Applied AI Lab
Sungkyunkwan University

ICML
International Conference
On Machine Learning
2025

One-Step Generalization Ratio Guided Optimization for Domain Generalization

Sumin Cho
Sungkyunkwan University
jsm0707@skku.edu

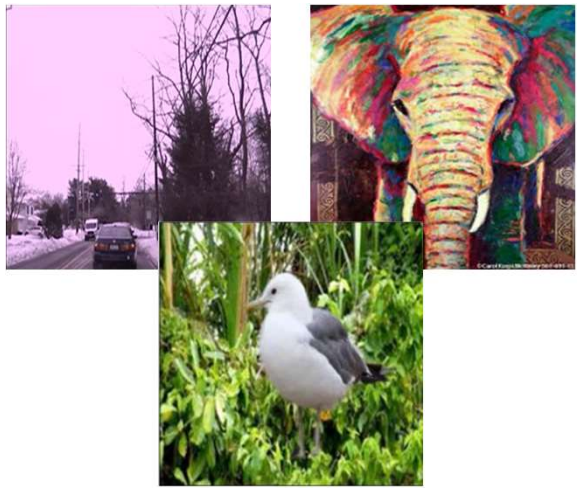
Dongwon Kim
Sungkyunkwan University
kdwaha@skku.edu

Kwangsu Kim
Sungkyunkwan University
kim.kwangsu@skku.edu

Motivation

Motivation : Domain Generalization

Training data

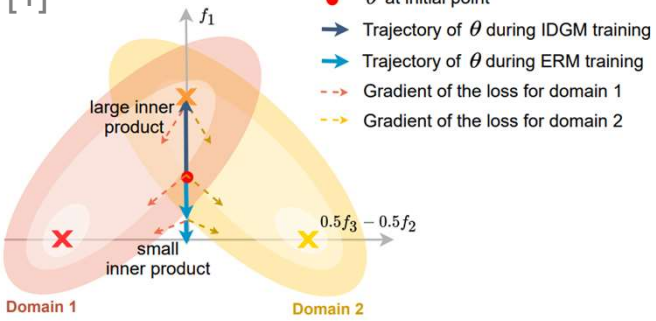


Test data

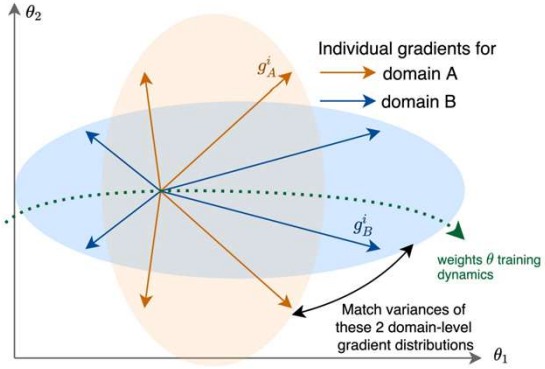
- Domain Generalization (DG) aims to train models that generalize well to unseen domains.
- The key challenge is to avoid overfitting to spurious correlations (e.g., background, color) and instead learn causal relationships (e.g., object, class) that are invariant across domains.

Motivation : Previous Research

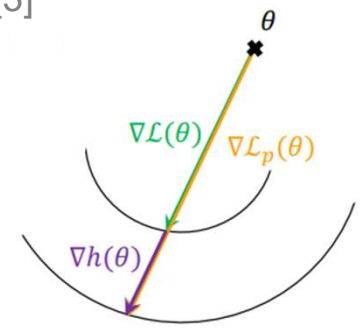
[1]



[2]



[3]



- Many DG methods align gradients from different domains to enforce consistent learning in a dominant direction.
- This dominant direction may be guided by overly predictive features, leading to reinforced spurious correlations.

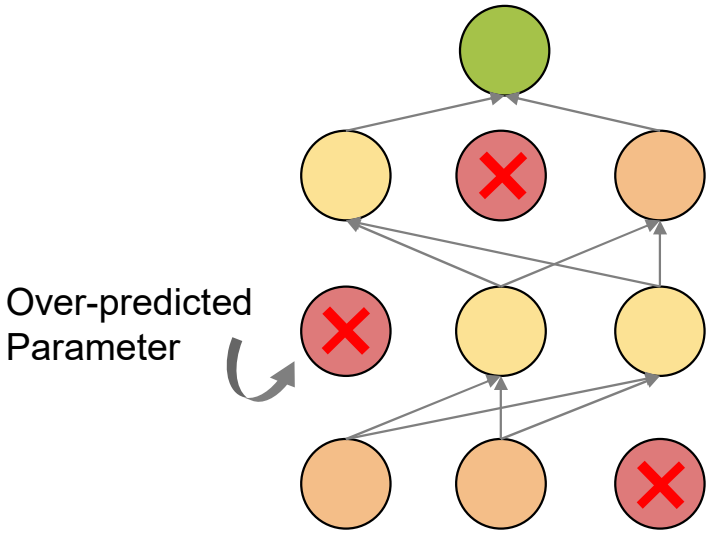
➢ **We need an alternative approach that goes beyond simple alignment and considers generalization sensitivity**.

