Learning to Compress Plasma Turbulence

Gianluca Galletti [1] **Gerald Gutenbrunner** [1] **Fabian Paischer** [1,3]
Sandeep S. Cranganore [1] **William Hornsby** [2] **Lorenzo Zanisi** [2]
Naomi Carey [2] **Stanislas Pamela** [2] **Johannes Brandstetter** [1,3]

[1] ELLIS Unit, Institute for Machine Learning, JKU Linz
[2] United Kingdom Atomic Energy Authority, Culham campus
[3] Emmi AI, Linz
galletti@ml.jku.at
github.com/ml-jku/neural-gyrokinetics

Abstract

Turbulence is a nonlinear phenomenon exhibiting chaotic, multiscale behavior. It is simulated with high-fidelity numerical solvers operating on fine grids, making the process both computationally demanding and storage intensive. A prime example is gyrokinetics, which simulates turbulence in a magnetized plasma. A single run can take weeks and produce up to tens of terabytes of data, making storage unfeasible even with standard compression algorithms. Raw data is typically discarded, which is wasteful and prevents routine post-hoc analysis of the simulations. To this end, we investigate neural compression methods capable of extreme compression ratios (up to $40{,}000\times$) while preserving reconstruction quality and physical fidelity. Our study focuses on *autoencoders* and *neural implicit fields*, specifically trained with novel physics-informed losses. This direction could enable practitioners to store high-fidelity turbulence simulations for downstream scientific analysis at a fraction of the volume.

1 Introduction

Turbulence is a nonlinear phenomenon characterized by chaotic, multiscale dynamics in space and time (38, 26, 44, 6, 22). Turbulent flows are defined by four properties: (**i**) random emergence of patterns (irregularity and *eddies*), (**ii**) dissipation into heat (*diffusivity*), (**iii**) dynamics spanning many scales (*scale separation* between viscous and inertial), and (**iv**) transport and mixing (*vorticity* fluctuations). For the Navier-Stokes equations, this leads to a *direct energy cascade*: energy injected at large scales transfers nonlinearly to progressively smaller scales until dissipated by viscosity.

An example of a system governed by turbulence is gyrokinetics (16, 27, 37). It describes turbulence in magnetised Plasmas, found for example in solar wind or in magnetically-confined nuclear fusion devices. Gyrokinetics models the time evolution of particles in a Plasma via a 5D distribution function $\boldsymbol{f} \in \mathbb{R}^{v_\parallel \times \mu \times s \times x \times y}$, where x, y and s are spatial coordinates and $v_\parallel$, μ are velocity space coordinates. More details and derivation in Appendix A. Importantly, the energy cascade in gyrokinetics is *bi-directional*, meaning that energy is redistributed from larger to smaller scales and vice-versa. Capturing turbulence requires simultaneously resolving large coherent structures and fine, rapidly evolving ones. Therefore, high-fidelity simulations are needed, but they become prohibitively expensive to store as a single snapshot can exceed hundreds of gigabytes of disk space. Researchers thus usually only keep derived quantities, making comprehensive temporal post-hoc analysis of turbulence harder or infeasible.

Accepted at the AI for Science workshop (NeurIPS 2025)

In this work, we explore learned neural compression techniques for turbulent gyrokinetics data, and compare them to traditional lossy methods. To this end, we evaluate two paradigms of neural compression: *monolithic* models (e.g. VQ-VAE (50)), where a single model is used to generalize across datasets, and *micromodels* (e.g. neural implicit representations (35, 36, 13)), which reduce individual snapshots by encoding them directly into network parameters. By training them on physics-informed objectives, we steer these methods toward capturing the underlying physical characteristics. Moreover, we evaluate whether the compressed snapshots accurately reproduce ground-truth temporal turbulent patterns and preserve physical quantities of interest, e.g. flux-trace integrals. To this end, it is imperative to disentangle *transient fluctuations* and *steady-state* quantities in evaluation. The former captures energy transfer across modes to evaluate whether the turbulent energy cascade is reproduced. The latter describes the properties of the system once statistical equilibrium is reached.

Our physical and turbulence-focused evaluation provides novel insights that are valuable for practitioners. We find that both neural approaches attain reconstruction errors comparable or better to traditional lossy compression techniques at similar compression ratios, while significantly outperforming them on physics preservation. Interestingly, at these high compression regimes, integral quantities and temporal turbulence metrics get significantly deteriorated for snapshots compressed with traditional approaches, while learned models with physics-inspired constraints maintain higher physical fidelity. These findings highlight the potential of neural methods for extreme compression in scientific computing.

2 Related Work

Compression of spatiotemporal data is not a novel topic, and fields such as numerics and High-Performance Computing (HPC) have produced a great deal of research in this direction (18, 11, 28, 30, 51, 1, 3). Advanced research (2) exists in the domain of computational plasma physics, and in particular from the neighboring field of Particle-In-Cell (PIC) simulations (5, 49). The most relevant approaches include ISABELA (28), which is an advanced spline method that promises up to $7\times$ compression with almost no loss of information; and VAPOR (9), a deep learning method which uses autoencoders to compress expensive PIC data to a latent space. Concurrent to our work, Kelling et al. (24) proposes steaming ML pipelines for petascale PIC simulations, enabling models to learn directly from data *in-transit*, without intermediate storage. Unlike PIC, which tracks individual particles (Lagrangian representation), gyrokinetics is a distributional approach which models the dynamics in a phase space (Eulerian specification).

Physics-Informed Neural Networks (PINNs) combine neural network training with physical constraints, including them in the loss function (41, 7). This is done to ensure that physical constraints such as boundary conditions and conservation laws are respected in the learned solutions, and more generally that they remain consistent with the underlying equations. Sitting at the intersection of PINNs and neural compression, EinFields (10) use neural fields and "Sobolev" training (45) to achieve impressive compression and accuracy on storage intensive general relativity data. On the other hand, we employ problem specific integral loss terms to improve the physical consistency under highly lossy compression.

3 Methods

We identify two dominant approaches to learned compression, depending of a few key features and where weight sharing happens. The first, which we denominate as *"monolithic models"*, shares its weights across different trajectories and time, and a single model Γ_θ is trained. In other words, they are *single-model-many-data*: a large model is trained once on a dataset, and compression is applied to unseen samples. A representative of the first class is VQ-VAEs (50), where compression explicitly occurs in latent space at the bottleneck between encoder and decoder.

Table 1: Monolithic and micromodel.

Property	**Mono**	**Micro**
Cheap compress	✓	✗
Cheap reconstruct	✓	✓
Cheap training	✗	✓
Dataset-free	✗	✓
Resolution invariant	~	✓
OOD generalization	~	✓
Time dynamics	?	?

