Transform Once
Efficient Operator Learning in Frequency Domain

Anonymous Author(s)
Affiliation
Address
email

Abstract

Spectrum analysis provides one of the most effective paradigms for information–preserving dimensionality reduction in data, often, a simple description of naturally occurring *signals* can be obtained via few terms of periodic basis functions. Neural operators designed for frequency domain learning – *frequency domain models* (FDMs) – are based on complex–valued transforms i.e. *Fourier Transforms* (FT), and layers that perform computation on the spectrum and input data separately. This design introduces considerable computational overhead: for each layer, a forward and inverse FT. Instead, this work introduces a blueprint for frequency domain learning through a single transform: *transform once* (`T1`). To enable efficient, direct learning in the frequency domain we develop a variance preserving weight initialization scheme and address the open problem of choosing a transform. Our results noticeably streamline the design process of FDMs, pruning redundant transforms, and leading to speedups of 3 x to 10 x that increase with data resolution and model size. We perform extensive experiments on learning to solve partial differential equations, including incompressible Navier–Stokes, turbulent flows around airfoils and high-resolution video of smoke dynamics. `T1` models improve on the test performance of SOTA FDMs while requiring significantly less computation, with over 20% reduction in predictive error across tasks.

1 Introduction

Nature uses only the longest threads to weave her patterns, so that each small piece of her fabric reveals the organization of the entire tapestry. (Feynman, 2017)

Naturally occurring *signals* are often sparse when projected on periodic basis functions (Strang, 1999). Central to recently–introduced instances of frequency–domain neural operators (Li et al., 2020; Tran et al., 2021), which we refer to as *frequency–domain models* (FDMs), is the idea of learning to modify specific frequency components of inputs to obtain a desired output in data space. With a hierarchical structure that blends learned transformations on frequency domain coefficients with regular convolutions, FDMs are able to effectively approximate global, long–range dependencies in higher resolution signals without requiring prohibitively deep architectures. Yet, state–of–the–art variants of these models suffer from several drawbacks:

1. **Poor parameter scaling:** FDMs employ complex–valued transforms, followed by parameter–inefficient complex–valued layers[1].
2. **Slow inference:** every layer of an FDM performs a forward and inverse frequency domain transform, introducing a considerable computational overhead.
3. **Incompatibility:** parameter initialization schemes and layers devised to learn directly in data space can be highly suboptimal when introduced without modifications to FDMs.

[1] A single complex–valued parameter requires separate real and imaginary parts.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Although some attempts to improve performance (Gupta et al., 2021) or alleviate the above–mentioned limitations (Tran et al., 2021), scaling FDMs to larger data resolutions and model sizes remains fundamentally challenging. In example, instantiating a medium–sized FDM (tens of layers with a non–trivial number of channels) requires a parameter count in the hundred millions.

In this work, we start by posing the question:

To reap the benefits of learning on frequency domain representations, is it necessary to construct hierarchical deep models that perform forward and inverse frequency transforms at each layer?

We provide the answer in *Transform Once* (`T1`), a model that builds representations directly in frequency domain, after a *single* forward transform. Each aspect of `T1` addresses a specific limitation of existing FDMs:

1. **Favourable scaling:** `T1` employs a single real–valued transform (*Discrete Cosine Transform* – DCT), which reduces parameter scaling overhead w.r.t complex–valued layers.
2. **Fast:** by performing a single forward transform and optimizing directly on DCT coefficients of target data, `T1` iterations are *at least* 3x to 10x faster. When scaling to larger models and higher resolutions, the relative speedups increase as the overhead of each transform grows.
3. **Enhanced compatibility:** removing redundant transforms streamlines the design space for `T1` architectures compared to existing FDMs, allowing direct introduction of optimized layers developed for other applications e.g. UNets (Ronneberger et al., 2015).

In §2.1 we provide a (short) history on frequency domain approaches in deep learning, followed by background on FDMs in §2.3. In Section §3, we describe how to train `T1` directly in frequency domain, in §3.1 we motivate the choice of DCT, in §3.2 we derive a loss decomposition for reduced–order FDMs and in §3.3 we introduce a simple variance–preserving weight initialization scheme for all FDMs. Finally, in §4 we evaluate `T1` on a suite of benchmarks related to learning solution operators for a variety of dynamics: incompressible Navier–Stokes, flow around different airfoil geometries, and high–resolution videos of turbulent smoke plumes (Eckert et al., 2019).

Across tasks, `T1` is $3\times$ to $10\times$ faster and reduces predictive errors by 20% on average. Training `T1` models on high resolution videos (600 x 1062) of turbulent dynamics is significantly faster, requiring 5 hours instead of 32 hours (FNOs) for the same number of iterations.