[1] Matching for Domain Generalization (ICLR2022)

[2] Fishr: Invariant Gradient Variances for Out-of-Distribution Generalization (ICML2022)

[3] Sharpness-Aware Gradient Matching for Domain Generalization (CVPR2023)

Motivation : Previous Research



Gradient Signal to Noise Ratio (GSNR)

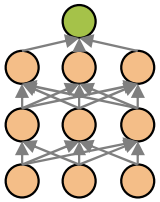
$$\frac{\tilde{\mathbf{g}}^2(\theta_j)}{\rho^2(\theta_j)}$$

⇐ *Signal* $\tilde{\mathbf{g}}(\theta) = \mathrm{E}_{(x,y)\sim\mathcal{Z}}(\mathbf{g}(x,y,\theta))$

⇐ *Noise* $\rho^2(\theta) = \mathrm{Var}_{(x,y)\sim\mathcal{Z}}(\mathbf{g}(x,y,\theta))$

- Prior work use GSNR to drop overly predictive parameters to mitigate spurious correlations.
- But dropout is a binary. It can't quantify each parameter's contribution to generalization.

➢ **A finer approach is needed—like an optimizer that dynamically adapts update strength based on generalization feedback.**



To improve generalization, balance the contribution of parameters.

How

What

Optimizer

- The imbalance across parameters arises from the way standard optimizers operate.
- We propose an optimizer that directly adjusts each parameter's OSGR.

One-Step Generalization Ratio (OSGR)

- OSGR quantifies how much a single update step contributes to generalization.
- It reflects both the convergence speed and the generalization ability of each parameter.

GENIE

Generalization-ENhancing Iterative Equalizer

Optimizer that regulates each parameter's OSGR consistently throughout training.

Method

Method : Method Component

Algorithm 1 Algorithm for GENIE

Input: Mini-batches $\{\mathcal{B}_t\}_{t=1}^{T}$, Learning Rate α, Total Steps T.
Hyperparameters: $\beta \in [0, 1]$, Dropout Probability p
Initialize: Parameters θ_0, $m_0 \leftarrow 0$, $v_0 \leftarrow 0$.
for $t = 1$ to T **do**
Compute Gradient:

$$g_t = \nabla \mathcal{L}(\theta_t; \mathcal{B}_t)$$

Update Moving Averages:

$$m_t \leftarrow \beta m_{t-1} + (1-\beta) g_t, \quad v_t \leftarrow \beta v_{t-1} + (1-\beta) g_t^2$$

Calculate GSNR and Preconditioning: ①

$$\sigma_t^2 = v_t - m_t^2, \quad r_j = \tanh(\frac{1}{\sigma_t^2}) m_t^2$$

$$\hat{g}_t \leftarrow \frac{m_t}{1-\beta^t} \cdot \frac{1}{v_t} \cdot r_t$$

Noise Injection: ②

$$Noise_t \leftarrow \xi_t [1 - \tanh(\frac{1}{\sigma_t^2})], \quad \xi_t \sim \mathcal{N}(0, \sigma^2)$$

Random Mask: ③

$$M_j \sim Bernoulli(p)$$

$$\hat{g}_t \leftarrow (\hat{g}_t + Noise_t) \odot M$$

Update Parameters:

$$\theta_{t+1} \leftarrow \theta_t - \alpha \tilde{g}_t$$

end for
Output: Final parameters θ_{T+1}.

Main Method

(1) Preconditioning

- OSGR-based weighting leads to balanced updates across parameters.

