Scale-consistent learning with neural operators

Anonymous Author(s)
Affiliation
Address
email

Abstract

Data-driven models have emerged as a promising approach for solving partial differential equations (PDEs) in science and engineering. Previous machine learning (ML) models typically cover only a narrow distribution of PDE problems; for example, a trained ML model for the Navier-Stokes equations usually works only for a fixed Reynolds number and domain size. To overcome these limitations, we propose a data augmentation scheme based on scale-consistency properties of PDEs and design a scale-informed neural operator that can model a wide range of scales. Our formulation (i) leverages the fact that many PDEs possess a scale consistency under rescaling of the spatial domain, and (ii) is based on the discretization-convergent property of neural operators, which allows them to be applied across arbitrary resolutions. Our experiments on the 2D Darcy Flow, Helmholtz equation, and Navier-Stokes equations show that the proposed scale-consistency loss helps the scale-informed neural operator model generalize to Reynolds numbers ranging from 250 to 10000. This approach has the potential to significantly improve the efficiency and generalizability of data-driven PDE solvers in various scientific and engineering applications.

1 Introduction

Natural phenomena often exhibit strong relationships across a wide range of scales. A canonical example is the Koch snowflake, a fractal where the same generating rule applies at various scales, as illustrated in figure 1. Such multi-scale behavior is also seen in solutions of partial differential equations (PDEs), which model various phenomena in science and engineering. For instance, the Navier-Stokes equation, a classical model describing fluid motion, applies to both large-scale problems such as weather forecasting [1], and small-scale problems such as airfoils [2]. Despite the diversity in behaviors and frequency ranges, these problems can be reformulated in a dimensionless manner using scale parameters such as the Reynolds number in the Navier-Stokes equation and the wavenumber in the Helmholtz equations, leading to broad applicability.

Data-driven models have become a common methodology to complement or augment numerical solvers for physical simulation [3]. However, existing data-driven models are typically targeted to a single input variable, such as the coefficient function or initial condition, while other parameters remain fixed, including the domain size, boundary condition, and forcing term [4]. Recently, general foundation models have been proposed to capture various datasets under a wide range of conditions, or even multiple families of PDEs [5–11]. However, they do not explicitly capture relationships across a wide range of scales seen in physical systems. It is challenging for standard neural networks to capture different scales. Multi-scale physical phenomena exhibit varying intrinsic complexities and frequency ranges, often generated at different resolutions. In general, separate neural network models are trained for capturing each scale, making it cumbersome to couple them together and impose constraints across scales. In prior works, symmetry-based data augmentation has led to improved generalization and data efficiency [12–14]. For 2D PDEs, the symmetry group includes translation,

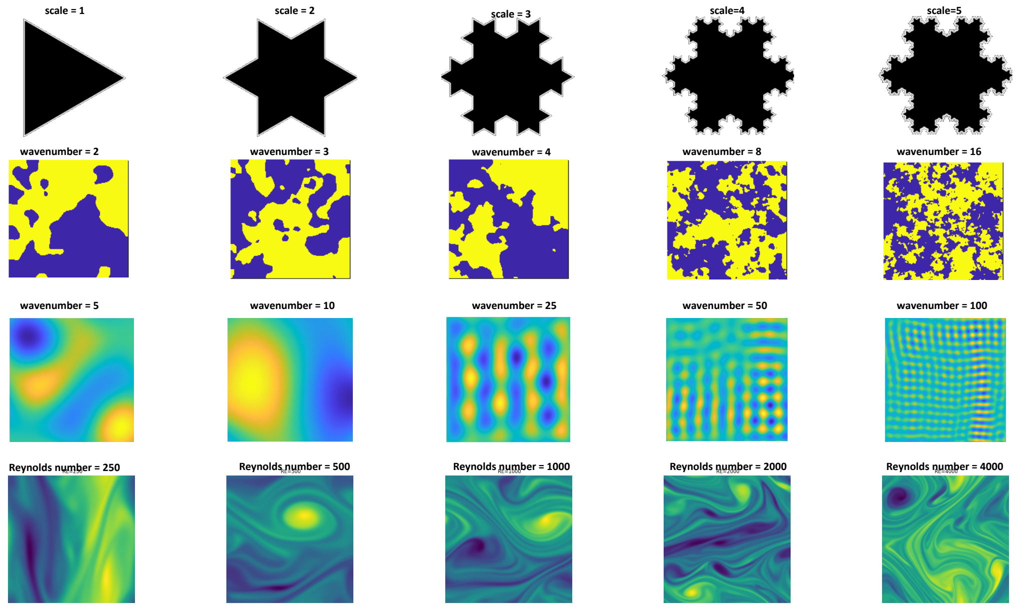

Figure 1: **Row 1**: Koch snowflake, a toy example of a problem at different scales. We consider three sets of PDEs with different scales. Large-scale problems are intrinsically more complex: **Row 2**: Darcy Flows $O(1)$, **Row 3**: Helmholtz Equation $O(k^2)$, **Row 4**: Navier Stokes equation $O(Re^3)$. In this work, we aim to design a learning framework to capture the consistency across the scales.

rotation, Galilean boost, and scaling. Among them, scale symmetry has been the least successful in improving performance [13, 14].

A possible reason is that the previous formulations of scale-symmetry are defined as global scaling, which does not introduce new scales. In this work, instead, we use sub-sampling (i.e., restricting to a subdomain) to create new data instances of various scales, which broadens the training distribution and allows the model to capture the solution operator over a wide range of scales. Neural operators are ideal models for imposing such constraints across multiple scales. Neural operators are designed to learn the solution operator of PDEs in a dimensionless manner [15–18]. They automatically rescale problems to a unit domain size without the need for interpolation. Neural operators are discretization-convergent, maintaining accuracy across various resolutions and converging as the resolution refines, making them ideal backbone models for addressing problems at various scales.

Our approach. In this work, we extend the notion of scale-symmetry to scale-consistency across problems with different intrinsic scales. The solution operators of PDEs are **scale-consistent**, meaning that applying the model on a large domain and then restricting it to its subdomain should yield the same result as directly applying the model to the subdomain. Based on scale-consistency, we propose a data augmentation scheme to generate instances with different scales. As shown in Figure 2, given a data instance, we sub-sample the domain to obtain new data at a smaller scale, and we calculate the new input coefficients, boundary conditions, and corresponding output solution, which are then rescaled to unit domain size by adjusting the scale parameter. The loss is defined as the difference between the sub-domain model output and the sub-sampled ground truth data. Furthermore, when the ground truth output solution is unknown, we can use the global model output as an estimate and define the loss as the difference between the models evaluated at different scales, which can be used to sample larger unseen scales.

