PNAS Nexus, 2023, 2, 1–14
https://doi.org/10.1093/pnasnexus/pgad015
Advance access publication 23 January 2023
Research Report

Explaining the physics of transfer learning in data-driven turbulence modeling

Adam Subel[a,1], Yifei Guan[a], Ashesh Chattopadhyay[a] and Pedram Hassanzadeh[a,b,*]

[a]Department of Mechanical Engineering, Rice University, Houston, TX 77005, USA
[b]Department of Earth, Environmental and Planetary Sciences, Rice University, Houston, TX 77005, USA
*To whom correspondence should be addressed: Email: pedram@rice.edu
[1]Present address: Courant Institute of Mathematical Sciences, New York University, 10012 NY, USA
Edited By: Yannis Yortsos

Abstract

Transfer learning (TL), which enables neural networks (NNs) to generalize out-of-distribution via targeted re-training, is becoming a powerful tool in scientific machine learning (ML) applications such as weather/climate prediction and turbulence modeling. Effective TL requires knowing (1) how to re-train NNs? and (2) what physics are learned during TL? Here, we present novel analyses and a framework addressing (1)–(2) for a broad range of multi-scale, nonlinear, dynamical systems. Our approach combines spectral (e.g. Fourier) analyses of such systems with spectral analyses of convolutional NNs, revealing physical connections between the systems and what the NN learns (a combination of low-, high-, band-pass filters and Gabor filters). Integrating these analyses, we introduce a general framework that identifies the best re-training procedure for a given problem based on physics and NN theory. As test case, we explain the physics of TL in subgrid-scale modeling of several setups of 2D turbulence. Furthermore, these analyses show that in these cases, the shallowest convolution layers are the best to re-train, which is consistent with our physics-guided framework but is against the common wisdom guiding TL in the ML literature. Our work provides a new avenue for optimal and explainable TL, and a step toward fully explainable NNs, for wide-ranging applications in science and engineering, such as climate change modeling.

Keywords: transfer learning, neural networks, subgrid-scale parameterization, turbulence modeling, climate modeling

Significance Statement

The use of deep neural networks (NNs) in critical applications such as weather/climate prediction and turbulence modeling is growing rapidly. Transfer learning (TL) is a technique that enhances NNs' capabilities, e.g. enabling them to extrapolate from one system to another. This is crucial in applications such as climate change prediction, where the system substantially evolves in time. For effective and reliable TL, we need to (a) understand physics that is learned in TL, and (b) have a framework guiding the TL procedure. Here, we present novel analysis techniques and a general framework for (a)–(b) applicable to a broad range of multi-scale, nonlinear dynamical systems. This is a major step toward developing interpretable and generalizable NNs for scientific machine learning.

Introduction

There are ever-growing efforts focused on using machine learning (ML), particularly the powerfully expressive deep neural networks (NNs), to improve simulations or predictions of nonlinear, multi-scale, high-dimensional systems. For example, in thermo-fluid sciences and in weather/climate modeling, a number of different approaches using NNs have shown significant promise for fully data-driven forecasting, subgrid-scale (SGS) closure modeling, and novel ways of solving partial differential equations (PDEs) [12, 3, 6, 11, 2, 1, 5, 10, 13, 4, 14, 7–9]. However, one major challenge facing such efforts is the inability of NNs, and more broadly ML techniques, to *generalize* out-of-distribution, i.e. to perform equally well when tested on a dataset whose distribution (or some measure of its statistics) is different from the training set [16, 15].[a] Some degree of such out-of-distribution generalization is essential for NNs to be practically useful in many applications. For instance, NN-based SGS closures (i.e. data-driven parameterizations) should work accurately for a range of climates to be useful for global warming projections. If this were not the case, once some parameters (e.g. sea-surface temperature or forcing) change, the data-driven closures may lead to unstable or inaccurate simulations [11, 18, 17]. Studies have found a similar challenge arising across thermo-fluid applications [22, 19, 20, 23, 21].

Transfer learning (TL) provides a powerful and flexible framework for improving the out-of-distribution generalization of NNs, and has shown success in various ML applications [24, 25, 16]. Consider an NN that is already trained on a large-enough number of training samples (M_{tr}) from a *base system* and makes predictions with sufficient out-of-sample accuracy. We hereafter refer to this network as a base NN (BNN). The goal of TL is to build a new NN

Competing Interest: The authors declare no competing interest.
Received: September 12, 2022. **Revised:** December 9, 2022. **Accepted:** January 12, 2023

from a BNN that works with similar accuracy for a *target system* whose statistical properties could be different from those of the base system. For instance, this could be because of a change in physical properties (e.g. in the context of turbulence, an increase in Reynolds number, *Re*) or in external forcing (e.g. in the context of climate change, a higher radiative forcing due to increased greenhouse gases). We refer to this network as a TLNN. In TL, a (usually small) number of the layers of the BNN are re-trained, starting from their current weights, with a *small* number of re-training samples from the target system (e.g. $M_{tr}/10$ or $M_{tr}/100$ samples). The TL procedure, if properly formulated (as discussed later), can produce a TLNN whose out-of-sample accuracy for the target system is comparable to that of the BNN, despite using only a small amount of re-training data from the target system.

In thermo-fluid sciences and weather/climate modeling, a few studies have reported such success with TL for SGS closure modeling and spatio-temporal forecasting [18, 28, 22, 21, 27, 29, 26]. For example, in data-driven closure modeling with a convolutional NN (CNN) for large-eddy simulation (LES) of decaying 2D turbulence, Guan et al. [21] showed stable and accurate *a posteriori* (online)[b] LES using only $M_{tr}/100$ re-training samples from a target system that had a 16× higher *Re* number. Aside from enabling generalization for one system when parameters change, TL can also be used to effectively blend datasets of different quality and length for training, e.g. a *large*, high-fidelity training set from high-resolution simulations and a *very small* but *higher*-quality re-training set from observations/experiments or much *higher*-resolution simulations [5, 32, 31, 30]. Such an application of TL in blending large climate model outputs and small observational datasets has shown promising results in forecasting El Niño–Southern Oscillation and daily weather [5, 32, 33]. Even further, TL has been suggested as a way to improve the training of physics-informed NNs, a novel PDE-solving technique [35, 34].

In the TL procedure, there is one critical decision to make: Which layer(s) to re-train? This is an important question, considering that the goal of TL is to find the best-performing TLNN given the constraint imposed by the limited availability of *re-training samples* from the *target system*. Finding the best layer(s) to re-train via trial-and-error can become intractable for deep NNs, given that hyperparameter tuning and *a priori* (offline) and *a posteriori* (online) tests would be needed for each trial (i.e. a combination of re-trained layers). So far, all of the aforementioned studies using TL for turbulence or weather/climate modeling have followed the conventional wisdom from the ML community [16, 36, 37], which is to re-train the *deepest*, i.e. near the output, layers (or have re-trained all layers or most layers in an ad-hoc fashion). The idea here, mainly developed based on experiments and analyses using static images and classification tasks, is that the shallow layers learn general features of images while the deep layers learn features specific to the images in a given training set [38]. Thus, for effective TL to an out-of-distribution set of images, these deepest layers are the best to re-train [16]. Following this idea of re-training, the deepest layers has yielded good results in the aforementioned studies on turbulence and weather/climate modeling, e.g. to generalize to canonical flows with 10–16 times higher *Re* numbers [21]. However, given the increasing interest in using TL, its broad applications in these areas, and the need for effective TL in more complex systems, the best practices and the learned physics should be understood and readily accessible. Specifically, the question of the best layer(s) for re-training should be more deeply investigated for the types of data and networks relevant to turbulence and weather/climate modeling applications. Here, we report on such an investigation for the first time.

In this paper, we use CNN-based non-local SGS closure modeling for LES of several setups of forced 2D turbulence as the test case. We first demonstrate the power of TL in enabling out-of-distribution generalization to 100× higher *Re* numbers, and even more challenging target flows. We further show that here, against the conventional wisdom in the ML literature, the *shallowest* layers are the best to re-train. Next, we leverage the fundamentals of turbulence physics and recent theoretical advances in ML to

1. explain what is learned during TL to a different turbulent flow, which is based around changes in the convolution kernels of the BNN after re-training to the TLNN, and these kernels' physical interpretation,
2. explain why the shallowest layers, rather than the deepest ones, are the best to re-train in these setups,
3. introduce a general framework to guide TL of similar systems based on a number of analysis steps that could be performed before re-training any TLNN.

While we use the SGS modeling of canonical 2D turbulence as the test case, the methods used for (1)–(2) and the framework in (3) can be readily applied to any other TL applications in turbulence or weather/climate modeling. More broadly, this framework can be used for TL applications beyond SGS modeling and for any multi-scale, nonlinear, high-dimensional dynamical systems.

2D turbulence: DNS and LES

The dimensionless governing equations of 2D turbulence in a doubly periodic square domain are:

$$\frac{\partial \omega}{\partial t} + \underbrace{\frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y}}_{\mathcal{N}(\omega,\psi)} = \frac{1}{Re}\nabla^2\omega - \underbrace{m_f \cos(m_f x) + n_f \cos(n_f y)}_{f(x,y)} - r\omega, \tag{1a}$$

$$\nabla^2 \psi = -\omega, \tag{1b}$$

where ψ is the stream-function, ω is the vorticity, and $\mathcal{N}(\omega, \psi)$ is the advection term. r is the linear drag coefficient and $f(x, y)$ is a time-independent external forcing at wavenumbers m_f and n_f. This system, with different combinations of f and r, is a fitting prototype for a variety of large-scale geophysical and environmental flows and has been widely used to test novel techniques including data-driven SGS closures [40, 21, 7, 41, 42, 39].

For direct numerical simulations (DNS), Eqs. 1a–1b are solved using a pseudo-spectral solver with high resolution (N_{DNS} collocation grid points in each direction), resolving all relevant spatio-temporal scales (see Materials and methods for the solver's details). Filtering Eqs. 1a–1b yields equations for LES (Eqs. 2–3). In the LES equations, an SGS term, $\Pi = \mathcal{N}(\overline{\omega}, \overline{\psi}) - \overline{\mathcal{N}(\omega, \psi)}$, arises and has to be explicitly represented in terms of the resolved flow $(\overline{\omega}, \overline{\psi})$ via an SGS closure. Here, $\overline{(\cdot)}$ denotes filtering and coarse-graining (see Materials and methods for details). The same pseudo-spectral solver, but with a lower spatio-temporal resolution (e.g. $N_{LES} = N_{DNS}/8$ and a 10× larger time step), is used to solve the LES equations (2–3). While the LES solver is computationally much cheaper, it requires an accurate closure for $\Pi(\overline{\omega}, \overline{\psi})$, a long-standing challenge in every discipline of science and engineering dealing with turbulent flows.

Here, to build data-driven closures, we train CNNs on filtered and coarse-grained DNS (FDNS) data[c]: The input of the CNNs is $(\bar{\psi}, \bar{\omega})$ and the output is Π (see Materials and methods for details).