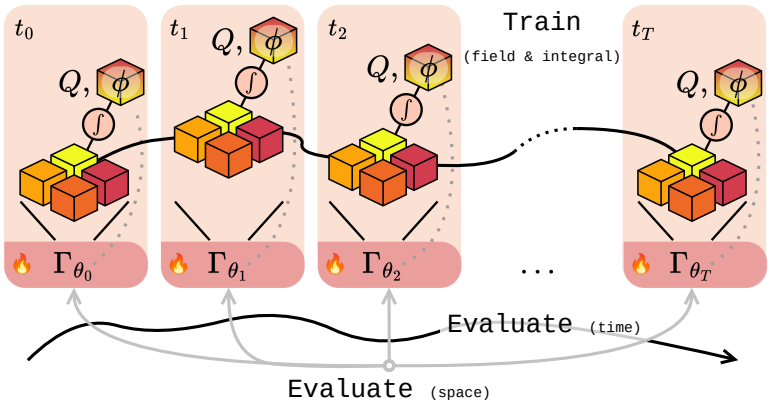

Figure 1: Sketch of the training and evaluation for neural compression models. Training is done at individual time snapshots for scalability reasons, while evaluation considers turbulence characteristics, taking both spatial and temporal information into account.

In contrast, *"micromodels"* work by overfitting an independent and compact set of parameters at each datapoint, for example in time $[\Gamma_{\theta_t}]_{(0\ldots T)}$, making them *many-models-many-data*. Unlike autoencoders, compression occurs implicitly in weight-space and so reconstruction happens without an explicit representation. Coordinate-conditioned neural implicit fields (13, 43) are an instance of *micromodels*, for example trained to represent a 3D scene. Figure 1 outlines our training and evaluation for a time trajectory: for both cases, time is not explicitly used in compression but only in evaluation. Table 1 provides a broad comparison of the characteristics of these two approaches, using a tick (✓) where they excel, a cross (✗) where they fail, and ~or **?**for properties that are sometimes present or unknown.

3.1 5D autoencoders

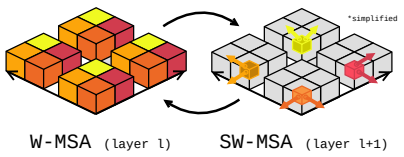

Figure 2: 5D swin attention.

Because gyrokinetic data is 5-dimensional, there is no standard architecture for such dimensionalities. We use nD swin layers (17), based on Shifted Window Attention (32), which promise good scaling to higher dimensions. They work by first partitioning the domain in non-overlapping *windows*, then performing attention only locally within the window. Swin layers scale linearly with the sequence length as opposed to the quadratic behaviour of attention (32, 33, 25). Figure 2 displays the 5-dimensional `W-MSE` and `SW-MSE` layers, where blocks of the same color visualize the receptive field of the local attention.

The autoencoder structure is derived from hierarchical vision transformers (12, 32): the encoder $\mathcal{E}$ first tiles the field into (non-overlapping) patches and projects it to an embedding space, then applies interleaved swin and downsampling layers. At the bottleneck, the channels are downprojected to increase the compression level. The decoder $\mathcal{D}$ mirrors the encoder, with swin layers and patch expansions to up-project the fields until the original resolution is restored.

Given an autoencoder $\Gamma(\boldsymbol{f}) = \mathcal{D} \circ \mathcal{E}(\boldsymbol{f})$, the distribution function $\boldsymbol{f}$ is compressed by the encoder $\mathcal{E}$, which maps to a low-dimensional latent space. Namely, the compressed representation $Z = \mathcal{E}(\boldsymbol{f})$. Similarly, decompression is performed by the decoder $\mathcal{D}$, which transforms latent representations to the 5D physical space. In particular, we experiment with both standard autoencoders and Vector-Quantized Variational Autoencoders (VQ-VAE) (50). In VQ-VAEs, the encoder $\mathcal{E}$ outputs indices of a *codebook*, learned end-to-end during autoencoder training. The discrete and compact nature of the codebook enables a very high compression ratio in the latent representation. Finally, the decoder $\mathcal{D}$ reconstructs the input by mapping the quantized latent codes back to the data space.

3.2 Neural implicit fields

Neural fields implicitly represent continuous signals as coordinate-based learnable functions (35, 36, 13). In general, neural fields map input coordinates to the respective values of a physical field. They are usually implemented as MLPs with special activation functions (43, 42, 14). Most importantly, this representation is independent of the data resolution and offers flexible compression methods, while allowing for continuous interpolation.

The distribution function $\boldsymbol{f}$ is a complex matrix indexed by a five-tuple of coordinates $(v_\parallel, \mu, s, x, y)$. We train a separate neural field $\Gamma_{\theta_{t,c}}$ to overfit each $\boldsymbol{f}_t^c$, a snapshot at time t of a trajectory configured as c, for performance, compute and scalability reasons. Coordinates are first embedded in a sinusoidal positional embedding, then fed in an MLP such as SIREN (43). Fitting a neural field $\Gamma_{\theta_{t,c}}$ takes $\sim$5 minutes (NVIDIA H100), and since we use independent networks for each timestep and trajectory, training is easily parallelizable or can be performed in a staggered, pipelined fashion.

3.3 Physics-informed training

To ensure conservation of important characteristics, we include supervision on physical quantities. In particular, we train on the (scalar) heat flux $Q \in \mathbb{R}$ and electrostatic potential $\boldsymbol{\phi} \in \mathbb{R}^{x \times s \times y}$ integrals. These are integrals of the distribution function $\boldsymbol{f}$ and are formulated as

$$\boldsymbol{\phi} = \mathbf{A} \int \mathbf{J_0} \boldsymbol{f} \, \mathrm{d}v_\parallel \mathrm{d}\mu, \quad Q = \int \mathbf{B} \int \mathbf{v}^2 \boldsymbol{\phi} \boldsymbol{f} \, \mathrm{d}v_\parallel \mathrm{d}\mu \; \mathrm{d}x \mathrm{d}y \mathrm{d}s,$$

where $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{x \times s \times y}$ encompass geometric and physical parameters, $\mathbf{v}$ is the particle energy, and $\mathbf{J_0}$ denotes the Bessel function. The electrostatic potential $\boldsymbol{\phi}$ is obtained by integrating velocity-space from $\boldsymbol{f}$, while the heat flux Q depends on both $\boldsymbol{f}$ and $\boldsymbol{\phi}$. The corresponding losses are defined as deviations from the ground truth, for both the 5D distribution function $\boldsymbol{f}$ and its integrals:

$$\mathcal{L}_{recon} = \sum_{v_\parallel, \mu, x, s, y} \left\| \boldsymbol{f}^{\text{pred}} - \boldsymbol{f}^{\text{GT}} \right\|^2, \quad \mathcal{L}_Q = \left| Q_{\text{pred}} - Q_{\text{GT}} \right|, \quad \mathcal{L}_{\boldsymbol{\phi}} = \sum_{x, s, y} \left\| \boldsymbol{\phi}^{\text{pred}} - \boldsymbol{\phi}^{\text{GT}} \right\|^2.$$

$\mathcal{L}_Q$ and $\mathcal{L}_{\boldsymbol{\phi}}$ losses rely on *global* quantities, as the Q and $\boldsymbol{\phi}$ integrals require information on the spectral structure of $\boldsymbol{f}$, or in general spanning the entire phase space domain. This slightly complicates their application as losses.