To capture a wide range of scales, we propose a new architecture called the scale-informed neural operator, as shown in Figure 3. Since neural operators can handle any scale by design, we incorporate the scale parameter as additional input and embed the scale features in the Fourier space, helping the model capture different frequencies corresponding to different scale parameters. Similar to [19], we use a weight-sharing parameterization, where a single weight network is shared across all frequency modes. This approach not only reduces the size of the model parameters but also enables the handling

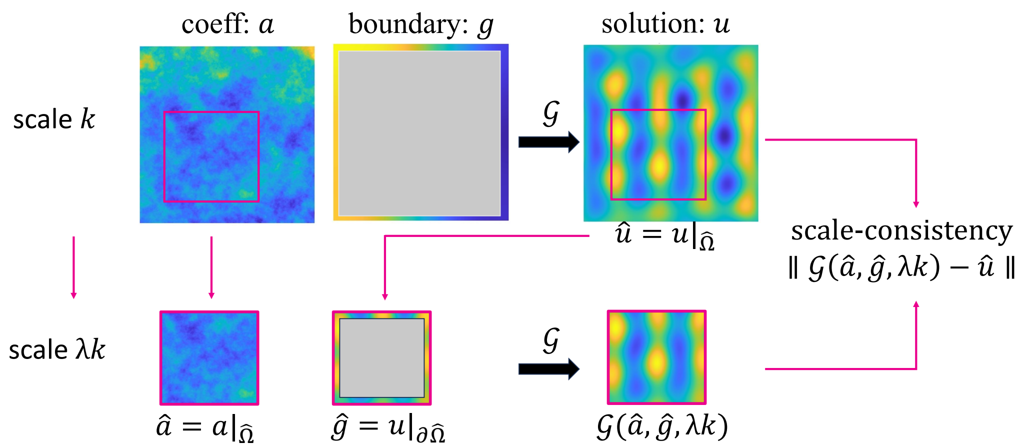

Figure 2: Self-consistency: given data instance of input coefficient a, boundary g, scale parameter k, and solution u, we restrict them to a sub-domain $\hat{\Omega}$ to obtain new data instance $\hat{a}, \hat{g}, \hat{u}$, which is rescaled to unit length by adjusting the parameter to $k \mapsto \lambda k$. The loss is defined as $\|u|_{\hat{\Omega}} - \mathcal{G}(a|_{\hat{\Omega}}, u|_{\partial\hat{\Omega}}, \lambda k)\|$ (4). When the solution u is unavailable, it can be approximated by $G(a, g, k)$ and the loss becomes $\|\mathcal{G}(a, g, k)|_{\hat{\Omega}} - \mathcal{G}(a|_{\hat{\Omega}}, \mathcal{G}(a, g, k)|_{\partial\hat{\Omega}}, \lambda k)\|$ (3), resulting in a self-supervised loss.

of higher frequencies without truncation. Additionally, it employs a multi-band U-shaped architecture that optimizes channel dimensions, using larger dimensions for lower frequency bands and smaller dimensions for higher frequency bands, reducing the weight tensor size in the original FNO. In summary, our main contributions are:

- We propose a data augmentation scheme based on scale-consistency that creates data instances with various intrinsic scales via sub-sampling and super-sampling.
- We show a theorem for Darcy equation that if the model is scale-consistent and it matches the target operator at simple instances, then it matches the target operator at all instances.
- We design a scale-informed neural operator that takes the scale parameter as input with weight-sharing parameterization and adaptive U-shape architecture to capture a wide range of scales.
- Our experiments on 2D Darcy Flow, Helmholtz equations, and Navier-Stokes equations demonstrate that the scale-consistency loss helps the scale-informed neural operator model extrapolate to wider scales with a 25% error reduction compared to baseline models.

The proposed approach has the potential to significantly improve the efficiency and generalizability of data-driven PDE solvers in various scientific and engineering applications, reducing the need for extensive training data and enabling the development of more flexible and foundational models.

2 Related work

Neural operator. Data-driven methods have become increasingly popular in learning Partial Differential Equations (PDEs) for scientific computing [1, 20]. As the solutions of PDEs live on infinite dimensional function space, the neural operators are constructed in a continuous manner to learn the underlying solution operators [15, 21, 17]. Among these, the Fourier Neural Operator (FNO) stands out as one of the most effective models [22] with numerous variants [23–25]. Specially AFNO [19] has an attention-like layer using FFT, and UNO [26] has a U-shape architecture. In this work, we use FNO as the backbone model.

Symmetry. In previous works [12–14], the scaling symmetry has been used for data augmentation. In the context of a dynamical system, symmetry is a relationship between the spatial domain, temporal domain, and magnitude. For the example of simplified Navier-Stokes equation $\partial_t u + u\nabla u = \nu \Delta u$, u is the velocity field, and ν is viscosity. If we transform the velocity field as $\mathcal{T}_\lambda : u(x,t) \mapsto \lambda u(\lambda x, \lambda^2 t)$, the equation becomes $\lambda^2(\partial_t u + u\nabla u) = \lambda^2 \nu \Delta u$ which is still satisfied with ν unchanged. The scaling symmetry form an equivariance $\mathcal{T}_\lambda \circ \mathcal{G}(u) = \mathcal{G} \circ \mathcal{T}_\lambda(u)$ with operator $\mathcal{G}$. However, it has been reported as not helpful in [13] and [14]. One reason may be that continuous

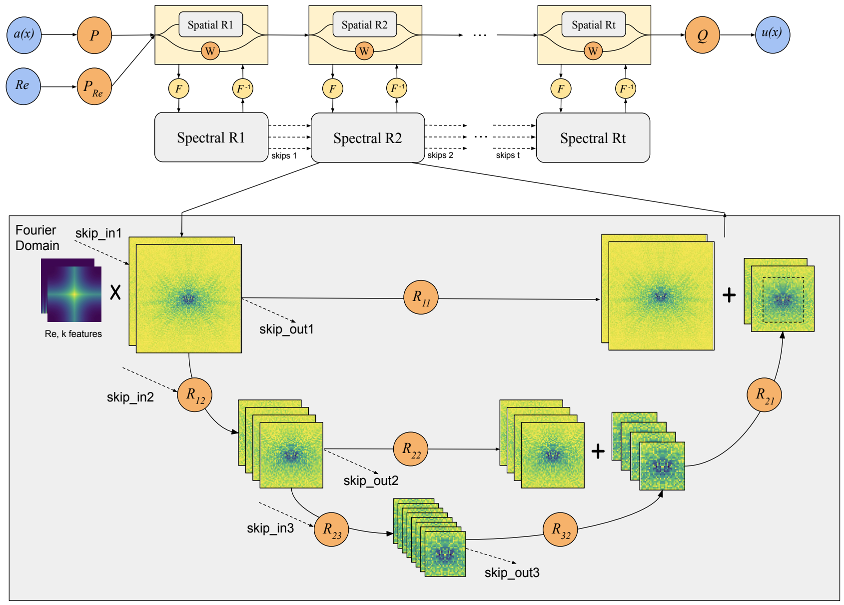

Figure 3: Diagram: scale-informed neural operator has a U-shape structure on the Fourier space. In the down block, the input tensors are truncated and lifted by complex layer R; in the up block, the tensors are projected and added to the inputs. Skip connections are added across the blocks.

scaling symmetry is ill-defined on periodic domains considered in [13]. Another reason could be that scaling the magnitude of velocity λu changes the range of inputs. In most scenarios, such as weather forecasts, the magnitude of velocity lies in a constant range. In this work, we consider a generalized scaling equivariance considering the scaling parameter $Re := LU/\nu$, where L is the domain size and U is the mean magnitude of the velocity field. The scale-transform not only scale the input function (vorticity), but also scale the parameter, $\mathcal{T}_\lambda : (\omega(x), Re) \mapsto (\omega(\lambda x), \lambda^2 Re)$. $\mathcal{T}_\lambda \circ \mathcal{G}(\omega, Re) = \mathcal{G} \circ \mathcal{T}_\lambda(\omega, Re)$. In the new formulation, L scales as λx as before, but the magnitude $U = O(1)$ and dt are unchanged.