2 Related Work and Background

2.1 Learning and Frequency Domain: A Short History

Links between frequency–domain signal processing and neural network architectures have been explored for decades, starting with the original CNN designs (Fukushima and Miyake, 1982). Mathieu et al. (2013); Rippel et al. (2015) proposed replacing convolutions in pixel space with element–wise multiplications in Fourier domain. In the context of learning to solve *partial differential equations* (PDEs), *Fourier Neural Operators* (FNOs) (Li et al., 2020) popularized the state–of–the–art FDM layer structure: forward transform $\rightarrow$ learned layer $\rightarrow$ inverse transform. Similar architectures had been previously proposed for generic image classification tasks in (Pratt et al., 2017; Chi et al., 2020). Modifications to the basic FNO recipe are provided in (Tran et al., 2021; Guibas et al., 2021; Wen et al., 2022). A frequency domain representation of convolutional weights has also been used for model compression (Chen et al., 2016). Fourier features of input *domains* and periodic activation functions play important roles in deep implicit representations (Sitzmann et al., 2020; Dupont et al., 2021; Poli et al., 2022) and general–purpose PERCEIVERS (Jaegle et al., 2021).

2.2 Learning to Solve Differential Equations

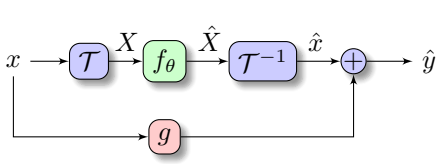
A variety of deep learning approaches have been developed to solve differential equations: neural operators and physics–informed networks (Long et al., 2018; Raissi et al., 2019; Lu et al., 2019; Karniadakis et al., 2021), specialized architectures (Wang et al., 2020; Lienen and Günnemann, 2022), hybrid neural–numerical methods (Poli et al., 2020; Kochkov et al., 2021; Mathiesen et al., 2022), and FDMs (Li et al., 2020; Tran et al., 2021), the focus of this work.

2.3 Frequency–Domain Models

Let $\mathcal{D}_n$ (n-space) to be the set of real–valued discrete signals[2] of resolution N. Our objective is to develop efficient neural networks to process discrete signals $x \in \mathcal{D}_n$,

$$x_0, x_1, \ldots, x_{N-1}, \quad x_n \in \mathbb{R}.$$

We define a layer of FDMs mapping x to an output signal $\hat{y} \in \mathcal{D}_n$ as the structured operator:



$$\begin{aligned} X &= \mathcal{T}(x) && \text{Forward Transform} \\ \hat{X} &= f_\theta(X) && \text{Learned Map} \\ \hat{x} &= \mathcal{T}^{-1}(\hat{X}) && \text{Inverse Transform} \\ \hat{y} &= \hat{x} + g(x) && \text{Residual} \end{aligned} \tag{1}$$

where $\mathcal{T}$ is an orthogonal (possibly *complex*) linear operator. We denote the $\mathcal{T}$–transformed n-space with $\mathcal{D}_k$ (k-space) so that $\mathcal{T} : \mathcal{D}_n \to \mathcal{D}_k$. Typically, we assume $\mathcal{T}$ to be a *Fourier–type* transform[3] (Oppenheim, 1999, Chapter 8) so that the k-space corresponds to the *frequency domain* and its elements form the *spectrum* of the input signal x.

The learned parametric map $f_\theta : \mathcal{D}_k \to \mathcal{D}_k$ is the stem of a FDM layer: it maps the k-space into itself and is typically chosen to be rank–deficient in the linear case, e.g. $f_\theta(X) = S_m^\top A(\theta) S_m X,\ A(\theta) \in \mathbb{C}^{m \times m}$ ($m \leq N$). The matrix $S_m \in \mathbb{R}^{n \times m}$ selects m desired elements of X, setting the rest to zero. In the case of frequency domain transforms, this allows (1) to preserve or modify only specific frequencies of the input signal x.

Residual connections or residual convolutions g (Li et al., 2020; Wen et al., 2022) are optionally added to reintroduce frequency components filtered by the S_m. An FDM mixes global transformations applied to coefficients of the chosen transform to local transformations g i.e. convolutions with finite kernel sizes. To ensure that such models can approximate generic nonlinear functions, nonlinear activations are introduced after each inverse transform.

Fourier Neural Operators Layers of the form (1) appear in recent FDMs such as *Fourier Neural Operators* (FNOs) (Li et al., 2020) and variants (Tran et al., 2021; Guibas et al., 2021; Wen et al., 2022).

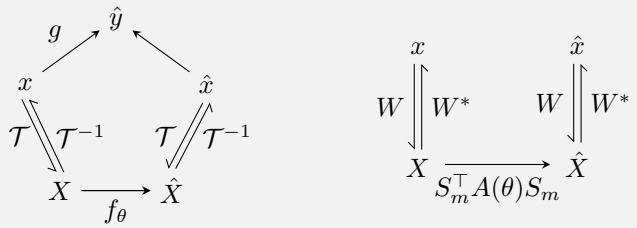
In example, an FNO is recovered from (1) by letting $\mathcal{T}$ be a *Discrete Fourier Transform* (DFT)

$$\hat{x} = \mathcal{T}^{-1} \circ f_\theta \circ \mathcal{T}(x) = W^* S_m^\top A(\theta) S_m W x$$

where $W \in \mathbb{C}^{N \times N}$ is the standard N-dimensional DFT matrix and W^* its conjugate transpose. The *Discrete Fourier Transforms* (DFTs) is a natural choice of $\mathcal{T}$ as it can be computed in $O(N \log N)$ via *Fast Fourier Transform* (FFT) algorithms (Oppenheim, 1999, Chapter 9.2).

