Learning Large-scale Neural Fields via Context Pruned Meta-Learning

Paper Link

GitHub Link

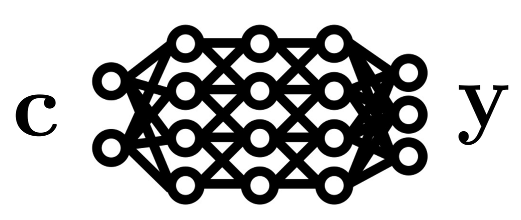
Jihoon Tack[1], Subin Kim[1], Sihyun Yu[1], Jaeho Lee[2], Jinwoo Shin[1], Jonathan Richard Schwarz[3]

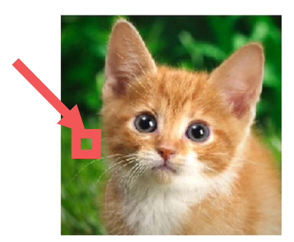
KAIST[1], POSTECH[2], University College London[3]

TL;DR. We propose an efficient meta-learning framework for scalable neural fields learning that involves online data pruning of the context set

Neural Fields

Neural Fields (NFs) represent each data as a neural network approximating a coordinate-to-signal mapping function.





$$f_{\mathrm{img}}\left(\tfrac{53}{255}, \tfrac{152}{255}\right) = (0, 1, 0)$$

(R, G, B)

NF has the potential to be a popular form of data representation by showing versatile usage such as compression, generation, classification, and unseen view synthesis.

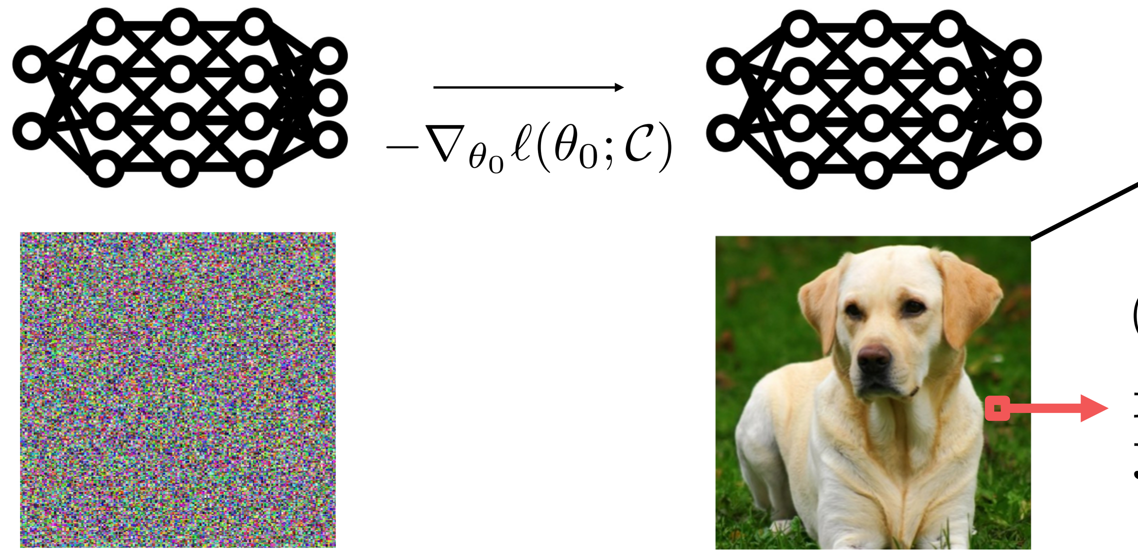
🤔 **Challenges** when learning NFs on large-scale datasets

- Training time inefficiency
- Memory inefficiency

Meta-learning: Time efficiency

Meta-learning is a popular way to fasten the training of NFs

- Find θ_0 such that **few-step gradients** can fit the signal (optimization-base meta-learning; used for versatile applications)