3 Scale-Consistency

Many PDEs possess symmetries, which are reflected by the fact that the equations remain invariant under transformations such as translation, rotation, or re-scaling. An example is the Darcy flow problem on a domain $\Omega \in \mathbb{R}^d$.

$$\begin{cases} -\nabla \cdot (a(x)\nabla u(x)) = 0, \quad (x \in \Omega), & \text{(1a)} \\ u(x) = g(x), \quad (x \in \partial\Omega). & \text{(1b)} \end{cases}$$

Then the associated solution operator $\mathcal{G}$ can be viewed as a mapping

$$(a(x), g(x)) \mapsto \mathcal{G}(a, g)(x) := u(x).$$

Re-scale symmetry. Let $\mathcal{T}_\lambda$ be the re-scaling operator defined by $(\mathcal{T}_\lambda a)(x) := a(\lambda x)$. In the absence of boundary conditions, the scale symmetry implies an equivariance property of $\mathcal{G}$:

$$\mathcal{G}(\mathcal{T}_\lambda a, \dots) = \mathcal{T}_\lambda \mathcal{G}(a, \dots).$$

The boundary condition (or simply the fact that the PDE is defined on a bounded domain Ω) breaks the scale symmetry; if $u : \Omega \to \mathbb{R}$ is defined on the domain Ω, then $\mathcal{T}_\lambda u$ is defined on the rescaled domain $\Omega_\lambda = \{\lambda^{-1}x | x \in \Omega\}$, and we are generally lacking information about the boundary condition on the boundary of the re-scaled domain $\partial\Omega_\lambda$. Thus, the presence of boundaries in most problems of practical interest makes it difficult to leverage the underlying symmetry properties of the equations in a straightforward way.

Nevertheless, under some conditions on the domain Ω (e.g. $\Omega = [0,1]^d$ is a cube), the formal scale symmetry of the solution operator of (1) implies that if $u(x)$ solves (1) with coefficient field $a(x)$ and with boundary condition $g(x)$, then the rescaled function $u_\lambda(x) = \mathcal{T}_\lambda u(x) = u(\lambda x)$, solves

$$\begin{cases} -\nabla \cdot (a_\lambda(x) \nabla u_\lambda(x)) = 0, & (x \in \Omega_\lambda), \\ \qquad\qquad u_\lambda(x) = \mathcal{T}_\lambda u(x), & (x \in \partial\Omega_\lambda). \end{cases}$$

i.e. $u_\lambda(x)$ is a solution of the Darcy flow problem on domain Ω_λ, with coefficient field $a_\lambda = \mathcal{T}_\lambda a$, and boundary condition $(\mathcal{T}_\lambda u)|_{\partial\Omega_\lambda}$. Another operation we can perform is the restriction from Ω_λ to Ω when $\lambda \leq 1$. Intuitively, this condition expresses the fact that the solution operator of (1) is **scale-consistent**: The solution on a smaller subdomain $\Omega \subset \Omega_\lambda$ can either be obtained

1. by solving the PDE over the entire domain Ω_λ and then restricting the solution u to the smaller domain $u|_\Omega$.
2. $u|_\Omega$ can be obtained by solving the PDE directly on the domain Ω, and imposing consistent boundary condition $u|_{\partial\Omega}$.

Combining the scale symmetry with restriction, we obtain the following identity in terms of the solution operator $\mathcal{G}$:

$$[\mathcal{T}_\lambda \mathcal{G}(a,g)]|_\Omega = \mathcal{G}([\mathcal{T}_\lambda a]|_\Omega, [\mathcal{T}_\lambda u]|_{\partial\Omega}) \equiv \mathcal{G}([\mathcal{T}_\lambda a]|_\Omega, [\mathcal{T}_\lambda \mathcal{G}(a,g)]|_{\partial\Omega}), \quad (\lambda \leq 1). \tag{2}$$

For the solution operator, this identity holds for arbitrary inputs $a(x)$ and $g(x)$. The scale-consistency (2) can be used as a loss to train solution operators. Informally, if an operator satisfies (2), then it must be the target solution operator. The proof can be found at B.

Theorem 3.1 (Scale-consistency (informal)) *If an operator Ψ satisfies the scale-consistency (2) and it matches the ground truth solution operator $\mathcal{G}$ on nearly constant functions, then $\Psi \equiv \mathcal{G}$.*

Scale-consistency loss. The first way to impose such a constraint is by introducing a loss of the form

$$L(a,g) = \|\mathcal{T}_\lambda \Psi(a,g) - \Psi(\mathcal{T}_\lambda a, \mathcal{T}_\lambda \Psi(a,g)|_{\partial\Omega})\|. \tag{3}$$

Note that this is an unsupervised loss term that doesn't require access to $\mathcal{G}$. It only requires producing input function samples (a,g). When solution data u is available, the scale-consistency loss simplifies to

$$L(a,g) = \|\mathcal{T}_\lambda u - \Psi(\mathcal{T}_\lambda a, \mathcal{T}_\lambda u|_{\partial\Omega}))\|. \tag{4}$$

Infinitesimal scale-consistency. Another way to impose this constraint is by taking the λ-derivative of (2), leading to:

$$\partial_\lambda \mathcal{T}_\lambda \mathcal{G}(a,g) = \partial_\lambda \left[\mathcal{G}(\mathcal{T}_\lambda a, \mathcal{T}_\lambda \mathcal{G}(a,g)|_{\partial\Omega})\right].$$

We note that if $a(x)$ is a function, then the derivative $\partial_\lambda \mathcal{T}_\lambda a$ evaluated at $\lambda = 1$, is given by

$$\partial_\lambda \mathcal{T}_\lambda a|_{\lambda=1} = [\partial_\lambda a(\lambda x)]_{\lambda=1} = x \cdot \nabla a(x),$$

i.e., a radial spatial derivative of a. Substitution of this identity, and noting that $\mathcal{T}_{\lambda=1} a = a$ and $\mathcal{T}_{\lambda=1} \mathcal{G}(a,g)|_{\partial\Omega} = g$, implies that

$$x \cdot \nabla_x [\mathcal{G}(a,g)](x) = \left\langle \frac{\delta \mathcal{G}(a,g)}{\delta a}, x \cdot \nabla_x a \right\rangle + \left\langle \frac{\delta \mathcal{G}(a,g)}{\delta g}, x \cdot \nabla_x [\mathcal{G}(a,g)] \right\rangle. \tag{5}$$

We observe that while (2) is highly non-linear, the infinitesimal constraint is quadratic in $\mathcal{G}$.

3.1 Scale-dependent problem: extension beyond scale symmetry

The scale-consistency constraint can be written in greater generality, even if the underlying PDE has no scale symmetry. In this case, the domain could be an input to the operator, and the relevant scale-consistency would be

$$\mathcal{G}(a, g; \Omega)|_{\Omega'} = \mathcal{G}(a|_{\Omega'}, \mathcal{G}(a, g, \Omega)|_{\partial\Omega'}; \Omega'), \quad (\Omega' \subset \Omega).$$

In some cases, this is equivalent to scaling certain parameters in the PDE, as explained below.

Helmholtz equation. An example not satisfying scale symmetry is the Helmholtz equation,

$$\Delta u(x) + k^2 u(x) = f(x). \tag{6}$$

In this case, a rescaling of the spatial variable corresponds to a rescaling of the frequency k^2, i.e. $u_\lambda(x) = u(\lambda x)$ solves $\Delta u_\lambda(x) + \lambda^{-2}k^2 u_\lambda(x) = \lambda^{-2} f(\lambda x)$, or

$$\Delta u_\lambda(x) + k_\lambda^2 u_\lambda(x) = f_\lambda(x),$$

with $k_\lambda := \lambda^{-1}k$, $f_\lambda(x) := \lambda^{-2} f(\lambda x)$. Thus, the scale-consistency constraint involves the whole family of PDEs, $\Delta u + k^2 u = f$, for $k > 0$, with the transform on parameter $\mathcal{T}_\lambda(k) = \lambda k$.