Still, these losses can be included in both autoencoder and neural field training, with two caveats: First, because they rely on mode composition of $\boldsymbol{f}$, they are effective only once the reconstruction is already *"reasonably close"* to the ground truth. To address this, we apply $\mathcal{L}_Q$ and $\mathcal{L}_{\boldsymbol{\phi}}$ only after the predicted $\boldsymbol{f}$s have converged, and with a reduced learning rate (generally $100\times$ smaller than the one used for $\boldsymbol{f}$). Second, due to their global nature, they cannot be applied on small coordinate batches normally used for neural field training. Instead, the full field must be reconstructed to evaluate them.

4 Results

Setup. The neural fields are SIREN networks (43), with 64 latent dimension, 5 layers and skip connections. They are fit using AdamW (34) with learning rate decaying between $[1e-3, 1e-12]$ (details in Appendix C.3). Autoencoders have latent dimension of 1024 and a total of 152M parameters. They are trained on a dataset of 12,985 distribution function $\boldsymbol{f}$ time snapshots, amounting to around 1TB of data (data information in Appendix B), and compression/reconstruction is subsequently expected to happen *out of distribution*, to unseen trajectories. Autoencoder training takes $\sim$60 hours (300 epochs, $4\times$ NVIDIA H100) for both AE and VQ-VAE, trained using Muon (23) with learning rate decaying between $[1e-4, 1e-7]$ (details in Appendix C.4).

We compare with traditional compression based on different techniques: ZFP (30), a very popular compression method for scientific data relying on block-quantization, Wavelet-based compression and spatial PCA. Baselines are tuned to achieve compression rates of around $1{,}000\times$ (99.9% size

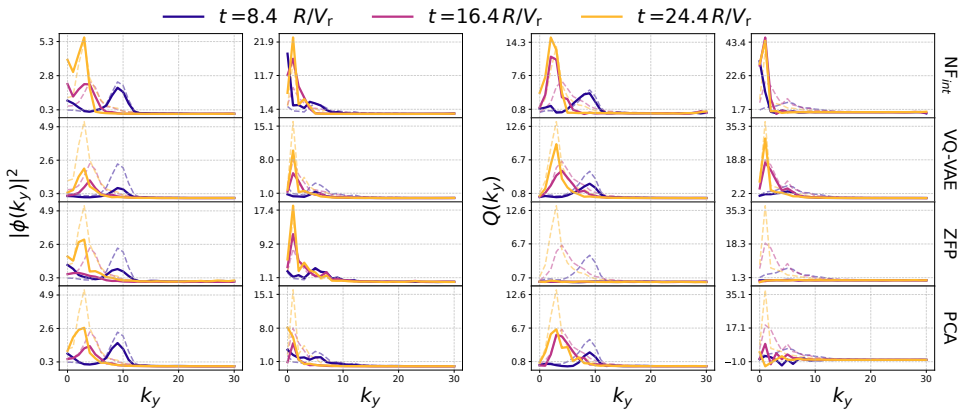

Figure 3: Energy cascade visualized as the energy transfer from higher to lower modes as turbulence develops. The three time snapshots at $[8.4, 16.4, 24.4]R/V_r$ are specifically sampled in the transitional phase where mode growth and energy cascade happens, before reaching the statistically stable phase. Pairs of columns are two different trajectories for k_y^{spec} (left pair) and Q^{spec} (right pair), rows are compression methods, with first row being ground truth. Lines of varied colors are the spectras at specific times (non time-averaged), and background lines are respective ground truth.

reduction), comparable with neural fields and vanilla autoencoders. For reference, *off-the-shelf* traditional techniques such as `gzip` achieve a lossless compression ratio of $\sim$1.1x (8% reduction). More information on baselines can be found in Appendix C.2.

Interpretation. Table 2 quantitatively summarizes the results of our analysis. Baselines are evaluated on both traditional compression metrics, integral and turbulence errors. To measure $\boldsymbol{f}$ reconstruction quality after compression, L1 error, Peak Signal-to-Noise-Ratio (PSNR) and Bits-per-Pixel (BPP) (defined in Appendix C.1) are reported. Integral errors are mean absolute error of flux Q and potential ϕ obtained by integration of the reconstructed distribution field according to Section 3.3. As for turbulence, Pearson correlation (PC) for *time-averaged* (TA) $\overline{k_y^{\text{spec}}}$ and $\overline{Q^{\text{spec}}}$ are reported. k_y^{spec} and Q^{spec} are spatially averaged spectras along the k_y coordinates of the potential ϕ and the flux field (Q before second integral) respectively. Accurately capturing them is a requirement for properly reconstructing turbulence.

At this regime, neural fields and autoencoders are comparable or better than traditional methods on the compression metrics, while providing a clear promotion on integrated quantities. The difference

Table 2: Comparison between neural compression and traditional methods on compression metrics and physical measures (integrals, steady-state turbulence metrics). Evaluation on 30 total $\boldsymbol{f}$s (10 different turbulent trajectories, 3 random time snapshots), sampled in the statistically steady phase. $()_{\text{int}}$ models are trained with physics-informed integral training. L1 errors are not normalized. Best result in bold, second best underlined.

		Compression $\boldsymbol{f}$			Integrals Q, ϕ		Turbulence $Q^{\text{spec}}, k_y^{\text{spec}}$	
	CR	L1 ↓	PSNR ↑	BPP ↓	L1(Q) ↓	L1(ϕ) ↓	PC($\overline{k_y^{spec}}$) ↑	PC($\overline{Q^{spec}}$) ↑
NF	988×	<u>0.43</u>	<u>34.19</u>	0.097	29.65	414.24	0.90	0.90
NF$_{\text{int}}$	988×	**0.39**	**34.88**	0.097	**2.80**	**118.82**	**0.91**	0.91
NF$_{\text{int}}$ + ZFP	2,581×	**0.39**	**34.88**	<u>0.037</u>	<u>3.51</u>	256.91	0.90	0.91
AE	1,208×	0.55	32.94	0.079	56.51	272.88	0.90	0.91
AE$_{\text{int}}$	1,208×	0.77	32.13	0.079	24.66	536.06	0.89	<u>0.96</u>
VQ-VAE	38,684×	0.41	33.06	**0.002**	37.01	<u>121.74</u>	<u>0.91</u>	**1.00**
VQ-VAE$_{\text{int}}$	38,684×	0.55	32.60	**0.002**	9.71	593.56	0.89	0.78
ZFP	982×	0.65	28.65	0.103	91.01	1064.73	0.90	-0.19
Wavelet	920×	0.44	33.05	0.127	90.63	691.64	0.89	-0.93
PCA	1,020×	0.47	31.95	0.094	63.38	425.49	0.90	0.67

in integral error between NF and NF_{int} in Table 2 highlights the advantage of training on $\mathcal{L}_Q$ and $\mathcal{L}_\phi$, considerably boosting physical soundness. At the same time, comparing "plain" autoencoders with those trained with integral losses reveals the challenges of balancing the loss terms, as training instabilities disrupt the $\boldsymbol{f}$ reconstruction quality when jointly optimizing $\mathcal{L}_{recon}$, $\mathcal{L}_Q$ and $\mathcal{L}_\phi$.

