Stochastic Preconditioning for Neural Field Optimization

Michael Belyaev

Abstract

This report explores and evaluates the benefits of stochastic preconditioning (SP) as presented in "Stochastic Preconditioning for Neural Field Optimization" [1].

Neural Fields and Coordinate-based Neural Networks

Coordinate-based neural networks have recently attracted significant attention due to their ability to achieve accurate approximations of fields, which are quantities defined for all spacial and/or temporal coordinates. A field, typically represented as $\boldsymbol{y} = \boldsymbol{\Phi}(\boldsymbol{x})$, is simply a function that maps a coordinate $\boldsymbol{x}$ to some quantity, typically a scalar, vector or tensor.

A typical field $\boldsymbol{y} = \boldsymbol{\Phi}(\boldsymbol{x})$ isn't defined by a simple analytical expression or formula, like the gravitational field generated by a solid ball. Instead, it may be described as a set of parameters, $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_n)$ where the parameters are either hand-crafted, optimised or learned. Fields learned using coordinate-based neural networks are called NEURAL FIELDS.

Implicit Neural Representations

We say that a field, $\boldsymbol{y} = \boldsymbol{\Phi}(\boldsymbol{x})$, is defined implicitly if its values are not given directly, but are instead described by one or several equations. These equations relate the field's value at a given coordinate $\boldsymbol{x}$ to properties like its slope or curvature, as shown in the general form

$$\mathcal{C}(\boldsymbol{x}, \boldsymbol{\Phi}(\boldsymbol{x}), \nabla_{\boldsymbol{x}}\boldsymbol{\Phi}(\boldsymbol{x}), \nabla^2_{\boldsymbol{x}}\boldsymbol{\Phi}(\boldsymbol{x}), \dots) = 0.$$

Following the definition used in [3], we say that a neural field $\boldsymbol{y} = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{\theta})$ provides an IMPLICIT NEURAL REPRESENTATION if it is defined implicitly.

Representing Shapes with Distance Functions

Consider a surface $\mathcal{S}$ in three-dimensional space. Let $d_{\mathcal{S}}(\boldsymbol{x})$ represent the minimal distance from a point $\boldsymbol{x}$ to the surface. The equation $d_{\mathcal{S}}(\boldsymbol{x}) = 0$ then implicitly represents the surface $\mathcal{S}$. This is useful when $\mathcal{S}$ is given as a set of points from real-world measurements. In such cases, a properly trained neural field $\Phi(\boldsymbol{x}, \boldsymbol{\theta})$ can approximate the distance function $d_{\mathcal{S}}(\boldsymbol{x})$ and the zero-level set, where $\Phi(\boldsymbol{x}, \boldsymbol{\theta}) = 0$, approximates the surface $\mathcal{S}$.

Let's assume that the surface $\mathcal{S}$ is two-sided and $d_{\mathcal{S}}(\boldsymbol{x})$ is its signed distance function (SDF). The SDF not only measures distance but also indicates which side of the surface a point is on (e.g., inside or outside). While the SDF may have singularities where its gradient is not defined, at non-singular points it is a well-behaved function that satisfies a set of fundamental mathematical equations:

$$|\nabla_{\boldsymbol{x}} d_{\mathcal{S}}(\boldsymbol{x})| = 1 \text{ outside } \mathcal{S}, \quad d_{\mathcal{S}}(\boldsymbol{x}) = 0 \text{ on } \mathcal{S}, \quad \nabla_{\boldsymbol{x}} d_{\mathcal{S}}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}) = 1 \text{ on } \mathcal{S}. \tag{1}$$

Here, $\boldsymbol{n}(\boldsymbol{x})$ is the unit normal vector of the surface $\mathcal{S}$. The first equation, $|\nabla_{\boldsymbol{x}} d_{\mathcal{S}}(\boldsymbol{x})| = 1$, is the eikonal equation. Unfortunately, these equations do not uniquely define the distance function, as shown in Fig.1.

