:1810.04650.

Dmitry Senushkin, Nikolay Patakin, Arseny Kuznetsov, and Anton Konushin. Independent component alignment for multi-task learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. arXiv:2305.19000.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017. arXiv:1701.06538.

Hao-Jun Michael Shi, Tsung-Hsien Lee, Shintaro Iwasaki, Jose Gallego-Posada, Zhijing Li, Kaushik Rangadurai, Dheevatsa Mudigere, and Michael Rabbat. A distributed data-parallel PyTorch implementation of the distributed Shampoo optimizer for training neural networks at-scale, 2023. URL https://arxiv.org/abs/2309.06497.

Dahun Shin, Dongyeop Lee, Jinseok Chung, and Namhoon Lee. Sassha: Sharpness-aware adaptive second-order optimization with stable hessian approximation. In *International Conference on Machine Learning (ICML)*, pp. 55145–55171, 2025. URL https://proceedings.mlr.press/v267/shin25d.html.

Matthew Staib, Sashank Reddi, Satyen Kale, Sanjiv Kumar, and Suvrit Sra. Escaping saddle points with adaptive gradient methods. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5956–5965, 2019. URL https://proceedings.mlr.press/v97/staib19a.html.

Trevor Standley, Amir R. Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning (ICML)*, 2020. arXiv:1905.07553.

Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your ViT? data, augmentation, and regularization in vision transformers. *Transactions on Machine Learning Research*, 2022. URL https://arxiv.org/abs/2106.10270.

Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. AdaShare: Learning what to share for efficient deep multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/634841a6831464b64c072c8510c7f35c-Abstract.html.

Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. Progressive layered extraction (PLE): A novel multi-task learning (MTL) model for personalized recommendations. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pp. 269–278, 2020. doi: 10.1145/3383313.3412236.

Nikolaos Tselepidis, Jonas Kohler, and Antonio Orvieto. Two-level K-FAC preconditioning for deep learning, 2020. URL https://arxiv.org/abs/2011.00573.

Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3614–3633, 2022. doi: 10.1109/TPAMI.2021.3054719. URL https://doi.org/10.1109/TPAMI.2021.3054719.

Oriol Vinyals and Daniel Povey. Krylov subspace descent for deep learning. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pp. 1261–1268, 2012. URL https://proceedings.mlr.press/v22/vinyals12.html.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. SOAP: Improving and stabilizing Shampoo using Adam for language modeling. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=IDxZhXrpNf.

Xiao Wang, Yuying Han, Dazi Li, Fei Zhang, and Min Tang. From gradient volume to Shapley fairness: Towards fair multi-task learning. In *International Conference on Learning Representations*, pp. 129987–130040, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/d32d6793969af58872b6e0394a94ee48-Abstract-Conference.html.

Zedong Wang, Siyuan Li, and Dan Xu. Rep-MTL: Unleashing the power of representation-level task saliency for multi-task learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3413–3423, 2025. URL https://arxiv.org/abs/2507.21049.

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2010.05874.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Enneng Yang, Junwei Pan, Ximei Wang, Haibin Yu, Li Shen, Xihua Chen, Lei Xiao, Jie Jiang, and Guibing Guo. AdaTask: A task-aware adaptive learning rate approach to multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10745–10753, 2023. doi: 10.1609/aaai.v37i9.26275. URL https://ojs.aaai.org/index.php/AAAI/article/view/26275.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael W. 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. doi: 10.1609/aaai.v35i12.17275. URL https://ojs.aaai.org/index.php/AAAI/article/view/17275.

Jui-Nan Yen, Sai Surya Duvvuri, Inderjit S. Dhillon, and Cho-Jui Hsieh. Block low-rank preconditioner with shared basis for stochastic optimization. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/389cfad711d2b1e2128e931feee80230-Abstract-Conference.html.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2001.06782.

Amir R. Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3712–3722, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Zamir_Taskonomy_Disentangling_Task_CVPR_2018_paper.html.

Xiaoyuan Zhang, Xi Lin, and Qingfu Zhang. PMGDA: A preference-based multiple gradient descent algorithm, 2024. URL https://arxiv.org/abs/2402.09492.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2202.09368.

Appendix contents

A Proofs . . . 18
A.1 Notation, regularity, and coupled saddles . . . 18
A.2 Proof of Theorem 1 and extension to MoE . . . 18
A.3 Proof of Theorem 2 and finite-time escape . . . 22
B Additional related work . . . 24
B.1 Task structure and additional multi-task optimization methods . . . 24
B.2 Tensor-wise statistics and structured preconditioning . . . 25
B.3 Subspace optimization and cross-block corrections . . . 25
B.4 Curvature estimation and nonconvex geometry . . . 25
C Implementation details of CoPEN . . . 26
D Controlled toy experiments . . . 26
D.1 Toy setups and tuning . . . 26
D.2 Toy results . . . 29
D.3 Cross-group coupling diagnostics . . . 30
E Experimental setup for ViT and MoE . . . 32
E.1 Datasets and architectures . . . 32
E.2 Baselines and hyperparameter search . . . 33
E.3 Training schedules and evaluation . . . 33
F Additional experiments and analysis . . . 35
F.1 Task-wise validation results . . . 35
F.2 Training curves . . . 35
F.3 Constant-learning-rate results . . . 37
F.4 Computational cost . . . 38

A PROOFS

After fixing the notation, Section A.2 presents the common local-convergence lemma, followed by the proof of Theorem 1 and its MoE extension. Section A.3 gives the proof of Theorem 2 and the finite local escape analysis.

A.1 NOTATION, REGULARITY, AND COUPLED SADDLES

Vector norms are Euclidean and unqualified matrix norms are spectral. For a symmetric positive-definite (SPD) matrix P, define

$$\|x\|_P = (x^\top P x)^{1/2}, \qquad \|T\|_P = \sup_{x\neq 0} \frac{\|Tx\|_P}{\|x\|_P} = \left\|P^{1/2} T P^{-1/2}\right\|_2 .$$

All Hessians refer to the fixed scalarized objective L. A stationary point satisfies $\nabla L(\theta) = 0$. Individual task gradients need not vanish. We write $L \in C^2$ when L is twice continuously differentiable. Its Hessian is β-Lipschitz on a domain if

$$\left\|\nabla^2 L(x) - \nabla^2 L(y)\right\|_2 \le \beta \|x - y\|$$

for every x, y in that domain.

Definition A.1 (Coupled saddles)**.** For a fixed partition $\mathcal{B}$, a coupled saddle is a stationary point satisfying

$$H_{bb}(\theta_*) \succ 0 \quad \text{for every } b \in \mathcal{B}, \qquad \lambda_{\min}(H(\theta_*)) < 0.$$

Each diagonal block is positive definite, while the full Hessian has a joint negative-curvature direction.

A.2 PROOF OF THEOREM 1 AND EXTENSION TO MOE

A.2.1 LOCAL CONVERGENCE LEMMA

This standard lemma supplies the spectral identities and the nonlinear convergence argument used in Theorem 1 and Proposition A.1.

Lemma A.1 (Local contraction of preconditioned updates)**.** *Let L be C^2 near θ_*, with $\nabla L(\theta_*) = 0$ and $H_* = \nabla^2 L(\theta_*) \succ 0$. Let $M(\theta)$ be continuous and SPD there, with $M_* = M(\theta_*)$. For a fixed step size $\eta > 0$, define the iteration*

$$F(\theta) = \theta - \eta M(\theta)^{-1} \nabla L(\theta)$$

and let

$$q_* = \max_i |1 - \eta \lambda_i(M_*^{-1/2} H_* M_*^{-1/2})|$$

denote the H_*-norm of its linearization at θ_*. If $q_* < 1$, then for every $\bar{q} \in (q_*, 1)$ there is $\delta > 0$ such that

$$\|F(\theta) - \theta_*\|_{H_*} \le \bar{q} \|\theta - \theta_*\|_{H_*} \quad \textit{whenever } \|\theta - \theta_*\|_{H_*} \le \delta.$$

Every initialization in this ball remains in it and satisfies $\|\theta_t - \theta_*\|_{H_*} \le \bar{q}^t \|\theta_0 - \theta_*\|_{H_*}$ and $L(\theta_t) - L(\theta_*) = O(\bar{q}^{2t})$.

Proof. Step 1: Linear contraction factor. For SPD A, P, the matrices $A^{1/2}P^{-1}A^{1/2}$ and $P^{-1/2}AP^{-1/2}$ have the same positive eigenvalues. Hence

$$\begin{aligned}\left\|I - \eta P^{-1}A\right\|_A &= \left\|I - \eta A^{1/2} P^{-1} A^{1/2}\right\|_2 \\ &= \max_i |1 - \eta \lambda_i(P^{-1/2} A P^{-1/2})|. \end{aligned} \tag{17}$$

To obtain the best contraction achievable for a fixed preconditioner P, let $0 < a \le b$ be the smallest and largest eigenvalues of $P^{-1/2}AP^{-1/2}$. Because $|1 - \eta\lambda|$ is convex in λ, its maximum over the

eigenvalues in $[a, b]$ is attained at one of the endpoints. Hence the factor in Eq. (17) reduces to $\max\{|1-\eta a|, |1-\eta b|\}$. Minimizing this quantity over $\eta > 0$ gives

$$\inf_{\eta>0} \left\|I - \eta P^{-1}A\right\|_A = \frac{b-a}{b+a}, \qquad \eta_{\text{opt}} = \frac{2}{a+b}. \tag{18}$$

For $A = H_*$ and $P = M_*$, Eq. (17) equals q_*.

Step 2: Local nonlinear convergence. Let $e = \theta - \theta_*$ denote the error from the stationary point. Since L is C^2, its gradient admits the local expansion

$$\nabla L(\theta_* + e) = H_* e + o(\|e\|).$$

Continuity of M^{-1} similarly gives

$$M(\theta_* + e)^{-1} = M_*^{-1} + o(1).$$

Substituting these two expansions into the iteration F yields

$$F(\theta_* + e) - \theta_* = (I - \eta M_*^{-1} H_*)e + r(e), \qquad \|r(e)\|_{H_*} = o(\|e\|_{H_*}). \tag{19}$$

where $r(e)$ collects the higher-order terms. Thus, sufficiently close to θ_*, the nonlinear iteration is governed by the linearized update from Step 1 up to a higher-order remainder. Since $\|r(e)\|_{H_*} = o(\|e\|_{H_*})$, for any $\bar{q} \in (q_*, 1)$ we can choose $\delta > 0$ such that

$$\|r(e)\|_{H_*} \le (\bar{q} - q_*) \|e\|_{H_*} \tag{20}$$

whenever $\|e\|_{H_*} \le \delta$. Applying the triangle inequality to Eq. (19), then using the operator-norm identity in Eq. (17) and the remainder bound in Eq. (20), gives

$$\begin{aligned} \|F(\theta_* + e) - \theta_*\|_{H_*} &\le \left\|(I - \eta M_*^{-1} H_*)e\right\|_{H_*} + \|r(e)\|_{H_*} \\ &\le q_* \|e\|_{H_*} + \|r(e)\|_{H_*} \\ &\le \bar{q} \|e\|_{H_*}. \end{aligned}$$

Because $\bar{q} < 1$, an iterate initialized in this ball remains in the ball, and repeated application of the bound yields

$$\|\theta_t - \theta_*\|_{H_*} \le \bar{q}^t \|\theta_0 - \theta_*\|_{H_*}.$$

Finally, a second-order Taylor expansion of L at the stationary point gives

$$L(\theta_* + e) - L(\theta_*) = \tfrac{1}{2} \|e\|_{H_*}^2 + o(\|e\|_{H_*}^2),$$

which implies $L(\theta_t) - L(\theta_*) = O(\bar{q}^{2t})$. □

A.2.2 Proof of Theorem 1

Theorem 1 (Cross-group curvature and local contraction)**.** *At a local minimum θ_* with $A \succ 0$, let D, C, ρ, and M_* be defined as above. If $\|C - \widehat{C}\|_2 < \rho(1 - \|\widehat{C}\|_2)$, then*

$$\|I - M_*^{-1}A\|_A = \varepsilon < \rho = \inf_{\eta>0} \|I - \eta D^{-1}A\|_A. \tag{12}$$

Furthermore, if L is C^2 near θ_* and $M(\theta)$ is a continuous SPD preconditioner with $M(\theta_*) = M_*$, then every $\varepsilon < q < \rho$ admits a neighborhood in which the unit-step iteration satisfies $\|\theta_t - \theta_*\|_A \le q^t \|\theta_0 - \theta_*\|_A$ for all sufficiently close initializations.