Recovering turbulence. Figure 3 shows how well different models capture the direct energy cascade phenomena across different simulations (energy shifting to lower modes over time), by visualizing the *per-timestep* spectras k_y^{spec} and Q^{spec}. Note that this figure provides a qualitative comparison of turbulence recovery on the temporal axis, in contrast to the steady-state statistics reported in Table 2. The time snapshots examined in Figure 3 ($[8.4, 16.4, 24.4]R/V_{\text{r}}$) are sampled in the transitional phase where turbulence grows, at the so called *overshoot*. These are different to those in Table 2 (random in the saturated phase). On k_y^{spec}, traditional compression methods already achieve reasonable performance in most cases, but on Q^{spec} they produce severely non-physical results (flat curves, negative numbers). In contrast, neural fields and VQ-VAE can reproduce the overall profiles consistently, with VQ-VAE excelling at the flux spectra. However, both often fail to capture the higher-frequency magnitudes, especially autoencoders. The behaviors can be attributed to the spectral bias of neural networks (40, 48), where low-frequency (high-energy) components are favored over high-frequencies.

Hybrid compression. Finally, we additionally showcase an example to further push the idea of high-compression: neural fields can be compressed by applying traditional techniques in weight space, with very minor degradation in performance ("$\text{NF}_{\text{int}} + \text{ZFP}$" row in Table 2). Similarly to how data can be compressed into a compact neural field representation, the network weights themselves are redundant and also lie on a lower-dimensional manifold. This is related to pruning (29, 19), network compression (21), and the lottery ticket hypothesis (15).

5 Conclusions

Our study provides compelling evidence that for gyrokinetic simulations of plasma turbulence, neural compression can preserve underlying characteristics of the physical system while achieving extreme compression levels of up to $40{,}000\times$. This is accomplished by constraining the compression during training to maintain integral quantities over the 5D fields. We expect that similar approaches are extendable to large-scale scientific datasets of other disciplines, enabling practitioners to store compressed simulations that accurately capture specified physical phenomena across time and space, which was previously infeasible due to storage requirements. These tools may provide considerable impact in simplifying accessibility and transfer of data, hence accelerating research in the scientific community: our results are a first step in adopting machine learning for physics-preserving, on-the-fly (*in-situ*) compression of simulation snapshots, enabling more comprehensive post-hoc analysis.

Future work should systematically explore the trade-off between compression rate, reconstruction quality, and the preservation of physical quantities, over the different neural compression methodologies. A more thorough ablation of different modeling choices, such as WIRE (42) and SIREN (43) networks as neural fields, is an essential direction for future work. Further, at the moment, our current physics training relies on heuristics such as two-stage training to balance compression and reconstruction of physics constraints. Especially for autoencoders, which showed instabilities when training on integral losses, gradient normalization (8, 31) or adaptive loss weighting (4) could help. Finally, so far we only evaluate transient turbulent phenomena quantitatively and for a limited setup. A larger scape, quantitative and systematic evaluation of time dynamics reconstruction would be beneficial, especially for applications.

Acknowledgments and Disclosure of Funding

This work has been funded by the Fusion Futures Programme. As announced by the UK Government in October 2023, Fusion Futures aims to provide holistic support for the development of the fusion sector. The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. We thank the projects FWF AIRI FG 9-N (10.55776/FG9), AI4GreenHeatingGrids (FFG- 899943), Stars4Waters (HORIZON-CL6-2021-CLIMATE-01-01), FWF Bilateral Artificial Intelligence (10.55776/COE12).

References

[1] Mark Ainsworth, Ozan Tugluk, Ben Whitney, and Scott Klasky. Multilevel techniques for compression and reduction of scientific data—the multivariate case. SIAM Journal on Scientific Computing, 41(2):A1278–A1303, 2019.

[2] Rushil Anirudh, Rick Archibald, M. Salman Asif, Markus M. Becker, Sadruddin Benkadda, Peer-Timo Bremer, Rick H. S. Budé, C. S. Chang, Lei Chen, R. M. Churchill, Jonathan Citrin, Jim A. Gaffney, Ana Gainaru, Walter Gekelman, Tom Gibbs, Satoshi Hamaguchi, Christian Hill, Kelli Humbird, Sören Jalas, Satoru Kawaguchi, Gon-Ho Kim, Manuel Kirchen, Scott Klasky, John L. Kline, Karl Krushelnick, Bogdan Kustowski, Giovanni Lapenta, Wenting Li, Tammy Ma, Nigel J. Mason, Ali Mesbah, Craig Michoski, Todd Munson, Izumi Murakami, Habib N. Najm, K. Erik J. Olofsson, Seolhye Park, J. Luc Peterson, Michael Probst, David Pugmire, Brian Sammuli, Kapil Sawlani, Alexander Scheinker, David P. Schissel, Rob J. Shalloo, Jun Shinagawa, Jaegu Seong, Brian K. Spears, Jonathan Tennyson, Jayaraman Thiagarajan, Catalin M. Tico, Jan Trieschmann, Jan van Dijk, Brian Van Essen, Peter Ventzek, Haimin Wang, Jason T. L. Wang, Zhehui Wang, Kristian Wende, Xueqiao Xu, Hiroshi Yamada, Tatsuya Yokoyama, and Xinhua Zhang. 2022 review of data-driven plasma science. IEEE Transactions on Plasma Science, 51(7):17501838, July 2023. ISSN 1939-9375. doi: 10.1109/tps.2023.3268170.

[3] Rafael Ballester-Ripoll, Peter Lindstrom, and Renato Pajarola. Tthresh: Tensor compression for multidimensional visual data. IEEE Transaction on Visualization and Computer Graphics, to appear, 2019. arXiv:1806.05952.

[4] Shamsulhaq Basir and Inanc Senocak. An adaptive augmented lagrangian method for training physics and equality constrained artificial neural networks, 2023.

[5] C. K. Birdsall and A. B. Langdon. Plasma physics via computer simulation. New York: Taylor and Francis, first edition, 2005.

[6] Steven L. Brunton. What is turbulence? turbulent fluid dynamics are everywhere. Video, Data Driven Fluid Dynamics, Cassyni, 2021. Accessed 2025-08-07.

[7] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review, 2021.

[8] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks, 2018.

[9] Jong Choi, Michael Churchill, Qian Gong, Seung-Hoe Ku, Jaemoon Lee, Anand Rangarajan, Sanjay Ranka, Dave Pugmire, CS Chang, and Scott Klasky. Neural data compression for physics plasma simulation. In Neural Compression: From Information Theory to Applications – Workshop @ ICLR 2021, 2021.

[10] Sandeep Suresh Cranganore, Andrei Bodnar, Arturs Berzins, and Johannes Brandstetter. Einstein fields: A neural perspective to computational general relativity, 2025.