+

Enhancing Preconditioning

(2) Noise Injection

- Noise stimulates under-trained parameters

  → boosts exploration and update diversity

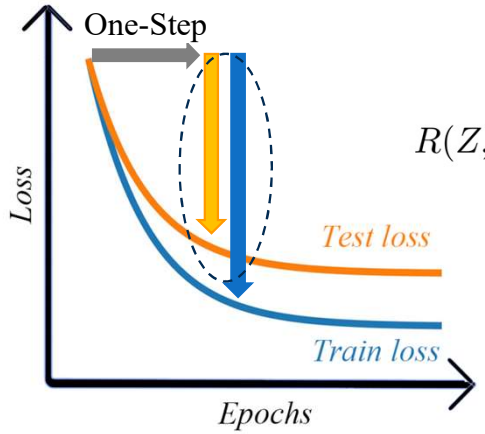
(3) Random Mask

- Randomly omitting updates

  → prevents overfitting and promotes balanced parameter

Method : Preconditioning

One Step Generalization Ratio (OSGR)



$$R(Z,n) = \frac{\mathbb{E}_{D,D' \sim Z^n} \Delta L_{D'}}{\mathbb{E}_{D \sim Z^n} \Delta L_D}$$

According to Theorem [1]

$$R(Z,n) = 1 - \frac{1}{n} \sum_j \underbrace{\frac{\mathbb{E}_{D \sim Z^n}[g_j^2]}{\sum_{j'} \mathbb{E}_{D \sim Z^n}[g_{j'}^2]}}_{\text{Gradient Update Ratio}} \cdot \underbrace{\frac{1}{r_j + \frac{1}{n}}}_{\text{Generalization Ability}}$$

- High OSGR means the corresponding update direction promotes generalization.
- Standard optimizers repeatedly update parameters with large gradient magnitudes.
- Biased updates toward dominant gradients reduce OSGR and hurt generalization.

Z: Data distribution
D': Test Data
D: Training Data
g: Gradient
j: j-th parameter
r: GSNR
n : #Sample

[1] Understanding Why Neural Networks Generalize Well Through GSNR of Parameters (ICLR2020)

Method : Preconditioning

***According to** Theorem in [1]*

Previous OSGR

$$R(Z,n) = 1 - \frac{1}{n}\sum_j \frac{\mathbb{E}_{D\sim\mathcal{Z}^n}[g_j^2]}{\sum_{j'}\mathbb{E}_{D\sim\mathcal{Z}^n}[g_{j'}^2]} \cdot \frac{1}{r_j + \frac{1}{n}}$$

Gradient Update Ratio — Generalization Ability

GENIE (our)

Parameter Update

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{P}(t) \nabla_{\boldsymbol{\theta}_t} L(f_{\boldsymbol{\theta}_t})$$

$$R'(Z,n) = 1 - \frac{1}{n}\sum_j \frac{p_j \mathbb{E}_{D\sim\mathcal{Z}^n}[g_j^2]}{\sum_{j'} p_{j'}\mathbb{E}_{D\sim\mathcal{Z}^n}[g_{j'}^2]} \cdot \frac{1}{r_j + \frac{1}{n}}$$

Precondition Factor

$$p_j = \frac{1}{\mathbb{E}_{D\sim\mathcal{Z}^n}[g_j^2]}\left(r_j + \frac{1}{n}\right)$$

Our OSGR

$$R'(Z,n) = 1 - \frac{1}{n}\sum_j \frac{1}{\sum_{j'}\left(r_{j'} + \frac{1}{n}\right)}$$

Uniform Generalization Ability

Remove update dominance

- Precondition the optimizer to cancel parameter-wise update imbalance.

Promote uniform generalization

- Ensure all parameters contribute equally to generalization—not just dominant ones.

[1] Understanding Why Neural Networks Generalize Well Through GSNR of Parameters (ICLR2020)

Theoretical Analysis

Theoretical Analysis : OSGR based Intuition

Corollary 1 (Preconditioning and OSGR)

If each parameter j applies an arbitrary preconditioner p_j, the OSGR can be expressed as:

$$R'(Z,n) = 1 - \frac{1}{n}\sum_j \frac{p_j \mathbb{E}_{D\sim\mathcal{Z}^n}[g_j^2]}{\sum_{j'} p_{j'} \mathbb{E}_{D\sim\mathcal{Z}^n}[g_{j'}^2]} \cdot \frac{1}{r_j + \frac{1}{n}}$$

Corollary 2 (OSGR of GENIE)

The OSGR of GINIE is:

$$\mathcal{R}_{Ours} = 1 - \frac{1}{n\mathbb{E}_{j\in J}\left(r_j + \frac{1}{n}\right)}$$

Replacing weighted average $\sum_j \frac{p_j \mathbb{E}_{D\sim\mathcal{Z}^n}(g_{D,j}^2)}{\sum_{j'} p_j \mathbb{E}_{D\sim\mathcal{Z}^n}(g_{D',j}^2)}$ with a uniform average $\sum W_j = 1$ with $W_j = \frac{1}{|J|}$

and applying **Jensen's inequality**, we obtain the following bound:

$$0 \le 1 - \frac{1}{n}\sum_{j\in J} W_j \left(\frac{1}{r_j + \frac{1}{n}}\right) \le 1 - \frac{1}{n\mathbb{E}_{j\in J}\left(r_j + \frac{1}{n}\right)} \le 1.$$

$$0 \le R_{\text{precondition}} \le R_{\text{ours}} \le 1$$

Higher OSGR

Theoretical Analysis : Generalization Bound

PAC Bayes bound.