Proof. All Hessian blocks are evaluated at θ_*. The shared block is A_{11} and the stacked-task-specific block is A_{22}. With the notation of Eqs. (9) and (11), write

$$\bar{A} = D^{-1/2} A D^{-1/2} = \begin{pmatrix} I & C \\ C^\top & I \end{pmatrix}, \qquad \bar{M} = D^{-1/2} M_* D^{-1/2} = \begin{pmatrix} I & \widehat{C} \\ \widehat{C}^\top & I \end{pmatrix}.$$

Step 1: Optimal block-diagonal contraction factor. Let $Cv_i = \sigma_i u_i$ and $C^\top u_i = \sigma_i v_i$ be a singular-vector pair. Then

$$\bar{A} \begin{pmatrix} u_i \\ \pm v_i \end{pmatrix} = (1 \pm \sigma_i) \begin{pmatrix} u_i \\ \pm v_i \end{pmatrix}.$$

Hence the paired eigenvalues of $\bar{A}$ are $1 \pm \sigma_i$, with any remaining eigenvalues equal to one. Thus

$$a = \lambda_{\min}(\bar{A}) = 1 - \rho > 0, \qquad b = \lambda_{\max}(\bar{A}) = 1 + \rho.$$

Applying Eq. (18) with $P = D$ gives

$$\inf_{\eta>0} \left\|I - \eta D^{-1}A\right\|_A = \frac{(1+\rho)-(1-\rho)}{(1+\rho)+(1-\rho)} = \rho, \qquad \eta_{\text{opt}} = 1. \tag{21}$$

Step 2: Coupling-aware contraction factor. We first verify that the coupling-aware preconditioner is well defined. By the same singular-vector argument as in Step 1, the eigenvalues of $\bar{M}$ are $1 \pm \sigma_i(\widehat{C})$, together with any remaining eigenvalues equal to one. Since $\left\|\widehat{C}\right\|_2 < 1$, all of these eigenvalues are positive, so $\bar{M}$ is SPD. Because $M_* = D^{1/2}\bar{M}D^{1/2}$ with $D \succ 0$, M_* is SPD as well.

To isolate the error introduced by approximating the cross-group curvature, define

$$E = \bar{A} - \bar{M} = \begin{pmatrix} 0 & C - \widehat{C} \\ (C - \widehat{C})^\top & 0 \end{pmatrix}.$$

Thus E contains only the difference between the true and approximated cross-group blocks. For the unit-step update, Eq. (17) gives

$$\begin{aligned} \left\|I - M_*^{-1}A\right\|_A &= \max_i |1 - \lambda_i(M_*^{-1/2}AM_*^{-1/2})| \\ &= \left\|M_*^{-1/2}(A - M_*)M_*^{-1/2}\right\|_2 = \varepsilon. \end{aligned} \tag{22}$$

Using $\bar{A} = D^{-1/2}AD^{-1/2}$ and $\bar{M} = D^{-1/2}M_*D^{-1/2}$, the same contraction factor can be written as

$$\varepsilon = \left\|\bar{M}^{-1/2}E\bar{M}^{-1/2}\right\|_2.$$

We can therefore bound

$$\varepsilon \le \left\|\bar{M}^{-1/2}\right\|_2^2 \|E\|_2.$$

Here $\|E\|_2 = \left\|C - \widehat{C}\right\|_2$, while $\lambda_{\min}(\bar{M}) = 1 - \left\|\widehat{C}\right\|_2$, so

$$\varepsilon \le \frac{\left\|C - \widehat{C}\right\|_2}{1 - \left\|\widehat{C}\right\|_2} < \rho.$$

The final inequality is exactly the approximation condition in Theorem 1. Together with Eqs. (21) and (22), this proves Eq. (12).

Step 3: Local nonlinear convergence. Under the additional assumptions of the theorem, L is C^2 near θ_* and $M(\theta)$ is a continuous SPD preconditioner with $M(\theta_*) = M_*$. We can therefore apply Lemma A.1 with $H_* = A$ and unit step size $\eta = 1$. By Eq. (22), the linearized factor in the lemma is $q_* = \varepsilon$. Step 2 gives $\varepsilon < \rho$, while Step 1 gives $\rho < 1$. Hence any q satisfying $\varepsilon < q < \rho$ also satisfies the lemma's requirement $q_* < q < 1$. The lemma then provides a sufficiently small invariant A-ball in which

$$\|\theta_t - \theta_*\|_A \le q^t \|\theta_0 - \theta_*\|_A,$$

which establishes the claimed local geometric convergence. □

A.2.3 COUPLING AND LOCAL CONVERGENCE IN MOE MODELS

Use the MoE parameters $\theta = (\xi, \phi, \psi)$ from Section 3.2. The partition $\mathcal{B}$ consists of the expert blocks $\phi_{\ell,e}$, router blocks ψ_ℓ, and the specified blocks of ξ, with mixed blocks covering expert–expert, expert–router, router–router, and ξ-related interactions. For hard top-r routing, we consider a neighborhood in which the selected expert sets remain unchanged, so the local objective is C^2.

For the MoE partition $\mathcal{B}$, D retains the within-block Hessian terms and removes all mixed blocks, while M_* retains the same diagonal blocks and may additionally include selected mixed blocks.

Proposition A.1 (Local contraction for general MoE partitions)**.** *Let L be C^2 near a stationary point θ_*, with $A = \nabla^2 L(\theta_*) \succ 0$. Define*

$$D = \mathrm{diag}(\{A_{bb}\}_{b\in\mathcal{B}}), \qquad (M_*)_{bb} = A_{bb} \quad \textit{for every } b \in \mathcal{B},$$

with $M_* \succ 0$, and let

$$\varepsilon_{\mathrm{MoE}} = \left\| M_*^{-1/2}(A - M_*)M_*^{-1/2} \right\|_2, \tag{23}$$

$$q_D = \frac{\lambda_{\max}(D^{-1/2}AD^{-1/2}) - \lambda_{\min}(D^{-1/2}AD^{-1/2})}{\lambda_{\max}(D^{-1/2}AD^{-1/2}) + \lambda_{\min}(D^{-1/2}AD^{-1/2})}. \tag{24}$$

Then

$$\left\| I - M_*^{-1}A \right\|_A = \varepsilon_{\mathrm{MoE}}, \qquad \inf_{\eta>0} \left\| I - \eta D^{-1}A \right\|_A = q_D.$$

The coupling-aware preconditioner has a smaller worst-case local contraction factor than scalar-step block-diagonal preconditioning if and only if $\varepsilon_{\mathrm{MoE}} < q_D$. If $M(\theta)$ is a continuous SPD preconditioner with $M(\theta_*) = M_*$ and $\varepsilon_{\mathrm{MoE}} < 1$, then every $\varepsilon_{\mathrm{MoE}} < q < 1$ admits a neighborhood in which $\|\theta_t - \theta_*\|_A \le q^t \|\theta_0 - \theta_*\|_A$ for all sufficiently close initializations.

Proof. Step 1: Block-diagonal and coupling-aware contraction factors. Let a, b be the smallest and largest eigenvalues of $D^{-1/2}AD^{-1/2}$. Since $A, D \succ 0$, $0 < a \le b$, and Eq. (18) gives

$$\inf_{\eta>0} \left\| I - \eta D^{-1}A \right\|_A = \frac{b-a}{b+a} = q_D < 1.$$

For the coupling-aware preconditioner, applying Eq. (17) with $P = M_*$ and $\eta = 1$ gives

$$\begin{aligned} \left\| I - M_*^{-1}A \right\|_A &= \left\| I - M_*^{-1/2}AM_*^{-1/2} \right\|_2 \\ &= \left\| M_*^{-1/2}(A - M_*)M_*^{-1/2} \right\|_2 = \varepsilon_{\mathrm{MoE}}. \end{aligned}$$

Thus q_D and $\varepsilon_{\mathrm{MoE}}$ are the contraction factors of the two updates being compared, so the coupling-aware preconditioner is better in this local worst-case sense if and only if $\varepsilon_{\mathrm{MoE}} < q_D$.

Step 2: Local nonlinear convergence. Assume now that $M(\theta)$ is a continuous SPD preconditioner with $M(\theta_*) = M_*$. We can then apply Lemma A.1 with $H_* = A$ and unit step size $\eta = 1$. Step 1 shows that the linearized factor in the lemma is $q_* = \varepsilon_{\mathrm{MoE}}$. Therefore, when $\varepsilon_{\mathrm{MoE}} < 1$, every q satisfying $\varepsilon_{\mathrm{MoE}} < q < 1$ meets the lemma's condition $q_* < q < 1$. The lemma then gives a sufficiently small invariant A-ball in which

$$\|\theta_t - \theta_*\|_A \le q^t \|\theta_0 - \theta_*\|_A,$$

establishing local geometric convergence. □

In a general MoE partition, $\varepsilon_{\mathrm{MoE}}$ aggregates all unrepresented expert–expert, expert–router, router–router, and ξ-related interactions. Unlike the two-group setting, the normalized Hessian need not have paired eigenvalues of the form $1 \pm \rho$, so independent pairwise coupling strengths do not characterize the overall local contraction. The block-diagonal reference factor is therefore determined by the full spectrum in Eq. (24), and the coupling-aware preconditioner must satisfy $M_* \succ 0$.

A.3 Proof of Theorem 2 and finite-time escape

A.3.1 Proof of Theorem 2

Theorem 2 (Cross-group curvature and unstable-mode amplification)**.** *Let L be C^2 near a stationary point $\theta_\dagger$ with $H_{11}, H_{22} \succ 0$ and normalized coupling strength $\rho > 1$. Using M_κ from Eq.* (13) *with $\eta_\kappa = c_0(1+\kappa)/(1+\rho)$ for $0 < c_0 < 2$, every positive-curvature mode contracts under the linearized update, while perturbations along the most unstable joint direction are amplified by the per-step factor*

$$a_\kappa = 1 + c_0 \frac{\rho - 1}{\rho + 1} \frac{1+\kappa}{1-\kappa}. \tag{14}$$

This amplification factor $a_\kappa > 1$ increases strictly with κ.