[11] James Diffenderfer, Alyson L Fox, Jeffrey A Hittinger, Geoffrey Sanders, and Peter G Lindstrom. Error analysis of zfp compression for floating-point data. SIAM Journal on Scientific Computing, 41(3):A1867–A1898, 2019.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[13] Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. arXiv preprint arXiv:2201.12204, 2022. Preprint.

[14] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, 2017.

[15] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

[16] EA Frieman and Liu Chen. Nonlinear gyrokinetic equations for low-frequency electromagnetic waves in general plasma equilibria. The Physics of Fluids, 25(3):502–508, 1982.

[17] Gianluca Galletti, Fabian Paischer, Paul Setinek, William Hornsby, Lorenzo Zanisi, Naomi Carey, Stanislas Pamela, and Johannes Brandstetter. 5d neural surrogates for nonlinear gyrokinetic simulations of plasma turbulence, 2025.

[18] Zhenhuan Gong, Terry Rogers, John Jenkins, Hemanth Kolla, Stephane Ethier, Jackie Chen, Robert Ross, Scott Klasky, and Nagiza F Samatova. MLOC: Multi-level layout optimization framework for compressed scientific data exploration with heterogeneous access patterns. In Proceedings of the 41st International Conference on Parallel Processing (ICPP), pp. 239–248. IEEE, 2012.

[19] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2015.

[20] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[21] Moshik Hershcovitch, Andrew Wood, Leshem Choshen, Guy Girmonsky, Roy Leibovitz, Ilias Ennmouri, Michal Malka, Peter Chin, Swaminathan Sundararaman, and Danny Harnik. Zipnn: Lossless compression for ai models. arXiv preprint arXiv:2411.05239, 2024.

[22] Richard D. J. G. Ho, Daniel Clark, and Arjun Berera. Chaotic measures as an alternative to spectral measures for analysing turbulent flow, 2024.

[23] Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

[24] Jeffrey Kelling, Vicente Bolea, Michael Bussmann, Ankush Checkervarty, Alexander Debus, Jan Ebert, Greg Eisenhauer, Vineeth Gutta, Stefan Kesselheim, Scott Klasky, Vedhas Pandit, Richard Pausch, Norbert Podhorszki, Franz Poschel, David Rogers, Jeyhun Rustamov, Steve Schmerler, Ulrich Schramm, Klaus Steiniger, Rene Widera, Anna Willmann, and Sunita Chandrasekaran. The artificial scientist – in-transit machine learning of plasma simulations, 2025.

[25] Peter Kim, Junbeom Kwon, Sunghwan Joo, Sangyoon Bae, Donggyu Lee, Yoonho Jung, Shinjae Yoo, Jiook Cha, and Taesup Moon. Swift: Swin 4d fmri transformer. Advances in Neural Information Processing Systems, 36:42015–42037, 2023.

[26] A. N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers. Proceedings: Mathematical and Physical Sciences, 434(1890):9–13, 1991. Accessed: 2025-08-07.

[27] John A. Krommes. The gyrokinetic description of microturbulence in magnetized plasmas. Annual Review of Fluid Mechanics, 44(Volume 44, 2012):175–201, 2012. ISSN 1545-4479. doi: https://doi.org/10.1146/annurev-fluid-120710-101223.

[28] Sriram Lakshminarasimhan, Neil Shah, Stephane Ethier, Scott Klasky, Rob Latham, Rob Ross, and Nagiza F. Samatova. Compressing the incompressible with isabela: In-situ reduction of spatio-temporal data. In Emmanuel Jeannot, Raymond Namyst, and Jean Roman (eds.), Euro-Par 2011 Parallel Processing, pp. 366–379, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-23400-2.

[29] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. Advances in neural information processing systems, 2, 1990.

[30] Peter Lindstrom. Fixed-rate compressed floating-point arrays. IEEE Transactions on Visualization and Computer Graphics, 20(12):2674–2683, 2014.

[31] Qiang Liu, Mengyu Chu, and Nils Thuerey. Config: Towards conflict-free training of physics informed neural networks. In The Thirteenth International Conference on Learning Representations, 2024.

[32] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, pp. 9992–10002. IEEE, 2021. doi: 10.1109/ICCV48922.2021.00986.

[33] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pp. 3192–3201. IEEE, 2022. doi: 10.1109/CVPR52688.2022.00320.

[34] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[35] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In European Conference on Computer Vision (ECCV), pp. 405–421. Springer, 2020. doi: 10.1007/978-3-030-58452-8_24.

[36] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation, 2019.

[37] A.G. Peeters, Y. Camenen, F.J. Casson, W.A. Hornsby, A.P. Snodin, D. Strintzi, and G. Szepesi. The nonlinear gyro-kinetic flux tube code gkw. Computer Physics Communications, 180 (12):2650–2672, 2009. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2009.07.001. 40 YEARS OF CPC: A celebratory issue focused on quality software for high performance, grid and novel computing architectures.

[38] Stephen B. Pope. Turbulent Flows. Cambridge University Press, 2000.

[39] Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free, 2025.

[40] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2019.

[41] Maziar Raissi, Paris Perdikaris, and George Em 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. doi: 10.1016/j.jcp.2018.10.045.

[42] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G. Baraniuk. Wire: Wavelet implicit neural representations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023. arXiv preprint arXiv:2301.05187.

[43] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In NeurIPS, 2020. arXiv preprint arXiv:2006.09661.

[44] Alexander J. Smits. Lectures in fluid mechanics: Viscous flows and turbulence. Lecture Notes, Department of Mechanical Engineering, Princeton University.

[45] Hwijae Son, Jin Woo Jang, Woo Jin Han, and Hyung Ju Hwang. Sobolev training for physics informed neural networks, 2021.

[46] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.

[47] Zoran Šverko, Miroslav Vrankić, Saša Vlahinić, and Peter Rogelj. Complex pearson correlation coefficient for eeg connectivity analysis. Sensors, 22(4):1477, 2022. doi: 10.3390/s22041477.

[48] Damien Teney, Armand Nicolicioiu, Valentin Hartmann, and Ehsan Abbasnejad. Neural redshift: Random networks are not random functions, 2025.

[49] David Tskhakaya. The Particle-in-Cell Method, pp. 161–189. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-74686-7. doi: 10.1007/978-3-540-74686-7_6.

[50] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Advances in Neural Information Processing Systems (NeurIPS), volume 30, 2017.

[51] Jialing Zhang, Xiaoyan Zhuo, Aekyeung Moon, Hang Liu, and Seung Woo Son. Efficient encoding and reconstruction of hpc datasets for checkpoint/restart. In 2019 35th Symposium on Mass Storage Systems and Technologies (MSST), pp. 79–91. IEEE, 2019.