Navier-Stokes equation. Another example is the two-dimensional incompressible Navier-Stokes equation in vorticity formulation,

$$\partial_t \omega(x,t) + u(x,t) \cdot \nabla \omega(x,t) = \nu \Delta \omega(x,t), \tag{7}$$

describing the evolution of the vorticity $\omega = \mathrm{curl}(u)$ of an underlying flow velocity field u. Rescaling the spatial variable x corresponds to rescaling the viscosity ν; $\omega_\lambda(x,t) = \omega(\lambda x, t)$ solves

$$\partial_t \omega_\lambda(x,t) + u_\lambda(x,t) \cdot \nabla \omega_\lambda(x,t) = \nu_\lambda \Delta \omega_\lambda(x,t),$$

where $\nu_\lambda := \lambda^{-2}\nu$, and where u_λ is the flow field associated with ω_λ, s.t. $\mathrm{curl}(u_\lambda) = \omega_\lambda$.

3.2 Main algorithms

Remark: neural operator automatically rescales input to unit length. For standard neural networks such as convolution neural networks, re-scaling $\mathcal{T}$ needs to be implemented as interpolation. However, in the design of neural operators such as FNO, the domain size is implicitly re-scaled to unit length, where the Fourier basis is defined with length $[0, 1]$. Given $\mathcal{T}_\lambda f$ defined on domain $[0, \lambda]$, Fourier neural operator Ψ automatically rescale it to unit length,

$$\Psi(\mathcal{T}_\lambda f, \dots) := \Psi(\mathcal{T}_{1/\lambda}\mathcal{T}_\lambda f, \dots) = \Psi(f, \dots).$$

where f is defined on unit size $[0, 1]$. Therefore, the re-scaling $\mathcal{T}$ is omitted in the algorithm.

Scale-down. The down-algorithm is based on equation (4), where we use sub-sampling (i.e., restrict to sub-domain) to obtain instance with smaller scale $\lambda k < k$. Given the input and output data $\{(a, g, k), u\}$ defined on domain Ω, we truncate the domain into a smaller sub-domain $\hat{\Omega}$. The input and output restrict to the sub-domain, along with the re-scaled parameter, become a new data instance $\{(\hat{a}, \hat{g}, \hat{k}), \hat{u}\}$. We compute the consistency loss as the difference between the model evaluated on restricted input $\Psi(\hat{a}, \hat{g}, \hat{k})$ and the restricted output $\hat{u}$.

Algorithm 1 Scale-down via sub-sampling

1: **input**: data pair $\{(a, g, k), u\}$ on domain $\Omega = [0, 1]^2$, model Ψ, and sampling rate $\lambda < 1$.
2: sample the sub-domain $\hat{\Omega} = [w, w + \lambda] \times [h, h + \lambda]$, where $w, h \sim Unif[0, 1 - \lambda]$.
3: define new instance with $(\hat{a} = a|_{\hat{\Omega}}, \hat{g} = u|_{\partial\hat{\Omega}}, \hat{k} = \lambda\, k), \hat{u} = u|_{\hat{\Omega}}$.
4: **output**: consistency loss $= \|\Psi(\hat{a}, \hat{g}, \hat{k}) - \hat{u}\|$.

Scale-up. The up-scaling algorithm is based on equation (3), where we sample new instances corresponding to larger scale $\lambda k > k$. Given the distributions μ for a and ν for g, we can sample new instance a, g with larger scale λk and apply Algorithm 1. Different from 1, we do not have the ground truth output u on the larger scale. Instead, we estimate using the model $u = \Psi(a, g, \lambda k)$.

Algorithm 2 Scale-up

1: **input**: distributions of inputs μ, ν, model Ψ, scale parameter k, and sampling rate $\lambda > 1$.
2: sample new instances $a \sim \mu, g \sim \nu$. Define new scale as λk.
3: estimate the solution of new domain $u = \Psi(a, g, \lambda k)$.
4: call Algorithm 1 with input $\{(a, g, \lambda k), u\}$ with scale $1/\lambda$.
5: **output**: consistency loss $= \|\Psi(a|_{\hat{\Omega}}, \Psi(a, g, \lambda k)|_{\partial\hat{\Omega}}, k) - \Psi(a, g, \lambda k)|_{\hat{\Omega}}\|$.

4 Scale-informed Neural Operator

The scale-informed neural operator is based on the FNO [22], where convolution is implemented as a pointwise multiplication in the Fourier space. Since FNO automatically rescales its input to unit length, we design a scale embedding in the Fourier space to inform the model of the scale parameter k. Further, we design a U-shaped architecture to optimize the channel dimension.

4.1 Scale-informed MLP on Fourier Space

In the previous FNO, the weight tensor R is defined as a $(M_1 \times \cdots \times M_d \times C_{in} \times C_{out})$-tensor, which is sufficient for lower-dimensional problems with fewer total modes M. For larger-scale problems, such as highly turbulent flows, the weight tensor R becomes prohibitively large. Therefore, we propose an implicit representation of the weight tensor similar to AFNO [19], where the complex weight R with the shape $(C_{in} \times C_{out})$ is shared across all modes $(M_1 \times \cdots \times M_d)$.

Different from AFNO, we further define the features of scale k and mode index ξ as input, so that the transform R can behave correspondingly with respect to different scales k and modes ξ. Let C_k be the embedding channel dimension; we define scale features as $h(k)_i = k^{i/(C_k-1)}$ for $i = 0, 1, \ldots, C_k - 1$, which covers a wide range from $k^{0/(C_k-1)} = 1$ to $k^{(C_k-1)/(C_k-1)} = k$. The input $f_t(\xi) \in \mathbb{C}^{C_{in}}$ is first element-wise multiplied with the features of the scale parameter and wavenumber $h(k, \xi)$, and then multiplied with R, followed by a group normalization and a complex activation σ as defined in Section C.4. The transform $\mathcal{K}$ can be viewed as a kernel function defined on the Fourier space:

$$(\mathcal{K} f_{t+1})(\xi) = \sigma\big(R(f_t(\xi) \odot h(k, \xi))\big). \tag{8}$$

4.2 Multi-band Architecture

The Fourier signal usually follows an ordered structure, where the energy decays as the wavenumber increases. Therefore, previous methods such as FNO [22] and SNO [27] choose to truncate to a fixed number of frequencies by omitting higher frequencies. Similar to previous works such as UNet [28] and UNO, we design a multi-band structure to gradually shrink the frequency bands, as shown in Figure 3. Different from UNO, which applies spectral convolutions at each down and up block, in this work, we define the U-shaped structure fully in the Fourier space. Given the initial channel dimension C, maximum input modes M, and a predefined number of levels L, we define C_l and M_l as $C_l = 2^l C$ and $M_l = 2^{-l} M$, where each block has shape $C_l^2 M_l^d$. For $d = 2$, $C_l^2 M_l^2 = C^2 M^2$, so each level has the same size. We define the first level in MLP formulation, where R_1 has the shape $(C_{in} \times C_{out})$, and higher levels in tensor formulation with $(M_l^d \times C_{in} \times C_{out})$.