Table 1. Physical and numerical parameters for the six different systems, which are divided into three cases, each with a base and a target system

System	Re	m_f	n_f	r	N_{DNS}	N_{LES}
Base (Case 1)	3.2×10^4	0	0	0	2,048	128
Target (Case 1)	1×10^4	4	0	0.1	1,024	128
Base (Case 2)	1×10^3	4	0	0.1	512	128
Target (Case 2)	1×10^5	4	0	0.1	2,048	128
Base (Case 3)	2×10^4	25	25	0.1	1,024	128
Target (Case 3)	2×10^4	4	4	0.1	1,024	128

See Fig. 1 for snapshots and some of the statistical properties of these distinctly different flows.

By changing Re, r, m_f, and n_f, we have created six distinctly different flows, divided into three cases, each with a base and a target system (Table 1 and Materials and methods). We have shown in previous studies that for various setups of 2D turbulence, CNNs trained on large training sets, or on small training sets with physics-constraints incorporated, produce accurate and stable data-driven closures in *a priori* (offline) and *a posteriori* (online) tests [21, 39]. These CNN-based closures were found to accurately capture both diffusion and backscattering, and to outperform widely used physics-based SGS closures such as the Smagorinsky, dynamic Smagorinsky, and mixed models in both *a priori* and *a posteriori* tests. In this paper, we focus on TL and addressing objectives (1)–(3) listed in the Introduction.

Closing the generalization gap using transfer learning

Before attempting to explain the physics of TL, we first show that TL enables our CNN-based SGS closures to effectively generalize between the base and target systems in each of the three cases. The first three rows of Fig. 1 demonstrate the differences in spatial scales between each pair of base and target systems. In Case 1, the base system is decaying turbulence while the target systems is forced turbulence. From the ω and Π snapshots, their spectra, and the kinetic energy (KE) spectra, it is clear that the two systems are different at both the large and small scales. As a results of these substantial differences across all scales, the LES of the target system using a BNN trained on the base system (BNN_{base}) produces a KE spectrum that does not agree with that of the target system's FDNS (the truth). This indicates that the BNN_{base} fails to generalize here, leading to a generalization gap that is the difference between the two KE spectra (most noticeable at wavenumbers, k, larger than 10). Note that comparing the KE spectra of FDNS and LES is the most common measure of the *a posteriori* (online) performance of SGS closures.

Similar failures of the BNN_{base} to generalize are seen for Cases 2 and 3, leading to large generalization gaps in the KE spectra. In Case 2, the base system has $Re=10^3$ and the target system has $Re=10^5$. This $100\times$ increase in the Re number leads to the development of more small-scale features in the target system, and changes the spectrum of Π in both large and small scales. In Case 3, the forcing of the base system is at wavenumber $m_f=n_f=25$, while the target system's forcing is at $m_f=n_f=4$. This decrease in forcing wavenumbers results in more (less) large-scale (small-scale) structures in the resolved flow, as seen in the spectra of both $\bar{\omega}$ and KE. This change in forcing wavenumber also leads to more large-scale structures in Π without any noticeable change in its small-scale structures. In short, Cases 1–3 represent 6 fluid flow systems that are different in terms of both the physics that drive the differences and the spatial scales of the resolved and SGS components.

In all three cases, TL closes the *out-of-distribution* generalization gap: LES of the target system using a TLNN (re-trained with $M_{tr}/10$ samples) produces a KE spectrum that matches that of the target system's FDNS. For the LES of the target system, the TLNN not only significantly outperforms the BNN_{base}, but is almost as good as the BNN trained on M_{tr} samples from the target system, BNN_{target} (see the insets in Fig. 1).

Impact of re-training layer(s) on accuracy

Fig. 1 shows the power of TL in closing the generalization gaps. These results also show that in contrast to the conventional wisdom, the best layers to re-train are not the deepest, but rather, the *shallowest* ones. For each case, we have explored all possible combinations of 1, 2, and 3 hidden layers for re-training; i.e. each layer, each pair of layers, and each 3-layer combination. Based on the correlation coefficient of the Π terms from FDNS and TLNN, which is the most common metric for *a priori* (offline) tests, we have found that for Cases 2 and 3, re-training layer 2 alone is enough to get the best performance. For Case 1, re-training layers 2 and 5 provides the best performance, although most of the gap can be closed by re-training layer 2 alone.

To better understand the effects of "re-training layer" selection in TL, Fig. 2 shows the offline and online performance of $TLNN^{\ell}$ as a function of an individual re-trained hidden layer ℓ. In Case 1, the offline performance of TLNNs substantially declines as deeper layers are used for re-training (top row). As a result, TL with deepest layers is completely ineffective; for example, LES with $TLNN^{10}$ is as poor as LES with BNN_{base}, leaving a large generalization gap in the KE spectrum for $k>10$ (bottom row). In contrast, LES with $TLNN^{2}$ has a KE spectrum that closely matches that of the FDNS and only has a small generalization gap for $k>40$ (as shown in Fig. 1, this gap is further closed when both layers 2 and 5 are re-trained). Similarly, in Case 3, the offline performance of TLNNs declines as ℓ increases. That said, in this case, TL with even the worst layer to re-train ($\ell=10$) is effective in closing the generalization gap in the online test. Still, LES with $TLNN^{2}$ is slightly better than LES with $TLNN^{10}$ (see the inset). In these two cases, there are substantial changes in the large scales of the inputs and outputs between the base and target systems (see the spectra of $\bar{\omega}$ and Π in Fig. 1). The offline results show a clear deterioration of the performance when moving from shallow to deep layers, which is due to the inability of the deeper layers to learn about changes in large scales during TL, as shown later.

In Case 2, the offline performance of TL is not a monotonic function of ℓ, though $\ell=2$ is still the best layer to re-train ($\ell=7$ is the worst), based on both offline and online results. The non-monotonicity emerges because changes between the base and target systems' $\bar{\omega}$ and Π occur predominantly at smaller scales (see their spectra in Fig. 1), which deeper layers are also able to learn during TL. For this case, as in Case 1, there is a noticeable difference in the online performance of the LES with TLNNs that use the best and worst performing re-trained layers.

The above analysis demonstrates that a poor selection of the re-training layer can lead to poor offline and/or online performance of the TLNN. This analysis also shows that in all three cases, re-training the shallowest layers consistently yields the best-performing TLNNs. This is in contrast to the conventional wisdom of TL, which is predominantly built on studies on classification of static images, which often do not have a broad continuous spectrum of spatial scales [16, 25, 43].

A spectral approach to interpreting transfer learning

Failure of deep layers to learn changes in large scales during transfer learning

To understand why different re-training layers lead to different TL performance, next, we conduct a spectral analysis of the CNNs in this section and the next one. The mathematical representation of CNNs is discussed in Materials and methods. Explained briefly, in our CNNs, inputs $\mathbf{u} = (\bar{\omega}, \bar{\psi})$ are passed through 11 sequential convolutional layers to predict outputs, Π (Fig. 3). The hidden layers each have 64 channels. The output of channel j of layer ℓ, called activation g^j_ℓ, is computed using Eq. 4: 64 kernels perform convolution on $g^j_{\ell-1}$ of each of the 64 channels, j, and the outcome of these linear operations is sent through a ReLU nonlinear activation function, σ. Fig. 3 shows examples of g^j_ℓ, which are 128×128 matrices (the size of the LES grid). Note that these 64^2 kernels in each hidden layer extract information from the activations through spatial convolution, and their weight matrices $W^{\beta,j}_\ell \in \mathbb{R}^{5\times5}$ are the main parameters that are learned during the training of a CNN.

In the second row of Fig. 3, we compare the all-channels-averaged Fourier spectra of activations of the last hidden layer $\langle \hat{g}^j_{10} \rangle$ from a fully trained BNN_{base}, TLNN^2, and TLNN^{10} ($\langle \cdot \rangle$ represents averaging over all channels and $\hat{\cdot}$ means Fourier transform). The spectrum of $\langle \hat{g}^j_{10} \rangle$ from TLNN^2 differs from that of the BNN_{base} at most wavenumbers including the small wavenumbers. This indicates that re-training layer 2 *can* account for differences in the output (Π) from the base and target flows at all scales, including the large scales. In contrast, the spectra from TLNN^{10} are almost the same as those from BNN_{base} at all scales (Case 1) or at large scales $k < 10$ (Cases 2 and 3). This indicates that re-training layer 10 *cannot* account for differences in the output from the base and target flows at large scales. Given that in all three cases there