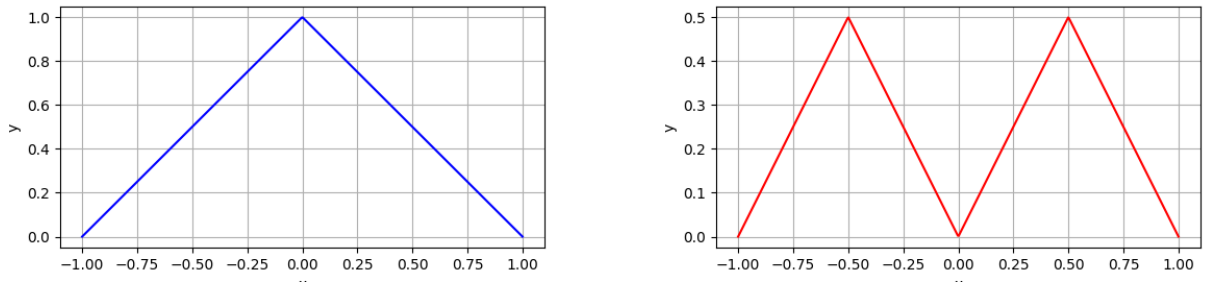


Figure 1: Left: the graph of the distance function for $\mathcal{S} = \{-1, +1\}$. Right: graph of another 1D functions which satisfies (1) almost everywhere.

Stochastic Preconditioning

Stochastic preconditioning (SP) is a method [1] to improve neural field optimization by adding Gaussian noise to the input coordinates during training. The process works by querying the neural field, $\boldsymbol{\Phi}_{\boldsymbol{\theta}}$, at a perturbed location, $\boldsymbol{x} + \delta$, instead of the original location, $\boldsymbol{x}$, where δ is a Gaussian-distributed offset with a standard deviation of α. This technique is extremely easy to apply to existing neural field-based methods.

Neural Surface Reconstruction from Oriented Point Clouds

Point clouds are a common way to represent 3D surface geometry, where a collection of data points each has a corresponding 3D coordinate $\boldsymbol{x}$ and a unit normal vector $\boldsymbol{n}(\boldsymbol{x})$ that indicates the surface's orientation at that point. The goal of this task is to train a neural field to approximate the signed distance function (SDF) of the surface. This way, the surface can be reconstructed as the zero-level set of the neural field, where $\Phi(\boldsymbol{x}, \boldsymbol{\theta}) = 0$.

The loss function used for these experiments follows the recommendation of [3], which are also followed by the stochastic preconditioning paper. The four main components are:

$$\mathcal{L}_{\text{surface}} = \int_{\Omega_0} |\Phi(\boldsymbol{x})| \, d\boldsymbol{x} \approx \frac{1}{n} \sum_{i=1}^{n} |\Phi(\boldsymbol{x}_i, \boldsymbol{\theta})|, \tag{2}$$

$$\mathcal{L}_{\text{normal}} = \int_{\Omega_0} \left[1 - \nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x})\right] \, d\boldsymbol{x} \approx \frac{1}{n} \sum_{i=1}^{n} \left[1 - \nabla_{\boldsymbol{x}_i} \Phi(\boldsymbol{x}_i, \boldsymbol{\theta}) \cdot \boldsymbol{n}(\boldsymbol{x}_i)\right], \tag{3}$$

$$\mathcal{L}_{\text{eik}} = \int_{\Omega} \left| |\nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x}, \boldsymbol{\theta})| - 1 \right| \, d\boldsymbol{x}, \tag{4}$$

$$\mathcal{L}_{\text{offsurface}} = \int_{\Omega \setminus \Omega_0} e^{-\gamma |\Phi(\boldsymbol{x}, \boldsymbol{\theta})|} \, d\boldsymbol{x}. \tag{5}$$

Here, Ω_0 represents the surface $\mathcal{S}$ to be reconstructed, and Ω is the 3D domain containing it. The total loss function is then a weighted sum of these terms:

$$\mathcal{L}_{\text{total}} = \lambda_0 \cdot \mathcal{L}_{\text{surface}} + \lambda_1 \cdot \mathcal{L}_{\text{normal}} + \lambda_2 \cdot \mathcal{L}_{\text{eik}} + \lambda_3 \cdot \mathcal{L}_{\text{offsurface}}. \tag{6}$$

Experiments

To evaluate the benefits of stochastic preconditioning, I conducted a series of experiments using an **INGP Hashgrid MLP** [2] as the neural field representation. The model was trained for 500 iterations to reconstruct surfaces from oriented point clouds.