Down Blocks. At each level, the input tensor is transformed into shape $(B, C_l, M_{1,l}, \ldots, M_{d,l})$ with the down blocks. The down block consists of two steps:

- Truncation: Truncate the modes from M_l to M_{l+1}.
- $\mathcal{K}$ Layer: Apply $R_{l,l+1}$ to lift the channel dimension from C_l to C_{l+1}, followed by a complex activation function.

After reaching the lowest level, we have collected the input $\{f, f_1, \ldots, f_L\}$.

Up Blocks. Conversely, the up blocks lift the tensor back to the original shape.

- $\mathcal{K}$ Layer: Apply $R_{l,l-1}$ to project the channel dimension from C_l to C_{l-1}, followed by a complex activation function.
- Summation: Combine the output of mode M_l with the inputs f_l of M_{l-1} by adding corresponding modes.

Optionally, we have the middle block $R_{l,l}$ applied on f_l. The complex kernel consists of learned complex matrices $\{R_{l,l+1}\}_1^{L-1}$ and $\{R_{l,l-1}\}_2^{L}$.

Skip Connection. Furthermore, we define skip connections in the Fourier space. After the down block, we save the intermediate tensors $\{f, f_1, \ldots, f_L\}$ and pass them to the next layer. The skip-out tensor at layer t will be added back at the next layer $t + 1$ in the down block.

Boundary. For boundary value problems, we take the boundary as an additional input. For a 1-dimensional boundary on a 2-dimensional square domain, we extend the boundary to 2D by repeating along the other dimension. For Dirichlet-type boundaries, it is known that the boundary is the restriction of the solution, and their magnitudes should be similar. Therefore, we define a normalization at the end of the model that multiplies the output by the magnitude of the boundary.

5 Experiments

We generated datasets for the Darcy Flow, Helmholtz equation, and Navier-Stokes equation, each spanning a wide range of scales. For each test case, we trained the models on a narrow range of scales and compared the performance with and without self-consistency augmentation. All experiments

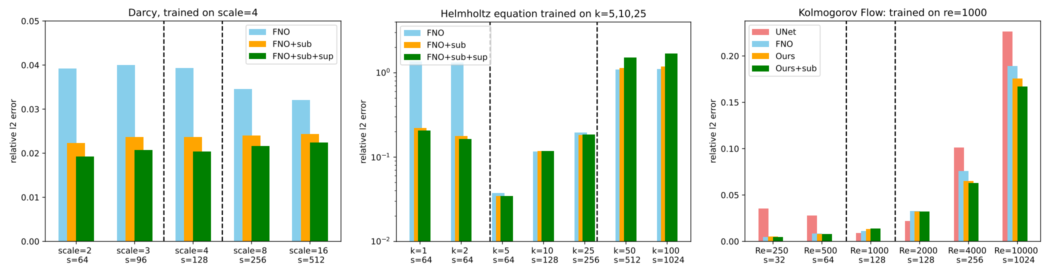

Figure 4: Supervised training versus self-supervised training evaluated on multiple scales. **Left**: Darcy Flows. **Mid**: Helmholtz Equation. **Right**: Navier-Stokes equation. The range of the black dashed line shows the region of training scales. It is shown that self-consistency loss helps generalization to unseen scales.

were run on Nvidia A100 (40GB) or P100 (16GB) GPUs. We used the Adam optimizer with a default learning rate of $1e-3$, weight decay of $1e-4$, and a cosine annealing scheduler, trained for 100 epochs. The error metric is relative L2 error. The choice of hyperparameters can be found in Appendix D.1. The results show that self-consistency augmentation helps the model generalize better to unseen scales.

5.1 Self-consistency loss

In the first part, we compare FNO, UNet, and our models, with and without the self-consistency loss. For Darcy and Helmholtz equations, where the input distribution is given as a Gaussian random field, we apply both sub-sampling 1 and super-sampling 2. For the Navier-Stokes equation, the input distribution is unknown, so we only apply sub-sampling. The detailed data generation can be found at A.

Darcy Flow. We considered the Darcy Flow (1) with a non-zero Dirichlet boundary. The input coefficient a was sampled as $a = 2 + 10 \cdot \mathbb{1}_{[\hat{a}>0]}$ representing two types of media with values 2 and 10, where $\hat{a}$ is sampled from a Gaussian random field $\mathcal{N}(0, \mathcal{C})$. The covariance kernel $\mathcal{C}$ has Fourier coefficients $\exp(-\sigma|\xi|^{1/2})$. We considered wave lengths $\sigma = 2, 4/3, 1, 1/2, 1/4 := 1/k$, which is inverse to the scales. The resolutions were $s = 64, 96, 128, 256, 512$, respectively. We train 1024 instances for training and 128 for testing. The data generation details can be found in Appendix A.1. We used $\sigma = 1$ for training. Since Darcy has no scale parameters, we used FNO with and without scale-consistency. As shown in Figure 4 (left) and Table 2, FNO with scale-consistency reduced the error by half compared to the baseline.

Helmholtz. We considered the Helmholtz equation (6) with a non-zero Dirichlet boundary. The input coefficients a, g were sampled from a fixed Gaussian random field, with varying wavenumbers $k = 1, 2, 5, 10, 25, 50, 100$. The resolutions were $64, 64, 64, 128, 256, 512, 1024$, respectively. We train 1024 instances for training and 128 for testing. The data generation details can be found in Appendix A.2. We used $k = 5, 10, 25$ for training. As shown in Figure 4 (middle) and Table 3, the scale-informed FNO with scale-consistency reduced the error by half compared to the baseline FNO on smaller wavenumbers $k = 1, 2$, but neither model captured larger scales $k = 50, 100$, since Helmholtz equation has very different behaviors on larger scales.

Navier-Stokes. We considered the Navier-Stokes equation (7) defined on sub-domain similar to applications in climate. The input is the vorticity field of the previous ten time frames ω_0. We considered Reynolds numbers ranging from $Re = 250, 500, 1000, 2000, 4000, 10000$. The resolutions were $32, 64, 128, 128, 256, 512$, respectively. We train 50 trajectories for training and 5 (per each Re) for testing, where each trajectory consists of 300 time steps, with $dt = 0.1$. The data generation details can be found in Appendix A.3. We used $Re = 1000$ for training. As shown in Figure 4 (right) and Table 1, the scale-informed multi-band neural operator with scale-consistency reduced the error by 1/4 compared to the baseline UNet on unseen $Re = 250, 500, 4000, 10000$.

5.2 Ablation study

Further, we conducted an ablation study on the proposed model in the standard supervised learning setting with periodic Navier-Stokes equation with fixed scales $Re = 5000$ (with forcing) and $Re = 10000$ (zero forcing) as in [29]. For baselines, we consider FNO [22], UNet [28], FNO-UNet

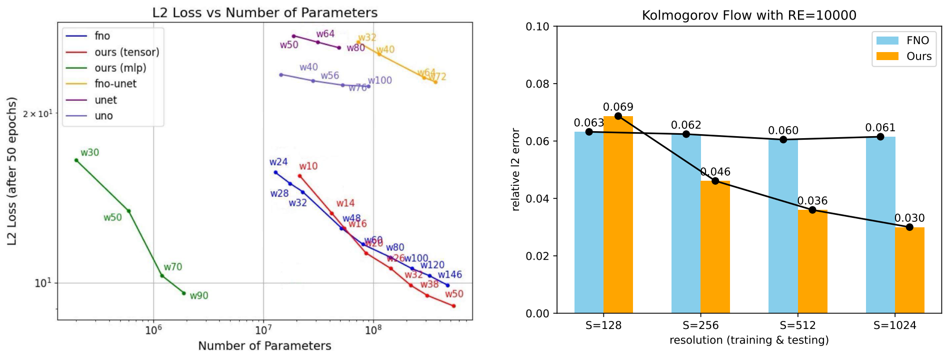

Figure 5: Ablation study. **left**: Cost Accuracy: Kolmogorov Flow with RE=5000 on resolution $s = 128 \times 128$. We use full modes for all the models. Our model (tensor) converges faster than baseline models. Further, the model (mlp) achieves comparative accuracy with 1/10 of the parameters. **right**: discretization convergence: the proposed model does not truncate to a fixed bandwidth. As the training resolution increases, the model's error converges while the baseline FNO remains the same.