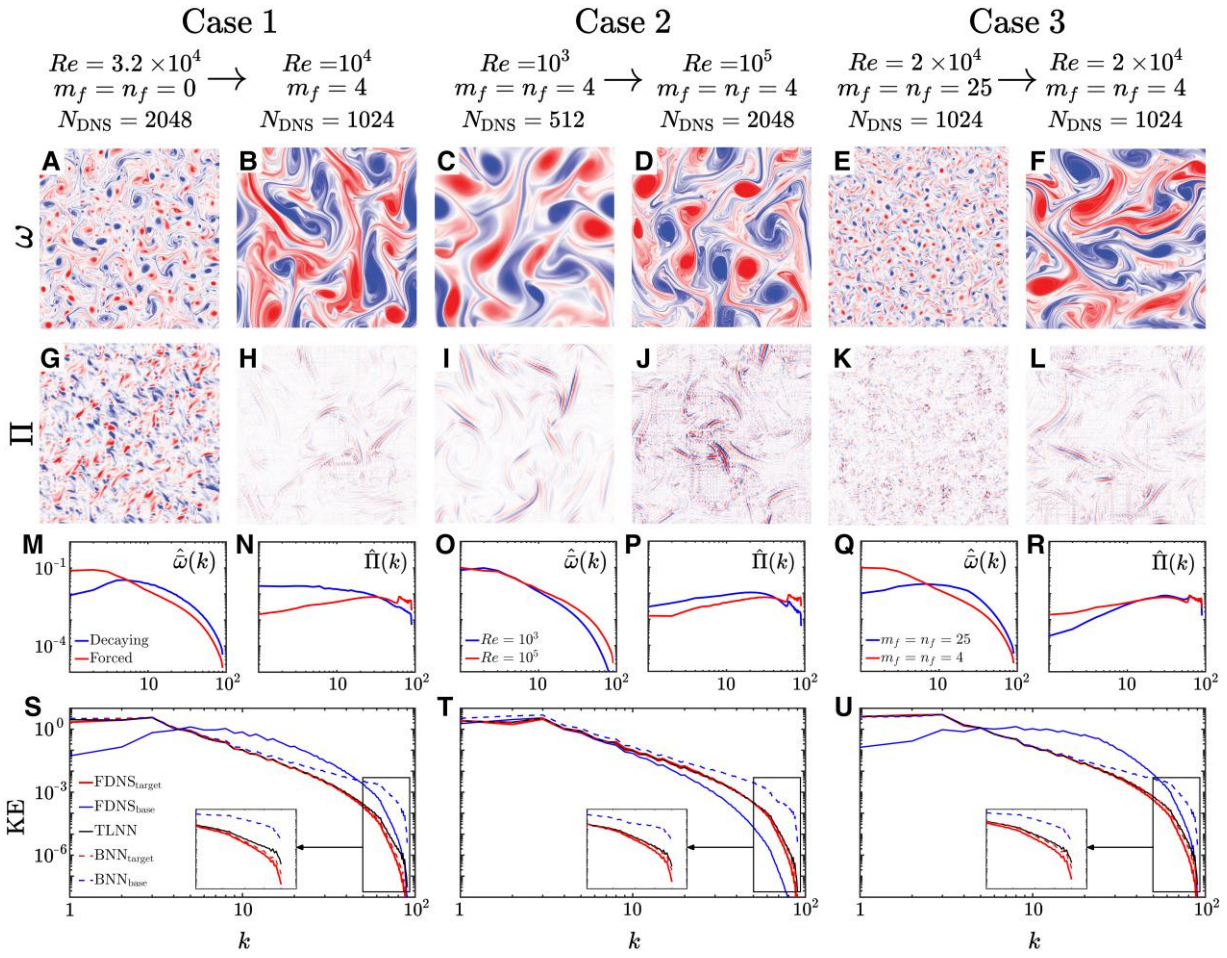

Fig. 1. Some comparisons between the base and target systems of the three cases (rows 1–3) and the ability of TL to close the generalization gaps in *a posteriori* (online) LES (row 4). Parameters of the six systems are listed in Table 1, and these cases are further described in Materials and methods. Each case consists of a base (left column) and a target (right column) system. The first and second rows show, respectively, the DNS snapshots of one of the inputs to the CNNs, ω, and the snapshots of the SGS terms, Π, the output of the CNNs (note that $N_{\text{LES}} = 128$ for all systems). These rows visualize the substantial differences in the length scales dominating the base and target systems in each case. To further demonstrate these differences in spatial scales, using the entire training sets and solid blue lines for base (top of legend) and solid red lines (bottom of legend) for target systems, we show the angle-averaged spectra of $\bar{\omega}$ (left) and Π (right) in the third row, and the KE spectra of FDNS in the fourth row. In these panels, the horizontal axis is wavenumber $k = \sqrt{k_x^2 + k_y^2}$, where k_x and k_y are the wavenumbers in x and y directions. The fourth row also shows the *out-of-sample* accuracy of the NN-based closures: The KE spectra are from *a posteriori* LES of the *target* systems using SGS closures that are BNNs trained on M_{tr} samples from the *base* systems (BNN_{base}, dashed blue lines) or from the *target* systems (BNN_{target}, dashed red lines), or from the TLNN (black lines) re-trained using $M_{tr}/10$ samples (see Materials and methods for details of TL). In all three cases, there is a large generalization gap (difference between the dashed blue and solid red lines), particularly for $k > 10$. In each case, TL closes this gap (black and solid red lines almost overlap for all k). Note that for the TL here, layers 2 and 5 are re-trained for Case 1, and layer 2 is re-trained for Cases 2 and 3 (see Section "Impact of re-training layer(s) on accuracy" and Fig. 2 for more discussions).

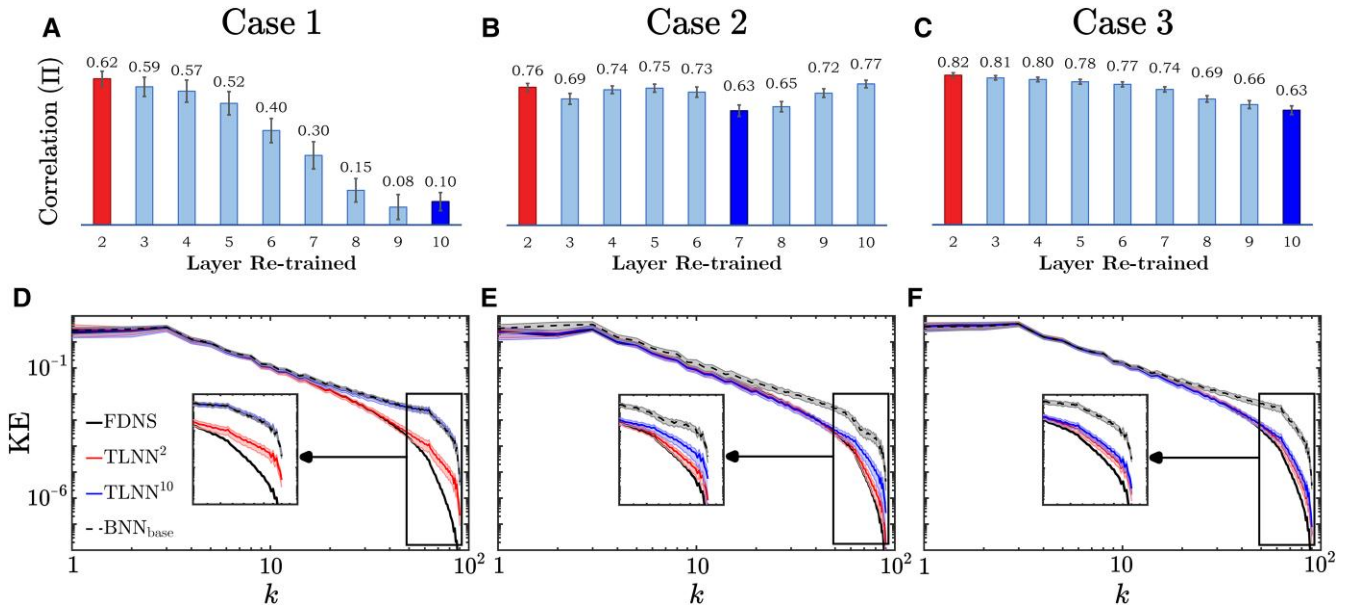

Fig. 2. Online and offline performance of TLNNs as a function of the individual re-trained layer. For each individual layer re-trained with $M_{tr}/10$ samples, the top row shows the most common measure of *a priori* (offline) accuracy of a SGS model: the correlation coefficient between Π from FDNS (truth) and from the TLNN. The vertical lines on the bar plots show uncertainty measured as the standard deviation calculated over 100 random samples from the testing set. The bottom row shows the KE spectra of the target systems' FDNS and the KE spectra from *a posteriori* (online) LES with BNN_{base} or $TLNN^{\ell}$, where ℓ indicates the re-trained layer. These KE spectra are calculated using five long integrations, each equivalent to $10^6\Delta t_{DNS}$. Shading shows uncertainty, estimated as 25th–75th percentiles of standard error calculated from partitioning each of the 5 runs into 10 sub-intervals. For each case, the best (worst) individual layer to re-train is shown in red (blue) in both rows. The best- and worst-performing layers here are chosen based on the online performance, i.e. how closely the KE spectrum matches that of the FDNS. Note that in Fig. 1, both layers 2 and 5 are re-trained during TL for Case 1, leading to a better TLNN with LES' KE spectrum matching that of the FDNS even at the highest wavenumbers. See Fig. S5 for the offline results of Case 3 with the base and target systems switched.

are large-scale differences in the Π terms between the base and target flows (Fig. 1), this analysis explains why re-training layer 10 (or other deep layers) leads to ineffective TL, while re-training layer 2 leads to the best TL performance.

To further understand what controls the spectra of g_ℓ^j, we have examined Eq. 8, which is the analytically derived Fourier transform of Eq. 4. As discussed in Materials and methods, this analysis shows that the Fourier spectrum of g_ℓ^j depends on the spectrum of $\hat{g}_{\ell-1}^j \in \mathbb{C}^{128\times128}$, the spectra of the weight matrices $\tilde{\hat{W}}_\ell^{\beta,j} \in \mathbb{C}^{128\times128}$ (and constant biases $\hat{b}_\ell^j \in \mathbb{R}$), as well as where linear activation $h_\ell^j(x, y) > 0$ (defined in Eq. 7). The latter is a result of the Fourier transform of the ReLU activation function, the only source of nonlinearity in the calculation of g_ℓ^j. In Fig. S1, we have compared the spectra of activations from layers 2 and 10 before and after applying the ReLU activation function. From this, we find that in all three cases, linear changes due to updating the weights substantially alter the spectra of the activations, while nonlinear changes only play a significant role in Case 1. These results (and further discussions in Materials and methods) suggest that a deeper insight into TL might be obtained by examining the spectra of the weight matrices, $\tilde{\hat{W}}_\ell^{\beta,j}$, and how they change from BNN_{base} to TLNN, as done next.

Spectral analysis of the kernels' weights

Before investigating how TL changes the spectra of kernels' weights, let us first look at the spectra from the BNN_{base} of the three cases. A close examination of $|\tilde{\hat{W}}_\ell^{\beta,j}|$ in different layers shows that the learned kernels are a combination of a number of known spectral filters. While visualizing all the 64^2 kernels in each layer is futile, we realize that the similarity across the spectra of many kernels allows us to meaningfully cluster them using the k-means algorithm. Fig. S2 presents the cluster centers (in Fourier space) for $\ell = 2$ and 10 for each case. This figure shows that the learned kernels are a combination of coherent low-pass filters (row 1), high-pass filters (row 8), as well as band-pass and Gabor filters. It should be pointed out that learning Gabor filters by CNNs has been reported in the past for a number of applications such as text recognition [44]. Even more broadly, the emergence of such filters for learning multi-scale, oriented, localized features has been reported in the sparse coding and vision literature [45].

Since deep CNNs contain a very large number of parameters ($O(10^6)$), it is often intractable to isolate the effect of each convolution kernel for either a BNN or TLNN. Moreover, investigating the learned convolution kernels in physical space ($W_\ell^{\beta,j} \in \mathbb{R}^{5\times5}$) does not lead to any meaningful physical understanding. Above, we show that examining the kernels in the spectral space ($\tilde{\hat{W}}_\ell^{\beta,j} \in \mathbb{C}^{128\times128}$) leads to physically interpretable insight into their role as spectral filters. Still, due to the large number of parameters and the impact of nonlinearities, it is currently challenging to understand the physics learned by the entire BNN. Fortunately, due to the over-parameterized nature of these deep CNNs, TL occurs in the *lazy training regime* [46]. In this regime, significant changes occur in only a small number of kernels, as shown below. This opens an avenue for *explaining what is learned in TL* through examining the spectra of the few kernels with the largest changes.