We identify two major limitations of FDMs in the form (1); each layer performs $\mathcal{T}$ and $\mathcal{T}^{-1}$, and DFTs are complex–valued, requiring parameter–inefficient complex–valued layers.

With `T1`, we aim to develop an FDM that does not require more than a single $\mathcal{T}$, while preserving or improving on predictive accuracy. Ideally, the transform in `T1` should be (1) real–valued, to avoid introducing parameter scaling overhead, (2) universal, to allow the representation of target signals, and (3) approximately sparse or structured, to allow dimensionality reduction.



Commutative diagrams for FDM layers (1) and linear FNOs (frequency domain part).

[2]For clarity of exposition, models and algorithms proposed in the paper are introduced without loss of generality for one–dimensional scalar signals (i.e. $\mathcal{D}_n \equiv \mathbb{R}^n$).

[3]e.g. *discrete Fourier transform* (DFT), *discrete cosine transform* (DCT), etc.

3 Transform Once: The T1 Recipe

With T1, we introduce major modifications to the way FDMs are designed and optimized. In particular, T1 is defined, inferred and trained directly in the frequency domain with only a **single** direct transform required to process data. Hence follows the name: *transform once* (T1).

Direct learning in the frequency domain Consider two signals $x \in \mathcal{D}_n$, $y \in \mathcal{D}_n$ and suppose there exists a function $\varphi : \mathcal{D}_n \to \mathcal{D}_n$ mapping x to y, i.e.

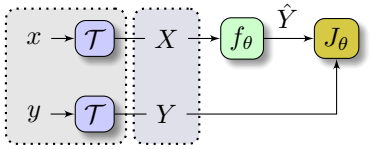
$$y = \varphi(x).$$

Then, there must also exist another function $\psi : \mathcal{D}_k \to \mathcal{D}_k$ that relates the spectra of the two signals, i.e. $Y = \psi(X)$ being $X = \mathcal{T}(x)$ and $Y = \mathcal{T}(y)$. In particular,

$$\varphi(x) = \mathcal{T}^{-1} \circ \psi \circ \mathcal{T}(x) \quad \Leftrightarrow \quad \mathcal{T} \circ \varphi(x) = \psi \circ \mathcal{T}(x)$$

It follows that, from a learning perspective, we can aim to approximate ψ directly in the k-space rather than φ in the n-space. To do so, we define a learnable parametric function $f_\theta : \mathcal{D}_k \to \mathcal{D}_k$ and train it to minimize the approximation error J_θ of the output signal spectrum Y in the k–space. Given a distribution $p(x)$ of input signals, T1 is characterized by the following nonlinear program

$$\begin{aligned} \min_{\theta} \quad & \mathbb{E}_x \left[\| \mathcal{T}(y) - \hat{Y} \| \right] \\ \text{subject to} \quad & \hat{Y} = f_\theta \circ \mathcal{T}(x) \\ & x \sim p(x) \\ & y = \varphi(x) \end{aligned} \tag{2}$$



If $\mathcal{T}$ is a DFT, the above turns out to be a close approximation (or equivalent, depending on the function class of f_θ) to the minimization of $\|y - \hat{y}\|$ in n-space by the *Parseval-Plancherel identity*.

Theorem 3.1 (Parseval-Plancherel Identity (Stein and Shakarchi, 2011, pp. 223)). *Let $\mathcal{T}$ be the normalized DFT. Given a signal $v \in \mathcal{D}_n$ and its transform $V = \mathcal{T}(v)$, it holds $\|v\| = \|V\|$.*

This result also applies to any other norm–preserving transform $\mathcal{T}$, e.g. a normalized type–II DCT (Oppenheim, 1999, pp. 679). For the linear transforms considered in this work, $\mathcal{T}(x) = Wx$, $W \in \mathbb{C}^{N \times N}$, condition for Th. 3.1 to hold is W to be orthonormal, i.e. $W^* W = \mathbb{I}$.

Note that T1 retains, in principle, the same universal approximation properties of FNOs (Kovachki et al., 2021) as f_θ is allowed to operate on the entirety of the input spectrum. Given enough capacity, f_θ can arbitrarily approximate ψ, implicitly reconstructing φ via $\mathcal{T}^{-1} \circ f_\theta \circ \mathcal{T}$.

Speedup measurements We provide a concrete example of the effect of pruning redundant transforms on computational costs. We measure wall–clock inference time speedups of depth d T1

$$\texttt{T1}(x) := f_d \circ \cdots \circ f_2 \circ f_1 \circ \mathcal{T}(x)$$

over an equivalent depth d FNO with layers (1). The only difference concerns the application of transforms between layers.

Figure 3.1 provides the speedups on two–dimensional signals: on the left, we fix model depth $d = 6$ and investigate the scaling in signal width (i.e. number of channels) and signal resolution. On the right, we fix signal width to be 32 and visualize the interaction of model depth and signal resolution. For common experimental settings e.g. resolutions of 64 or 128, 6 layers and width 32, T1 is at least 10 x faster than other FDMs. It will later be shown (§4) that T1 also preserves or improves on predictive accuracy of other FDMs across tasks.