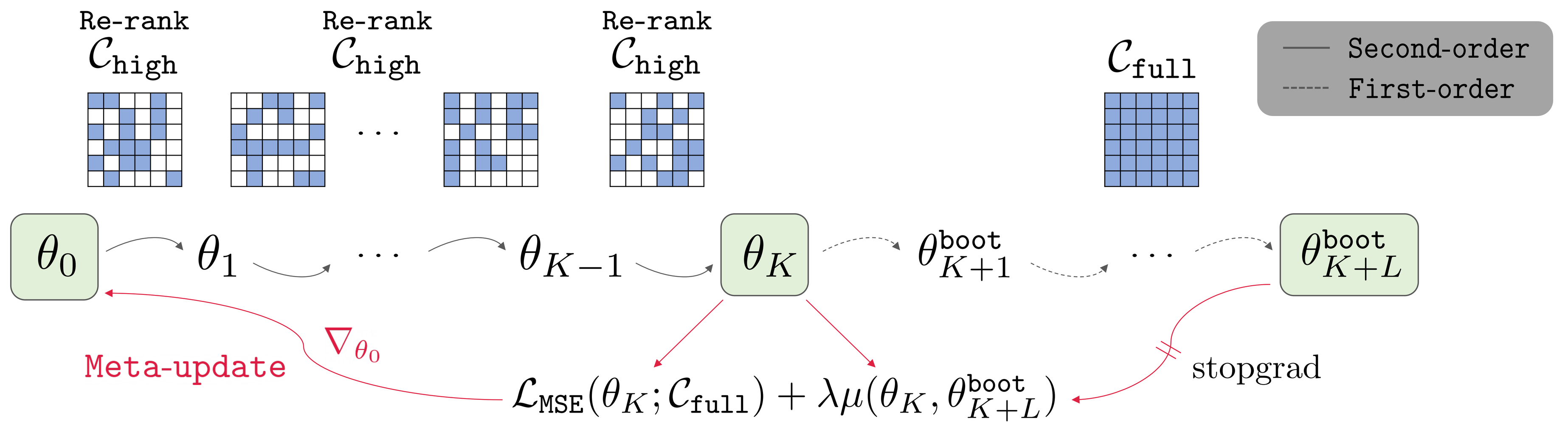


$$\theta_0 = \mathrm{argmin}_{\theta_0} \frac{1}{N}\sum_{i=1}^{N} \ell\left(\theta_0 - \boxed{\alpha\nabla_{\theta_0}\ell(\theta_0;\mathcal{C}^{(i)})}; \mathcal{C}^{(i)}\right)$$

memory scales linearly with the number of context set

Gradient Norm-based Context Pruning

We propose **Gradient Norm-based Context Pruning**



1. Online context pruning

$$\begin{aligned}
&\ell(\theta_k;\{(\mathbf{x},\mathbf{y})\}) - \ell(\theta_k';\{(\mathbf{x},\mathbf{y})\}) && \text{(Immediate improvment in model quality)}\\
&\approx \ell(\theta_k;\{(\mathbf{x},\mathbf{y})\}) - \ell(\theta_k - \alpha g_k;\{(\mathbf{x},\mathbf{y})\}) && \text{(Last layer update only)}\\
&\approx \ell(\theta_k;\{(\mathbf{x},\mathbf{y})\}) - \Big(\ell(\theta_k;\{(\mathbf{x},\mathbf{y})\}) - \alpha g_k^\top \nabla_{\theta_k}\ell(\theta_k;\{(\mathbf{x},\mathbf{y})\})\Big) && \text{(Taylor approximation)}\\
&= \alpha g_k^\top \nabla_{\theta_k}\ell(\theta_k;\{(\mathbf{x},\mathbf{y})\}) = \alpha\|g_k\|_2^2 && \text{(Last layer gradient norm)}
\end{aligned}$$