For each case, we quantify the change in each kernel by computing the Frobenius norm of the difference between $\tilde{\hat{W}}_\ell^{\beta,j}$ from the BNN_{base} and $TLNN^{\ell}$ for $\ell = 2$ and 10. As demonstrated in Fig. S3, in each case and each layer, there are a few kernels with substantial changes, much larger than the changes in the rest of the 64^2 kernels. Fig. 4 shows the spectra of the four most-changed

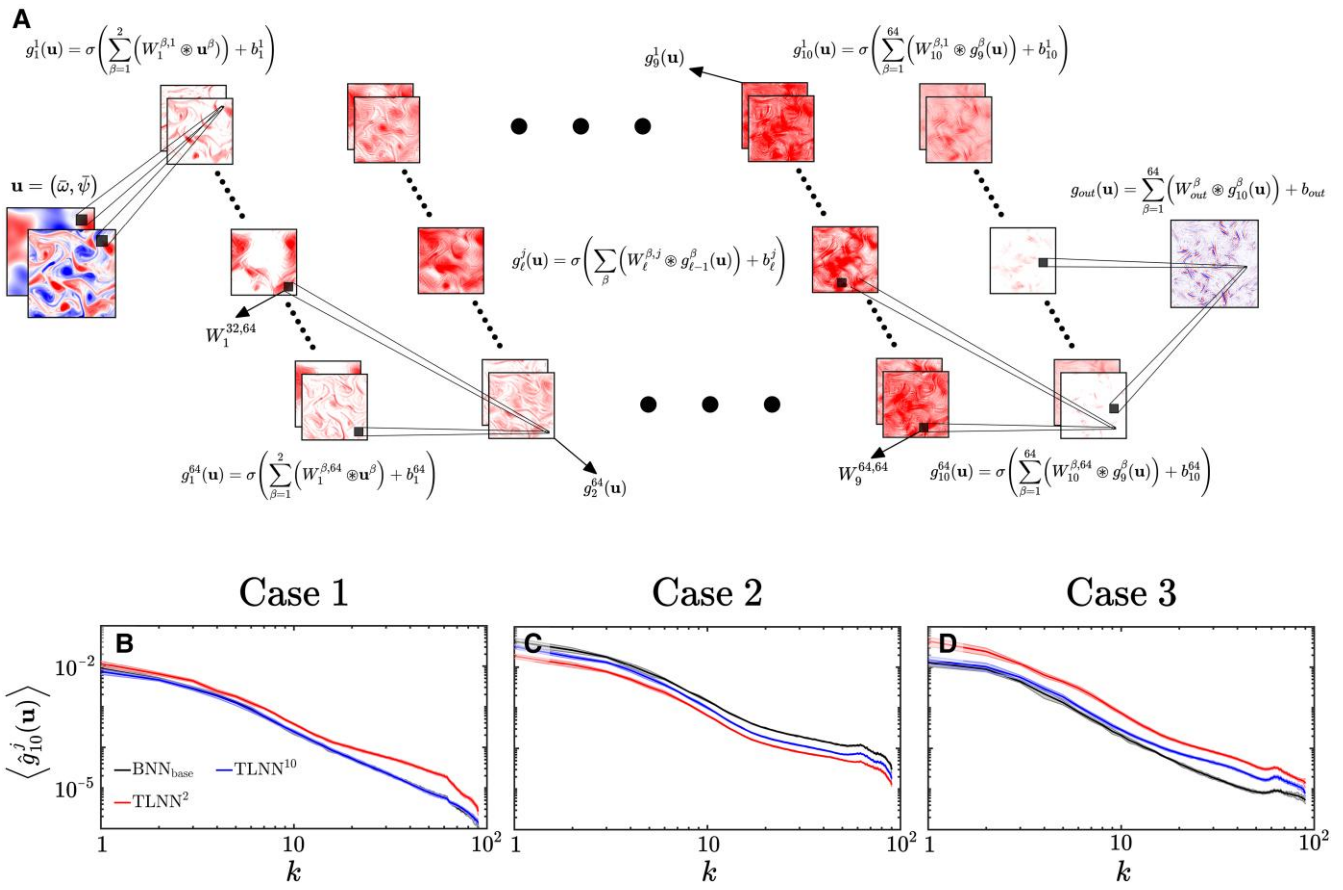

Fig. 3. The top row shows a schematic of the CNN architecture and its governing equations. Examples of activations $g_\ell^j \in \mathbb{R}^{128\times128}$ of some of the layers ℓ and channels j are shown as red shading (with σ being the ReLU nonlinear function, the values of these activations are all positive). Note that training a CNN means learning the convolution kernels' weight matrices $W_\ell^{\beta,j} \in \mathbb{R}^{5\times5}$ and biases' constant matrices $b_\ell^j \in \mathbb{R}^{128\times128}$ (for hidden layers $\ell = 2\ldots10$, $\beta \in \{1, 2\ldots64\}$ and $j \in \{1, 2\ldots64\}$). See Materials and methods for a detailed discussion of the CNN and its mathematical representation. In the bottom row, the effects of re-training layer 2 versus layer 10 on the Fourier spectrum of the averaged activation of the last hidden layer ($\ell = 10$) are compared (note that the output layer $\ell = 11$ has a linear activation function). The averaging is done over all channels, denoted by $\langle\cdot\rangle$. Shading shows uncertainty, estimated as 25th–75th percentiles of the averaged activation spectra computed with 20 random input samples.

kernels (due to TL) in layers 2 and 10 from BNN$_{base}$ and TLNN$^\ell$. We see that in all three cases, re-training layer 2 converts a few relatively inactive kernels into clear low-pass filters (one exception is the 4th most-changed kernel in Case 1, discussed later). In contrast, re-training layer 10 turns inactive or complex filters into other complex (often less coherent) filters, though some of them can be identified as band- or high-pass filters. The two panels on the right further show that the kernels learned in TL act as their spectra suggest: the new low-pass filter learned from re-training layer 2 produces activation g_2^j that is different from that of the BNN$_{base}$ (for the same input $\mathbf{u}$) only in large scales, while the most-changed kernel from re-training layer 10 (a high-pass filter) produces activation g_{10}^j that is different from that of the BNN$_{base}$ mainly in the small scales.

We remind the reader of the earlier discussion in this section: TL needs to capture changes in large scales of the output Π between the base and target systems, and the inability of the re-trained layer 10 to do so is the reason for the ineffectiveness of TLNN^{10}. Based on the above analysis, we can now explain the reason of this ineffectiveness (and the effectiveness of layer 2): layer 10 fails to learn new low-pass filters, which are essential for capturing changes in the large scales, especially at the end of the network right before the *linear* output layer. In contrast, layer 2 is capable of learning new low-pass filters to capture these changes in the large scales of the base and target systems' outputs. Admittedly, the nonlinearity and subsequent layers after $\ell = 2$ could impact the outcome of a low-pass filter, but it is possible to separate out the impact of the nonlinearity. Fig. 4 and Fig. S1 show the impact of the ReLU nonlinearity by comparing the spectrum of the activation before and after ReLU is applied. In Case 1, where the ReLU function plays an important role in changing the activations' spectra after TL, we find that in addition to low-pass filters, TLNN^2 also learns more complex filters, such as the 4th most-changed kernel in Fig. 4, that impact the sign of the linear activations, h_2^j.

The analyses presented so far provide answers to objectives 1-2 from the Introduction. To address objective 3 (develop a general framework to guide TL), we need to understand why layer 10 cannot learn the filters needed for the TL in these cases while layer 2 can. This question is investigated next by leveraging recently developed ideas in theoretical ML.

Loss landscapes: sensitivity of kernels to perturbations and re-training data

So far, we have presented *post-hoc* analyses, investigating changes in the spectra of activations and weights, as well as the learned physics, after a BNN$_{base}$ has been re-trained to obtain a TLNN. Here, we present a *non-intrusive* method for gaining insight into

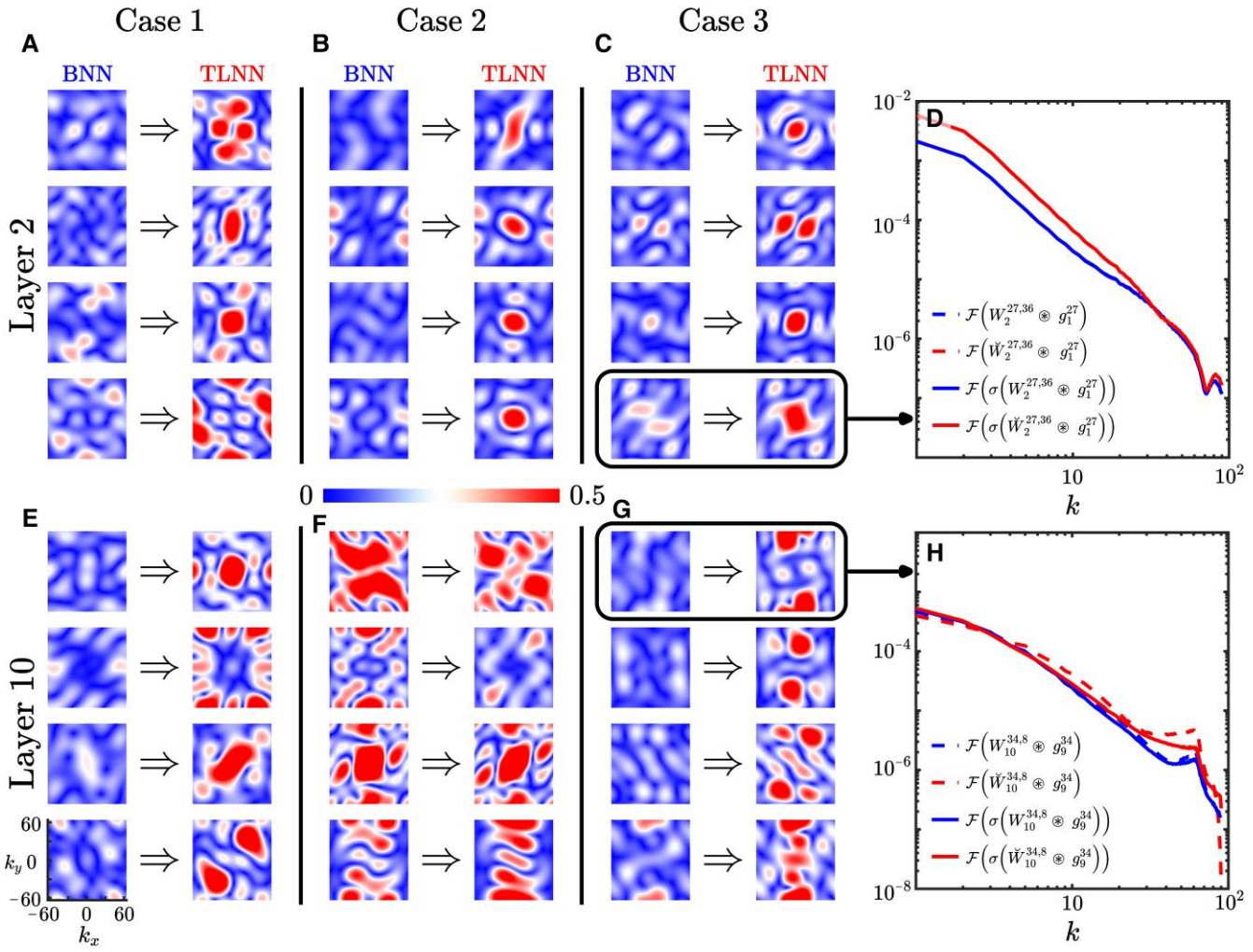

Fig. 4. The three left columns compare the Fourier spectra $|\tilde{W}_\ell^{\beta,j}|$ of the four convolution kernels that have changed the most between BNN_{base} and $TLNN^2$ (top row) and $TLNN^{10}$ (bottom row). The change in each kernel is quantified using the Frobenius norm $\|\mathcal{F}(\check{\tilde{W}}_\ell^{\beta,j}) - \mathcal{F}(\tilde{W}_\ell^{\beta,j})\|_F$, where $\mathcal{F}$ indicates the Fourier transform (Eq. 5) and $\check{\cdot}$ indicates that the weight matrix is from a TLNN (absence of $\check{\cdot}$ in this figure means that the matrix is from a BNN_{base}). The two panels on the right show examples of how changes in one kernel of layer 2 and one kernel of layer 10 affect the activations' spectra of layer 10 by comparing $\hat{g}_{10}^j$ from BNN_{base} (solid blue) with that from the $TLNN^{\checkmark}$ (solid red). We also show the activations before the application of ReLU nonlinearity σ with dashed lines. Note that the inputs to the networks ($\mathbf{u}$) are the same and from the *target* system. The top panel shows that the newly learned kernel in layer 2 substantially changes the activation in low wavenumbers ($k \leq 20$) without affecting the higher wavenumbers, as expected from a low-pass filter. Here, nonlinearity has little impact: the solid and dashed lines coincide. The bottom panel shows that the newly learned kernel in layer 10 only changes the activation at high wavenumbers and that in this case, the ReLU nonlinearity has a contribution.