For any $\lambda > 0$ and any data-dependent probability measure $\tilde{p}$,

$$\mathbb{E}_S \mathbb{E}_{\theta\sim\tilde{p}}[R(\theta)] \;\leq\; \mathbb{E}_S \mathbb{E}_{\theta\sim\tilde{p}}\Big[\underbrace{L(\theta)}_{T_1} + \frac{\lambda C^2}{8n} + \underbrace{\frac{\mathrm{KL}(\tilde{p}\,\|\,\pi)}{\lambda}}_{T_2}\Big] \qquad \left(\begin{array}{l} \theta_{t+1} = \theta_t - q \odot g \\ \tilde{p} = \mathcal{N}(\theta_{t+1}, \Sigma_{\tilde{p}}) \text{ and } \pi = \mathcal{N}(\theta_t, \Sigma_\pi) \end{array} \right)$$

T_1: Sensitivity to perturbation T_2: Prior-Posterior Divergence

Previous Method

While SAM focuses only on minimizing T_1

$$L_{\mathcal{D}}(\theta) \;\leq\; \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\rho)}\left[L_{\mathcal{D}}(\theta+\epsilon)\right] \;\leq\; \max_{\|\epsilon\|_2\leq\rho}\left[L_{\mathcal{D}}(\theta+\epsilon)\right]$$

GENIE (our)

GENIE adjusts both T_1 and T_2 using OSGR

$$E[||q \odot g - E[g]||_2^2] = \sum_j q_j^2 E[g_j^2] - 2q_j E[g_j]^2 + E[g_j]^2$$

When $q_j = \frac{\mathbb{E}[g]^2}{\mathbb{E}[g_j^2]}$,

$$[\nabla_{\theta_t} KL(\tilde{p}\|\pi)]_j = \underbrace{\frac{1}{\mathbb{E}[g_j^2]} \cdot \frac{\mathbb{E}[g_{j,t}]^2}{\rho_j^2}}_{\text{GENIE}} \cdot g_{j,t}$$

Tighter Generalization Objective

q_j : Variance adaptation factor
ρ_j : Deviation of gradient

Theoretical Analysis : Convergence Analysis

Basic Assumptions

1. Bounded Gradient

$$\|\nabla \mathcal{L}(\theta_t)\| \leq G \quad \text{for all } t$$

2. Lipschitz-smooth

$$\|\nabla \mathcal{L}(\theta_1) - \nabla \mathcal{L}(\theta_2)\| \leq L\|\theta_1 - \theta_2\|$$

3. Non-zero variance

$$\mathbb{E}[\|g_t - \nabla \mathcal{L}(\theta_t)\|^2] \geq 1/S_u, \quad \forall t$$

Convergence Rate of GENIE

$$\mathbb{E}[\|\nabla L(\theta)\|^2] \leq O\left(\frac{1}{P_l}\left(1 + \frac{G \cdot S_u^2}{2}\right)\frac{1}{\sqrt{\hat{T}}}\right)$$

Fast Convergence (similar to SGD)

G: Upper bound of gradient
L: Lipschitz constant
$1/S_u$: Lower bound of variance
P_l : Lower bound of Preconditioning

Experiments

Experiments : Comparison of Optimizers on DG

Domain Generalization

OPT.	PACS	VLCS	OFFICE HOME	TERRA INC	DOMAIN NET	AVG.	
ADAM*	84.2	77.3	67.6	44.4	43.0	63.3	- 5.69% (DG baseline optimizer)
ADAMW*	83.6	77.4	68.8	45.2	43.4	63.7	
SGD*	79.9	78.1	68.5	44.9	43.2	62.9	- 6.36%
YOGI*	81.2	77.6	68.3	45.4	43.5	63.2	
ADABELIEF*	84.6	78.4	68.0	45.2	43.5	63.9	
ADAHESSIAN*	84.5	78.6	68.4	44.4	**44.4**	64.1	
SAM*	85.3	78.2	68.0	45.7	43.4	64.1	- 4.37%
GAM*	86.1	78.5	68.2	45.2	43.8	64.4	
FAD*	**88.2**	78.9	69.2	45.7	**44.4**	65.3	
GENIE	87.8	**80.7**	**69.7**	**52.0**	44.1	**66.9**	

- Consistently outperforms existing optimizers on average across all datasets.
- Shows significant improvement on VLCS, which often suffers from early convergence and overfitting, and on TerraIncognita, a real-world, challenging dataset