Table 1: Navier-Stokes equation trained on RE1000, zero-shot test on various RE.

Model **Resolution**	parameters	re250 32	re500 64	re1000 128	re2000 128	re4000 256	re10000 512
UNet	25M	0.03554	0.02754	**0.00897**	**0.02169**	0.10120	0.22642
UNet+aug	25M	0.04181	0.03164	0.06964	0.08320	0.09960	0.18662
FNO	8M	0.00482	0.00819	0.01122	0.03260	0.07586	0.18902
FNO+aug	8M	**0.00348**	**0.00608**	0.00929	0.02793	0.07282	0.19559
Ours	34M	0.00508	0.00781	0.01305	0.03232	0.06502	0.17567
Ours+aug	34M	0.00459	0.00791	0.01388	0.03201	**0.06293**	**0.16715**

[30], and UNO [26]. The results show that our model achieves a smaller error rate with one-tenth of the parameters compared to the previous FNO at the cost of longer runtime, as shown in Figure 5 (left). Since the model does not truncate the maximum Fourier frequency, its accuracy improves as the resolution refines, as shown in Figure 5 (right).

Limitations. While sub-sampling 1 is generally helpful, super-sampling 2 requires input distribution known to sample new instances. For the example of the Navier-Stokes equation, it is challenging to sample input history from the unseen distribution (attractor) of higher Reynolds numbers. It could be an interesting direction to combine with generative models [31] to sample virtual inputs. Besides, our implementation of sub-sampling and super-sampling is limited to simple topology and cannot create periodic boundary domains such as torus or sphere.

6 Conclusion

In this paper, we consider the scale consistency for learning solution operators on partial differential equations (PDEs) across various scales. By leveraging the scale-consistency properties of PDEs and designing a scale-informed neural operator, we demonstrated the ability to model a wide range of scales. Experimental results showed significant improvements in generalization to unseen scales, with better generalization errors compared to baseline models. The proposed self-supervised training scheme further enhances model performance by creating virtual instances via sub-sampling and super-sampling. This approach holds promise for improving the efficiency and generalizability of data-driven PDE solvers, reducing the need for extensive training data, and enabling the development of more flexible and foundational models for scientific and engineering applications.

References

[1] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[2] Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Christian Hundt Maximilian Stadler, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3d pdes. *arXiv preprint arXiv:2309.00583*, 2023.

[3] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

[4] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[5] Shashank Subramanian, Peter Harrington, Kurt Keutzer, Wahid Bhimji, Dmitriy Morozov, Michael W Mahoney, and Amir Gholami. Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior. *Advances in Neural Information Processing Systems*, 36, 2024.

[6] Liu Yang, Siting Liu, Tingwei Meng, and Stanley J Osher. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39):e2310142120, 2023.

[7] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, et al. Multiple physics pretraining for physical surrogate models. *arXiv preprint arXiv:2310.02994*, 2023.

[8] Zhanhong Ye, Xiang Huang, Leheng Chen, Hongsheng Liu, Zidong Wang, and Bin Dong. Pdeformer: Towards a foundation model for one-dimensional partial differential equations. *arXiv preprint arXiv:2402.12652*, 2024.

[9] Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training. *arXiv preprint arXiv:2403.03542*, 2024.

[10] Junhong Shen, Tanya Marwah, and Ameet Talwalkar. Ups: Towards foundation models for pde solving via cross-modal adaptation. *arXiv preprint arXiv:2403.07187*, 2024.

[11] Md Ashiqur Rahman, Robert Joseph George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A Yeh, Jean Kossaifi, et al. Pretraining codomain attention neural operators for solving multiphysics pdes. *arXiv preprint arXiv:2403.12553*, 2024.

[12] Rui Wang, Robin Walters, and Rose Yu. Incorporating symmetry into deep dynamics models for improved generalization. *arXiv preprint arXiv:2002.03061*, 2020.

[13] Johannes Brandstetter, Max Welling, and Daniel E Worrall. Lie point symmetry data augmentation for neural pde solvers. In *International Conference on Machine Learning*, pages 2241–2256. PMLR, 2022.

[14] Grégoire Mialon, Quentin Garrido, Hannah Lawrence, Danyal Rehman, Yann LeCun, and Bobak Kiani. Self-supervised learning with lie symmetries for partial differential equations. *Advances in Neural Information Processing Systems*, 36:28973–29004, 2023.

[15] Nikola B Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *J. Mach. Learn. Res.*, 24(89):1–97, 2023.

[16] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, mar 2021.

[17] Georgios Kissas, Jacob H Seidman, Leonardo Ferreira Guilhoto, Victor M Preciado, George J Pappas, and Paris Perdikaris. Learning operators with coupled attention. *Journal of Machine Learning Research*, 23(215):1–63, 2022.

[18] Bogdan Raonic, Roberto Molinaro, Tobias Rohner, Siddhartha Mishra, and Emmanuel de Bezenac. Convolutional neural operators. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023.

[19] John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers. *arXiv preprint arXiv:2111.13587*, 2021.

[20] Gege Wen, Zongyi Li, Qirui Long, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. Real-time high-resolution co 2 geological storage prediction using nested fourier neural operators. *Energy & Environmental Science*, 2023.

[21] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, Mar 2021.

[22] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[23] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 2021.

[24] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *arXiv preprint arXiv:2207.05209*, 2022.

[25] Haydn Maust, Zongyi Li, Yixuan Wang, Daniel Leibovici, Oscar Bruno, Thomas Hou, and Anima Anandkumar. Fourier continuation for exact derivative computation in physics-informed neural operators. *arXiv preprint arXiv:2211.15960*, 2022.

[26] Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-no: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 2022.

[27] Vladimir Fanaskov and Ivan Oseledets. Spectral neural operators. *arXiv preprint arXiv:2205.10573*, 2022.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[29] Zongyi Li, Miguel Liu-Schiaffini, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Learning chaotic dynamics in dissipative systems. *Advances in Neural Information Processing Systems*, 35:16768–16781, 2022.

[30] Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

[31] Jae Hyun Lim, Nikola B Kovachki, Ricardo Baptista, Christopher Beckham, Kamyar Azizzadenesheli, Jean Kossaifi, Vikram Voleti, Jiaming Song, Karsten Kreis, Jan Kautz, et al. Score-based diffusion models in function space. *arXiv preprint arXiv:2302.07400*, 2023.

[32] Maarten V de Hoop, Daniel Zhengyu Huang, Elizabeth Qian, and Andrew M Stuart. The cost-accuracy trade-off in operator learning with neural networks. *arXiv preprint arXiv:2203.13181*, 2022.

[33] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[34] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. *arXiv preprint arXiv:1705.09792*, 2017.

[35] Ziyuan Liu, Yuhang Wu, Daniel Zhengyu Huang, Hong Zhang, Xu Qian, and Songhe Song. Spfno: Spectral operator learning for pdes with dirichlet and neumann boundary conditions. *arXiv preprint arXiv:2312.06980*, 2023.