which layers of a BNN_{base} are the best (or worst) to re-train for a given target system before performing any actual re-training. This analysis exploits the concept of "loss landscapes" [43, 47, 48] and examines, for a given CNN input $\mathbf{u}$, the sensitivity of the loss function $\mathcal{L}$ to perturbations of the weights (and biases) of the layer(s) to be re-trained. Training a deep CNN requires solving a high-dimensional non-convex optimization problem, for which the smoothness of the loss function can be a significant factor in the success of training. Previous studies [48, 43, 47, 49] show that even one- or two-dimensional approximations of the loss landscape can provide meaningful information about how easily a deep neural network, such as a CNN, can be trained. In this study, leveraging recent work in theoretical ML [43], we extend the application of loss landscape analysis to studying TL; see Materials and methods for more details and discussions about computing the loss landscapes.

Fig. 5 (rows 1 and 2) shows the loss landscape calculated for perturbations along two random directions in parameter space of shallow or deep layers for the BNN_{base} with data from the target system as the input. Fig. S4 presents the loss landscapes obtain using a second method (based on perturbations along the eigenvectors of the Hessian of the loss). These loss landscapes provide insight to indicate if a layer is receptive to change when re-trained with new data during TL. Two important characteristics of these landscapes are their convexity and the magnitude. Notably, the landscapes in row 1 (re-training layer 2, or 2 and 5) are both smooth and of much lower magnitude than those in row 2 (deep layers). For Case 1, we show results for combinations of two layers as this yields better performance than re-training a single layer, and this also demonstrates that the method is robust beyond perturbations of individual layers. This analysis indicates that these shallow BNN_{base} layers are easier to re-train for these target systems' data, and that the loss function will likely reach a better optimum during TL. This loss landscape analysis is consistent with our previous findings of $TLNN^2$'s ability ($TLNN^{10}$'s inability) to perform well in these TL tasks.

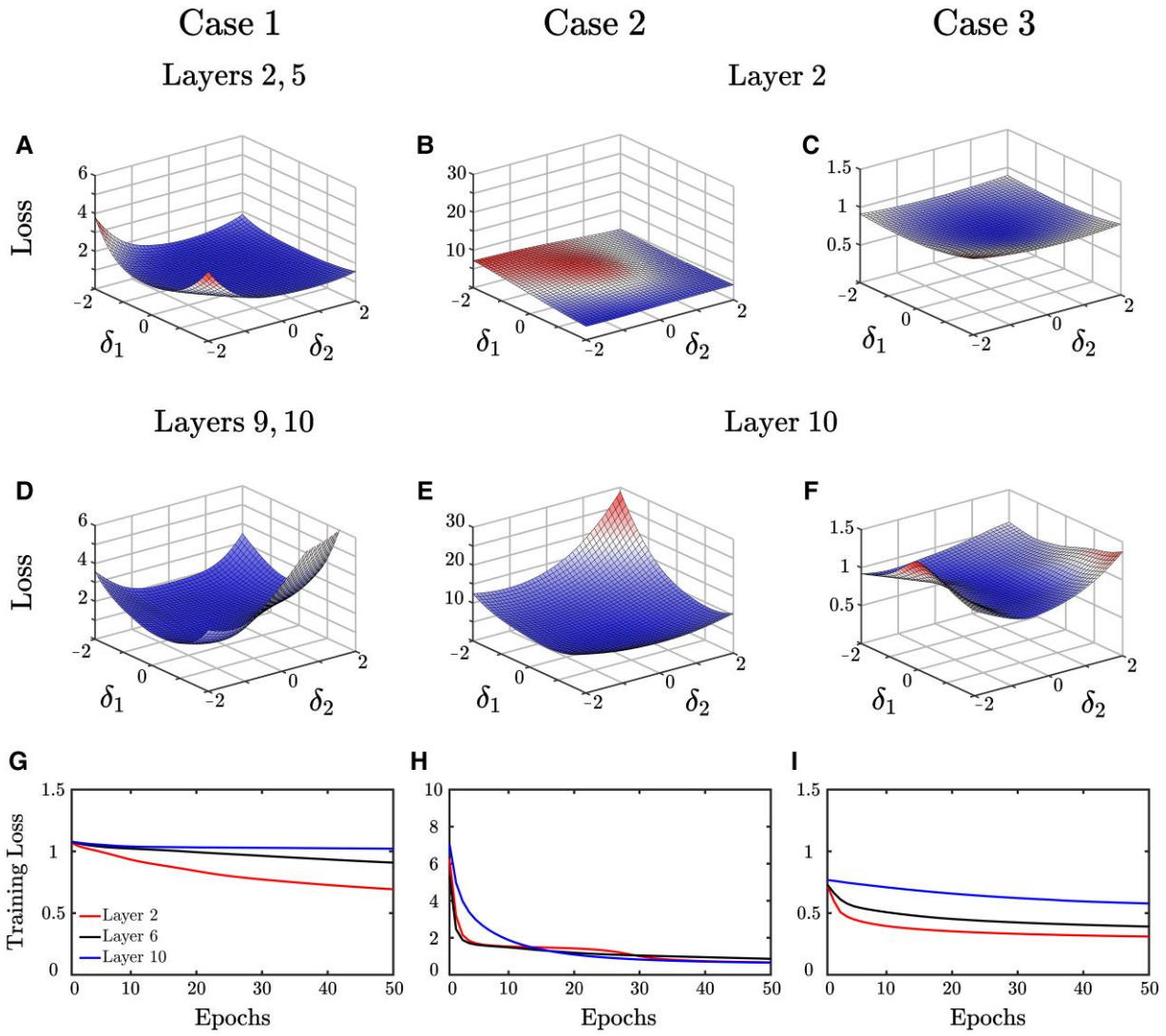

Fig. 5. The top two rows present the loss landscape $\mathcal{L}_{(\delta_1,\delta_2)}$ computed from Eq. 9. In row 1, the weights and biases of layers 2 and 5 (Case 1) or 2 (Cases 2 and 3) from the BNN_{base} are perturbed in two random directions by amplitudes δ_1 and δ_2; see Materials and methods for details. Similarly, in row 2, the deepest layers are perturbed. Row 3 shows the convergence of the training loss when individual shallow, middle, and deep layers are re-trained for TL. In all calculations, the inputs are from the *target* system.

Additionally, Fig. 5 (bottom row) shows how quickly the loss decreases as a function of the number of epochs during re-training layer 2, 6, or 10 of the BNN_{base} using the target system's data. For all three cases, TLNN^2 converges the fastest. This is a direct consequence of the structure of the loss landscapes shown in rows 1 and 2 of Fig. 5: landscapes obtained from perturbing layer 2 are more favorable for convergence (an absence of pathological non-convexities) as compared to the landscapes obtained from perturbing layer 10.

As a final note, we point out that the concept of "spectral bias" [50, 51] from theoretical ML suggests that layer 2, which converges faster, is learning the large scales while the slow-converging layer 10 is learning the small scales. This is consistent with the conclusions of our earlier analyses of the weights' spectra.

Discussion

In Section '2D turbulence: DNS and LES', we present a number of novel analysis steps, ranging from a) the most intrusive, computationally expensive ones to gain insight into the learned physics, to b) non-intrusive, inexpensive analysis, which can effectively guide TL for any new problem. For (a), we examine the BNNs' and TLNNs' activations and weights (done after re-training), revealing that the newly learned kernels are meaningful spectral filters, consistent with the physics of the base and target systems and their difference in the spectral space. To the best of our knowledge, this is the first full interpretation of CNNs' kernels in an application for turbulence or weather/climate modeling. For (b), we introduce a novel use of loss landscapes, shedding light on which layers are most receptive to learn the new filters in re-training.

These steps connect the spectral analysis of turbulent flows[d] and CNNs, and further connect them to the most recent advances in analyzing deep NNs. The above analyses show that the shallowest layers are the best to re-train here, and shed light on the learned physics and the inner workings of TL for these three test cases. Admittedly, some or all of these findings, in terms of learned physics and best layer(s) to re-train, are likely specific to these three cases, our specific NN architecture, and the SGS modeling application. However, the analysis methods we introduce or employ are all general and can be used for any base-target systems,