Proof. Step 1: Local linearization. Center the analysis at $\theta_\dagger$ and normalize by the within-group curvature with $y = D^{1/2}(\theta - \theta_\dagger)$, whose inverse is $\theta = \theta_\dagger + D^{-1/2}y$. In this coordinate, define

$$\mathcal{L}(y) := L(\theta_\dagger + D^{-1/2}y).$$

The Hessian and preconditioner at the saddle are

$$\bar{H} = D^{-1/2} H D^{-1/2}, \qquad \bar{M}_\kappa = D^{-1/2} M_\kappa D^{-1/2}.$$

Because L is C^2 and $\theta_\dagger$ is stationary, $\nabla\mathcal{L}(0) = 0$, and a first-order Taylor expansion of the gradient around $y = 0$ gives

$$\nabla\mathcal{L}(y) = \bar{H}y + q(y), \qquad \|q(y)\| = o(\|y\|), \tag{25}$$

where $q(y)$ is the higher-order remainder. The resulting nonlinear update can be written as

$$y_{t+1} = (I - \eta_\kappa \bar{M}_\kappa^{-1}\bar{H})y_t + r_\kappa(y_t), \qquad r_\kappa(y) = -\eta_\kappa \bar{M}_\kappa^{-1} q(y) = o(\|y\|). \tag{26}$$

Step 2: Preconditioned spectrum. As in the proof of Theorem 1, the paired singular-vector directions of C simultaneously diagonalize $\bar{H}$ and $\bar{M}_\kappa$, with respective eigenvalues $1 \pm \sigma_i$ and $1 \pm \kappa\sigma_i/\rho$. Hence the two matrices commute and $B_\kappa = \bar{M}_\kappa^{-1}\bar{H}$ is symmetric. The eigenvalues of $\bar{M}_\kappa$ satisfy $1 \pm \kappa\sigma_i/\rho \geq 1 - \kappa > 0$, so M_κ is SPD. The corresponding paired eigenvalues of B_κ are

$$\lambda_i^+ = \frac{1+\sigma_i}{1+\kappa\sigma_i/\rho}, \qquad \lambda_i^- = \frac{1-\sigma_i}{1-\kappa\sigma_i/\rho}. \tag{27}$$

If the two parameter groups have unequal dimensions, the remaining orthogonal directions have eigenvalue one. To identify the largest positive eigenvalue and the most negative eigenvalue of B_κ, we examine how these two branches vary with σ_i. Both λ_i^+ and $|\lambda_i^-|$ for $\sigma_i > 1$ increase with σ_i, since

$$\frac{d}{d\sigma}\frac{1+\sigma}{1+\kappa\sigma/\rho} = \frac{1-\kappa/\rho}{(1+\kappa\sigma/\rho)^2} > 0, \qquad \frac{d}{d\sigma}\frac{\sigma-1}{1-\kappa\sigma/\rho} = \frac{1-\kappa/\rho}{(1-\kappa\sigma/\rho)^2} > 0.$$

Since $\rho = \max_i \sigma_i$, and both λ_i^+ and $|\lambda_i^-|$ for $\sigma_i > 1$ increase with σ_i, the extreme eigenvalues occur at $\sigma_i = \rho$. Therefore

$$\Lambda_\kappa = \lambda_{\max}(B_\kappa) = \frac{1+\rho}{1+\kappa}, \qquad \Gamma_\kappa = -\lambda_{\min}(B_\kappa) = \frac{\rho-1}{1-\kappa}. \tag{28}$$

Step 3: Stability and unstable-mode amplification. With $\eta_\kappa = c_0/\Lambda_\kappa$ and $0 < c_0 < 2$, each positive eigenvalue of B_κ satisfies

$$0 < \eta_\kappa\lambda_i \leq c_0 < 2 \quad \Longrightarrow \quad |1 - \eta_\kappa\lambda_i| < 1,$$

so all positive-curvature modes are stable under the linearized update. For $\sigma_i = \rho$, the minus direction has eigenvalue $1 - \rho < 0$ in $\bar{H}$ and eigenvalue $-\Gamma_\kappa$ in B_κ. Its multiplier under the linearized update is therefore

$$1 - \eta_\kappa(-\Gamma_\kappa) = 1 + \eta_\kappa\Gamma_\kappa = 1 + c_0\frac{\rho-1}{\rho+1}\frac{1+\kappa}{1-\kappa} = a_\kappa > 1.$$

Finally,

$$\frac{da_\kappa}{d\kappa} = \frac{2c_0(\rho-1)}{(\rho+1)(1-\kappa)^2} > 0,$$

which proves the theorem. □

A.3.2 FROM LOCAL AMPLIFICATION TO FINITE ESCAPE

Theorem 2 shows that increasing κ strengthens the linearized growth along the unstable joint directions. To determine when this growth forces the nonlinear iterates to leave a neighborhood of the same saddle, we use the coordinates $y_t = D^{1/2}(\theta_t - \theta_\dagger)$, learning rate η_κ, and matrix $B_\kappa = \bar{M}_\kappa^{-1}\bar{H}$ from its proof. The nonlinear update in Eq. (26) is

$$y_{t+1} = (I - \eta_\kappa B_\kappa)y_t + r_\kappa(y_t). \tag{29}$$

The term r_κ is the nonlinear Taylor remainder of this update. Let U be the negative eigenspace of $\bar{H}$, which is also the negative eigenspace of B_κ, and let P_U and $P_{U^\perp}$ denote the corresponding orthogonal projections. By Eq. (27), these unstable directions correspond to singular values $\sigma_i(C) > 1$. Along such a direction, the linearized update multiplier is

$$1 - \eta_\kappa \lambda_i^- = 1 + \eta_\kappa \frac{\sigma_i - 1}{1 - \kappa\sigma_i/\rho}.$$

To obtain a growth bound that holds for every direction in U, we use the weakest unstable singular value and define

$$\tau = \min\{\sigma_i(C) : \sigma_i(C) > 1\}, \qquad \chi_\kappa = \eta_\kappa \frac{\tau - 1}{1 - \kappa\tau/\rho} > 0. \tag{30}$$

Since the excess expansion increases with σ_i, χ_κ is the minimum excess expansion over the unstable subspace. Hence $I - \eta_\kappa B_\kappa$ expands every vector in U by at least $1 + \chi_\kappa$, while it does not increase the norm of vectors in $U^\perp$.

Proposition A.2 (Finite local escape with a coupled preconditioner)**.** *Under the hypotheses of Theorem 2, suppose $\nabla^2\mathcal{L}$ is β_D-Lipschitz on $\|y\| \le R_0$, and let $c_\kappa := \eta_\kappa\beta_D/[2(1-\kappa)]$. Choose $\|y_0\| \le R \le R_0$ with $a_0 = \|P_U y_0\| > 0$, $\|P_{U^\perp} y_0\| \le a_0$, and $c_\kappa R \le \chi_\kappa/(2\sqrt{2})$. Then the first exit time $T_\kappa = \inf\{t \ge 0 : \|y_t\| > R\}$ satisfies*

$$T_\kappa \le \left\lfloor \frac{\log(R/a_0)}{\log(1 + \chi_\kappa/2)} \right\rfloor + 1. \tag{31}$$

For a common initialization and radius satisfying these conditions for every compared κ, this upper bound is nonincreasing in κ.

Proof. Step 1: Quadratic remainder bound. Recall from Eq. (25) that $q(y) = \nabla\mathcal{L}(y) - \bar{H}y$ is the gradient Taylor remainder. Since $\nabla^2\mathcal{L}$ is β_D-Lipschitz,

$$\|q(y)\| \le \frac{\beta_D}{2}\|y\|^2. \tag{32}$$

Since $\lambda_{\min}(\bar{M}_\kappa) = 1 - \kappa$ and $\bar{M}_\kappa$ is SPD, $\left\|\bar{M}_\kappa^{-1}\right\|_2 = (1-\kappa)^{-1}$. Combining this identity with $r_\kappa(y) = -\eta_\kappa\bar{M}_\kappa^{-1}q(y)$ from Eq. (26) and the bound in Eq. (32) gives

$$\begin{aligned}\|r_\kappa(y)\| &\le \eta_\kappa \left\|\bar{M}_\kappa^{-1}\right\|_2 \|q(y)\| \\ &\le \frac{\eta_\kappa\beta_D}{2(1-\kappa)}\|y\|^2 = c_\kappa\|y\|^2.\end{aligned}$$

Step 2: Growth of the unstable component. Write $u_t = P_U y_t$ and $v_t = P_{U^\perp} y_t$. The initialization satisfies $\|v_0\| \le \|u_0\|$, and we show inductively that this relation persists while the trajectory remains inside the radius-R ball. Suppose $\|y_t\| \le R$ and $\|v_t\| \le \|u_t\|$. By the remainder bound from Step 1,

$$\|r_\kappa(y_t)\| \le c_\kappa\|y_t\|^2 \le c_\kappa R\|y_t\|.$$

Because $u_t \perp v_t$ and $\|v_t\| \le \|u_t\|$,

$$\|y_t\|^2 = \|u_t\|^2 + \|v_t\|^2 \le 2\|u_t\|^2,$$

so using the assumption $c_\kappa R \le \chi_\kappa/(2\sqrt{2})$

$$\|r_\kappa(y_t)\| \le c_\kappa R\sqrt{2}\|u_t\| \le \tfrac{1}{2}\chi_\kappa\|u_t\|.$$

Since U and $U^\perp$ are invariant subspaces of B_κ, projecting Eq. (29) gives

$$u_{t+1} = (I - \eta_\kappa B_\kappa)u_t + P_U r_\kappa(y_t), \qquad v_{t+1} = (I - \eta_\kappa B_\kappa)v_t + P_{U^\perp} r_\kappa(y_t).$$

The linear part expands vectors in U by at least $1 + \chi_\kappa$ and does not increase the norm of vectors in $U^\perp$. Hence

$$\|u_{t+1}\| \geq (1 + \chi_\kappa)\,\|u_t\| - \|r_\kappa(y_t)\| \geq (1 + \chi_\kappa/2)\,\|u_t\|\,,$$
$$\|v_{t+1}\| \leq \|v_t\| + \|r_\kappa(y_t)\| \leq (1 + \chi_\kappa/2)\,\|u_t\| \leq \|u_{t+1}\|\,.$$

Thus $\|v_t\| \leq \|u_t\|$ persists until exit, while repeated application of the first bound gives

$$\|u_t\| \geq (1 + \chi_\kappa/2)^t a_0.$$

To obtain an upper bound on the first exit time, choose

$$N = \left\lfloor \frac{\log(R/a_0)}{\log(1 + \chi_\kappa/2)} \right\rfloor + 1,$$

so that $(1 + \chi_\kappa/2)^N a_0 > R$. If the trajectory exits before step N, then $T_\kappa \leq N$ immediately. Otherwise, it remains in the radius-R ball through step N, so the growth bound applies up to that step and gives $\|u_N\| > R$. Since $u_N = P_U y_N$ is an orthogonal component of y_N, this implies $\|y_N\| > R$, contradicting that the trajectory remained in the ball through step N. Hence $T_\kappa \leq N$, which proves Eq. (31).