A Dataset

A.1 Darcy Flow

We use a finite element solver with a resolution of 1024 to generate the dataset. The dataset is similar to the one used in [22], but with non-zero Dirichlet boundary conditions.

A.2 Helmholtz Equation

We prepare the data using a finite element solver with a resolution of 1024. It is worth noting that for physical equations, the Helmholtz equation is often paired with an impedance boundary condition, namely a Robin boundary condition:

$$\nabla u \cdot \nu = ik\beta u + g$$

For simplicity, we use Dirichlet boundary conditions for operator learning in this work. It is important to note that the Helmholtz equation with Dirichlet boundary conditions is a wave scattering problem, which may have multiple solutions as studied in [32].

A.3 Navier-Stokes Equation

We consider a partially observed Navier-Stokes equation, inspired by practical applications in weather forecasting and oceanography, where a specific subdomain of the globe is of interest. We generate an isotropic Navier-Stokes equation on a periodic domain $[0, 1]^2$ and truncate it to a $[0, 0.5] \times [0, 0.5]$ subdomain. For convenience, we set the forcing term to zero and study the decay of turbulence.

Since the underlying system is defined on a periodic boundary, we generate the data using a pseudo-spectral solver with Crank-Nicolson time updates.

B Proof of Theorem 3.1

Theorem B.1 *Suppose that the neural network solution Ψ is scale-consistent and is accurate for constant inputs. Namely if*

1. *For almost constants a,*
$$\Psi(a, g) = \mathcal{G}(a, g),$$
2. *Ψ satisfies (2) exactly along with translation symmetry,*
3. *Ψ satisfies the boundary condition exactly.*

then we must necessarily have $\Psi \equiv \mathcal{G}$.

Proof B.1 *We use an overlapping partition of the domain Ω into subdomains and zoom in. Suppose $\Omega = \cup_{i\in I}\Omega_i$ is an overlapping partition of Ω, such that each one of Ω_i is a rescaling and shifting of Ω and is of size h. For sufficiently small h, the coefficient a is almost constant in each one of the Ω_i. Consider a partition of unity $1 = \sum_{i\in I} \chi_i$ such that χ_i has support in Ω_i. By scale-consistency (2), we know that Ψ is exact when restricted to Ω_i. Thus we have by the weak formulation that*

$$(a\nabla\Psi, \nabla v_i) = (a\nabla\mathcal{G}, \nabla v_i) = 0$$

for any v_i supported in Ω_i. Therefore for any v supported in Ω, we can take $v_i = \chi_i v$ and summing up the weak formulation for all i and arrive at

$$(a\nabla\Psi, \nabla v) = (a\nabla\mathcal{G}, \nabla v) = 0$$

Therefore Ψ is a weak solution with the desired boundary condition, and thus $\Psi = \mathcal{G}$.

C Implementation Details

C.1 Fourier Neural Operator

The neural operator, proposed in [33], is formulated as an iterative architecture $f_0 \mapsto f_1 \mapsto \ldots \mapsto f_T$, where f_j for $j = 0, 1, \ldots, T-1$ is a sequence of functions, each taking values in $\mathbb{R}^C$. The input $a \in \mathcal{A}$ is first lifted to a higher-dimensional representation $f_0(x) = P(a(x))$ by the local transformation P, which is usually parameterized by a shallow fully-connected neural network. The output $u(x) = Q(f_T(x))$ is the projection of f_T by the local transformation $Q : \mathbb{R}^C \to \mathbb{R}^{d_u}$. In each iteration, the update $f_t \mapsto f_{t+1}$ is defined as the composition of a non-local integral operator $\mathcal{K}$ and a local, nonlinear activation function σ.

$$\mathcal{G}_\theta := \mathcal{Q} \circ (W_L + \mathcal{K}_L) \circ \cdots \circ \sigma(W_1 + \mathcal{K}_1) \circ \mathcal{P} \tag{9}$$

Denote the layer $\sigma(W_l + \mathcal{K}_l)$ mapping the representation $f_t \mapsto f_{t+1}$ by

$$f_{t+1}(x) := \sigma\Big(W f_t(x) + \big(\mathcal{K}(a;\phi) f_t\big)(x)\Big), \tag{10}$$

where $\mathcal{K}$ maps to bounded linear operators on $\mathcal{U}(D;\mathbb{R}^C)$ and is parameterized by $\phi \in \Theta_{\mathcal{K}}$, $W : \mathbb{R}^C \to \mathbb{R}^C$ is a linear transformation, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function whose action is defined component-wise.

In FNO, the kernel integral operator in $\mathcal{K}$ is defined as a convolution operator in Fourier space. Let $\mathcal{F}$ denote the Fourier transform of a function $f : D \to \mathbb{R}^C$ and $\mathcal{F}^{-1}$ its inverse, then

$$\hat{f}(k) = (\mathcal{F} f)_j(k) = \int_D f_j(x) e^{-2i\pi\langle x,k\rangle} \mathrm{d}x,$$
$$f(x) = (\mathcal{F}^{-1} f)_j(x) = \int_D \hat{f}_j(k) e^{2i\pi\langle x,k\rangle} \mathrm{d}k,$$

C.2 Tensor Parameterization and MLP Parameterization

The spectral convolution is defined as

$$\big(\mathcal{K}(\phi) f_t\big)(x) = \mathcal{F}^{-1}\Big(R_\phi \cdot (\mathcal{F} f_t)\Big)(x) \qquad \forall x \in D, \tag{11}$$

where R_ϕ is the learnable weight matrix or weight tensor.

Weight Tensor parameterization. Assuming the domain D is discretized with $n \in \mathbb{N}$ points, we have $f_t \in \mathbb{R}^{n \times C}$ and $\mathcal{F}(f_t) \in \mathbb{C}^{n \times C}$. Since we convolve f_t with a function that only has M_{max} Fourier modes, we may simply truncate the higher modes to obtain $\mathcal{F}(f_t) \in \mathbb{C}^{M_{\text{max}} \times C}$, where $M_{\text{max}} = M_1 \times \ldots \times M_d$. Multiplication by the weight tensor $R \in \mathbb{C}^{M_{\text{max}} \times C \times C}$ is defined as

$$\big(R \cdot (\mathcal{F} f_t)\big)_k = \sum_{j=1}^{C} R_{k,j} (\mathcal{F} f_t)_{k,j}. \tag{12}$$

Weight MLP parameterization. Multiplication by the weight matrix $R \in \mathbb{C}^{C \times C}$ is defined as

$$\big(R \cdot (\mathcal{F} f_t)\big)_k = \sum_{j=1}^{C} R_j (\mathcal{F} f_t)_{k,j}. \tag{13}$$

For the MLP parameterization, it is optional to add a bias term $b \in \mathbb{C}^C$.

Combining MLP and tensor parameterization. The multi-band structure is designed in a robust manner, allowing the specification of the channel dimension C_l and bandwidth M_l to any size. It is also flexible to combine the tensor parameterization (12) and matrix parameterization (13). In practice, we use the first level as MLP to have a full-frequency convolution, and the rest of the levels as tensor versions.

C.3 Frequency Encoding

The wavenumber $k \in \mathbb{Z}$ is encoded to a frequency feature $\mathbb{C}^{C_k}$ by a frequency encoding layer before being fed into the kernel network. For C_k channels, we define

$$k_j = k^{\frac{i}{(C_k - 1)}}, \qquad j = 0, 1, \ldots C_k - 1. \tag{14}$$

We note that k_j is unbounded and can become very large. As $k \to \infty$, $k_j \to \infty$. Since the input signal decays exponentially, $\hat{f}_t(k) = O(\exp(-\alpha k))$, a larger feature will help the model capture smaller signals.