A Gyrokinetics

Gyrokinetics (16, 27, 37) is a reduced form of Plasma kinetics that is computationally more efficient and can be use to locally simulate Plasma behaviour within a so-called *flux tube* in the torus. Local gyrokinetics is a theoretical framework to study plasma behavior on perpendicular spatial scales comparable to the gyroradius, i.e., the radius of circular motion exhibited by charged particles in a magnetic field, and frequencies much lower than the particle cyclotron frequencies, i.e., the frequency at which charged particles spiral around magnetic field lines due to the Lorentz force. Gyrokinetics models the time evolution of electrons and ions via the distribution function $\boldsymbol{f}$, which is based on 3D coordinates, their parallel and perpendicular velocities, together with the angle w.r.t. the field lines. However, the latter dimension is averaged out by modelling only the so-called guiding center of a particle instead of its gyral movement. Furthermore, instead of modelling the perpendicular velocity, usually only its magnitude is considered, which is also referred to as the magnetic moment μ. Hence, the 5D gyrokinetic distribution function can be written as $\boldsymbol{f} = \boldsymbol{f}(k_x, k_y, s, v_\parallel, \mu)$, where k_x and k_y are spectral coordinates, s is the toroidal coordinate along the field line, and $v_\parallel$ the parallel velocity component. The perturbed time-evolution of $\boldsymbol{f}$, for each species (ions and electrons), is governed by

$$\underbrace{\frac{\partial \boldsymbol{f}}{\partial t} + (v_\parallel \boldsymbol{b} + \boldsymbol{v}_D) \cdot \nabla \boldsymbol{f} - \frac{\mu B}{m} \frac{\boldsymbol{B} \cdot \nabla B}{B^2} \frac{\partial \boldsymbol{f}}{\partial v_\parallel}}_{\text{Linear}} + \underbrace{\boldsymbol{v}_\chi \cdot \nabla \boldsymbol{f}}_{\text{Nonlinear}} = S \,, \tag{1}$$

where $v_\parallel \boldsymbol{b}$ is the motion along magnetic field lines, $\boldsymbol{b} = \boldsymbol{B}/B$ is the unit vector along the magnetic field $\boldsymbol{B}$ with magnitude B[1], $\boldsymbol{v}_D$ the magnetic drift due to gradients and curvature in $\boldsymbol{B}$, and $\boldsymbol{v}_\chi$ describes drifts arising from the $\boldsymbol{E} \times \boldsymbol{B}$ force, a key driver of plasma dynamics. Finally, S is the source term that represents the external supply of energy. The term $\boldsymbol{v}_\chi \cdot \nabla \boldsymbol{f}$ models the nonlinear interaction between the distribution function $\boldsymbol{f}$ and its velocity space integral $\boldsymbol{\phi}$, and it describes turbulent advection. The resulting nonlinear coupling constitutes the computationally most expensive term.

A.1 Derivation of the Gyrokinetic equation

We begin with the Vlasov equation for the distribution function $\boldsymbol{f}(\boldsymbol{r}, \boldsymbol{v}, t)$:

$$\frac{\partial \boldsymbol{f}}{\partial t} + \mathbf{v} \cdot \nabla \boldsymbol{f} + \frac{q}{m} (\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}) \cdot \nabla_v \boldsymbol{f} = 0 \tag{2}$$

The Vlasov equation describes the conservation of particles in phase space in the absence of collisions. Here, $\boldsymbol{r} = (x, y, z)$ and $\boldsymbol{v} = (v_x, v_y, v_z)$ correspond to coordinates in the spatial and the velocity domain, respectively. Hence the Vlasov equation is a 7D (including time) PDE representing the density of particles in phase space at position $\boldsymbol{r}$, velocity $\boldsymbol{v}$, and time. The term $\nabla_{\boldsymbol{v}} \boldsymbol{f}$ describes the response of the distribution function to accelerations of particles and $\frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B})$ denotes the Lorentz force, which depends on particle charge q and mass m, as well as electric field $\boldsymbol{E}$ and magnetic field $\boldsymbol{B}$. Finally, the advection (or convection) term $\boldsymbol{v} \nabla \boldsymbol{f}$ describes transport of the distribution functon through space due to velocities.

To derive the *gyrokinetic equation*, we transform from particle coordinates to guiding center coordinates $(\mathbf{R}, v_\parallel, \mu, \theta)$, where $\mu = \frac{m v_\perp^2}{2B}$ is the magnetic moment, θ the gyrophase, which describes the position of a particle around its guiding center as it gyrates along a field line, and $\boldsymbol{R}$ is the coordinate of the guiding center.

Assuming the time scale L at which the background field changes is much longer than the gyroperiod with a small Larmor radius $\rho \ll L$, we can *gyroaverage* to remove the dependency on the gyrophase θ, yielding:

$$\frac{\partial \boldsymbol{f}}{\partial t} + \dot{\boldsymbol{R}} \cdot \nabla \boldsymbol{f} + \dot{v}_\parallel \frac{\partial \boldsymbol{f}}{\partial v_\parallel} = 0 \tag{3}$$

[1]We adopt uppercase notation for vector fields $\boldsymbol{E}$ and $\boldsymbol{B}$ to adhere with literature.

A.1.1 Linear Terms

The unperturbed (background) motion of the guiding center is governed by:

$$\dot{\boldsymbol{R}} = v_{\parallel}\boldsymbol{b} + \boldsymbol{v}_D \tag{4}$$

$$\dot{v}_{\parallel} = -\frac{\mu}{m}\boldsymbol{b}\cdot\nabla\boldsymbol{B} \tag{5}$$

Here, $\mathbf{b} = \boldsymbol{B}/B$ is the unit vector along the magnetic field, and $\boldsymbol{v}_D$ represents magnetic drifts. Substituting into the kinetic equation yields

$$\frac{\partial \boldsymbol{f}}{\partial t} + (v_{\parallel}\boldsymbol{b} + \boldsymbol{v}_D)\cdot\nabla\boldsymbol{f} - \frac{\mu}{m}\boldsymbol{b}\cdot\nabla\boldsymbol{B}\frac{\partial \boldsymbol{f}}{\partial v_{\parallel}} = 0 \tag{6}$$

We can express the magnetic gradient term using:

$$\boldsymbol{b}\cdot\nabla\boldsymbol{B} = \frac{\boldsymbol{B}\cdot\nabla\boldsymbol{B}}{B} \tag{7}$$

so that:

$$\frac{\mu}{m}\boldsymbol{b}\cdot\nabla\boldsymbol{B} = \frac{\mu B}{m}\frac{\boldsymbol{B}\cdot\nabla B}{\boldsymbol{B}^2} \tag{8}$$

A.1.2 Nonlinear Term

Fluctuating electromagnetic potentials $\delta\boldsymbol{\phi}, \delta\boldsymbol{A}$ induce E×B and magnetic flutter drifts. We define the gyroaveraged generalized potential as

$$\chi = \langle \boldsymbol{\phi} - \frac{v_{\parallel}}{c}A_{\parallel}\rangle, \tag{9}$$

where $\boldsymbol{A}_{\parallel}$ is the parallel component of the vector potential, $\langle\cdot\rangle$ denotes the gyroaverage, and c is the speed of light, which is added to ensure correct units. $\boldsymbol{\phi}$ is the electrostatic potential, the computation of which involves an integral of $\boldsymbol{f}$ over the velocity space (see eq. 1.41 in the GKW manual [2] for a complete description).