Step 3: Exit-time comparison. The subspace U is independent of κ, and

$$\chi_\kappa = \frac{c_0(\tau - 1)}{1 + \rho} \frac{1 + \kappa}{1 - \kappa\tau/\rho}, \qquad \frac{d}{d\kappa} \frac{1 + \kappa}{1 - \kappa\tau/\rho} = \frac{1 + \tau/\rho}{(1 - \kappa\tau/\rho)^2} > 0.$$

For fixed R, a_0, increasing χ_κ cannot increase the bound in Eq. (31). □

When only one singular value exceeds one, $\tau = \rho$, and the sufficient radius can be chosen independently of κ as $R \leq (\rho - 1)/(\sqrt{2}\beta_D)$, with $R \leq R_0$ still required. With multiple unstable modes, the common admissible neighborhood may shrink as κ increases. Accordingly, the proposition compares guaranteed exit-time upper bounds for a common admissible neighborhood rather than ordering the actual exit times of all nonlinear trajectories.

B Additional related work

B.1 Task structure and multi-task optimization

Beyond balancing tasks for a fixed model, multi-task systems can alter which computations are shared. Cross-stitch networks (Misra et al., 2016) learn feature combinations across tasks, while attention-based sharing (Liu et al., 2019) and AdaShare (Sun et al., 2020) select features or layers for each task. Standley et al. (2020) and Fifty et al. (2021) study which task groupings support joint learning, while selective task-group updates (Jeong & Yoon, 2025) use task affinity to organize updates during training. Multi-gate mixture-of-experts (Ma et al., 2018) and progressive layered extraction (Tang et al., 2020) combine expert representations through task-specific gates, linking architectural sharing to task relationships. Sparsely gated experts (Shazeer et al., 2017), Switch Transformers (Fedus et al., 2022), and expert-choice routing (Zhou et al., 2022) instead make participation input-dependent. These designs determine which parameters participate together, whereas CoPEN studies their joint update geometry for a given architecture.

For a given sharing structure, multi-task gradients can also be coordinated beyond scalar loss weighting. Uncertainty weighting (Kendall et al., 2018) adapts task loss scales, while IMTL (Liu et al., 2021b), Aligned-MTL (Senushkin et al., 2023), and RotoGrad (Javaloy & Valera, 2022) address gradient balance, conditioning, or homogenization. GradDrop (Chen et al., 2020) uses sign-based masking, Gradient Vaccine (Wang et al., 2021) encourages gradient alignment, and MoCo (Fernando et al., 2023) corrects stochastic multi-objective gradient bias. Task priority (Jeong & Yoon, 2024) brings this coordination to the parameter level by modifying gradients according to task priority. This differs from modeling cross-group curvature for a fixed aggregate objective, without assigning task priorities to individual parameters.

Preference-aware methods address a different choice of task trade-off. EPO (Mahapatra & Rajan, 2020) combines descent with controlled ascent to seek preference-aligned Pareto solutions, while Pareto-front hypernetworks (Navon et al., 2021) and PMGDA (Zhang et al., 2024) learn or select preference-dependent solutions. CoPEN does not replace these choices of task weighting, gradient coordination, or preference. It addresses the parameter-space geometry of the resulting fixed training objective.

B.2 Tensor-wise statistics and structured preconditioning

Natural gradient (Amari, 1998) motivates conditioning through a parameter-space metric, but gradient second moments should not generally be identified with the Fisher or Hessian (Kunstner et al., 2019). Shampoo (Gupta et al., 2018) uses tensor-structured gradient statistics, and SOAP (Vyas et al., 2025) applies adaptive moment normalization in their eigenbasis. Scalable and distributed implementations address the cost of maintaining these matrix preconditioners (Anil et al., 2020; Shi et al., 2023). Sketchy (Feinberg et al., 2023) reduces this cost through low-rank sketches, while Yen et al. (2023) approximate diagonal blocks with low-rank matrices that share a basis. Sharing a basis across blocks does not itself retain off-diagonal interactions between those blocks.

Recent analyses further distinguish the statistics being estimated from how they transform the update. Lin et al. (2024) examine the role of matrix roots in adaptive methods. Purifying Shampoo (Eschenhagen et al., 2025) separates eigenvalue and eigenbasis updates to study scaling and staleness, while Lin et al. (2026) reinterpret Shampoo and SOAP through Kullback–Leibler covariance estimation. These works refine within-tensor conditioning. CoPEN retains this stage and uses a separate projected curvature model to adjust selected cross-group directions.

B.3 Subspace optimization and cross-block corrections

Structured curvature methods need not treat parameter blocks independently. K-FAC (Martens & Grosse, 2015) develops both block-diagonal and block-tridiagonal inverse approximations, while EKFAC (George et al., 2018) refines scaling in a Kronecker-factored eigenbasis. Kronecker-factored quasi-Newton methods (Goldfarb et al., 2020) provide another route to layerwise curvature approximation. More directly, two-level K-FAC (Tselepidis et al., 2020) supplements local preconditioning with a coarse correction that captures inter-layer interactions. Such corrections are not uniformly beneficial: Koroko et al. (2025) find limited gains from several coarse spaces in their experiments. These findings motivate identifying a small set of optimization-relevant cross-group interactions rather than treating all off-diagonal structure uniformly.

Matrix-free and subspace curvature methods also have a broader history. Hessian-free optimization (Martens, 2010) uses iterative curvature-vector solves, Krylov subspace descent (Vinyals & Povey, 2012) optimizes within a curvature-informed subspace, and SONIA (Jahani et al., 2021) combines a second-order-type subspace step with scaled steepest descent in its orthogonal complement. CoPEN instead constructs a joint reduced curvature model from directions supported on explicit structural groups, so its off-diagonal blocks directly represent interactions between those groups. The resulting correction acts only within this joint subspace on top of the tensor-wise update, while leaving its orthogonal complement unchanged rather than constructing a new full-space search direction.

B.4 Curvature estimation and nonconvex geometry

Matrix-free products avoid forming a full curvature matrix (Pearlmutter, 1994; Schraudolph, 2002). Stochastic quasi-Newton methods (Byrd et al., 2016) collect curvature less frequently than parameter updates. This cost–freshness trade-off is related to, but distinct from, CoPEN's periodically refreshed projected metric.

Hierarchical parameterizations can produce local minima and plateaus (Fukumizu & Amari, 2000), while adaptive conditioning can influence stochastic saddle escape (Staib et al., 2019). Our analysis instead isolates cross-group Hessian coupling that destabilizes individually stable groups. This mechanism differs from CoPEN's practical positive-semidefinite metric, which models joint output sensitivity rather than detecting negative Hessian curvature.

C IMPLEMENTATION DETAILS OF COPEN

Adjustment scope and grouping. For ViT, the adjustment scope contains the final three transformer blocks, the final normalization, and all ten task heads. The blocks form separate basis groups, with the final normalization assigned to the last block, and the heads form another group. For MoE, the scope contains the experts and routers of all routed layers and all ten heads. We evaluate both layer-wise grouping and variants that separate experts and routers into distinct groups. All parameters are trained. The scope limits only the cross-group adjustment.

Tensor-wise update and basis construction. The tensor-wise update b_t uses SOAP (Vyas et al., 2025): adaptive first and second moments are maintained in eigenbases derived from Shampoo statistics (Gupta et al., 2018), and the update is transformed back to the original coordinates. Tensor-wise moments use $\beta_1 = \beta_2 = 0.95$ and $\epsilon = 10^{-8}$, with eigenbasis refreshes every 10 updates and maximum preconditioned dimension 2,048. These tensor-wise eigenbases are maintained independently of the coupling basis Q_t. For each structural group, the candidate directions from b_t, g_t, and the displacement since the previous coupling refresh are orthonormalized as described in Section 5, retaining up to three basis directions per group.

Projected-curvature estimation. For squared-error and cross-entropy tasks, the output metric uses the analytic loss curvature. For absolute-error tasks, a valid residual r_j receives an IRLS weight proportional to $1/\max(|r_j|, 0.1)$, including task and valid-element normalization (Holland & Welsch, 1977). This changes only the curvature surrogate, not the objective or gradient. It is neither the Hessian nor the native GGN of the absolute-error loss. Router auxiliary losses contribute to the training gradient, but not to the output metric.

Curvature is estimated on an eight-image subset of the training minibatch, preserving complete MoE routing groups. During curvature differentiation, the discrete expert assignments are held fixed, while the selected gate weights remain differentiable.

Optimization settings. The CoPEN configurations in Table 1 use $\eta = 10^{-4}$ for ViT and $\eta = 1.25 \times 10^{-4}$ for MoE. Both use $\lambda_c = 0.1$, $\gamma = 0.5$, and decoupled weight decay 10^{-6}. Global gradient clipping uses norm 1. The reduced solve uses absolute damping λ_c, rather than scaling it by the mean projected curvature. The main full-model experiments refresh the coupling basis Q_t and projected curvature K_t every 25 updates. Table 8 lists the search ranges.

D CONTROLLED TOY EXPERIMENTS

We use three controlled toys to study coupling: a shared-MLP toy for shared–head preconditioning, a coupled-saddle toy for escape dynamics, and an MoE toy for expert–router preconditioning. Within each toy, methods share the data, initialization, and evaluation step budget.

D.1 TOY SETUPS AND TUNING

D.1.1 SHARED-MLP TOY

This toy isolates coupling between shared layers and task-specific heads. For $K = 4$ tasks and inputs $x \in \mathbb{R}^6$, the model and objective are

$$\begin{aligned} h_1(x) &= \tanh(W_1 x + b_1), \qquad h_2(x) = \tanh(W_2 h_1(x) + b_2), \\ f_k(x) &= a_k^\top h_2(x) + c_k, \qquad L(\theta) = \frac{1}{2KN} \sum_{k=1}^{K} \sum_{i=1}^{N} (f_k(x_i) - y_{ki})^2. \end{aligned} \tag{33}$$

Here $W_1 \in \mathbb{R}^{8\times 6}$ and $W_2 \in \mathbb{R}^{8\times 8}$. The two shared layers contain 128 parameters in total, and the four linear task heads contain 36. We use $N = 256$ inputs sampled from $\mathcal{N}(0, I_6)$ and reuse them across tasks. Targets are noiseless predictions of a teacher with the same architecture. Teacher weights use fan-in-scaled Gaussian initialization with scale one and bias standard deviation 0.1. The teacher heads are jointly rescaled to unit root-mean-square output on 2,048 independent Gaussian

inputs to set the overall target scale. Student weights use scale 0.3, with bias standard deviation 0.03.

For CoPEN, the two shared layers and four task heads form six separate parameter groups. Block-diag. uses the same groups with cross-group entries removed, so the comparison isolates the effect of retaining coupling. The damping is scaled by the mean curvature $\mathrm{tr}(G)/n$, where G is the exact output-Jacobian Gram matrix and n is the number of parameters.

D.1.2 COUPLED-SADDLE TOY

To isolate saddle escape caused purely by cross-group coupling, we use

$$L(n,t) = \tfrac{1}{2}An^2 + \Lambda nt + \tfrac{1}{2}Bt^2 + \tfrac{1}{4}\mu t^4, \qquad H(0) = \begin{pmatrix} A & \Lambda \\ \Lambda & B \end{pmatrix}, \tag{34}$$

where n and t represent shared and task-specific parameter groups. We set $A = 3$, $B = 0.5$, $\Lambda = 2.6$, and $\mu = 1$. Although each coordinate has positive curvature in isolation, $\Lambda^2 > AB$ makes the joint Hessian at the origin indefinite. The quartic term bounds the objective away from the saddle and yields finite minima after escape.