→ This score can be calculated with only a single forward pass

$$R_k^{\mathrm{GradNCP}}(\mathbf{x},\mathbf{y}) := \Big\|\big(\mathbf{y} - f_{\theta_k}(\mathbf{x})\big)\big[\phi_{\theta_k^{\mathrm{base}}}(\mathbf{x}), 1\big]^\top\Big\|$$

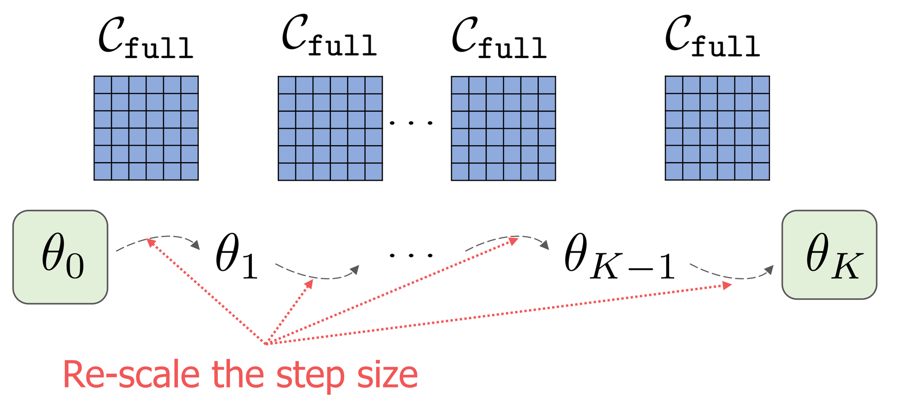
$\phi(.)$: penultimate feature
$f(.)$: network output
k: inner adaptation step

2. Bootstrapped correction

- Generate bootstrapped target model with **the full context set**
- Minimize the parameter distance between two models

3. Gradient re-scaling

- For meta-testing, we can use first-order gradients for adaptation
- The **gradient step size deviates a lot** from meta-train and test



$$g_t^{\mathrm{test}} \leftarrow \boxed{\frac{\|\nabla_{\theta_t}\mathcal{L}_{\mathrm{MSE}}(\theta_t;\mathcal{C}_{\mathrm{high}})\|}{\|\nabla_{\theta_t}\mathcal{L}_{\mathrm{MSE}}(\theta_t;\mathcal{C}_{\mathrm{full}})\|}} \nabla_{\theta_t}\mathcal{L}_{\mathrm{MSE}}(\theta_t;\mathcal{C}_{\mathrm{full}})$$

Scale-factor

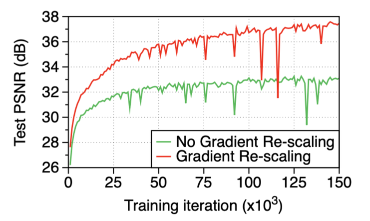


Figure 7: Effect of meta-test time gradient re-scaling. We apply re-scaling when adapting with full context set on SIREN trained with GradNCP.

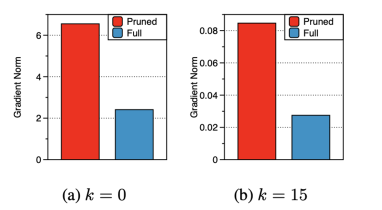
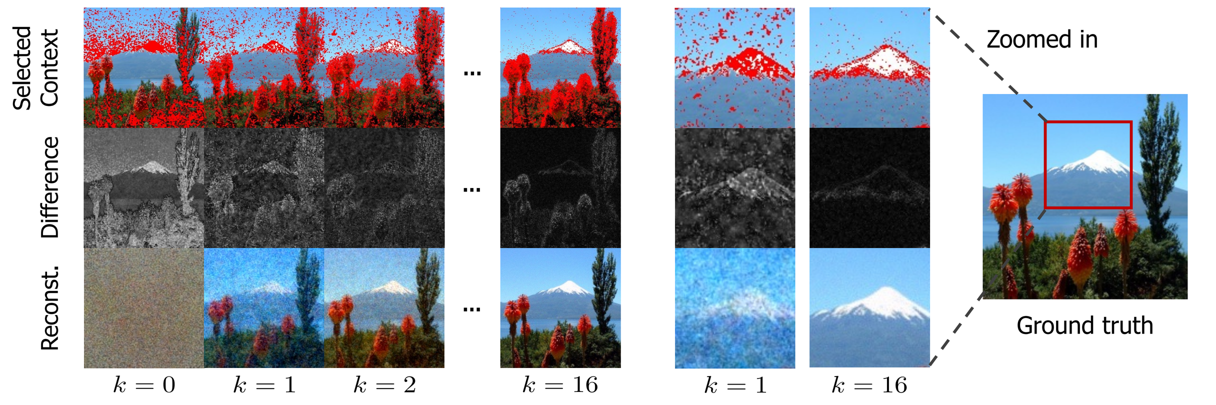