Table 2: Darcy equation trained on $\sigma = 1$, zero-shot test on various scale.

Model **Resolution**	$\sigma = 2$ 64	$\sigma = 4/3$ 96	$\sigma = 1$ 128	$\sigma = 1/2$ 256	$\sigma = 1/4$ 512
FNO	0.03922	0.03999	0.03936	0.03451	0.03209
FNO+sup	0.02226	0.02365	0.02361	0.02396	0.02431
FNO+sub+sup	**0.01918**	**0.02075**	**0.02035**	**0.02159**	**0.02237**

Table 3: Helmholtz equation trained on $k = 5, 10, 25$

Model **Resolution**	$k = 1$ 64	$k = 2$ 64	$k = 5$ 64	$k = 10$ 128	$k = 25$ 256	$k = 50$ 512	$k = 100$ 1024
FNO	1.35418	1.31057	0.03755	**0.11684**	0.19516	**1.09210**	**1.09897**
Ours+sup	0.22004	0.17735	0.03455	0.11810	**0.18307**	1.13492	1.17544
Ours+sub+sup	**0.20662**	**0.16226**	**0.03448**	0.11796	0.18365	1.51171	1.67784

C.4 Activation Functions on Complex Space

R is a complex kernel neural network $R : \mathbb{C}^{C_{in}+C_k} \to \mathbb{C}^{C_{out}}$. We use a complex GeLU as the activation function, which applies GeLU to the real and imaginary parts separately, similar to the complex ReLU in [34]. This choice empirically provides the best performance.

$$\text{cGeLU}(\hat{f}) : \text{GeLU}(\text{real}(\hat{f})) + i\text{GeLU}(\text{imag}(\hat{f})). \tag{15}$$

D Experimental Details

D.1 Scale Consistency Loss

We test scale consistency on the Darcy Flow, Helmholtz equation, and Navier-Stokes equation.

Darcy Flow In the Darcy Flow problem, since the solution is smooth and low-frequency, we use FNO as the baseline. As the domain is not periodic, we use domain padding similar to [35] and normalize the model output by the magnitude of boundary inputs, as discussed in Section 4. We use 20 Fourier modes, a width (channel dimension) of 64, and 4 layers for the runs with or without self-consistency. The super-sampling has an annealed learning rate with respect to the epoch, where we multiply the rate learn by $\alpha = ep/ep_{\max}$, where $ep = 0, 1, \ldots, ep_{\max}$.

Helmholtz Equation For the Helmholtz equation, we compare FNO with the scale-informed FNO. Again, we normalize the model output by the magnitude of boundary inputs. Since the Helmholtz equation has higher frequency components, we use 64 Fourier modes, a width (channel dimension) of 32, and 4 layers. We use an annealed learning rate $\alpha = ep/ep_{\max}$ for super-sampling.

Navier-Stokes Equation For the Navier-Stokes equation, we compare UNet, FNO, and the scale-informed neural operator. For UNet, we set 5 levels with channels ranging from 64 to 1024. For FNO, we set 32 Fourier modes and a width (channel dimension) of 32. For the scale-informed neural operator, we also set 32 Fourier modes and a channel dimension of 32, where the first level is MLP-based and the second level is tensor-based.

D.2 Cost versus Accuracy Study

We assess the trade-off between computational cost and accuracy by comparing the performance of various models on the Navier-Stokes flow with $Re = 5000$ to our baseline models at various memory consumption levels. Our comparison metric is the relative L2 loss, recorded after 50 epochs. We use the maximum number of modes for each model and vary the channel dimensions.

The results, as detailed in Figure 5 and Table 4, demonstrate that the proposed model shows superior performance, particularly at larger widths. Notably, the model can match the performance of FNO with one-tenth the number of parameters and exceeds the performance of the U-shaped variants by more than 15%, especially at higher memory consumption levels.

Table 4: Performance of different model configurations on the RE=5000 Navier Stokes dataset

Model	Channel (GB)	Mem (M)	Params (s)	time	Train h1	Test l2
FNO	w24	1.5	12.8	101	36.1%	15.7%
FNO	w28	1.8	17.5	119	34.8%	15.0%
FNO	w32	2.0	22.8	130	33.3%	14.5%
FNO	w48	3.2	51.3	201	28.9%	12.5%
FNO	w60	4.1	80.2	258	26.8%	11.7%
FNO	w80	6.7	142.6	403	25.1%	11.1%
FNO	w100	8.7	222.8	546	23.7%	10.6%
FNO	w120	10.7	320.9	691	22.7%	10.3%
FNO	w146	13.9	473.5	966	21.4%	9.9%
Ours (tensor)	w10	1.5	21.3	260	34.1%	15.5%
Ours (tensor)	w14	2.2	41.7	281	31.1%	13.3%
Ours (tensor)	w16	2.6	54.5	314	28.7%	12.5%
Ours (tensor)	w20	3.3	85.2	320	26.3%	11.3%
Ours (tensor)	w26	4.9	144.0	404	24.3%	10.6%
Ours (tensor)	w32	7.1	218.1	503	22.7%	9.9%
Ours (tensor)	w36	8.7	276.1	624	22.0%	9.6%
Ours (tensor)	w38	9.6	307.6	666	21.9%	9.5%
Ours (tensor)	w40	11.0	340.8	700	21.1%	9.3%
Ours (tensor)	w50	13.9	532.5	954	20.6%	9.1%
Ours (mlp)	w30	3.3	0.2	385	38.1%	16.5%
Ours (mlp)	w50	4.2	0.6	559	30.3 %	13.4%
Ours (mlp)	w70	5.8	1.2	900	23.6%	10.3%
Ours (mlp)	w90	7.4	1.9	1203	22.1%	9.6%
FNO-UNet	w32	2.7	72.3	245	48.6%	26.7%
FNO-UNet	w40	3.9	113	434	45.6%	25.4%
FNO-UNet	w64	9.2	289	742	40.0%	23.1%
FNO-UNet	w72	13.0	366	1138	38.4%	22.7%
UNet	w50	4.2	18.9	885	30.9%	27.4 %
UNet	w64	6.7	31.0	994	29.2%	26.7 %
UNet	w80	11.0	48.4	1437	27.3%	26.1 %
UNO	w40	4.6	14.5	271	50.6%	23.4%
UNO	w56	6.2	28.3	350	50.1%	22.8%
UNO	w76	8.2	52.0	511	49.6%	22.4%
UNO	w100	9.1	90.2	677	49.5%	22.3%

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims reflects the results in experiment sections.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.

- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitation is discussed at the end of experiment section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The theorem on generalization to all scales, with the assmpution of perfect scaling symmetry and accuracy on the coarse scale, and its proof are in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.

- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All of the hyperparameters are disclosed at appendix. The code is included in the supplements.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is included in the supplemental material. Both code and datasets will be released.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The details are discuseed in experiment section and appendix,

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Since the results differ by large margins, the error bars are not very necessary.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Information is reported in the experiments section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We stick to the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The work considers applications in partial differential equations. It does not have an immediate societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The work considers applications in partial differential equations. Our datasets have no immediate safety concern.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We create our own datasets.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Both code and dataset will be released and well-documented.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not conduct crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: We do not conduct crowdsourcing or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.