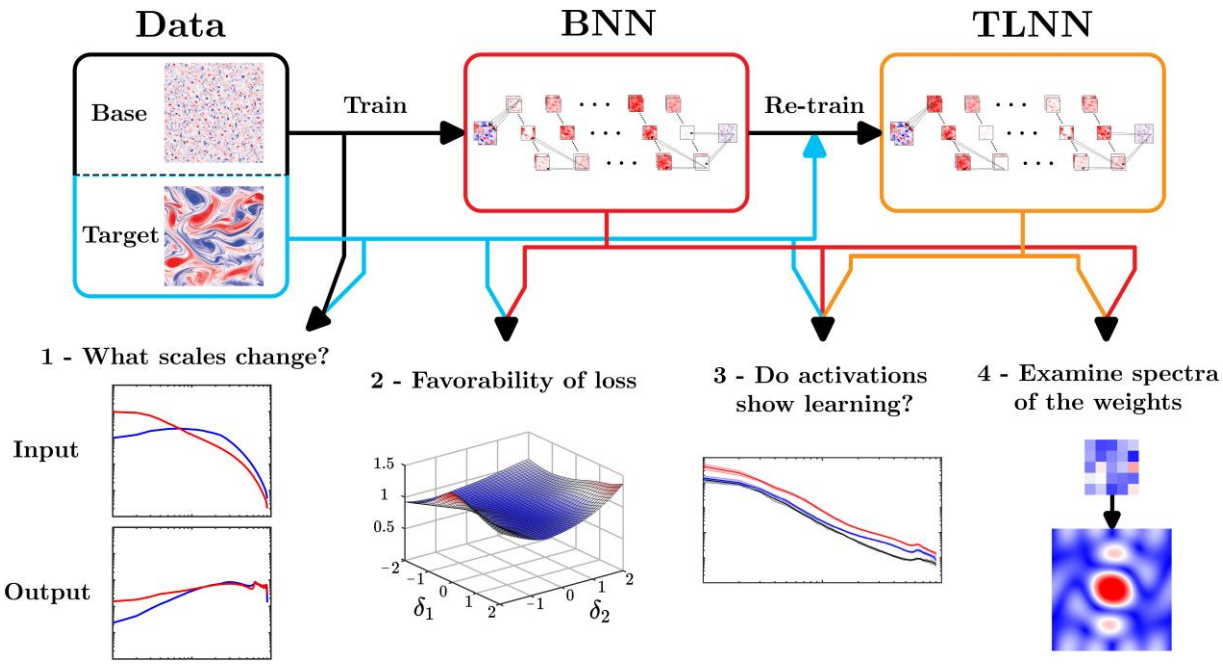

Fig. 6. Overview of the framework for guiding and explaining TL onto a new target system. The top row shows the steps of the TL process: acquiring a large amount of training data from the base system and a small amount from the target system, training a BNN_{base} using data from the base system, and re-training it using data from the target system to obtain a TLNN. On the bottom, we present the analyses involved in this framework, listed (left to right) in the order of when they should be used. The arrows indicate what is needed from each step of the TL process and the corresponding analyses. Here, the blue line represents data from the target system, the red line represents the trained BNN_{base}, and the orange line represents the re-trained TLNN.

applications (SGS modeling, data-driven forecasting, or blending training sets), and most CNN architectures.[e] Therefore, putting all these analysis steps together, below we propose a general framework for guiding and explaining TL, which we expect to benefit a broad range of applications involving multi-scale nonlinear dynamical systems.

The framework is shown schematically in Fig. 6. Assuming that we have a large number of training samples from the base system, an accurate BNN_{base} already trained on these samples, and a small number of re-training samples from the target system, the framework involves the following steps:

1. Compare the spectra of the input and output variables from the base and target systems. The three cases studied here have shown that the change of spatial scales between the base and target systems, particularly in the output variables, significantly impacts which layers are optimal for re-training.
2. Compute the loss landscapes of the BNN_{base} with target systems' data as various combinations of layers are chosen for re-training. Re-training layer(s) with favorable landscapes (smooth and small magnitudes) should be the first choices for TL. We further suggest examining the properly clustered weights' spectra of the BNN_{base} to see if they have clear interpretations as spectral filters.
3. Re-train a TLNN based on the outcome of Step 2. Examine the spectra of the activations from the re-trained layer(s) and the last hidden layer to see if the differences in the spatial scales identified in Step 1 are learned.
4. Examine the spectra of the most-changed kernels between BNN_{base} and TLNN. Investigate if the nature of the newly learned kernels (as spectral filters) is consistent with the outcome of Steps 1 and 3 in terms of spatial scales that need to be learned in TL.

Steps 1–2 are non-intrusive, inexpensive analyses that do not require any re-training, and will effectively guide Step 3, replacing expensive and time-consuming trial-and-error with many combinations of re-training layers. Steps 3–4 provide an explanation for what is learned in TL and act to validate decisions made based on Steps 1–2.

There are a few points about this framework that need to be further clarified. In general, turbulent flows have universal behavior in their smallest scales [52, 53] and vary in large scales due to forcing and geometry. This might seem to suggest that TL will always need to learn changes in large scales between a base and a target turbulent flow. This is not necessarily true, as even in Cases 1–2 here, in which the base and target flows are different in forcing and *Re* number, there are differences in small scales of Π too. Furthermore, in the broader applications of TL (e.g. in blending different datasets) and beyond just single-physics turbulent flows, there might be differences between the base and target systems at any scales. Step 1 is intended to identify these differences.

We also emphasize that currently there is no complete theoretical understanding of which layers of a CNN are better in learning what spatial scales. Our findings for Cases 1–3 and some other studies [43, 50] in the ML community suggest that the shallower layers are better in learning large scales. If further work confirms this behavior for a variety of systems and CNN architectures, then Steps 1–2 together would be able to even better guide TL in terms of the best layer(s) to re-train.

It should be noted that in more complex, an-isotropic, inhomogeneous systems (e.g. channel flows or ocean circulations), spectral analysis using other basis functions, such as Chebyshev or wavelets [54, 55], might be needed. Moreover, additional modifications of the spectral analysis component of the framework might be needed for some types of NN architectures, e.g. those

involving pooling layers, fully connected layers, or other activation functions. Recent work in the ML literature on spectral analysis of NNs, particularly on developing end-to-end analysis, could be leveraged in addressing these challenges [51, 56].

Aside from items (1)–(3) in the Introduction addressed in this study, another major question about TL is how much re-training data are needed to achieve a certain level of out-of-sample accuracy for the target system. Currently, there is no theoretical framework to answer this question, particularly for data from dynamical systems such as turbulent flows or the climate system. However, a few recent developments in the ML literature for TL error bounds of simple NNs (e.g. shallow or linear) could be leveraged as the starting point [57–59], and combined with extensive empirical explorations, may provide some insight into this critical question.

Finally, we point out that a number of recent studies have proposed improving out-of-distribution generalization via incorporating physics constraints into NNs (e.g. [60, 61]) or via data augmentation (e.g. [62, 63, 64]). The latter approach has shown promising results in image classification tasks, and could be potentially used in applications involving dynamical systems too. Incorporating physics has also shown promising results for specific applications; however, such an approach requires the existence of a physical constraint that is universal (e.g. a scaling law), otherwise, it could potentially deteriorate the performance of the NN. However, the availability of such constraints are very limited. In contrast, TL provides a flexible framework that beyond improving out-of-distribution generalization, is also broadly useful to blend disparate datasets for training, an important application on its own. Note that the aforementioned approaches can be combined with TL to possibly reduce the amount of re-training data.

To summarize, here we have presented the first full explanation of the physics learned in TL for multi-scale, nonlinear dynamical systems, and a novel general framework to guide and explain TL for such systems. This framework will benefit a broad range of applications in areas such as turbulence modeling and weather/climate prediction. Climate change modeling, which deals with an inherently non-stationary system and also involves combining various observational and model datasets, is an application that particularly needs TL, and can benefit from the framework proposed here.

Materials and methods

Numerical solvers for DNS and LES

We have performed DNS for all six systems used in this study (see Table 1 and below). In DNS, Eqs. 1a–1b are solved using a Fourier–Fourier pseudo-spectral solver with N_{DNS} collocation grid points and second-order Adams-Bashforth and Crank-Nicolson time-integration schemes with time step Δt_{DNS} for the advection and viscous terms, respectively. See Guan et al. [21, 39] for more details on the solvers and these simulations. For the base system in Case 1 (decaying 2D turbulence), following earlier studies [40, 21], the flow is initialized randomly using a vorticity field (ω_{ic}) with a prescribed power spectrum. Snapshots of (ω, ψ) in this system are obtained from 50–200τ, where τ is the initial eddy-turn-over time: $\tau = 1/\max(\omega_{ic})$. For the other five systems (forced 2D turbulence), once the randomly initialized flow reaches statistical equilibrium after a long-term spin-up, we take sequential snapshots of (ω, ψ) that are $1000\Delta t_{DNS}$ apart, in order to reduce the correlation between samples. We use the filtered and coarse-grained DNS data, referred to as FDNS data (details below), for training the CNN-based data-driven closures for Π and for testing their *a priori* (offline) and *a posteriori* (online) performance.

For LES, we solve Eqs. 2–3 employing the same numerical solver used for DNS, but with coarser grid resolutions ($N_{LES} = 128 < N_{DNS}$) and larger time steps ($\Delta t_{LES} = 10\Delta t_{DNS}$). To represent Π, a CNN-based closure that is trained on FDNS data is coupled to the LES solver.

Filtering and coarse-graining: LES equations and FDNS data

Filtering Eqs. 1a–1b yields the governing equations for LES [39, 53, 65]:

$$\frac{\partial \overline{\omega}}{\partial t} + \mathcal{N}(\overline{\omega}, \overline{\psi}) = \frac{1}{Re}\nabla^2\overline{\omega} - \overline{f} - r\overline{\omega} + \underbrace{\mathcal{N}(\overline{\omega}, \overline{\psi}) - \overline{\mathcal{N}(\omega, \psi)}}_{\Pi}, \tag{2}$$

$$\nabla^2\overline{\psi} = -\overline{\omega}. \tag{3}$$

In LES, only the large-scale structures ($\overline{\psi}$ and $\overline{\omega}$) are resolved using a coarser grid resolution (compared to DNS). The effects of the structures smaller than the grid spacing are included in the unclosed SGS term Π, which requires a closure in terms of the resolved flow, ($\overline{\psi}$, $\overline{\omega}$).

To obtain the FDNS data, we use the DNS snapshots of (ψ, ω), which are of size $N_{DNS} \times N_{DNS}$, to compute snapshots of $\overline{\psi}$, $\overline{\omega}$, and Π (defined in Eq. 2), where $\overline{(\cdot)}$ represents filtering and coarse-graining. The latter is needed to compute these variables on the LES grid (size: $N_{LES} \times N_{LES}$). Here, we use a Gaussian filter and then sharp spectral cutoff coarse-graining [21, 39]. For each system, the FDNS dataset is divided into completely independent training, validation, and testing sets [21, 39].

Cases 1–3: base and target systems

By changing Re, r, m_f, and n_f, we have created six distinct systems of 2D turbulence, which are grouped into three cases, each with a base and a target system (Table 1). Snapshots of ω and Π as well as the spectra of $\bar{\omega}$, Π, and KE of these systems are shown in Fig. 1 to demonstrate the rich variety of fluid flow characteristics among these systems, particularly between each case's base and target systems. Case 1 involves TL from decaying to forced 2D turbulence. From the ω and Π snapshots as well as their spectra shown in Fig. 1, it is clear that the two systems are different at both the large and small scales. The significant differences across all scales make this case the most challenging one, and result in the largest generalization gap as discussed in the main text.

Case 2 involves TL between two forced 2D turbulence systems: the base system has $Re = 10^3$ and the target system has a 100× higher Reynolds number ($Re = 10^5$), making this the largest extrapolation in Re using TL ever reported, to the best of our knowledge. The increase in Re adds more small-scale features in $\bar{\omega}$ (see the spectrum), and changes the spectrum of Π in both large and small scales. Case 3 involves decreasing the forcing wavenumbers of the system. Here, the base system has $m_f = n_f = 25$ while the target system has $m_f = n_f = 4$. This decrease in forcing wavenumbers, as expected, results in more (less) large-scale (small-scale) structures in the resolved flow; see the spectra of $\bar{\omega}$ and KE. Furthermore, more large-scale structures appear in Π without any noticeable change in the small-scale structures (see the power spectrum of Π).