These parameters are the default values used in the author's implementation of stochastic preconditioning[1], available at this GitHub repository.

The following coefficients were used for the loss function (6):

$$\lambda_0 = 3 \times 10^3, \quad \lambda_1 = 1 \times 10^2, \quad \lambda_2 = 5, \quad \lambda_3 = 1 \times 10^2,$$

which correspond to the surface, normal, eikonal, and off-surface losses, respectively. The off-surface loss parameter γ was set to 100.

For the stochastic preconditioning, the standard deviation of the added noise α was initialised to 2% of the diagonal of the bounding box and decayed exponentially to 0 over the first third of training iterations.

The ground truth and output model files generated for these experiments can be found in this Google Drive folder.

Standard Benchmark Models

Armadillo

On the armadillo model, stochastic preconditioning (SP) performs exceptionally well. As shown in comparing Fig 2a and Fig 2b, SP effectively removes artifacts from the arms and between the legs while making the ears, fingers, and toes noticeably smoother. My replication of the papers experiment calculating mean Chafmer distance achieves a comparable and slightly better results (5.01×10^{-4} vs. 7.10×10^{-4}).

(a) Hashgrid MLP (b) Hashgrid MLP + SP (c) Increased Normal Loss

Figure 2: Armadillo

Table 1: Armadillo Metrics

Method	**Chamfer**	**Hausdorff**	**IoU**
Hashgrid MLP	0.003977	0.291914	0.066706
Hashgrid MLP + SP	0.000501	0.038156	**0.283356**
Hashgrid MLP (Increased Normal Loss)	**0.000485**	**0.037472**	0.216273

I also investigated whether adjusting the loss coefficients could achieve a similar effect to stochastic preconditioning. By increasing the normal loss coefficient from 1×10^2 to 1×10^4, I was able to produce a reconstructed surface that also effectively removed artifacts, achieving results comparable to the SP method. However, my experiments with changing the other loss coefficients lead to significantly worse reconstructions.

The loss curves Fig 3 show an interesting training dynamic. Initially, the baseline model converges more rapidly. However, once the stochastic preconditioning noise, α, decays to zero, the SP model rapidly catches up and overtakes the baseline model.

(a) Total Loss (b) Surface Loss (c) Normal Loss

Figure 3: Logarithmic loss curves for the Armadillo model, comparing the baseline model (blue) with the model trained with stochastic preconditioning (red).

Dragon

The application of SP also improves the quality of the dragon model. As a visual comparison shows in Fig 4, the artifacts at the back of the tail are completely removed, and the overall reconstruction is noticeably smoother and less grainy.

(a) Hashgrid MLP (b) Hashgrid MLP + SP

Figure 4: Dragon

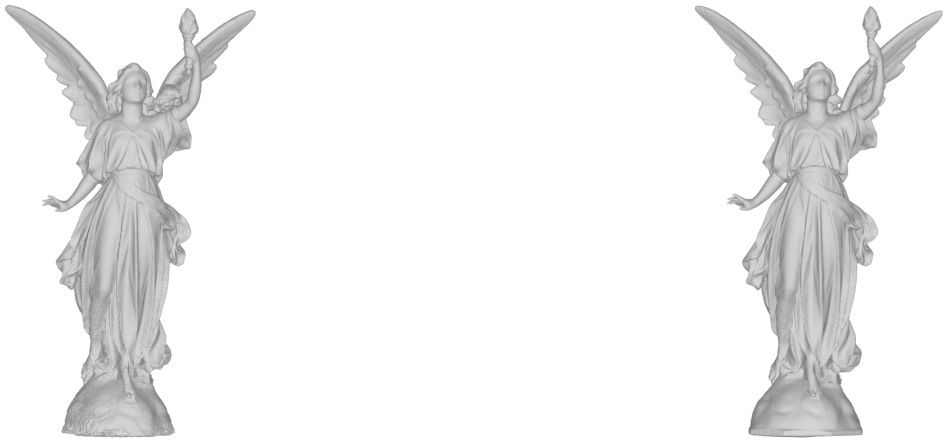
Lucy

I saw similar improvements with the Lucy model. The shoulder artifact is gone, and the feet and base are less grainy. These results show that stochastic preconditioning consistently produces higher-quality surface reconstructions across different models, as seen in Fig. 5.