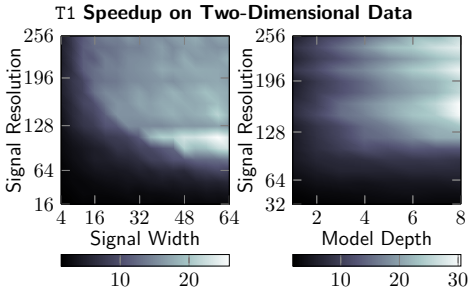


Figure 3.1: Speedup in a forward pass of T1 over FNOs sharing the same transform $\mathcal{T}$ (DFT) on two–dimensional signals of increasing resolution. The speedup for a given configuration (point on the plane) is shown as background color gradient. The improvement grows with signal width, resolution and model depth.

When T1 is not preceded by online preprocessing steps for inputs x, such as other neural networks or randomized data augmentations, the transform on $\mathcal{T}(x)$ can be done once on the dataset, amortizing the cost over training epochs, and increasing the speed of T1 further.

3.1 Choosing the right transform

The transform $\mathcal{T}$ in `T1` is chosen to be in the class of *Discrete Cosine Transforms* (DCTs) (Ahmed et al., 1974; Strang, 1999), in particular the normalized DCT–II,

$$X_k = 2 \sum_{n=0}^{N-1} \beta_k x_n \cos \frac{(2n+1)k\pi}{2N}, \quad \beta_k = \begin{cases} \frac{1}{2\sqrt{N}}, & k = 0 \\ \frac{1}{\sqrt{2N}}, & k = 1, \dots, N-1 \end{cases}.$$

DCT–II transforms can be computed in $O(N \log N)$ via an extended FFT (Makhoul, 1980), are real–valued, and can be shown to provide effective representations of smooth continuous signals leveraging classical results on function approximation with Chebyshev polynomials (Trefethen, 2019). Moreover, DCT–II transforms are the backbone of modern compression codecs for digital images i.e. JPEG, providing further evidence of approximate sparsity induced in $\mathcal{D}_k$ by this transform.

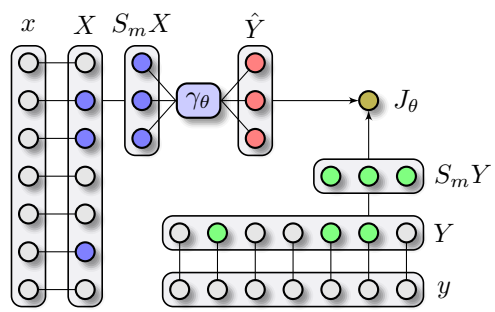
A real–valued transform allows `T1` to at least halve the required number of parameters by f_θ, avoiding the overhead of complex–valued parameters. Although our initial design of `T1` included training deep f_θ in FFT k-spaces, we observe poor performance and optimization instabilities, also noted by Li et al. (2020). Along with parameter inefficiency, these issues are resolved by letting $\mathcal{T}$ be a DCT–II.

3.2 Reduced–Order `T1` Model and Irreducible Loss Bound

We seek to leverage structure induced in $\mathcal{D}_k$ by $\mathcal{T}$. To this end we allow `T1`, similarly to (1), to modify specific elements of X and consequently affect only certain frequency components of x.

The reduced–order `T1` model is designed to operate only on $m < N$ elements (selected by $S_m \in \mathbb{R}^{N \times m}$) of the input k-space, i.e. on a *reduced* k-space $\mathcal{D}_m \equiv \mathbb{R}^m$ of lower dimension. Thus, we can employ a smaller neural network $\gamma_\theta : \mathcal{D}_m \to \mathcal{D}_m$ for mapping $S_m X$ to the corresponding m elements $S_m Y$ of the output k-space. Thus, training involves a truncated objective that compares predictions with elements in the output signal spectrum also selected by S_m:

$$\begin{aligned} \min_\theta \quad & \mathbb{E}_x \left[\| S_m \circ \mathcal{T}(y) - \hat{Y} \| \right] \\ \text{subject to} \quad & \hat{Y} = \gamma_\theta \circ S_m \circ \mathcal{T}(x) \\ & x \sim p(x) \\ & y = \varphi(x) \end{aligned} \tag{3}$$



Irreducible losses Without any loss of generality, let us assume to select with S_m the first m elements of the k-space, i.e.

$$S_m = m \left\{ \left[\overbrace{\begin{matrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{matrix}}^{m} \quad \overbrace{\begin{matrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{matrix}}^{T-m} \right] \right. .$$

The truncated training objective $J_\theta(X, Y)$ reads as

$$J_\theta(X, Y) = \| S_m Y - \hat{Y} \| = \sum_{k=0}^{m-1} \left| Y_k - \gamma_{\theta,k} \circ S_m(X) \right|,$$

However, J_θ does not represent a good metric to measure the real performance of the reduced–order `T1` model. For that we need to evaluate the complete loss L_θ of the approximation task, including

the $N - m$ elements of the output k-space discarded by our model, i.e.

$$L_\theta(X, Y) = \|Y - S_m^\top \hat{Y}\| = \underbrace{\sum_{k=0}^{m-1} |Y_k - \gamma_{\theta,k} \circ S_m(X)|}_{J_\theta(X,Y)} + \underbrace{\sum_{k=m}^{N-1} |Y_k - 0|}_{R_o(Y)}.$$

It follows that the overall loss L_θ is higher than `T1`'s training objective J_θ, i.e. $L_\theta = J_\theta + R_o > J_\theta$, whilst R_o represents the *irreducible* residual loss due to truncation of the predictions $\hat{Y}_k$.