Convolutional neural network and transfer learning

Building on the success of our earlier work [21, 39], to develop non-local data-driven SGS closure for each system, we train a CNN with input $\mathbf{u} = (\bar{\omega}(x, y), \bar{\psi}(x, y))$ to predict $\Pi(x, y)$ (output). These CNNs are built entirely from 11 sequential convolution layers, 9 of which are hidden layers each with 64^2 kernels of size 5×5 (note that these numbers are hyperparameters that have been optimized for this application to avoid underfitting or overfitting [21, 39]). The outputs of a convolutional layer are called activations. For channel j, of layer ℓ, the equation for activation $g_\ell^j \in \mathbb{R}^{N_{LES} \times N_{LES}}$ is:

$$g_\ell^j(\mathbf{u}) = \sigma\left(\sum_\beta (W_\ell^{\beta,j} \circledast g_{\ell-1}^\beta(\mathbf{u})) + b_\ell^j\right). \tag{4}$$

Note that $N_{LES} = 128$ for all systems (Table 1). Here, $\circledast$ represents spatial convolution and $\sigma(\cdot) = \max(0, \cdot)$ is the ReLU activation function (which is not present for the linear output layer, $\ell = 11$). $W_\ell^{\beta,j} \in \mathbb{R}^{5\times5}$ is the weight matrix of a convolution kernel, and $b_\ell^j \in \mathbb{R}^{128\times128}$ is the regression bias, a constant matrix. We have $\beta \in \{1, 2\ldots64\}$ and $j \in \{1, 2\ldots64\}$ for all layers with two exceptions: in the input layer ($\ell = 1$) $\beta \in \{1, 2\}$, and in the output layer ($\ell = 11$), $j = 1$, as the output is a single channel. The kernels' weights and biases together constitute the NN's trainable parameters, which we collectively refer to as $\theta \in \mathbb{R}^p$. Note that $g_{in} = g_0 = \mathbf{u}$ and $g_{out} = g_{11} = \Pi$.

A visualization of these networks as well as examples of activations in the hidden layers are presented in Fig. 3. An important distinction between these CNNs and traditional CNNs is that these do not include any max-pooling layers or dense layers such that they maintain the dimension of the input through all layers and channels in the network. Our earlier work and a few other studies have found such an architecture to lead to more accurate CNNs for SGS closures [21, 39, 66].

We train these CNNs using the Adam optimizer and a mean-squared-error (MSE) loss function $\mathcal{L}$. For BNNs, all their trainable parameters θ are *randomly* initialized, and each CNN is trained for 100 epochs using $M_{tr} = 2000$ samples from the training set of the base system.[f] Note that even when we use M_{tr} samples from the *training* set of the *target* system to train a CNN, we still call it a "BNN" for convenience (e.g. in Fig. 1). Subscripts on BNNs clearly indicate which system provided the M_{tr} training samples.

To appropriately train and evaluate the networks, for each of the six systems, we have created three independent training, validation, and testing sets from a long DNS dataset. To ensure independence, these subsets are chosen far apart and pattern correlations between $\mathbf{u}$ and between Π of samples are computed and found negligible. The training set is reserved solely for the actual training procedure, and the only metric calculated with this set is the MSE loss (during training) to assess the convergence of the network parameters, θ. The validation set is used to assess both convergence and overfitting during training: Alongside the training set, we compute the MSE loss on the validation set after each epoch to ensure that the network's performance is continuing to improve out-of-sample rather than overfitting. The testing set is used to evaluate the CNNs' *a priori* performance reported in Figs. 2–4. Furthermore, note that the FDNS data used in Figs. 1 and 2 are from the testing set of the corresponding system. No data from LES have been used during the training of any CNN.

To perform TL from a BNN, the weights and biases of the TLNN are initialized with those of the BNN. The layers to re-train are selected (trainable layers) and the remaining weights/biases are frozen (non-trainable layers). The TLNN is then re-trained using standard backpropagation and the same MSE loss function with $M_{tr}/10$ samples from the training set of the *target system*, updating the weights and biases of the trainable layers. The re-training continues until the loss plateaus (for TL, this happens at around 50 epochs), which helps avoid overfitting. Note that based on offline metrics such as the correlation coefficients for Π, we have not found any need for adjusting the hyperparameters such as the learning rate or adding additional layers between training a BNN and TLNN.

Spectral analysis of CNNs

The Fourier transform operator $\mathcal{F}$ is defined as

$$\hat{\cdot} = \mathcal{F}(\cdot), \quad \mathcal{F} : \mathbb{R}^{128\times128} \mapsto \mathbb{C}^{128\times128}. \tag{5}$$

To represent convolution as an operation in the spectral space, we first note that we can extend each kernel $W_\ell^{\beta,j} \in \mathbb{R}^{5\times5}$ to the full domain of the input by padding it with zeros, as done in practice for faster training [67], to obtain $\tilde{W}_\ell^{\beta,j} \in \mathbb{R}^{128\times128}$. Then, the convolution theorem yields

$$W_\ell^{\beta,j} \circledast g_{\ell-1}^\beta = \mathcal{F}^{-1}(\hat{\tilde{W}}_\ell^{\beta,j} \odot \hat{g}_{\ell-1}^\beta), \tag{6}$$

where $\odot$ is element-wise multiplication.

Next, we define linear activation h_ℓ^j, which contains all the linear operations in Eq. 4:

$$h_\ell^j = \sum_\beta (W_\ell^{\beta,j} \circledast g_{\ell-1}^\beta) + b_\ell^j. \tag{7}$$

Despite the nonlinearity of Eq. 4 due to the ReLU function, its Fourier transform can be written analytically. Using Eqs. 6 and 7 and the linearity of the Fourier transform we obtain

$$\begin{aligned} \hat{g}_\ell^j &= \sum_\alpha (e^{-i(k_x x_\alpha + k_y y_\alpha)}) \circledast \hat{h}_\ell^j \\ &= \sum_\alpha (e^{-i(k_x x_\alpha + k_y y_\alpha)}) \circledast \left\{\sum_\beta (\hat{\tilde{W}}_\ell^{\beta,j} \odot \hat{g}_{\ell-1}^\beta) + \hat{b}_\ell^j\right\}, \end{aligned} \tag{8}$$

where $(x_\alpha, y_\alpha) \in \{(x, y) \mid h_\ell^j(x, y) > 0\}$ and $i = \sqrt{-1}$. The term with sum over α is a result of the ReLU function and involves summing over grid points where $h_\ell^j > 0$ (note that this term is the Fourier transform of the Heaviside function). Also note that b_ℓ^j is a constant matrix, therefore, $\hat{b}_\ell^j$ is only non-zero at $k_x = k_y = 0$ (and is real). See [50, 51, 56] for more information and discussion about Fourier analysis of NNs.

Equation 8 shows that the spectrum of $\hat{g}_\ell^j$ depends on the spectrum of $\hat{g}_{\ell-1}^j$, the spectra of the weights $\hat{\tilde{W}}_\ell^{\beta,j}$ (and constant biases $\hat{b}_\ell^j$), and where $h_\ell^j > 0$ in the physical (grid) space. With TL, the weights and biases are updated, which changes their spectra as well as where $h_\ell^j > 0$. Understanding the full effects of all these changes on $\hat{g}_\ell^j$ is challenging. In Fig. S1, we have examined the spectra of activations of layers 2 and 10 from BNN_{base}, $TLNN^2$, and $TLNN^{10}$ before and after applying the ReLU activation function (i.e., compare the spectra of $\hat{h}_\ell^j$ and $\hat{g}_\ell^j$). This analysis shows that in all three cases, linear changes due to updating $\hat{h}_\ell^j$ substantially alter the spectra of the activations while nonlinear changes only play a significant role in Case 1. These results and Eq. 8 suggest that a deeper insight into TL might be obtained by investigating $\hat{\tilde{W}}_\ell^{\beta,j}$ and how they change from BNN_{base} to TLNN.

Calculating the loss landscape

Let us represent a CNN with input $\mathbf{u}$ and trainable parameters θ as $\mathcal{C}(\mathbf{u}, \theta)$. The MSE loss function of this CNN is a function of the output: $\mathcal{L}(\mathcal{C})$. The concept of loss landscape (of $\mathcal{L}$) has received much attention in recent years and is widely used to study the *training phase* of NNs [48, 47, 49]. Below, leveraging recent work in theoretical ML [43], we compute the loss landscape to study the *re-training phase* of NNs in order to gain insight into TL.

Suppose that $\theta_\ell \in \mathbb{R}^p$ are all the trainable parameters of a BNN_{base} from all layers ℓ. We define $\theta^*_L \in \mathbb{R}^{p^*}$ as the subset of parameters that are updated in TL, i.e. the weights and biases of the re-trained layer(s), L. Next, we follow two methodologies for constructing loss landscapes. In the first method, we follow Li et al. [48] and select two random direction vectors $\upsilon_1, \upsilon_2 \in \mathcal{R}^{p*}$ and normalize them with the 2-norm of θ^*. In the second method, we follow Yao et al. [68] and find the eigenvectors of the Hessian of $\mathcal{L}(\mathcal{C})$ computed with respect to θ^*_L. The first two eigenvectors with largest positive eigenvalues are chosen as υ_1 and υ_2.

Next, in both methods, we perturb θ^* along directions υ_1 and υ_2 by amplitudes δ_1 and δ_2, respectively ($\delta_1, \delta_2 \in [-2, 2]$ for method 1, $[-1, 1]$ for method 2). Finally, we compute

$$\mathcal{L}_{(\delta_1,\delta_2)} = \mathcal{L}(\mathcal{C}(\mathbf{u}_{target}, [\theta_{\ell \neq L} \;\; \theta^*_L + \delta_1\upsilon_1 + \delta_2\upsilon_2])) \tag{9}$$

to generate a 2D approximation of the loss landscape and plot the surface as a function of δ_1 and δ_2. Note that the input $\mathbf{u}$ is from the *target* system. Loss landscapes from the first (second) method are shown in Fig. 5 (Fig. S4).

In the context of TL, the shape of the loss landscape indicates how receptive the re-training layers, L, are to change for the new re-training samples from the target system. In practice, a *shallow*, *convex* landscape suggests that the network is in a favorable region of parameter space, and gradient descent will easily converge. Deviations from this in the form of pathological non-convexities or extremely large loss magnitudes can cause problems during training and prevent the network from converging to a useful optimum. See Li et al. [48] and Krishnapriyan et al. [47] for further discussions on the interpretation of loss landscapes for the common application where, in Eq. 9, $\mathbf{u}$ is from the base system and θ^* represent parameters still changing during the epochs of training.