We apply CoPEN with the two coordinates as separate groups. Block-diag. uses the same construction with the cross-group entry removed, isolating the effect of cross-group coupling. We use 32 paired random initializations of norm 10^{-4} around the origin and run both methods for 500 updates.

D.1.3 MOE TOY

This toy isolates coupling between experts and routers in a small densely routed MoE. Each of its two routed layers has four linear experts and a learned softmax router. For task k, the input is $h_0 = [x; \mathrm{onehot}(k)] \in \mathbb{R}^{10}$, with $x \sim \mathcal{N}(0, I_6)$. The model is

$$\begin{aligned} p_\ell(h) &= \mathrm{softmax}(R_\ell h + r_\ell), \qquad u_\ell(h) = \sum_{e=1}^{4} p_{\ell,e}(h)(W_{\ell,e}h + b_{\ell,e}), \\ h_1 &= \tanh(u_1(h_0)), \qquad f_k(x) = a^\top u_2(h_1). \end{aligned} \tag{35}$$

Both layers have width four, followed by a fixed unit-normalized linear readout a. The model has 256 expert parameters and 64 router parameters, with no trainable task-specific heads. Task identity provides input context rather than selecting an expert, and routing is dense, so all experts participate in every prediction.

The objective is the equal-task squared loss in Eq. (33). Targets are generated by the same width-eight shared-MLP teacher described above. We use 1,024 training inputs, 1,024 validation inputs, and 4,096 independent test inputs, each reused across the four task contexts. Expert weights use fan-in-scaled Gaussian initialization with scale 0.3 and bias standard deviation 0.03. Router weights use scale 0.5, with biases $(0.15, 0.05, -0.05, -0.15)$. Updates are full-batch and double precision, and all methods share the input splits and initialization.

For CoPEN, the eight experts and two routers form ten separate parameter groups. Block-diag. uses the same groups with cross-group entries removed.

Trajectory visualization. For experts and routers separately, we project each method's parameter displacement onto a one-dimensional axis v shared across methods,

$$q_{m,t} = v^\top(\theta_{m,t} - \theta_0). \tag{36}$$

Each axis maximizes the minimum retained squared displacement across methods. The resulting projections retain at least 13.7% of expert displacement and 24.9% of router displacement. At step t, the background at coordinate a evaluates

$$L(\theta_{m,t} + (a - q_{m,t})v)\,, \tag{37}$$

with all other coordinates held at the recorded iterate. The backgrounds are therefore time-dependent one-dimensional loss slices rather than a fixed two-dimensional loss landscape. Because these projections retain only part of the full parameter displacement, we use them as qualitative visualizations rather than quantitative measures of optimization progress.

D.1.4 HYPERPARAMETER TUNING

Table 3 lists the search spaces for the shared-MLP and MoE toys. Each method receives a budget of 24 configurations. We first evaluate a common step-size grid, including each method's default step size, and then refine around the best step size η_* using $0.3\eta_*$ and $3\eta_*$.

For the shared-MLP toy, method-specific hyperparameters are varied at the selected step size. For the MoE toy, method-specific configurations are additionally paired with $\{0.1, 0.3, 3, 10\}\eta_*$ until the 24-configuration budget is reached. MoE configurations are evaluated for 300 updates for hyperparameter selection, and the selected configuration is evaluated for 3,000 updates. All displayed runs use a constant learning rate. The coupled-saddle diagnostic is not part of this hyperparameter search.

MTO methods use Adam. The exact-diagonal control uses $d_j = -g_j/(|H_{jj}| + \lambda\,\overline{|H|})$, where $\overline{|H|}$ is the mean of $|H_{kk}|$.

Table 3: **Hyperparameter search spaces for the controlled toys.** Toy-specific settings are marked in parentheses.

Method	Hyperparameter	Candidates
All methods	Step size η	$\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1,$ $10, 100, 1000, 3000\}$ and each method's default
	Step-size refinement	$\{0.3\eta_*, 3\eta_*\}$ around the best η_*
	Search budget	24 settings per method
Optimizers		
Adam	First-moment decay β_1	$\{.8, .9, .95\}$
(Kingma & Ba, 2015)	Second-moment decay β_2	$\{.9, .99, .999\}$
Muon	Momentum	$\{.9, .95, .99\}$
(Jordan, 2024)	Adam-to-Muon step-size ratio for non-Muon parameters	$\{.1, .3, 1\}$
SOAP	Moment decays (β_1, β_2)	$\{(.9, .99), (.9, .999),$ $(.95, .95), (.95, .99)\}$
(Vyas et al., 2025)	Preconditioning frequency f	$\{1, 5, 10\}$
Shampoo (Gupta et al., 2018)	Initial factor scale ϵ ($L_0 = R_0 = \epsilon I$)	$\{10^{-8}, 10^{-6}, 10^{-4}\}$
K-FAC	Tikhonov damping λ (relative)	$\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$
(Martens & Grosse, 2015)	Momentum μ (fixed)	$\{0, .9\}$
AdaHessian	Second-moment decay β_2	$\{.9, .99, .999\}$
(Yao et al., 2021)	Denominator stabilizer ϵ	$\{10^{-4}, 10^{-3}\}$
Diagonal Hessian	Damping λ (relative)	$\{10^{-6}, 10^{-4}, 10^{-3}, 10^{-2},$ $10^{-1}, 1, 10\}$
Multi-task optimization (MTO), with Adam updates		
PCGrad (Yu et al., 2020)	Adam β_1 / β_2	$\{.8, .9, .95\}$ / $\{.9, .99, .999\}$
CAGrad	Conflict-averse radius c	$\{.2, .4, .8\}$
(Liu et al., 2021a)	Adam β_2	$\{.99, .999\}$
MGDA (Sener & Koltun, 2018)	Adam β_1 / β_2	$\{.8, .9, .95\}$ / $\{.9, .99, .999\}$
GradNorm	Asymmetry α	$\{.1, .5, 1, 1.5\}$
(Chen et al., 2018)	Task-weight learning rate	$\{.001, .01, .025\}$
Nash-MTL	Norm clipping of the shared gradient and update (0: none)	$\{0, 1, 10\}$
(Navon et al., 2022)	Adam β_2	$\{.99, .999\}$
FAMO	Task-logit decay γ	$\{0, .001, .01\}$
(Liu et al., 2023)	Task-logit learning rate β	$\{.001, .01, .025\}$
Coupling comparison		
CoPEN (Ours)	Damping coefficient λ (MLP)	$\{10^{-6}, 10^{-4}, 10^{-3}, 10^{-2},$ $10^{-1}, 1, 10\}$

Table 4: **MoE toy comparison.** Losses ($\times 10^{-3}$) after 3,000 updates, averaged over six runs.

Method	Train	Validation	Test
Optimizers			
Adam (Kingma & Ba, 2015)	4.581	5.800	5.965
K-FAC (Martens & Grosse, 2015)	17.745	20.899	20.045
Shampoo (Gupta et al., 2018)	6.762	8.189	8.087
AdaHessian (Yao et al., 2021)	5.973	7.461	7.380
Muon (Jordan, 2024)	5.779	7.269	7.056
SOAP (Vyas et al., 2025)	4.612	5.939	5.920
Diagonal Hessian	7.813	9.540	9.165
Multi-task optimization (MTO)			
GradNorm (Chen et al., 2018)	5.421	6.893	6.837
MGDA (Sener & Koltun, 2018)	7.290	8.490	8.392
PCGrad (Yu et al., 2020)	5.185	6.427	6.378
CAGrad (Liu et al., 2021a)	5.154	6.574	6.344
Nash-MTL (Navon et al., 2022)	4.894	5.968	5.920
FAMO (Liu et al., 2023)	8.531	10.064	10.021
Coupling comparison			
Block-diag.	5.689	6.837	6.897
CoPEN	**4.077**	**5.319**	**5.650**

Table 5: **Toy time-to-target comparison.** Targets are the reference methods' losses after 3,000 updates, and speedup is 3,000 divided by the CoPEN update count.

Toy (loss)	Reference	Target	CoPEN updates	Update speedup
Shared MLP (train)	Adam (Kingma & Ba, 2015)	1.292×10^{-4}	44	$68.18\times$
Shared MLP (train)	Block-diag.	8.152×10^{-7}	892	$3.36\times$
MoE (validation)	Adam (Kingma & Ba, 2015)	5.800×10^{-3}	600	$5.00\times$

D.2 Toy results

D.2.1 Shared-MLP toy

Across the six paired runs in Fig. 2, CoPEN attains a lower final training loss than Block-diag.

D.2.2 Coupled-saddle toy

Figure 1 shows the trajectory for the first of the 32 paired initializations. Starting from the same perturbation, CoPEN leaves the saddle neighborhood earlier than Block-diag., while both methods approach the same minimum. This behavior is consistent with the unstable-mode amplification in Theorem 2.

D.2.3 MoE toy

Table 4 summarizes six paired runs of 3,000 updates. CoPEN achieves the lowest mean training, validation, and test losses. Against Block-diag., it improves training and test loss in all six paired runs.

Updates to matched loss. Table 5 shows that CoPEN reaches Block-diag.'s 3,000-update Shared-MLP loss after 892 updates and Adam's 3,000-update MoE validation loss after 600 updates.

Figure 3 shows the corresponding training and test curves. After 2,000 updates, CoPEN's training loss continues to decrease while its test loss rises, so later optimization progress does not necessarily improve generalization. These are fixed-budget comparisons, not claims of full convergence.

Expert and router trajectories. Figure 4 provides a qualitative view of expert and router motion from the same initialization. The trajectories suggest that retaining expert–router coupling changes

how optimization motion is distributed across the two parameter types, rather than simply rescaling the same update path. This provides a qualitative counterpart to the coupling diagnostics below. As described in Section D.1.3, these projections retain only part of the full displacement and are not used as quantitative measures of optimization progress.

D.3 Cross-group coupling diagnostics

To diagnose cross-group coupling in the controlled toys, we measure two complementary quantities from the exact output-Jacobian Gram matrix G at recorded iterates. Output-effect overlap asks whether two parameter groups have aligned effects on the model outputs. Lost progress asks whether that interaction matters for local optimization by measuring the decrease lost when their cross-group curvature is removed. The groups are the two shared layers and four task heads in the shared-MLP toy, and the eight experts and two routers in the MoE toy.

Output-effect overlap. For groups i and j with n_i and n_j parameters, define

$$\begin{aligned} C_{ij} &= (G_{ii} + \lambda_i I)^{-1/2} G_{ij} (G_{jj} + \lambda_j I)^{-1/2}, \\ \lambda_b &= 10^{-3} \operatorname{tr}(G_{bb})/n_b, \\ \omega_{ij} &= \left(\frac{\|C_{ij}\|_F^2}{\min(n_i, n_j)} \right)^{1/2}. \end{aligned} \tag{38}$$

The singular values of C_{ij} quantify the regularized overlap between the output-space effects of the two groups. We report their root-mean-square value $\omega_{ij} \in [0, 1]$. This statistic is distinct from the spectral coupling strength $\rho = \|C\|_2$ used in Section 4.1. A value of zero indicates orthogonal output effects, while larger values indicate broader overlap across directions. We use the RMS statistic because the largest singular value is close to one for almost every pair.