This gives rise to the drift

$$\mathbf{v}_{\chi} = \frac{c}{B}\mathbf{b}\times\nabla\chi, \tag{10}$$

and yields the nonlinear advection term $\mathbf{v}_{\chi}\cdot\nabla\boldsymbol{f}$.

A.1.3 Final Equation

We arrive at the gyrokinetic equation in split form:

$$\frac{\partial \boldsymbol{f}}{\partial t} + (v_{\parallel}\mathbf{b} + \mathbf{v}_D)\cdot\nabla\boldsymbol{f} - \frac{\mu B}{m}\frac{\mathbf{B}\cdot\nabla B}{B^2}\frac{\partial \boldsymbol{f}}{\partial v_{\parallel}} + \mathbf{v}_{\chi}\cdot\nabla\boldsymbol{f} = S \tag{11}$$

Here, S represents external sources, collisions, or other drive terms. To enhance the tractability of Equation (1), the distribution function $\boldsymbol{f}$ is usually split into equilibrium and perturbation terms

$$\boldsymbol{f} = \boldsymbol{f}_0 + \delta\boldsymbol{f} = \underbrace{\boldsymbol{f}_0 - \frac{Z\boldsymbol{\phi}}{T}\boldsymbol{f}_0}_{\text{Adiabatic}} + \underbrace{\frac{\partial h}{\partial t}}_{\text{Kinetic}}, \tag{12}$$

where $\boldsymbol{f}_0$ is a background or equilibrium distribution function, T the particle temperature, Z the particle charge, $\boldsymbol{\phi}$ the electrostatic potential, and δf the total perturbation to the distribution function, which comprises of *adiabatic* and *kinetic* response. The adiabatic term describes rapid and passive responses to the electrostatic potential that do not contribute to turbulent transport, while the kinetic term governs irreversible dynamics that facilitate turbulence. Numerical codes, such as GKW (37), rely on solving for δf instead of $\boldsymbol{f}$. A common simplification is to assume that electrons are adiabatic, which allows us to neglect the kinetic term in the respective $\delta\boldsymbol{f}$. Hence, the respective $\boldsymbol{f}$ for electrons ($\boldsymbol{f}_e$) does not need to be modelled, effectively halving the computational cost.

[2] https://bitbucket.org/gkw/gkw/src/develop/doc/manual/

B Dataset

The simulations used for both the autoencoder training (49 trajectories) and the evaluation (10 trajectories) are generated with the numerical code GKW (37). They are sampled by varying four parameters: R/L_t, R/L_n, $\hat{s}$, and q, which significantly affect emerging turbulence in the Plasma.

- R/L_t is the ion temperature gradient, which is the main driver of turbulence.
- R/L_n is the density gradient, whose effect is less pronounced. It can have a stabilizing effect, but can sometimes also lead to increased turbulence.
- $\hat{s}$ denotes magnetic shearing, hence it usually has a stabilizing effect as more magnetic shearing results in better isolation of the Plasma.
- q denotes the so-called safety factor, which is the inverse of the rotational transform and describes how often a particle takes a poloidal turn before taking a toroidal turn.

We specify the ranges for sampling the four parameters as $R/L_T \in [1, 12]$, $R/L_n \in [1, 7]$, $q \in [1, 9]$, and $\hat{s} \in [0.5, 5]$. Additionally, we also vary the noise amplitude of the initial condition (within $[1e-5, 1e-3]$).

To make storage more feasible, simulations are time-coarsened by saving snapshot every 60. Each GKW run with the specified configurations takes around $\sim$6 hours (76 cores, Intel Ice Lake 4.1GHz CPU) and $\sim$20GBs of storage.

C Implementation details

C.1 Evaluation an training metrics

We evaluate reconstruction with spatial and physical metrics. Since gyrokinetic data is complex-valued, we can also apply complex-generalizations of common metrics.

Complex Mean Squared Error. Given two complex-valued fields $z_1, z_2 \in \mathbb{C}^N$, the complex mean squared error is:

$$\mathrm{cMSE}(z_1, z_2) = \left\langle \Re(z_1 - z_2)^2 + \Im(z_1 - z_2)^2 \right\rangle = \left\langle |z_1 - z_2|^2 \right\rangle$$

where $\langle \cdot \rangle$ denotes the average over all spatial or spatiotemporal dimensions.

Complex Pearson Correlation Coefficient. Given complex-valued vectors $x_1, x_2 \in \mathbb{C}^N$, the complex Pearson correlation is computed as (47):

$$r(x_1, x_2) = \frac{\sum_{n=1}^{N} (x_{1,n} - \bar{x}_1)\,(x_{2,n} - \bar{x}_2)^*}{\sqrt{\sum_{n=1}^{N} |x_{1,n} - \bar{x}_1|^2 \cdot \sum_{n=1}^{N} |x_{2,n} - \bar{x}_2|^2}}$$

where: - $\bar{x}_1, \bar{x}_2$ are the mean values of each series, - $(\cdot)^*$ denotes complex conjugation.

Peak Signal-to-Noise Ratio. The PSNR for complex-valued fields we defined as:

$$\mathrm{cPSNR}(z_1, z_2) = 10 \cdot \log_{10} \left(\frac{\max(|z_1|)^2}{\mathrm{cMSE}(z_1, z_2)} \right)$$

Bits Per Pixel (BPP). The BPP measures compression efficiency. Given a discrete representation of a field z and its compressed encoding, the bits per pixel is defined as

$$\mathrm{BPP} = \frac{\text{Total number of bits used to encode } z}{\text{Number of spatial points in } z}.$$

Lower BPP values indicate higher compression, while higher BPP generally corresponds to more faithful reconstruction.

C.2 Traditional compression

In the following paragraphs we briefly describe how the traditional compressions were implemented.

ZFP Compression. ZFP (30) is a compression library for numerical arrays designed for fast random access. It partitions the data into small blocks (typically $4 \times 4 \times 4$ elements for 3D data) and transforms them into a decorrelated representation using an orthogonal block transform. The transformed coefficients are quantized according to a user-specified tolerance, then entropy-coded to produce a compact bitstream. High-speed random access and tunable compression ratios are possible, making ZFP a very common choice for scientific data storage.

We rearrange $\boldsymbol{f}$ into a 3D array as $((v_\parallel \times \mu), (s \times y), x)$ for ZFP block-based compression scheme (up to 3D), and compress with ZFP with a specified absolute error tolerance. The compressed representation is a compact byte representation. Reconstruction is performed by decompressing with ZFP and reshaping the output back to the original tensor layout.

Wavelet Compression. Discrete wavelet transform (DWT) is applied using the level 1 Haar wavelet, on both spatial and temporal information. The multi-dimensional array is decomposed into wavelets (coefficient and slices). To achieve lossy compression, coefficients are pruned based on a fixed threshold dependent on the desired compression ratio, effectively discarding small high-frequency components. Reconstruction is performed by inverting the DWT.