Noisy Bunny

I experimented with adding a small amount of Gaussian noise to the ground truth point cloud. The specific noise parameters used are [TODO: Add noise parameters here].

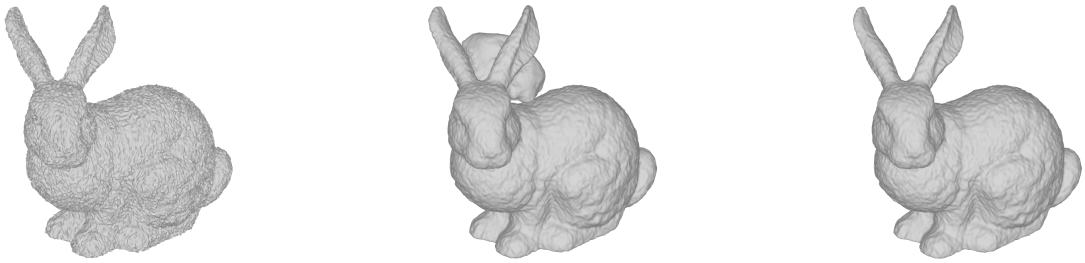
Both the baseline model and the SP model were able to reconstruct the surface regardless of the noise. However, the SP model removes the artifact between the ears, as seen in Fig. 6.



(a) Hashgrid MLP

(b) Hashgrid MLP + SP

Figure 5: Lucy



(a) Ground Truth

(b) Hashgrid MLP

(c) Hashgrid MLP + SP

Figure 6: Noisy Bunny

High-Genus Shapes

vbunny

The stochastic preconditioning (SP) technique, while effective on some models, shows a notable limitation when applied to high-genus shapes. My experiments on the vbunny model revealed that although SP successfully removes artifacts from the exterior of the model, it fails to do so for the interior artifacts.

I compared this result with another very recent method, HotSpot [4]. HotSpot introduces a new loss function called heat loss $\mathcal{L}_{\text{heat}}$, defined as

$$\mathcal{L}_{\text{heat}} = \frac{1}{2} \int_{\Omega} e^{-2\lambda |\Phi(\boldsymbol{x},\boldsymbol{\theta})|} \left(\|\nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x},\boldsymbol{\theta})\|^2 + 1 \right) d\mathbf{x}. \tag{7}$$

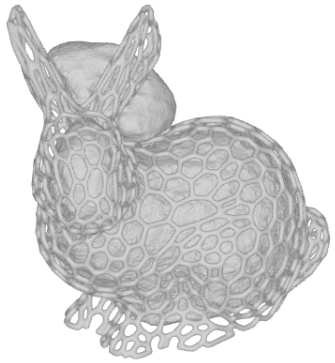
The HotSpot paper claims that the eikonal loss is a necessary but insufficient constraint for finding the true distance function.

The final total loss function for HotSpot combines this new heat loss with the surface and eikonal losses:

$$\mathcal{L} = \lambda_0 \cdot \mathcal{L}_{\text{surface}} + \lambda_1 \cdot \mathcal{L}_{\text{eik}} + \lambda_2 \cdot \mathcal{L}_{\text{heat}}, \tag{8}$$

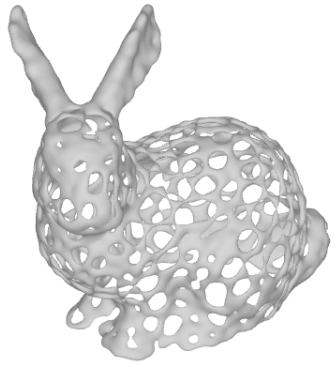
where λ_0, λ_1, and λ_2 are the weights for each loss term.