Predicted progress lost without coupling. For each pair, we restrict the damped local quadratic model to groups i and j while holding the remaining parameters fixed. We compare a joint direction that uses the full two-group curvature with a block-diagonal direction in which the cross-group block is removed. Let $\Delta_{ij}^{\text{joint}}$ and $\Delta_{ij}^{\text{block}}$ denote their predicted decreases. Each direction is evaluated at its best scalar step length so that the comparison reflects the direction rather than its scale. This is a diagnostic of the local quadratic model rather than the realized decrease of an optimizer step. We report

$$\text{Lost}_{ij} = 1 - \frac{\Delta_{ij}^{\text{block}}}{\Delta_{ij}^{\text{joint}}}. \tag{39}$$

Thus $\text{Lost}_{ij} = 0$ means that removing the cross-group block does not reduce the best predicted decrease, while larger values indicate greater dependence on the interaction. Pairwise statistics are averaged over pairs of the same type and six runs. We report ω_{ij} at initialization and after 3,000 Adam updates. We report Lost_{ij} after 3,000 updates along the Adam, CoPEN, and Block-diag. trajectories. The final row evaluates all groups jointly.

Table 6: **Cross-group diagnostics in the shared-MLP toy.** Overlap and predicted-progress loss are defined in Section D.3.

	Overlap ω		Lost progress at 3K		
Group pair	Init.	Adam 3K	Adam	CoPEN	Block-diag.
Shared layer 2 and a head	0.83	0.91	0.13	0.15	0.81
Shared layer 1 and a head	0.82	0.76	0.26	0.12	0.43
Shared layers 1 and 2	0.73	0.72	0.59	0.14	0.05
All groups jointly	–	–	0.18	0.58	0.01

In the shared-MLP toy (Table 6), shared–head interactions show high normalized overlap, especially for the second shared layer. Along the Block-diag. trajectory, removing the second-layer–head interaction discards a large fraction of the predicted local decrease.

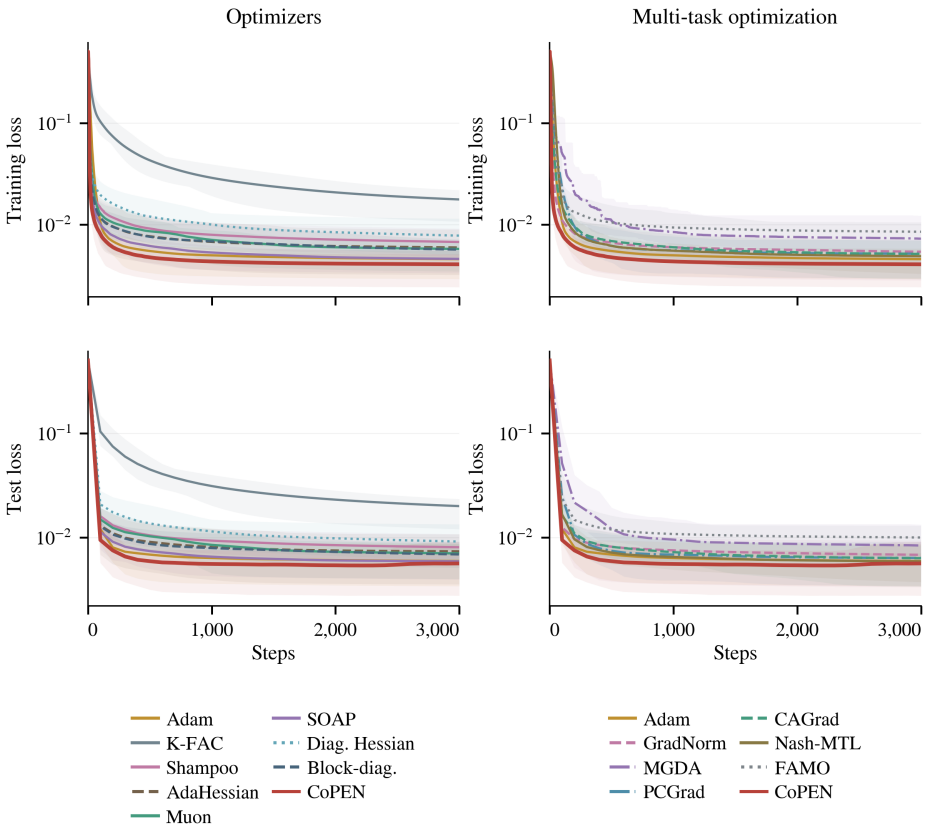


Figure 3: **Training and test loss on the MoE toy.** Left: optimizers. Right: MTO methods. Shaded regions show the range over six runs.

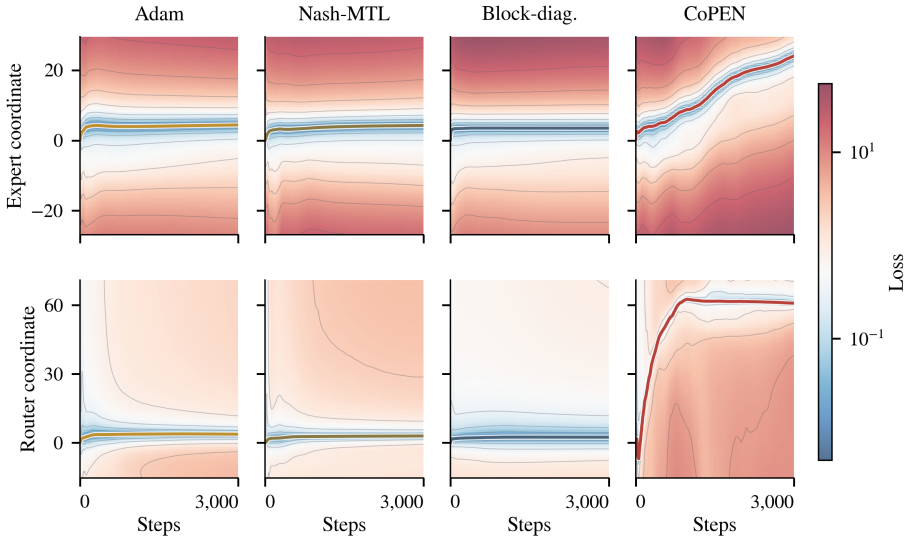


Figure 4: **Expert (top) and router (bottom) trajectories on the MoE toy.** Background colors show the time-dependent loss slices defined in Eq. (37).

Table 7: **Cross-group diagnostics in the MoE toy.** Overlap and predicted-progress loss are defined in Section D.3.

	Overlap ω		Lost progress at 3K		
Group pair	Init.	Adam 3K	Adam	CoPEN	Block-diag.
Same layer, expert and router	0.50	0.46	0.59	0.38	0.53
Same layer, two experts	0.57	0.39	0.54	0.26	0.60
Across layers, two routers	0.49	0.49	0.29	0.27	0.32
Across layers, expert and router	0.50	0.43	0.24	0.20	0.28
Across layers, two experts	0.46	0.33	0.18	0.15	0.33
All groups jointly	–	–	0.02	0.10	0.07

In the MoE toy (Table 7), within-layer expert–router and expert–expert pairs lose more predicted progress when coupling is removed than the corresponding cross-layer pairs. This indicates that the most optimization-relevant interactions are concentrated within routed layers, even though nonzero overlap is also present across layers.

E Experimental setup for ViT and MoE

E.1 Datasets and architectures

Taskonomy. The ViT and MoE experiments use the complete training and validation partitions of Taskonomy's tiny split (Zamir et al., 2018): 274,880 valid training images from 25 buildings and 51,753 valid validation images from five buildings. We retain images with valid files for all ten dense-prediction tasks:

- **Depth:** Euclidean depth (Depth-Euc.) and z-buffer depth (Depth-Z.).
- **Edges:** occlusion edges (Edge-Occ.) and texture edges (Edge-Tex.).
- **Keypoints:** 2D and 3D keypoint prediction (Keypt-2D and Keypt-3D).
- **Surface normals** (Normals).
- **Principal curvature** (Curv.).
- **Reshading** (Reshad.).
- **Semantic segmentation** (Sem.).

Images and targets are resized to 224×224. Inputs use the normalization of the pretrained backbone, and resized targets undergo task-specific normalization based on the Taskonomy reference implementation (Zamir et al., 2018).[1] We apply task-specific validity masks and use horizontal flips with corresponding target transformations during training. Within each architecture, all methods use identical preprocessing.

We use mean absolute error for eight regression tasks, mean squared error for principal curvature, and cross-entropy for semantic segmentation. Each task loss is averaged over its valid elements, including channels. The data objective is the equal mean of the ten task losses. For MoE, the router auxiliary loss is added once to the training objective and is excluded from the reported data-loss average.

Model architectures. The dense model is ViT-B/16 (Dosovitskiy et al., 2021), initialized from the ImageNet-21k AugReg checkpoint (Steiner et al., 2022). The routed model is a PyTorch port of V-MoE-B/16 (Riquelme et al., 2021), initialized from its ImageNet-21k checkpoint. Both have 12 transformer blocks, hidden width 768, and 12 attention heads. V-MoE replaces the feed-forward module in every second block with eight experts and top-2 routing. Routing groups contain eight images (1,576 tokens), with capacity factor 1.05 and training-time logit noise of standard deviation

[1] Taskonomy reference preprocessing code.

$1/8$. The router importance and load penalties each have coefficient 0.005 and are summed across routed layers. All MoE comparisons use the same FP32 dispatcher.

Each task has a newly initialized 3×3 convolutional head on the final 14×14 patch-feature map, followed by bilinear upsampling. The output-channel counts are one for each scalar task, three for normals, two for principal curvature, and 17 for segmentation. The entire backbone and all task heads are trained. The adjustment scope in Section C limits only the additional coupling adjustment.

E.2 Baselines and hyperparameter search

Baseline configurations. Table 8 lists the compared methods and their search spaces. All methods use the same hyperparameter-search budget of 24 configurations, and the configuration selected by periodic-validation loss is used for the reported comparison. The MTO methods use AdamW. PCGrad projects task gradients on the shared backbone while preserving task-head gradients, and the MoE router auxiliary gradient is added outside task-gradient projection.

Full-model hyperparameter search. Table 8 lists the searched hyperparameters of each method. The "Shared with AdamW" rows indicate which AdamW domains a method searches, and the remaining rows list its method-specific hyperparameters. These are varied within the listed ranges rather than exhaustively crossed.

E.3 Training schedules and evaluation

Common training setup. All methods update all backbone and task-head parameters on all 274,880 training images with batch size 32, identical pretrained initialization, preprocessing, and shuffled training stream. Every learning-rate schedule starts with 256 warmup updates.

Plateau-triggered cosine decay. After warmup, the learning rate stays at its peak until a validation-loss plateau, then decays by cosine to 1% of the peak. Training continues at this rate until validation loss stops improving. Table 1, Sections F.1 and F.2, and Fig. 5 use this schedule.

Constant-learning-rate schedule. After warmup, the learning rate stays constant. Each run's checkpoint is selected by its best periodic-validation loss within 24,000 updates. Section F.3 uses this schedule.