Principal Component Analysis Compression. $\boldsymbol{f}$ is reshaped into a 2D array $((v_\parallel \times \mu, s), (x \times y))$, by rearranging together the velocity space $v_\parallel, \mu$ with the field line s and the spatial coordinates x, y. PCA is applied on the flattened spatial components, retaining a fixed number of principal components dependent on the desired compression ratio ($N = 2$ for $1000\times$ from Table 2). The compressed representation consists of the principal components, the mean vector, and the explained variance. Reconstruction is achieved by projecting back to the original space, followed by reshaping to the original dimensions.

C.3 Neural fields

Neural fields are trained by representing the distribution function as a continuous signal, taking coordinates as inputs. A dataset consists, for a given simulation, of the 5D density function $\boldsymbol{f}$ at a specific timestep, and the 5D grid coordinates of each cell. Data normalization is applied both to the field values and to the coordinates.

A SIREN (43) model with $w_0^{\text{in}} = 1.0, w_0^{\text{h}} = 3.0, w_0^{\text{out}} = 3.0$, 64 hidden dimension, 5 layers, continuous sincos embedding for the coordinates and skip connections between the layers is optimized using AdamW (34), with cosine annealing learning rate scheduling decaying the learning rate from $1e-3$ to $1e-12$ and . Auxiliary optimizers can be used for additional integral losses, also with their scheduler that decays learning rate from $1e-5$ to $1e-12$. The neural field training loop iterates over batches of (2048) coordinates and field values. On a first pass of 20 epochs, the loss $\mathcal{L}_{recon}$ from Section 3.3 is fitted. Auxiliary integral losses are trained of such a pretrained model for 100 more epochs, with the whole 5D field as batch.

C.4 Autoencoders

The autoencoder and VQ-VAE baselines are both built on the 5D Swin Transformer (17). GELU activations are used (20). In particular, swin transformers are modified using the gated attention Qiu et al. (39) for stability. Relative Positional Bias (32) and RoPE (46) are used as positional encodings. Patch size is $(4, 8, 2, 5, 4)$ and window size is $(4, 1, 4, 9, 4)$. A stack of 4 swin blocks and 16 attention heads is applied after patching in the encoder and before unpatching in the decoder. Then, a single downsample level which further halves the resolution is applied before going in the bottleneck part. Here, a stack of 2 swin blocks is applied, which reduces the channels to increase the compression further. The decoder mirrors and inverts this.

The autoencoder has latent space of dimension 1024, with a bottleneck of 32, yielding a compression ratio of 1208. The model is trained with cMSE. For VQ-VAE, the latent dimension is also 1024, but the bottleneck effectively reduces to 1 with a codebook size of 8192, resulting in a much higher compression ratio of 38684. Training uses cMSE and VQ commit loss (50).

C.5 Extra results

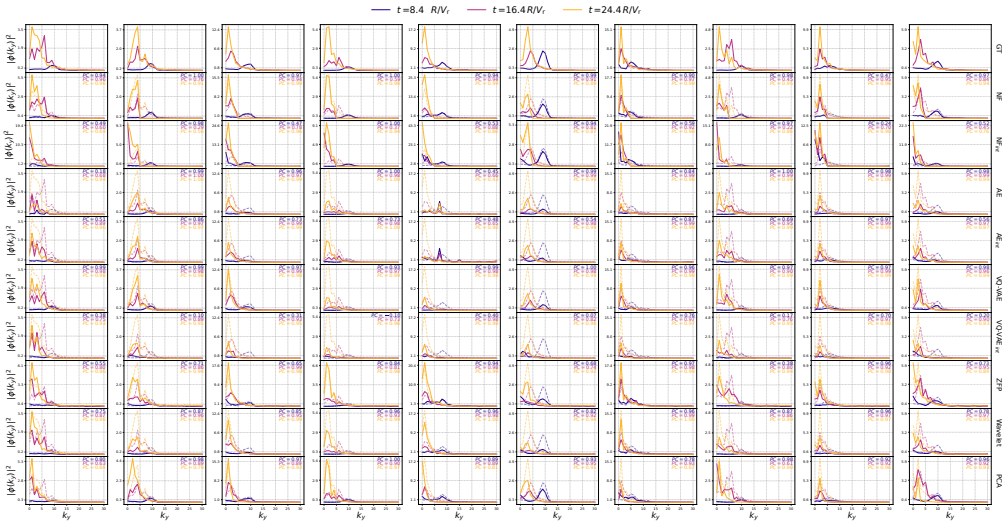

Figure 4: Extra models for the energy cascade (left Figure 3). The three time snapshots at $[8.4, 16.4, 24.4]R/V_{\mathrm{r}}$ are specifically sampled in the transitional phase where mode growth and energy cascade happens, before reaching the statistically stable phase. Visualized as the energy transfer from higher to lower modes as turbulence develops. Columns are different trajectories, rows are compression methods, lines of varied colors are the k_y^{spec} at specific timesteps, and transparent lines are respective ground truth.

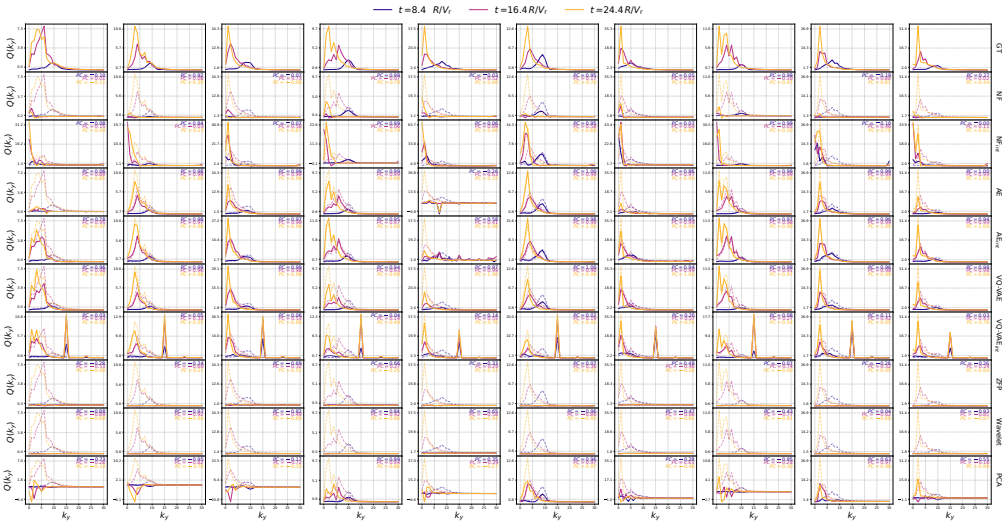

Figure 5: Extra models for the Q spectra (right Figure 3). The three time snapshots at $[8.4, 16.4, 24.4]R/V_{\mathrm{r}}$ are specifically sampled in the transitional phase where mode growth and energy cascade happens, before reaching the statistically stable phase. Visualized as the energy transfer from higher to lower modes as turbulence develops. Columns are different trajectories, rows are compression methods, lines of varied colors are the Q^{spec} at specific timesteps, and transparent lines are respective ground truth.