Experiments : Comparison of Optimizers on DG

Domain Generalization

Convergence Rate of GENIE

$$\mathbb{E}[\|\nabla L(\theta)\|^2] \leq O\left(\frac{1}{P_l}\left(1+\frac{G\cdot S_u^2}{2}\right)\frac{1}{\sqrt{\hat{T}}}\right)$$

Fast Convergence (similar to SGD)

Opt.	Iter.	Training Time (/s)	Avg. PACS	Avg. VLCS	Avg. Office Home
SGD	5000	5,273	69.8	76.7	51.3
	10000	10,546	73.9	77	62.5
	15000	15,783	75.8	77.7	63.9
Adam	5000	5,443	84.2	77	63.6
	10000	10,934	86.1	77	65.2
	15000	16,531	84.5	77	65.2
SAM	5000	5,775	82.4	79.4	69.4
	10000	11,500	83.5	80.3	69.6
	15000	17,191	84.1	80.4	**70**
GENIE	5000	4,292	**88.4**	**81.3**	**70**
	10000	8,582	87.1	**81.3**	69.2
	15000	12,876	86.9	**81.3**	69.1

Achieves an average of 1.3× faster training speed.

- The computational overhead of an optimizer is a critical factor for real-world applicability.
- GENIE outperforms other optimizers with just 5,000 iterations and demonstrates convergence speed consistent with theoretical analysis.

Experiments : Integration with DG Algorithms

Domain Generalization

Algorithm	PACS	VLCS	OfficeHome	TerraInc	Avg.
ERM†(Vapnik, 1999)	85.5	77.5	66.5	46.1	68.9
IRM†(Arjovsky et al., 2019)	83.5	78.6	64.3	47.6	68.5
GroupDRO†(Sagawa et al., 2020)	84.4	76.7	66.0	43.2	67.6
I-Mixup†(Xu et al., 2020)	84.6	77.4	68.1	47.9	69.5
MLDG†(Li et al., 2018a)	84.9	77.2	66.8	47.8	69.2
MMD†(Li et al., 2018b)	84.7	77.5	66.4	42.2	67.7
DANN†(Ganin et al., 2016)	83.7	78.6	65.9	46.7	68.7
CDANN†(Li et al., 2018c)	82.6	77.5	65.7	45.8	67.9
MTL†(Blanchard et al., 2021)	84.6	77.2	66.4	45.6	68.5
SagNet†(Nam et al., 2021)	86.3	77.8	68.1	48.6	70.2
ARM†(Zhang et al., 2021)	85.1	77.6	64.8	45.5	68.3
VREx†(Krueger et al., 2021)	84.9	78.3	66.4	46.4	69
Mixstyle*(Zhou et al., 2021)	85.2	77.9	60.4	44	66.9
MIRO*(Cha et al., 2022)	85.4	78.9	69.5	45.4	69.8
GENIE (ours)	**87.8**	**80.7**	**69.7**	**52.0**	**72.6**
RSC(Huang et al., 2020)+Adam*	84.5	77.9	65.7	44.5	68.2
RSC+AdamW*	83.4	77.5	66.3	45.1	68.1
RSC+SGD*	82.6	78.1	67	43.9	67.9
RSC+GENIE(ours)	**87.3**	**80.6**	**68.1**	**49.5**	**71.4**
CORAL(Sun & Saenko, 2016) + Adam*	86	78.9	68.7	43.7	69.3
CORAL+AdamW*	86.4	79.5	69.8	45.0	70.2
CORAL+SGD*	85.6	78.2	69.5	45.8	69.8
CORAL+GENIE(ours)	**87.9**	**80.7**	**70.6**	**48.4**	**71.9**

GENIE as a standalone method

GENIE as an optimizer

GENIE performs well as a standalone method and can also be used as an optimizer to improve the performance of existing DG algorithms without modifying the training procedure or model architecture.

Experiments : Single Domain Generalization (SDG)

Single Domain Generalization

ALGORITHM	PACS	VLCS	OFFICE HOME	TERRA INC	AVG.
ADAM	64.3	56.2	50.7	33.5	51.2
SGD	49.5	60.4	45.9	22.8	44.7
SAM	57.7	66.7	**59.2**	26.8	52.6
GENIE (OURS)	**69.5**	**69.9**	58.6	**36.0**	**58.5**
RSC+ADAM	56.8	51.6	2.1	31.6	35.5
RSC+SGD	22.2	39.8	1.7	17.6	20.3
RSC+GENIE(OURS)	**68.2**	**68.7**	**54.4**	**33.2**	**56.1**
CORAL+ADAM	64.3	56.2	50.7	33.5	51.2
CORAL+SGD	49.5	60.4	45.9	22.8	44.7
CORAL+GENIE(OURS)	**70.9**	**69.2**	**56.4**	**36.7**	**58.3**