Validation. Reported endpoint losses and Δ_m values use the complete validation split of 51,753 images from five buildings. Hyperparameter and checkpoint selection and plotted validation curves use fixed subsets of this split, evaluated every 512 updates. Plateau-triggered cosine-decay runs use an 11,094-image validation subset from two validation buildings for periodic evaluation, and constant-LR runs use a 2,048-image validation subset drawn across all five buildings. For each evaluation, we sum task-loss numerators and valid-element counts over all batches before division, then average the ten task losses equally. MoE padding that completes a routing group is excluded from this reduction. All task-wise columns in a row use the same checkpoint. Plateau-triggered cosine-decay training curves show means of 64 successive online minibatch losses. Constant-LR training curves show recorded minibatch means between evaluations, with no additional smoothing. All learning curves use a common display range of 24,000 updates.

Multi-task performance relative to single-task learning. Following the average relative-gain convention (Vandenhende et al., 2022), we use each task's validation loss as its performance measure and define

$$\Delta_m(\%) = \frac{100}{T} \sum_{k=1}^{T} \frac{L_k^{\mathrm{STL}} - L_k^{\mathrm{MTL}}}{L_k^{\mathrm{STL}}}. \tag{40}$$

All losses are lower-is-better, so positive values indicate an average improvement over single-task learning. We train ten separate single-task models for each architecture using AdamW, with the same backbone initialization, training data, and preprocessing as the multi-task runs. Each model optimizes only its designated task loss, with the router auxiliary loss retained for MoE. The learning

Table 8: **Model-scale hyperparameter search spaces.**

Method	Hyperparameter	Candidates
All methods	Search budget	24 settings per method
AdamW	Learning rate	$[10^{-5}, 10^{-1}]$, log scale
(Loshchilov & Hutter, 2019)	Weight decay	$\{0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$
	Moment decays β_1 / β_2	$\{.85, .9, .95\}$ / $\{.95, .99, .999\}$
	Denominator stabilizer ϵ	$\{10^{-8}, 10^{-6}, 10^{-4}\}$
Optimizers		
Shampoo	Shared with AdamW	Learning rate, weight decay, ϵ
(Gupta et al., 2018)	Preconditioner root refresh interval	$\{1, 5, 10\}$
K-FAC	Shared with AdamW	Learning rate, weight decay
(Martens & Grosse, 2015)	Relative Tikhonov damping λ	$[10^{-3}, 10^{-1}]$, log scale
	Inverse refresh interval T_3	$\{5, 10, 25\}$
AdaHessian	Shared with AdamW	All AdamW hyperparameters
(Yao et al., 2021)	Hessian-estimate refresh interval	$\{1, 5, 10\}$
Muon	Shared with AdamW	All AdamW hyperparameters (non-matrix parameters)
(Jordan, 2024)	Matrix learning rate	$[10^{-5}, 10^{-1}]$, log scale
	Momentum	$\{.9, .95, .99\}$
SOAP	Shared with AdamW	All AdamW hyperparameters
(Vyas et al., 2025)	Preconditioning frequency f	$\{5, 10, 25\}$
	Maximum preconditioned dimension	$\{2048, 4096\}$
SASSHA	Shared with AdamW	All AdamW hyperparameters
(Shin et al., 2025)	Perturbation radius ρ	$\{.01, .05, .1\}$
	Lazy Hessian update interval k	$\{1, 5, 10\}$
NorMuon	Shared with Muon	All Muon hyperparameters
(Li et al., 2025)	Neuron-wise second-moment decay β_2	Same draw as AdamW β_2
Trion	Shared with Muon	All Muon hyperparameters
(Modoranu et al., 2026)	Rank fraction r/d	$\{.125, .25, .5\}$
Multi-task optimization (MTO), with AdamW updates		
MGDA (Sener & Koltun, 2018)	Shared with AdamW	All AdamW hyperparameters
GradNorm	Shared with AdamW	All AdamW hyperparameters
(Chen et al., 2018)	Asymmetry α	$\{.1, .5, 1, 1.5\}$
	Task-weight learning rate	$\{.001, .01, .025\}$
PCGrad (Yu et al., 2020)	Shared with AdamW	All AdamW hyperparameters
CAGrad	Shared with AdamW	All AdamW hyperparameters
(Liu et al., 2021a)	Conflict-averse radius c	$\{.1, .2, .4, .8\}$
Nash-MTL	Shared with AdamW	All AdamW hyperparameters
(Navon et al., 2022)	Global gradient-norm clip (1 for other methods)	$\{1, 10, 10^8\}$ (10^8: none)
FAMO	Shared with AdamW	All AdamW hyperparameters
(Liu et al., 2023)	Task-logit learning rate β	$\{.005, .025, .1\}$
	Task-logit decay γ	$\{0, 10^{-5}, 10^{-3}\}$
NTKMTL-SR	Shared with AdamW	All AdamW hyperparameters
(Qin et al., 2025)	Weighting exponent (1/2 in the original)	$\{.25, .5, .75\}$
SVFair	Shared with AdamW	All AdamW hyperparameters
(Wang et al., 2026)	Shapley temperature τ	$\{.1, .5, 1, 2\}$
CoPEN (Ours)	Learning rate η	$\{1, 1.25, 2, 3.804312\} \times 10^{-4}$
	Damping λ_c	$\{.03, .1, .3\}$
	Mixing weight γ	$\{.5, .75\}$
	Adjustment scope (ViT)	Last three blocks and task heads, or all parameters
	Adjustment scope (MoE)	All routed layers' experts and routers, and heads
	Grouping (MoE)	Layer-wise, or separate experts and routers

rate and other AdamW hyperparameters follow the corresponding multi-task AdamW configuration. We freeze these references at their end-of-decay checkpoints and evaluate them on the same validation data as the multi-task losses they are compared with (Table 11).

Hardware. Experiments run on single NVIDIA H200 and NVIDIA RTX A6000 GPUs. Timing definitions are given in Section F.4.

F Additional experiments and analysis

This section provides additional model-scale results, a constant-learning-rate comparison, and computational-cost analysis. Training schedules, validation data, and checkpoint-selection rules are defined in Section E.3.

F.1 Task-wise validation results

Under the plateau-triggered cosine-decay schedule used for the main comparison, Tables 9 and 10 report the ten individual task losses at the checkpoints summarized in Table 1. CoPEN obtains the lowest loss on eight of ten tasks for ViT and seven of ten for MoE, in addition to the lowest mean loss for both architectures. The gains span depth, edge, keypoint, curvature, reshading, and semantic tasks rather than being concentrated in a single output. CoPEN is not uniformly best on every task: Edge-Tex. and Keypt-2D favor other methods for ViT, while Edge-Tex., Keypt-2D, and Normals do so for MoE.

Table 9: **Per-task validation loss on Taskonomy with ViT-B/16.** The best result in each column is marked in bold.

Method	Depth-Euc.	Depth-Z.	Edge-Occ.	Edge-Tex.	Keypt-2D	Keypt-3D	Normals	Curv.	Reshad.	Sem.	Mean
Optimizers											
AdamW	0.0132	0.0136	0.1163	0.1523	0.1531	0.0868	0.1270	0.5296	0.1686	0.0837	0.14442
Shampoo	0.0169	0.0175	0.1261	0.1552	0.1624	0.0941	0.1675	0.5463	0.1965	0.0986	0.15813
K-FAC	0.0427	0.0434	0.1907	0.1975	0.2063	0.1154	0.3571	0.6148	0.4592	0.3726	0.25997
AdaHessian	0.0231	0.0232	0.1343	0.1630	0.1737	0.0979	0.1924	0.5541	0.2141	0.0867	0.16624
Muon	0.0136	0.0141	0.1166	0.1521	0.1523	0.0871	0.1274	0.5311	0.1742	0.0858	0.14543
SOAP	0.0135	0.0139	0.1170	0.1520	**0.1519**	0.0873	0.1266	0.5323	0.1717	0.0919	0.14579
SASSHA	0.0541	0.0543	0.1641	0.1803	0.1935	0.1150	0.2405	0.5767	0.2734	0.1280	0.19799
NorMuon	0.0136	0.0140	0.1163	0.1520	0.1521	0.0870	0.1258	0.5308	0.1721	0.0857	0.14494
Trion	0.0136	0.0141	0.1162	0.1521	0.1520	0.0868	0.1268	0.5308	0.1734	0.0850	0.14506
Multi-task optimization (MTO)											
MGDA	0.0217	0.0224	0.1333	0.1560	0.1660	0.0962	0.1854	0.5425	0.2304	0.0994	0.16532
GradNorm	0.0162	0.0168	0.1213	0.1538	0.1554	0.0907	0.1361	0.5315	0.1908	0.0806	0.14933
PCGrad	0.0134	0.0138	0.1161	0.1519	**0.1519**	0.0868	0.1267	0.5302	0.1711	0.0876	0.14495
CAGrad	0.0145	0.0151	0.1187	0.1522	0.1523	0.0879	0.1315	0.5342	0.1815	0.0979	0.14859
Nash-MTL	0.0146	0.0152	0.1177	0.1522	0.1539	0.0876	0.1336	0.5298	0.1774	0.0817	0.14637
FAMO	0.0147	0.0151	0.1216	**0.1508**	0.1527	0.0891	0.1398	0.5397	0.1875	0.0963	0.15073
NTKMTL-SR	0.0139	0.0143	0.1178	0.1520	0.1520	0.0875	0.1290	0.5317	0.1761	0.0853	0.14596
SVFair	0.0142	0.0147	0.1171	0.1517	0.1521	0.0874	0.1340	0.5309	0.1794	0.0967	0.14782
CoPEN (Ours)	**0.0129**	**0.0133**	**0.1146**	0.1520	0.1524	**0.0862**	**0.1252**	**0.5280**	**0.1671**	**0.0789**	**0.14306**

F.2 Training curves

Figure 5 shows how the endpoint differences in Table 1 develop over training. Across both architectures, CoPEN moves rapidly into the low-loss range of the strongest baselines and remains competitive through the later training phase. Its validation loss flattens near the end of the cosine decay while the training loss can continue to decrease, so additional optimization after that point does not necessarily yield comparable validation improvement. The same qualitative pattern appears in both the optimizer and MTO comparisons, indicating that the reported endpoint gains are not isolated to a single comparison group or checkpoint.

Table 10: **Per-task validation loss on Taskonomy with V-MoE-B/16.** The best result in each column is marked in bold.