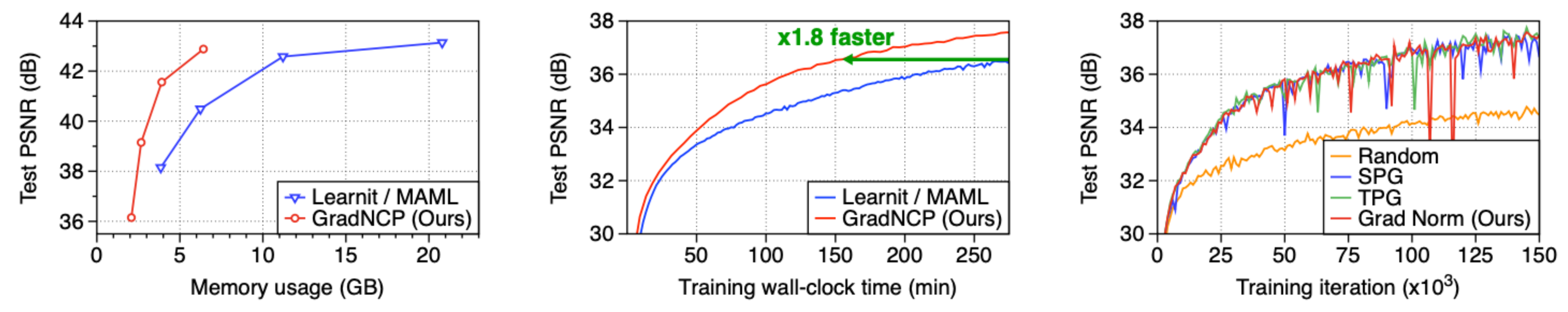
Figure 8: Gradient norm of the full context set $\mathcal{C}_{\mathrm{full}}$ and the gradient norm-based pruned context set $\mathcal{C}_{\mathrm{high}}^k$ at iteration k.

Experimental Results



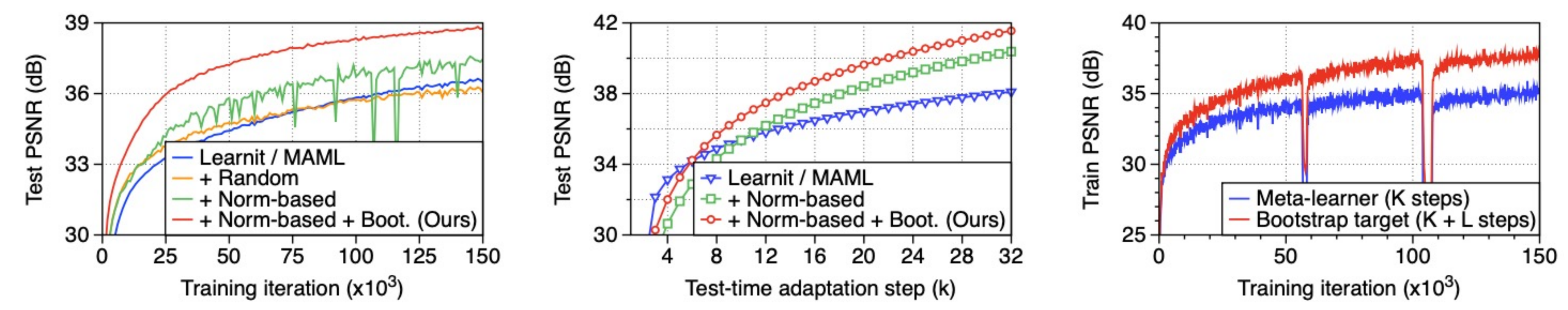
automatically **captures the global shape first**, then later captures the high-frequency details