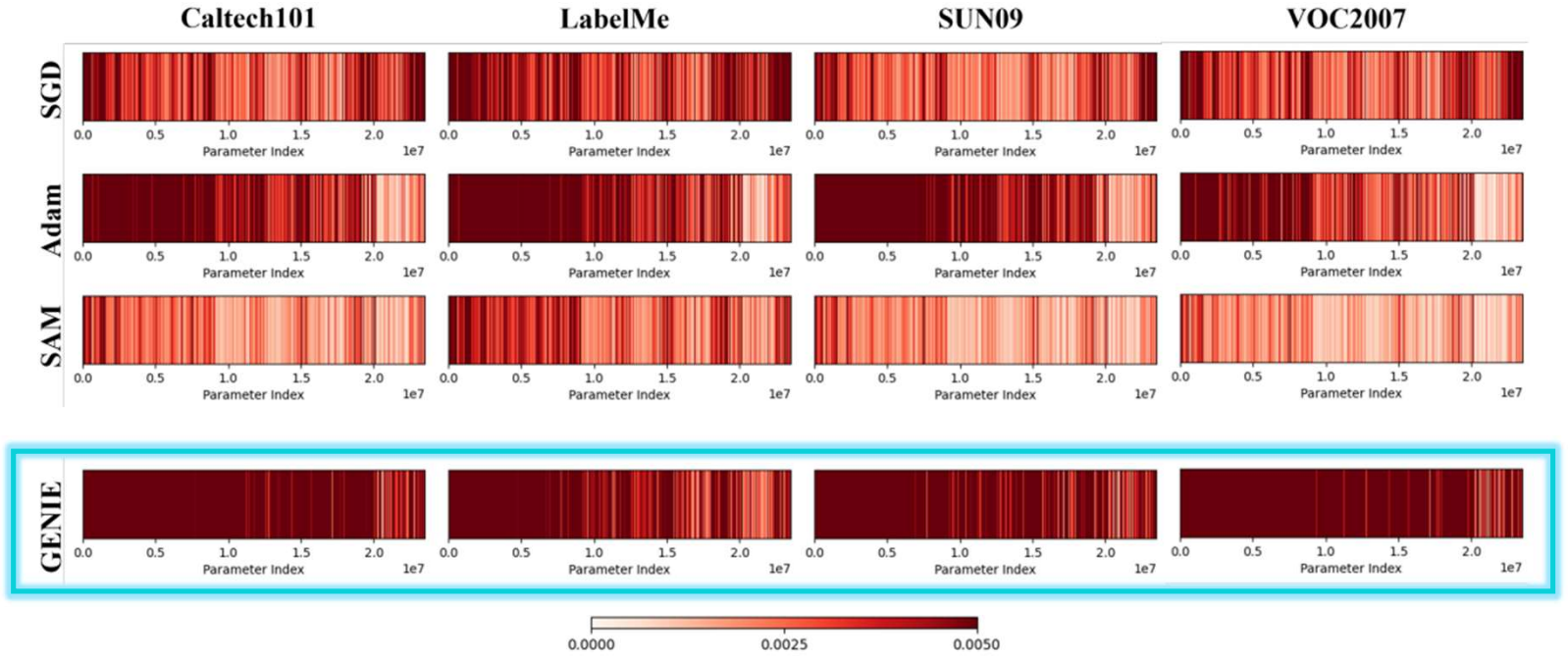
GENIE as a standalone method

GENIE as an optimizer

- In SDG, models are trained on a single source domain and tested on multiple unseen target domains — a more challenging yet realistic scenario.
- GENIE shows strong performance regardless of the number of training domains and provides additional gains when combined with existing methods.