The choice of elements and thus magnitude of irreducible loss is a strong inductive bias for reduced–order `T1`. Nonetheless, given a dataset of input–output signals it is possible to perform an *a priori* analysis on R_o to inform the choice of m and S_m. Often, we empirically observe the irreducible error R_o for reduced–order `T1` to be smaller than for non–reduced–order FDMs i.e $R_o < \sum_{k=m}^{K-1} \|Y_k - \mathcal{T}_k(\hat{y})\|$ with layers of type (1)[4].

We note further that the reachable component J_θ of the objective cannot always be minimized to zero regardless of the approximation power of γ_θ. For each $k < m$, S_m discards $N - m$ frequency components of the input signal which, if different than zero, likely contain the necessary information to approximate $\psi_k(X)$ exactly. Specifically, the irreducible lower bound on J_θ should depend on "how much" the output's m frequency components depend on the discarded $N - m$ input's elements. A rough quantification of such bound can be obtained by inspecting the mismatch between the gradients of $\psi_k - \gamma_{\theta,k} \circ S_m$ with respect to X. In particular, it holds

$$\sum_{j=0}^{N-1} \left| \frac{\partial \psi_k(X)}{\partial X_j} - \frac{\partial \gamma_{\theta,k}(S_m X)}{\partial X_j} \right| = \sum_{j=0}^{m-1} \left| \frac{\partial \psi_k(X)}{\partial X_j} - \frac{\partial \gamma_{\theta,k}(S_m X)}{\partial X_j} \right| + \sum_{j=m}^{N-1} \left| \frac{\partial \psi_k(X)}{\partial X_j} \right|,$$

Unless $\partial_{X_j} \psi_k(X) = 0$ holds for all $j = m, \ldots, N-1$ and $k = 0, 1, \ldots, N-1$ i.e. no dependency of the ground truth map in k–space on the truncated elements, there will be an irreducible overall gradient mismatch and thus a nonzero J_θ.

3.3 Weight Initialization for Reduced–Order FDMs

FDMs (Li et al., 2020; Tran et al., 2021; Wen et al., 2022) opt for a standard Xavier–like (Glorot and Bengio, 2010) initialization distribution that takes into account the input channels c to a layer i.e. $\mathcal{N}(0, \frac{1}{c})$. However, well–known variance preserving properties of Xavier schemes do not hold for FDM layers truncating $N - m$ elements of the k–space. Notably, Xavier schemes do not scale the variance of the weight initialization distribution based on the number of elements m kept after truncation of the spectrum performed by f_θ, leading to the *collapse* of the outputs to zero.

To avoid this issue in `T1` and other FDMs, we develop a simple *variance–preserving* (vp) that introduces a variance scaling factor based on m and the class of transform.

Theorem 3.2 (Variance Preserving (`vp`) initialization)**.** *Let $f_\theta(X) = S_m^\top A(\theta) S_m X$ be a k–space reduced–order layer and $\mathcal{T}$ a normalized DCT–II transform. Consider an input random variable $x \in \mathbb{R}^N$ distributed according to*

$$\mathbb{E}[x] = \mathbb{0}, \quad \mathbb{V}[x] = \sigma^2 \mathbb{I}.$$

Then,

$$A_{ij} \sim \mathcal{N}\left(0, \frac{T}{mc}\sigma^2\right) \implies \mathbb{V}[f_\theta(x)] = \mathbb{V}[x]$$

We report the proof in Appendix A, including some considerations for specific forms of f_θ.

Corollary 3.1 (`vp` initialization for DFTs)**.** *Under the assumption of Theorem 3.2, let $\mathcal{T}$ be a normalized Discrete Fourier Transform (DFT). Then, $A_{ij} \sim \mathcal{N}(0, \frac{T}{2mc}\sigma^2) \implies \mathbb{V}[f_\theta(x)] = \mathbb{V}[x]$.*

The collapse phenomenon is empirically shown in Figure 3.2 for $m = 24$, comparing a single layer of FNO and FFNO (with Xavier initialization) with FNO equipped with the proposed `vp` scheme. Under the assumptions of Corollary 3.1, we sample A and compute empirical variances of $\hat{x} = W^* S_m^\top A(\theta) S_m W x$ for several finite batches of input signals x. We repeat the experiment for signals of different lengths N. The `vp` scheme preserves unitary variances whereas the other layers concentrate output variances towards zero at a rate that grows with $N - m$.