Efficiency of GradNCP



(a) Memory consumption (b) Runtime efficiency (c) Context scoring quality

Effectiveness of Bootstrapped correction



(a) GradNCP component analysis (b) Myopia (c) Bootstrap target performance

GradNCP achieves **the state-of-the-art performance**

Image

	CelebA (178 × 178)	**Imagenette** (178 × 178)	**Text** (178 × 178)
Random Init.	19.94 / 0.532 / 0.708	18.57 / 0.443 / 0.810	15.37 / 0.574 / 0.755
TransINR [6]	32.37 / 0.913 / 0.068	28.58 / 0.850 / 0.165	22.70 / 0.898 / 0.085
IPC [29]	35.93 / - / -	38.46 / - / -	- / - / -
Learnit / MAML [61]	38.28 / 0.964 / 0.010	35.66 / 0.950 / 0.014	30.31 / 0.956 / 0.018
GradNCP (Ours)	**40.60 / 0.976 / 0.005**	**38.72 / 0.972 / 0.005**	**32.33 / 0.976 / 0.007**
	ImageNet (256 × 256)	**AFHQ** (512 × 512)	**CelebA-HQ** (1024 × 1024)
Random Init.	18.72 / 0.434 / 0.839	18.57 / 0.488 / 0.856	12.21 / 0.574 / 0.820
TransINR [6]	28.01 / 0.818 / 0.199	23.43 / 0.592 / 0.573	——— OOM ———
Learnit / MAML [61]	31.44 / 0.887 / 0.100	28.58 / 0.751 / 0.354	27.66 / 0.781 / 0.513
GradNCP (Ours)	**32.52 / 0.898 / 0.068**	**29.61 / 0.786 / 0.286**	**28.90 / 0.789 / 0.438**

Audio

Method	PSNR (↑) 1 sec	PSNR (↑) 3 sec
TransINR [6]	39.22	33.17
IPC [29]	40.11	35.38
Learnit / MAML [61]	39.55	31.39
GradNCP (Ours)	**43.25**	**36.24**

Manifold

Method	PSNR (↑)
Learnit / MAML [61]	64.91
GradNCP (Ours)	**75.11**

Video

Resolution	Network	Method	PSNR (↑)	SSIM (↑)	LPIPS (↓)
128×128×16	SIREN	TransINR [6]	15.14	0.360	0.636
		Learnit / MAML [61]	25.46	0.720	0.363
		GradNCP (Ours)	**26.92**	**0.781**	**0.223**
	NeRV	Learnit (MAML) [61]	28.86	0.871	0.140
		GradNCP (Ours)	**35.28**	**0.959**	**0.015**
256×256×32	SIREN	TransINR [6]	———	OOM	———
		Learnit / MAML [61]	———	OOM	———
		GradNCP (Ours)	**22.92**	**0.640**	**0.521**
	NeRV	Learnit / MAML [61]	23.75	0.659	0.422
		GradNCP (Ours)	**28.65**	**0.842**	**0.201**

NeRFs

Method	PSNR
Learnit [61]	22.80
TransINR [6]	23.78
GradNCP (Ours)	**24.06**

General (Few-shot)

Method	5-way 100-shot	10-way 50-shot
MAML [15]	66.03±0.82	48.95±0.52
GradNCP (Ours)	**73.45±0.83**	**55.71±0.49**