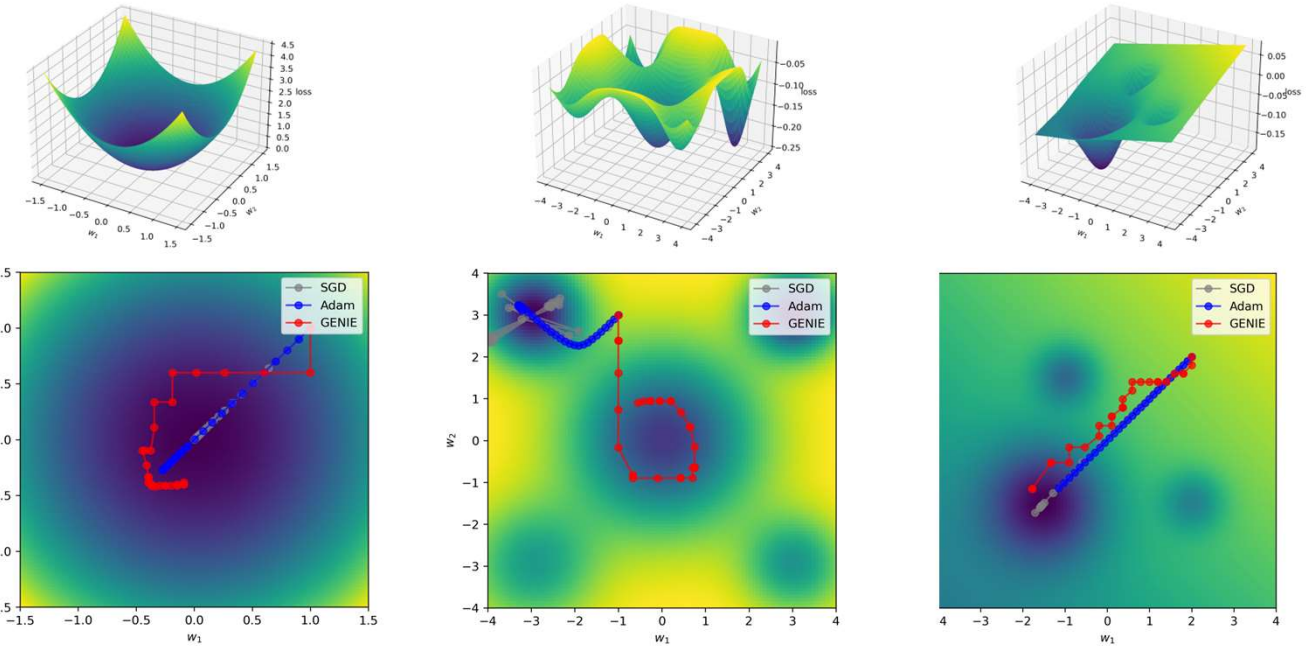
Experiments : Update Distribution Analysis

Domain Generalization



- Update magnitudes were normalized by parameter ID and visualized as a heatmap.
- Existing optimizers exhibited imbalanced update distributions, with updates concentrated on a small subset of parameters.
- In contrast, GENIE balances contributions across parameters, enabling well-distributed and unbiased optimization.

Experiments : Loss landscape



Domain Generalization

GENIE (our)

GENIE adjusts both T_1 and T_2 using OSGR

$$E[||q \odot g - E[g]||_2^2] = \sum_j q_j^2 E[g_j^2] - 2q_j E[g_j]^2 + E[g_j]^2$$

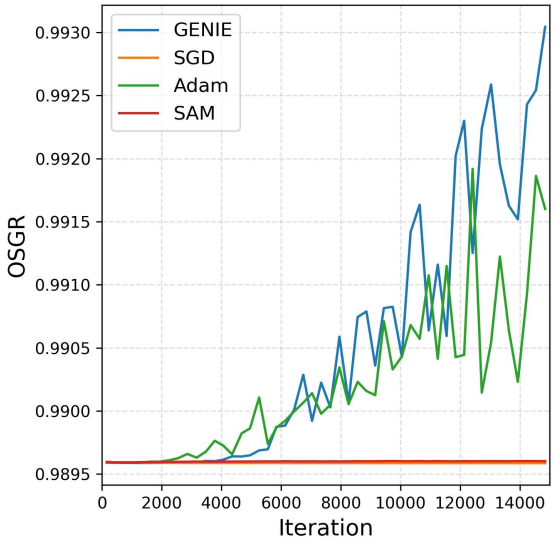
$$[\nabla_{\theta_t} KL(\tilde{p}\|\pi)]_j = \underbrace{\frac{1}{\mathbb{E}[g_j^2]} \cdot \frac{\mathbb{E}[g_{j,t}]^2}{\rho_j^2}}_{\text{GENIE}} \cdot g_{j,t}$$

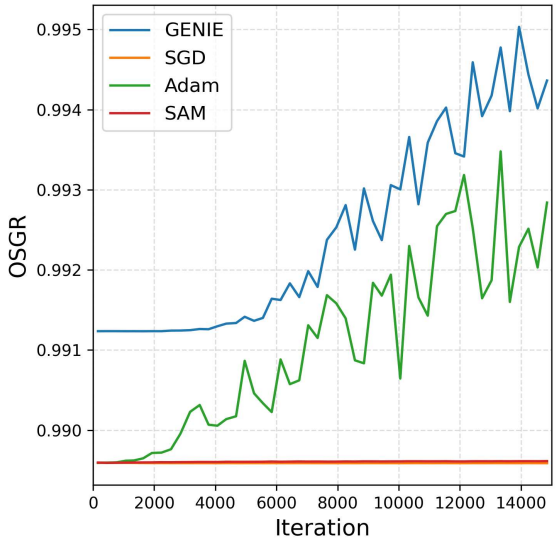
Tighter Generalization Objective

- SGD and Adam converge quickly along sharp directions, which often leads to overfitting on domain-specific features.
- GENIE converges toward flatter regions, aligning with the theoretical analysis based on PAC-Bayes.