[4] See Fig. 4.1 and Appendix B for experimental evidence in support of this phenomenon.

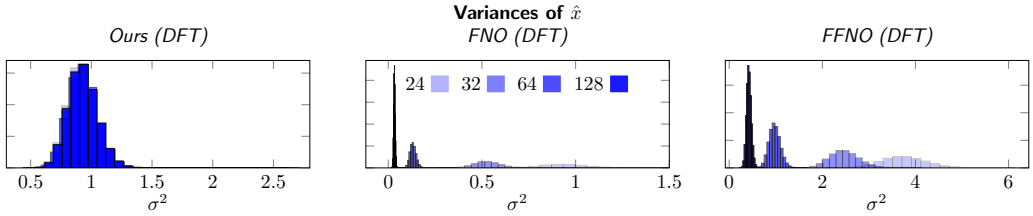


Figure 3.2: Output variance histogram in layer outputs $\hat{x} = W_m^* S_m^\top A(\theta) S_m W_N$, for a finite sample of inputs x and a single sample of θ. Color indicates signal resolution.

When the learned frequency–domain transformation f_θ is obtained, instead of the single low–rank linear layer $f_\theta = A(\theta) S_m X$, as the composition of several layers, preserving variances can be achieved by applying the `vp` scheme only to the first layer. For some variants of FDMs e.g. FNO that truncate the spectrum at each layer, `vp` initialization should instead be applied to all.

4 Experiments

We validate `T1` on learning to approximate solution operators of dynamical systems from images.

- In Section 4.1, we apply `T1` on the standard task of learning solution operators for incompressible Navier–Stokes, comparing against other FDMs. In 4.1.1 we perform a series of ablation experiments on each ingredient of the `T1` recipe, including weight initialization and architecture. In 4.1.2 we provide scaling laws.
- Section 4.2 we deal with fluid–solid interaction dynamics in the form of higher resolution images (128). We consider turbulent flows around varying airfoil geometries, benchmarking against current SOTA (Thuerey et al., 2020).
- In Section 4.3 we show how the computational efficiency of `T1` allows learning on unwieldy data without downsampling or building low–resolution meshes. We consider learning on high–resolution video (600×1062) capturing the turbulent dynamics of smoke (Eckert et al., 2019).

Configuration and model card details are reported in the supplementary material. Detailed *Weights & Biases* (`wandb`) (Biewald, 2020) training logs will be reported after the review phase.

4.1 Incompressible Navier–Stokes

We show that `T1` matches or outperforms SOTA FDMs with less computation on the standard incompressible Navier–Stokes benchmark. Losses are reported in n–space (signal space) for comparison.

Setup We consider two–dimensional Navier–Stokes equations for incompressible fluid in vorticity form as described in (Li et al., 2020). Given a dataset of initial conditions, we train all models to approximate the solution operator at time 50 seconds for high viscosity ($\nu = 1e^{-3}$) and at time 15 for lower viscosity ($\nu = 1e^{-4}$). As a metric, we report *normalized mean squared error* (N–MSE). Both initial condition as well as solution are provided as images of resolution 64.

We include as baseline established FDMs, such as Fourier Neural Operators (FNOs) (Li et al., 2020) and *Factorized Fourier Neural Operators* (FFNOs) (Tran et al., 2021). We indicate with the suffix *vp* models that employ the proposed variance preserving initialization scheme. All models truncate to $m = 24$, except FFNOs with $m = 32$.

Method	Param. (M)	Size (MB)	Step (ms)	high ν	low ν
FFNO (Tran et al., 2021)	8.9	35	294	1.002±0.011	1.016±0.010
FNO (Li et al., 2020)	14.2	56	31	0.379±0.006	0.327±0.004
FNOvp	14.2	56	32	0.351±0.007	0.314±0.005
`T1`+vp	10.2	40	19	**0.260**±0.011	**0.238**±0.003

Table 4.1: Benchmarks on incompressible Navier–Stokes. Direct long–range prediction errors (N–MSE) in n–space (signal space) of different models.

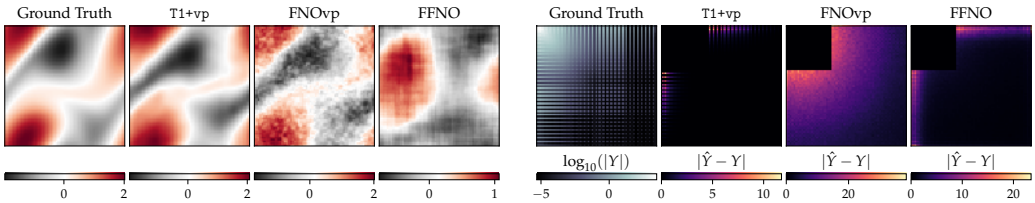


Figure 4.1: **[Left]** Direct predictions at $T = 50$s on high viscosity Navier–Stokes. **[Right]** Ground–truth spectrum and absolute errors in k–space (DCT–II). Despite predicting only the first $m = 24$ elements, reduced–order `T1` models produce smaller errors even in other regions of the k–space.