As shown in Fig. 7 and Table 2, the HotSpot method reconstructs the model exceptionally well, resolving both interior and exterior artifacts.

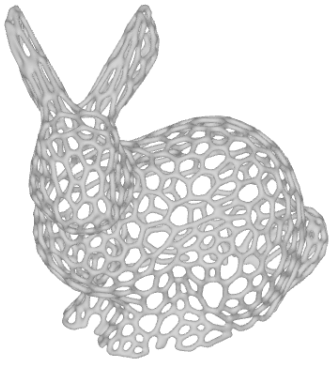


(a) Hashgrid MLP

(b) Hashgrid MLP + SP



(c) HotSpot (1k iter)



(d) HotSpot (10k iter)

Figure 7: vbunny

Table 2: vbunny

Method	Chamfer	Hausdorff	IoU
Hashgrid MLP	0.000422	0.084007	0.143944
Hashgrid MLP + SP	0.000572	0.079678	0.247838
HotSpot (1k iter)	0.000854	0.082362	0.280314
HotSpot (10k iter)	**0.000102**	**0.027738**	**0.465142**

Stochastic Preconditioning Hyperparameters

As mentioned earlier, the stochastic preconditioning (SP) paper [1] recommends initializing the noise standard deviation, α, to 1-2% of the bounding box diagonal and decaying it exponentially to zero over the first third of training. I experimented with alternative noise schedules.

Cow

After conducting experiments on the Cow model, I concluded that the specific method of decreasing stochastic preconditioning noise and the number of iterations over which it is applied are largely irrelevant to the final result. The quality of the reconstructed surface remains similar across different schedules. To demonstrate this, I compared three versions of SP:

- Fig. 8b **Constant:** Noise was applied at a constant magnitude for 167 of the 500 total iterations before being dropped to zero.
- Fig. 8c **400/500 iter:** An exponential decay schedule was applied for the first 400 iterations.
- Fig. 8d **100/500 iter:** An exponential decay schedule was applied for the first 100 iterations.

As seen in Fig. 8, all three versions of stochastic preconditioning achieved similar results and successfully removed the artifacts between the cows legs. The only minor difference was a slightly degraded tail on the model trained with the shortest noise schedule.

Increasing the Noise

Stochastic preconditioning can sometimes be sensitive to the initial amount of noise. As a visual comparison shows in Fig. 9, doubling the initial noise from the recommended 0.02 to 0.04 results in a worse reconstruction with significantly more artifacts inside the model.

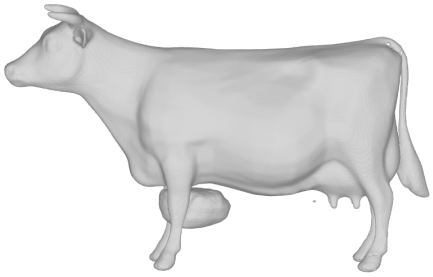
SP with Sinusoidal Noise

I experimented with an alternative noise schedule by applying a sinusoidal decay to the α parameter during training. The noise was set to oscillate and gradually decay to zero according to the function:

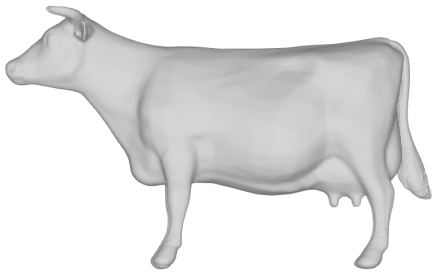
$$\alpha_{\text{noise}}(t) = \alpha_{\text{final}} + (\alpha_{\text{init}} - \alpha_{\text{final}}) \cdot (1 - t/T) \cdot \frac{1 + \cos(2\pi N_{\text{osc}}\, t/T)}{2}.$$

For these experiments, the total decay time was $T = 250$ iterations, with the alpha parameter starting at $\alpha_{\text{init}} = 0.4$ and decaying to $\alpha_{\text{final}} = 0$. The noise completed $N_{\text{osc}} = 5$ full oscillations. I chose these parameters for no particular reason.