Experiments : OSGR of Network Parameters Over Time

Domain Generalization





$$0 \leq 1 - \frac{1}{n}\sum_{j \in J} W_j \left(\frac{1}{r_j + \frac{1}{n}}\right) \leq 1 - \frac{1}{n\mathbb{E}_{j \in J}\left(r_j + \frac{1}{n}\right)} \leq 1.$$

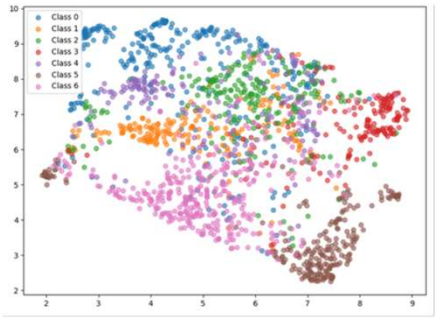
$$0 \leq R_{\text{precondition}} \leq R_{\text{ours}} \leq 1$$

Higher OSGR

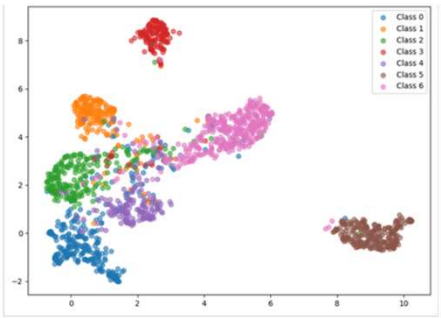
- We tracked the average OSGR across all parameters during training.
- GENIE maintains OSGR values closer to 1 compared to existing optimizers, indicating stronger generalization — consistent with our theoretical analysis.

Experiments : Feature Visualization

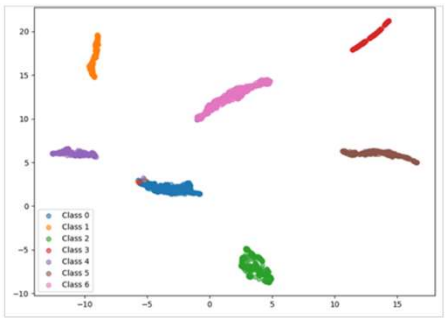
Domain Generalization



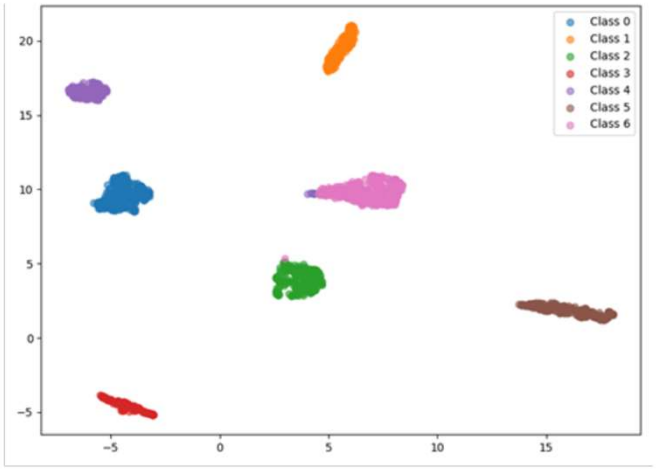
(A) Before Training



(B) SGD



(C) Adam



(D) GENIE

- We performed UMAP visualization on the PACS dataset.
- GENIE achieved clear class separation across domains, suggesting effective learning of domain-invariant features.

Conclusion

▪ Summary

- We propose **GENIE**, an optimizer that improves generalization performance by balancing parameters based on **OSGR**.
- GENIE combines **preconditioning**, **noise injection**, and **random masking** to modulate updates per parameter based on gradients.

▪ Contribution

- As a **plug-and-play optimizer**, it can be applied to any DG or SDG method without architectural changes.
- It introduces a **new optimization paradigm** that uses OSGR as a core indicator for generalization.
- GENIE shifts the focus in DG from *what* to learn to ***how*** to learn, offering a **new perspective**.

Thank you