Results We perform 20 training runs for each model and report mean and standard deviation in Table 4.1. `T1` reduces solution error w.r.t FNOs by over 20% and FFNOs by over 40%. A single forward pass of `T1` models is on average 2x faster than FNO and 10x than FFNOs. We note that FFNOs are designed to share parameters between layers, and thus require deeper architectures – and slower, due to more transforms. In particular, training time (500 epochs) for `T1` is cut to 20 minutes down from 40 of FNOs, matching the model speedup. Finally, we report an improvement in performance for FNOs with parameters initialized following our proposed scheme (FNOvp). Figure 4.1 provides sample predictions in n–space (left) to contextualize the task, in addition to prediction errors in frequency domain (right). Despite being a reduced order model with $m = 24$, `T1`+vp produces smaller errors on truncated k–space elements ($k > m$) compared to FNOvp and FFNO.

4.1.1 Ablations on weight scheme and architecture

We repeat the previous experiment and report prediction errors for four variants of `T1`: same architecture and weight initialization scheme as FNOs (`T1`), `T1` with our proposed vp scheme (`T1`vp), a reduced–order variant with k–space model f_θ defined as a UNet architecture (`T1`+), and `T1`+ with variance preserving scheme (`T1`+vp). The results in Table 4.2 provide empirical evidence in support of the vp scheme and its synergistic effect with the proposed architecture. In particular, combining vp scheme and UNet structure in frequency domain reduces error by half compared to the naive `T1` approach.

Method	high ν	low ν
`T1`	0.491	0.449
`T1`vp	0.304	0.280
`T1`+	0.295	0.260
`T1`+vp	**0.260**	**0.238**

Table 4.2: Ablation on the effect of the proposed weight initialization scheme and `T1` architecture.

4.1.2 Scaling laws

We verify whether the reduction in predictive error of `T1` over neural operator baselines is preserved as the size of training dataset grows. We perform 10 training runs on the Navier–Stokes $\nu = 1e^{-4}$ experiment, each time with a larger dataset size, and report the scaling laws in Figure 4.2. With additional data, the gaps in test errors narrow slightly, with noticeable improvements obtained by applying the vp scheme to both FNO and `T1`+.

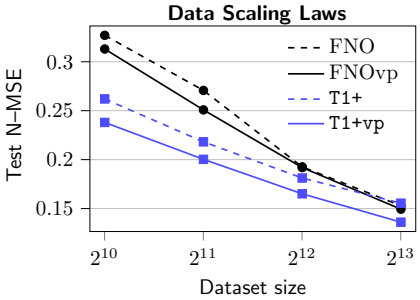


Figure 4.2: Scaling laws for N–MSE.

4.2 Flow Around Airfoils

We investigate the performance of `T1` in predicting steady–state solutions of flow around airfoils.

Setup We use data introduced in (Thuerey et al., 2020) in the form of 10000 training pairs of initial conditions, specifying freestream velocities and the airfoil mask, with the target steady–state velocity and pressure fields. This task introduces additional complexity in the form of higher resolution input images (128) and a full k–space due to the discontinuity in the field produced by the mask.

We compare a SOTA UNet architecture (DFPNet) introduced by (Thuerey et al., 2020) to FNOs and `T1` with vp initialization schemes. We perform a search on the most representative hyperparameters (detailed in the Appendix). Averages for 5 runs are reported in Table 4.3.

Method	N–MSE	Time (hrs)
DFPNet	0.026	2.1
FNO	0.019	6.8
T1+	0.020	2.2

Table 4.3: Test N–MSE and total training time on the flow around airfoil task.

Results All models are able to accurately predict steady–state solutions for different airfoils. FDMs and T1 offer a slight improvement over DFPNets (Thuerey et al., 2020), with FNOs introducing a significant computational overhead. Training of T1 is as fast as DFPNets and as accurate as FNOs. We note standard deviation of the results $\sigma = 0.001$; differences between FNO are explainable by natural variation T1 ($\leq 1\sigma$) whereas improvements over DFPNet are $\geq 6\sigma$.

4.3 Turbulent Smoke

We investigate the performance of T1 in predicting iterative rollouts from high–resolution video of real rising smoke plumes.

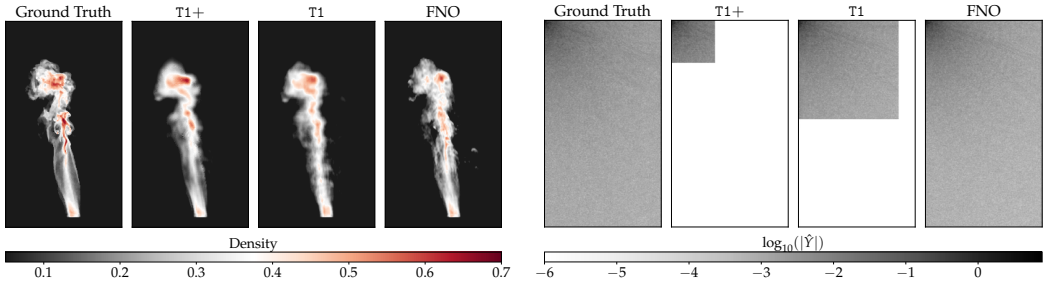


Figure 4.3: **[Left]** 10-step rollout predictions on ScalarFlow. FNOs produce non–physical artifacts and accumulate error more rapidly in time compared to T1 models **[Right]** Log–absolute values of predictions in k–space (DCT–II). Although T1 is limited to $m = 512$ and T1+ to $m = 224$ k–space elements, the predictions are overall more physically accurate in n–space.