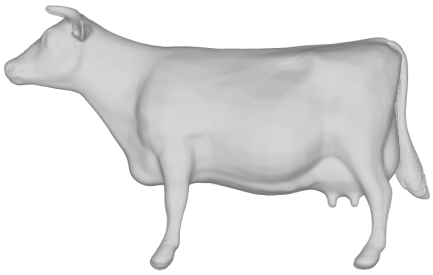
As illustrated in Figure 10, the baseline model generates artifacts both outside the object and on the cat's surface. While the standard exponential SP schedule removes the exterior artifacts, it leaves floating artifacts in the cat's torso and back. The sinusoidal noise schedule, however, successfully removes these torso artifacts and further reduces those on the back. It also achieves a lower total and surface loss.



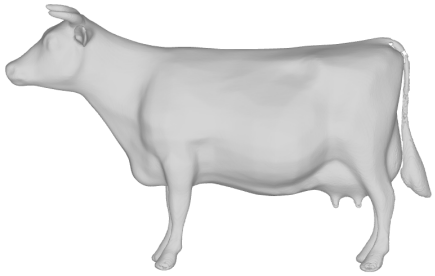
(a) Hashgrid MLP



(b) Hashgrid MLP + SP (Constant)



(c) Hashgrid MLP + SP (400/500 iter)

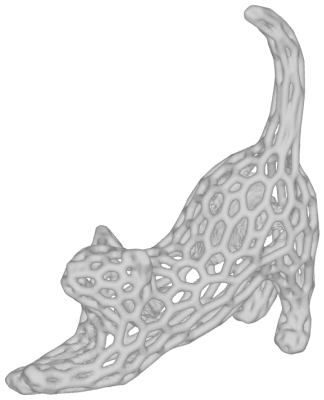


(d) Hashgrid MLP + SP (100/500 iter)

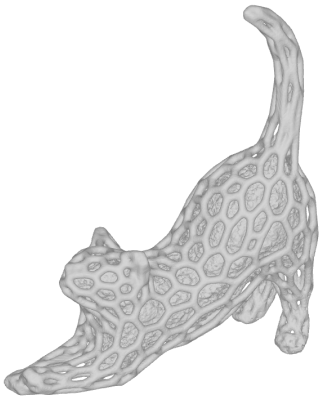
Figure 8: Cow

Table 3: Catstretch Losses

Model	Total Loss	Surface Loss	Normal Loss
Hashgrid MLP	4.63	2.41	**0.33**
Hashgrid MLP + SP (Exp)	3.53	1.53	0.49
Hashgrid MLP + SP (Sine)	**2.61**	**0.83**	0.67

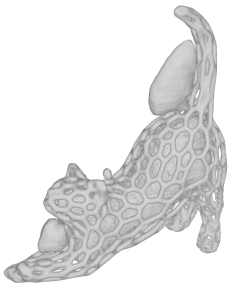


(a) Initial $\alpha = 0.02$

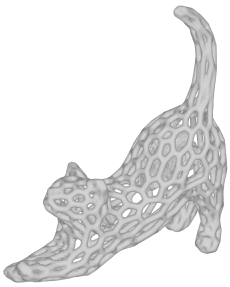


(b) Initial $\alpha = 0.04$

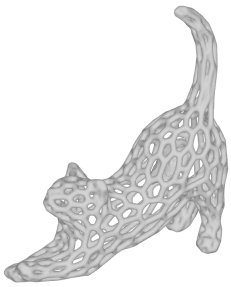
Figure 9: Catstretch



(a) Hashgrid MLP



(b) Hashgrid MLP + SP (Exp)



(c) Hashgrid MLP + SP (Sine)

Figure 10: Catstretch

References

[1] Selena Ling, Merlin Nimier-David, Alec Jacobson, and Nicholas Sharp. Stochastic preconditioning for neural field optimization. *ACM Transactions on Graphics*, 44(4):84:1–84:10, 2025.

[2] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022.

[3] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[4] Zimo Wang, Cheng Wang, Taiki Yoshino, Sirui Tao, Ziyang Fu, and Tzu-Mao Li. HotSpot: Signed distance function optimization with an asymptotically sufficient condition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2025)*, pages 1276–1286, 2025.