Method	Depth-Euc.	Depth-Z.	Edge-Occ.	Edge-Tex.	Keypt-2D	Keypt-3D	Normals	Curv.	Reshad.	Sem.	Mean
Optimizers											
AdamW	0.0137	0.0140	0.1183	0.1531	0.1544	0.0883	0.1341	0.5323	0.1732	0.0791	0.14606
K-FAC	0.2482	0.3098	0.5980	0.2962	0.3261	0.3011	0.3008	0.6213	0.2896	0.1815	0.34727
AdaHessian	0.0185	0.0188	0.1276	0.1578	0.1673	0.0936	0.1676	0.5452	0.1936	0.0837	0.15735
Muon	0.0138	0.0142	0.1165	0.1521	0.1522	0.0870	0.1276	0.5312	0.1738	0.0877	0.14562
SOAP	0.0135	0.0139	0.1169	0.1521	0.1520	0.0873	0.1275	0.5325	0.1721	0.0946	0.14624
SASSHA	0.0503	0.0521	0.1666	0.1813	0.1935	0.1149	0.2506	0.5797	0.2890	0.1273	0.20053
NorMuon	0.0134	0.0139	0.1158	0.1520	0.1519	0.0868	**0.1263**	0.5309	0.1718	0.0935	0.14563
Trion	0.0136	0.0140	0.1160	0.1522	0.1524	0.0869	0.1281	0.5306	0.1726	0.0853	0.14518
Multi-task optimization (MTO)											
MGDA	0.0254	0.0248	0.1383	0.1585	0.1679	0.0979	0.1889	0.5446	0.2366	0.1057	0.16886
GradNorm	0.0157	0.0161	0.1205	0.1526	0.1531	0.0890	0.1345	0.5319	0.1837	0.0854	0.14823
PCGrad	0.0138	0.0142	0.1171	0.1518	0.1519	0.0870	0.1274	0.5318	0.1749	0.0948	0.14646
CAGrad	0.0147	0.0153	0.1199	0.1521	0.1519	0.0880	0.1307	0.5355	0.1846	0.1047	0.14975
Nash-MTL	0.0137	0.0140	0.1170	0.1517	**0.1517**	0.0871	0.1303	0.5301	0.1755	0.0920	0.14631
FAMO	0.0144	0.0148	0.1208	**0.1508**	0.1524	0.0887	0.1401	0.5392	0.1855	0.0977	0.15042
NTKMTL-SR	0.0139	0.0143	0.1180	0.1520	0.1519	0.0876	0.1291	0.5327	0.1768	0.0899	0.14662
SVFair	0.0142	0.0147	0.1180	0.1518	0.1520	0.0878	0.1335	0.5313	0.1793	0.1004	0.14830
CoPEN (Ours)	**0.0129**	**0.0132**	**0.1157**	0.1521	0.1525	**0.0864**	0.1265	**0.5296**	**0.1670**	**0.0783**	**0.14344**

Single-task references. Table 11 reports the separately trained reference models used to compute Δ_m. Relative to these references, CoPEN improves six of ten individual task losses for ViT and eight of ten for MoE. Thus the positive aggregate Δ_m values summarize gains across most, but not all, tasks rather than implying uniform per-task improvement. These single-task models serve as references for Δ_m; the optimizer and MTO comparisons above are made directly on the multi-task validation losses.

Table 11: **Single-task reference losses on Taskonomy.**

Task	ViT	MoE
Depth-Euc.	0.01770	0.01515
Depth-Z.	0.01971	0.01583
Edge-Occ.	0.11820	0.12196
Edge-Tex.	0.15167	0.15208
Keypt-2D	0.15135	0.15395
Keypt-3D	0.09072	0.08827
Normals	0.12358	0.12565
Curv.	0.53117	0.53394
Reshad.	0.17441	0.17408
Sem.	0.07728	0.07848

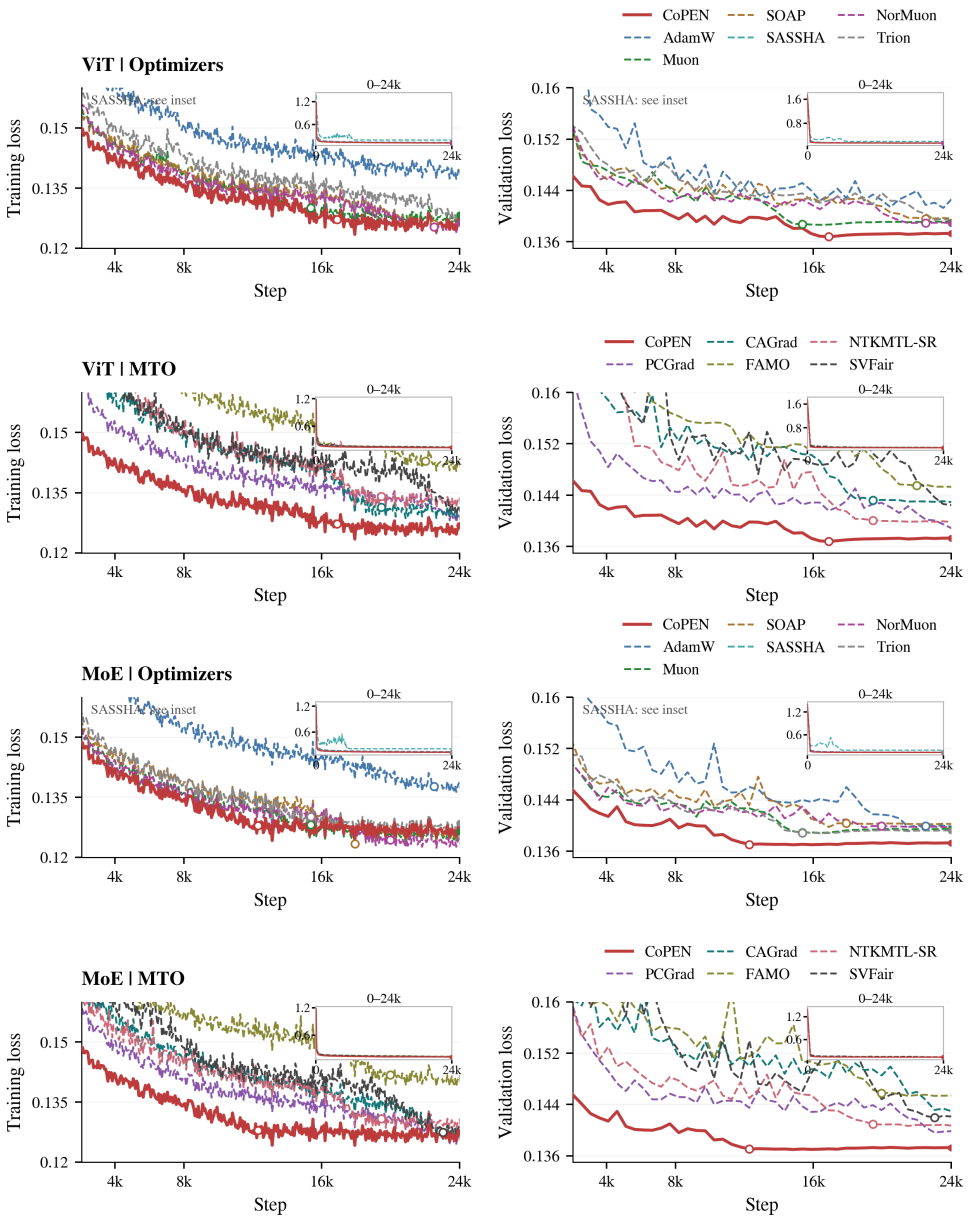


Figure 5: **Training and validation loss with plateau-triggered cosine decay.** Rows compare optimizers and MTO methods for ViT and MoE; left and right columns show training and validation loss, respectively. Insets show the complete 0–24K trajectories. Hollow circles mark the end of cosine decay when it falls within the displayed range.

F.3 Constant-learning-rate results

We additionally evaluate the constant-learning-rate schedule defined in Section E.3. Baselines reuse their plateau-triggered cosine-decay configurations with the constant schedule.

For CoPEN, the checkpoint selected by periodic-validation loss within 24,000 updates is evaluated on the full validation set of 51,753 images. Loss and Δ_m are computed at that checkpoint using full-validation single-task references for Δ_m.

A few ViT baselines use checkpoints slightly beyond 24,000 updates. All reported values use the same full-validation evaluation.

Table 12: **Constant-learning-rate comparison on Taskonomy.** The lowest validation loss and highest Δ_m are shown in bold.

	ViT		MoE	
Method	Val. loss ↓	$\Delta_m(\%)$ ↑	Val. loss ↓	$\Delta_m(\%)$ ↑
Optimizers				
AdamW (Loshchilov & Hutter, 2019)	0.14697	+0.03	0.14801	−1.38
Shampoo (Gupta et al., 2018)	0.16566	−17.55	0.15465	−6.50
K-FAC (Martens & Grosse, 2015)	0.14457	+5.49	0.17002	−28.08
AdaHessian (Yao et al., 2021)	0.16579	−29.46	0.16746	−26.09
Muon (Jordan, 2024)	0.14810	+0.81	0.14719	+0.17
SOAP (Vyas et al., 2025)	0.14747	+1.97	0.14748	−0.25
SASSHA (Shin et al., 2025)	0.21191	−137.33	0.26277	−280.48
NorMuon (Li et al., 2025)	0.14718	+1.93	0.14710	+0.33
Trion (Modoranu et al., 2026)	0.14762	+1.68	0.14720	−0.08
Multi-task optimization (MTO)				
MGDA (Sener & Koltun, 2018)	0.16831	−24.89	0.17540	−44.87
GradNorm (Chen et al., 2018)	0.15275	−4.24	0.15150	−5.36
PCGrad (Yu et al., 2020)	0.14747	+0.09	0.15021	−4.39
CAGrad (Liu et al., 2021a)	0.15258	−4.14	0.15501	−10.01
Nash-MTL (Navon et al., 2022)	0.15120	−5.36	0.14980	−5.07
FAMO (Liu et al., 2023)	0.15281	−4.11	0.15456	−9.36
NTKMTL-SR (Qin et al., 2025)	0.14954	−1.50	0.15097	−5.20
SVFair (Wang et al., 2026)	0.15263	−4.74	0.15211	−7.78
CoPEN (Ours)	**0.14413**	**+5.71**	**0.14444**	**+3.21**

F.4 COMPUTATIONAL COST

For group sizes d_b and basis widths k_b, the basis and reduced matrix require $O(\sum_b d_b k_b + r^2)$ storage, where $r = \sum_b k_b$. Each refresh uses r curvature products. Each adjustment requires $O(\sum_b d_b k_b)$ projection work and an $O(r^3)$ reduced factorization. These costs are additional to tensor-wise preconditioning and ordinary training.

Table 13 reports the measured cost of every method, and its seconds per step are the values shown in Table 1. Each method runs alone on one NVIDIA H200 GPU, starting from the pretrained checkpoint with the settings of Table 1. Seconds per step is the mean synchronized update time measured during training. It includes batch transfer and periodic preconditioner or curvature refreshes, and excludes data waiting, evaluation, and checkpoint writes. Memory is the peak GPU memory allocated by PyTorch during the run.

Table 13: **Computational cost on one NVIDIA H200 GPU.** Seconds per update and peak allocated GPU memory using the settings in Table 1.

	ViT		MoE	
Method	s/step	Mem. (GiB)	s/step	Mem. (GiB)
Optimizers				
AdamW	0.155	5.7	0.248	11.2
Shampoo	0.310	5.7	2.180	10.1
K-FAC	0.922	10.3	2.430	24.2
AdaHessian	0.564	13.3	0.909	23.9
Muon	0.258	5.4	0.473	10.1
SOAP	0.288	6.4	0.462	12.2
SASSHA	0.324	14.0	0.521	26.1
NorMuon	0.263	5.4	0.485	10.1
Trion	0.289	5.4	0.524	10.1
Multi-task optimization (MTO)				
MGDA	0.836	10.5	1.461	27.8
GradNorm	0.277	5.9	0.490	11.5
PCGrad	0.867	10.8	1.532	28.9
CAGrad	0.835	10.5	1.479	27.8
Nash-MTL	0.833	10.5	1.457	27.8
FAMO	0.194	5.7	0.318	11.2
NTKMTL-SR	0.457	6.0	0.538	11.4
SVFair	0.864	10.5	1.495	27.8
CoPEN (Ours)	0.327	6.7	0.672	25.1