Setup We use the ScalarFlow dataset introduced in (Eckert et al., 2019) consisting of 104 sequences of 150 frames each collected from video recordings of rising hot smoke plumes. The dataset consists of raw video data at high–resolution (600 × 1062) collected at 60 fps. This task scales up complexity by involving real–world high–definition data, capturing highly–turbulent dynamics. We perform rollouts iteratively based on previous predictions: all models models are trained on 3–step rollouts and evaluated over 10–steps extrapolation to test their generalization in time. We compare FNOs against T1 and T1+ of similar model sizes, after performing a search on most representative hyperparameters (Appendix B).

Results Figure 4.3 provides a sample rollout of different model predictions in k–space (DCT–II). T1 accumulates smaller errors over the rollout and is less prone to generation of non–physical artifacts by performing prediction only on a subset of the k–space (Table 4.4). Notably, T1 and T1+ are $4\times$ to $7\times$ faster, providing a reduction in training time from 32.4 hours to 4.7. In Appendix B, we include additional visualization, including prediction errors on k–space.

Method	N–MSE	Time (hrs)
FNO	0.220	32.4
T1	0.214	8.1
T1+	0.203	4.7

Table 4.4: Test 10–steps rollout n–space prediction errors (N–MSE) and total training time on the ScalarFlow dataset.

5 Conclusion

We present a streamlined class of *frequency domain models* (FDM): *Transform Once* (T1). T1 models are optimized directly in frequency domain, after a single transform, and achieve similar or improved predictive performance at a fraction of the computational cost (3x to 10x speedups across tasks). Further, a simple truncation–aware weight initialization scheme is introduced and shown to improve performance of T1 and existing FDMs.

References

N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

L. Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

W. Chen, J. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing convolutional neural networks in the frequency domain. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1475–1484, 2016.

L. Chi, B. Jiang, and Y. Mu. Fast fourier convolution. *Advances in Neural Information Processing Systems*, 33:4479–4488, 2020.

E. Dupont, A. Goliński, M. Alizadeh, Y. W. Teh, and A. Doucet. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021.

M.-L. Eckert, K. Um, and N. Thuerey. Scalarflow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019.

R. Feynman. *The Character of Physical Law, with new foreword*. MIT press, 2017.

K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

J. Guibas, M. Mardani, Z. Li, A. Tao, A. Anandkumar, and B. Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers. *arXiv preprint arXiv:2111.13587*, 2021.

G. Gupta, X. Xiao, and P. Bogdan. Multiwavelet-based operator learning for differential equations. *Advances in Neural Information Processing Systems*, 34, 2021.

A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118 (21), 2021.

N. Kovachki, S. Lanthaler, and S. Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22:Art–No, 2021.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

M. Lienen and S. Günnemann. Learning the dynamics of physical systems from sparse observations with finite element networks. *arXiv preprint arXiv:2203.08852*, 2022.

Z. Long, Y. Lu, X. Ma, and B. Dong. PDE-net: Learning PDEs from data. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3208–3216. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/long18a.html.

L. Lu, P. Jin, and G. E. Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

J. Makhoul. A fast cosine transform in one and two dimensions. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(1):27–34, 1980.

F. Mathiesen, B. Yang, and J. Hu. Hyperverlet: A symplectic hypersolver for hamiltonian systems. In *AAAI*, 2022.

M. Mathieu, M. Henaff, and Y. LeCun. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2013.

A. V. Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

M. Poli, S. Massaroli, A. Yamashita, H. Asama, and J. Park. Hypersolvers: Toward fast continuous-depth models. *Advances in Neural Information Processing Systems*, 33:21105–21117, 2020.

M. Poli, W. Xu, S. Massaroli, C. Meng, K. Kim, and S. Ermon. Self-similarity priors: Neural collages as differentiable fractal representations. *arXiv preprint arXiv:2204.07673*, 2022.

H. Pratt, B. Williams, F. Coenen, and Y. Zheng. Fcnn: Fourier convolutional neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 786–798. Springer, 2017.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

O. Rippel, J. Snoek, and R. P. Adams. Spectral representations for convolutional neural networks. *Advances in neural information processing systems*, 28, 2015.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

E. M. Stein and R. Shakarchi. *Fourier analysis: an introduction*, volume 1. Princeton University Press, 2011.

G. Strang. The discrete cosine transform. *SIAM review*, 41(1):135–147, 1999.

N. Thuerey, K. Weißenow, L. Prantl, and X. Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020.

A. Tran, A. Mathews, L. Xie, and C. S. Ong. Factorized fourier neural operators. *arXiv preprint arXiv:2111.13802*, 2021.

L. N. Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.

R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1457–1466, 2020.

G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson. U-fno–an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, page 104180, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] **See Appendix B**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]