Notes

a. Throughout this paper, we use "out-of-distribution" to indicate cases in which the training and testing datasets have different distributions. Furthermore, we use "out-of-sample" for accuracy computed using samples from a testing set that is completely independent from the training set, but has the same distribution.
b. Following the turbulence and climate literature [21, 65, 69], we use the terms "*a posteriori*" and "online" to refer to experiments/tests involving the data-driven closure *coupled* to the LES numerical solver. "*a priori*" and "offline" refer to experiments/tests involving the closure (e.g. the trained CNN) alone.
c. Whether the training FDNS data are from the base or target system or both is clearly explained for each analysis.
d. Spectral analysis has been the cornerstone of understanding turbulence physics since the pioneering work of Kolmogorov [52].
e. The weights' spectra analysis might have to be further modified for networks that involve dimension changes, e.g. via pooling layers. See the Discussions.
f. While $M_{tr} = 2000$ might seem like a small number of training samples, we are in fact here using a *big* training set, because these samples are chosen far apart to be weakly correlated, requiring a long DNS dataset (two million Δt_{DNS}). See Guan et al. [39] for further discussions about the big versus small training sets.

Acknowledgments

We thank Fabrizio Falasca and Laure Zanna for insightful discussions. We are grateful to three anonymous reviewers for helpful comments and suggestions.

Supplementary material

Supplementary material is available at *PNAS Nexus* online.

Funding

This work was supported by an award from the ONR Young Investigator Program (N00014-20-1-2722), a grant from the NSF CSSI program (OAC-2005123), and by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program. The authors also benefited form discussions at the KITP Program "Machine Learning and the Physics of Climate" supported by NSF grant PHY-1748958. Computational resources were provided by NSF XSEDE (allocation ATM170020) and NCAR's CISL (allocation URIC0004).

Authors Contributions

All authors designed the research and wrote the paper. A.S., A.C., and Y.G. contributed to the design of new analytic tools. A.S. performed the researched and analyzed the data.

Data availability

The data used for this work are available at https://zenodo.org/record/6621142. Codes used for transfer learning, testing, and analysis are available at https://github.com/envfluids/TL_for_SGS_Models.

References

1 Beck A, Flad D, Munz CD. 2019. Deep neural networks for data-driven LES closure models. *J Comput Phys*. 398:108910.
2 Bolton T, Zanna L. 2019. Applications of deep learning to ocean data inference and subgrid parameterization. *J Adv Model Earth Syst*. 11(1):376–399.
3 Brenowitz ND, Bretherton CS. 2018. Prognostic validation of a neural network unified physics parameterization. *Geophys Res Lett*. 45(12):6289–6298.
4 Brunton SL, Noack BR, Koumoutsakos P. 2020. Machine learning for fluid mechanics. *Annu Rev Fluid Mech*. 52:477–508.
5 Ham Y-G, Kim J-H, Luo J-J. 2019. Deep learning for multi-year ENSO forecasts. *Nature*. 573(7775):568–572.
6 Han J, Jentzen A, Weinan E. 2018. Solving high-dimensional partial differential equations using deep learning. *Proc Natl Acad Sci USA*. 115(34):8505–8510.
7 Kochkov D, *et al.* 2021. Machine learning–accelerated computational fluid dynamics. *Proc Natl Acad Sci USA*. 118(21): e2101784118.
8 Novati G, de Laroussilhe HL, Koumoutsakos P. 2021. Automating turbulence modelling by multi-agent reinforcement learning. *Nat Mach Intell*. 3(1):87–96.

9 Pathak J, *et al.* 2022. FourCastNet: a global data-driven high-resolution weather model using adaptive Fourier neural operators. arXiv, arXiv:2202.11214, preprint: not peer reviewed.
10 Raissi M, Perdikaris P, Karniadakis GE. 2019. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys*. 378:686–707.
11 Rasp S, Pritchard MS, Gentine P. 2018. Deep learning to represent subgrid processes in climate models. *Proc Natl Acad Sci USA*. 115-(39):9684–9689.
12 Schneider T, Lan S, Stuart A, Teixeira J. 2017. Earth system modeling 2.0: a blueprint for models that learn from observations and targeted high-resolution simulations. *Geophys Res Lett*. 44(24): 12–396.
13 Weyn JA, Durran DR, Caruana R. 2020. Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *J Adv Model Earth Syst*. 12(9): e2020MS002109.
14 Yuval J, O'Gorman PA. 2020. Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions. *Nat Commun*. 11(1):1–10.
15 Nagarajan V, Andreassen A, Neyshabur B. 2020. Understanding the failure modes of out-of-distribution generalization. arXiv, arXiv:2010.15775, preprint: not peer reviewed.
16 Yosinski J, Clune J, Bengio Y, Lipson H. 2014. How transferable are features in deep neural networks? arXiv, arXiv:1411.1792, preprint: not peer reviewed.
17 Beucler T, *et al.* 2021. Enforcing analytic constraints in neural networks emulating physical systems. *Phys Rev Lett*. 126(9): 098302.
18 Chattopadhyay A, Subel A, Hassanzadeh P. 2020. Data-driven super-parameterization using deep learning: experimentation with multiscale Lorenz 96 systems and transfer learning. *J Adv Model Earth Syst*. 12(11):e2020MS002084.
19 Chung WT, Mishra AA, Ihme M. 2021. Interpretable data-driven methods for subgrid-scale closure in LES for transcritical LOX/GCH4 combustion. *Combust Flame*. 239:111758.
20 Frezat H, Balarac G, Le Sommer J, Fablet R, Lguensat R. 2021. Physical invariance in neural networks for subgrid-scale scalar flux modeling. *Phys Rev Fluids*. 6(2):024607.
21 Guan Y, Chattopadhyay A, Subel A, Hassanzadeh P. 2022. Stable a posteriori LES of 2D turbulence using convolutional neural networks: backscattering analysis and generalization to higher Re via transfer learning. *J Comput Phys*. 458:111090.
22 Subel A, Chattopadhyay A, Guan Y, Hassanzadeh P. 2021. Data-driven subgrid-scale modeling of forced Burgers turbulence using deep learning with generalization to higher Reynolds numbers via transfer learning. *Phys Fluids*. 33(3):031702.
23 Taghizadeh S, Witherden FD, Girimaji SS. 2020. Turbulence closure modeling with data-driven techniques: physical compatibility and consistency considerations. *New J Phys*. 22(9):093023.
24 Tan C, *et al.* 2018. A survey on deep transfer learning. In: *International Conference on Artificial Neural Networks*. Springer. p. 270–279.
25 Zhuang F, *et al.* 2020. A comprehensive survey on transfer learning. *Proc of IEEE*. 109(1):43–76.
26 Goswami S, Kontolati K, Shields MD, Karniadakis GE. 2022. Deep transfer operator learning for partial differential equations under conditional shift. *Nat Mach Intell*. 4:1155–1164.
27 Guastoni L, *et al.* 2021. Convolutional-network models to predict wall-bounded turbulence from wall quantities. *J Fluid Mech*. 928: A27.
28 Inubushi M, Goto S. 2020. Transfer learning for nonlinear dynamics and its application to fluid turbulence. *Phys Rev E*. 102(4): 043301.
29 Yousif MZ, Yu L, Lim H-C. 2021. High-fidelity reconstruction of turbulent flow from spatially limited data using enhanced super-resolution generative adversarial network. *Phys Fluids*. 33(12): 125119.
30 Chattopadhyay A, Pathak J, Nabizadeh E, Bhimji W, Hassanzadeh P. 2022. Long-term stability and generalization of observationally-constrained stochastic data-driven models for geophysical turbulence. *Environ Data Sci*. 2:E1.
31 Mondal S, Chattopadhyay A, Mukhopadhyay A, Ray A. 2021. Transfer learning of deep neural networks for predicting thermoacoustic instabilities in combustion systems. *Energy and AI*. 5:100085.
32 Rasp S, Thuerey N. 2021. Data-driven medium-range weather prediction with a ResNet pretrained on climate simulations: A new model for weatherbench. *J Adv Model Earth Syst*. 13(2): e2020MS002405.
33 Hu J, *et al.* 2021. Deep residual convolutional neural network combining dropout and transfer learning for enso forecasting. *Geophys Res Lett*. 48(24):e2021GL093531.
34 Chakraborty S. 2021. Transfer learning based multi-fidelity physics informed deep neural network. *J Comput Phys*. 426:109942.
35 Karniadakis GE, *et al.* 2021. Physics-informed machine learning. *Nat Rev Phys*. 3(6):422–440.
36 Hussain M, Bird JJ, Faria DR. 2018. A study on CNN transfer learning for image classification. In: *UK Workshop on Computational Intelligence*. Springer. p. 191–202.
37 Talo M, Baran Baloglu U, Yıldırım Ö, Acharya UR. 2019. Application of deep transfer learning for automated brain abnormality classification using MR images. *Cogn Syst Res*. 54:176–188.
38 Zeiler MD, Fergus R. 2014. Visualizing and understanding convolutional networks. In: *European Conference on Computer Vision*. Springer. p. 818–833.
39 Guan Y, Subel A, Chattopadhyay A, Hassanzadeh P. 2023. Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate LES. *Physica D*. 443:133568.
40 Maulik R, San O, Rasheed A, Vedula P. 2019. Subgrid modelling for two-dimensional turbulence using neural networks. *J Fluid Mech*. 858:122–144.
41 Page J, Brenner MP, Kerswell RR. 2021. Revealing the state space of turbulence using machine learning. *Phys Rev Fluids*. 6(3): 034402.
42 Pawar S, San O, Rasheed A, Vedula P. 2023. Frame invariant neural network closures for Kraichnan turbulence. *Physica A Stat Mech App*. 609:128327.
43 Neyshabur B, Sedghi H, Zhang C. 2021. What is being transferred in transfer learning? arXiv, arXiv:2008.11687, preprint: not peer reviewed.
44 Goodfellow I, Bengio Y, Courville A. 2016. *Deep learning*. Cambridge (MA): MIT Press.
45 Olshausen BA, Field DJ. 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*. 381(6583):607–609.
46 Chizat L, Oyallon E, Bach F. 2019. On lazy training in differentiable programming. *Adv Neural Inf Process Syst*. 32:2937–2947.
47 Krishnapriyan A, Gholami A, Zhe S, Kirby R, Mahoney MW. 2021. Characterizing possible failure modes in physics-informed neural networks. *Adv Neural Inf Process Syst*. 34.
48 Li H, Xu Z, Taylor G, Studer C, Goldstein T. 2018. Visualizing the loss landscape of neural nets. arXiv, arXiv:1712.09913, preprint: not peer reviewed.

49 Mojgani R, Balajewicz M, Hassanzadeh P. 2023. Kolmogorov n–width and Lagrangian physics-informed neural networks: a causality-conforming manifold for convection-dominated PDEs. *Comput Methods Appl Mech Eng*. 404:115810.

50 Rahaman N, *et al.* 2019. On the spectral bias of neural networks. In: *International Conference on Machine Learning*. PMLR. p. 5301–5310.

51 Xu ZQJ, Zhang Y, Luo T. 2022. Overview frequency principle/spectral bias in deep learning. arXiv, arXiv:2201.07395, preprint: not peer reviewed.

52 Kolmogorov A. 1941. The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. *Cr Acad. Sci. URSS*. 30:301–305.

53 Pope SB. 2001. *Turbulent flows*. Cambridge: Cambridge University Press.

54 Bruna J, Zaremba W, Szlam A, LeCun Y. 2013. Spectral networks and locally connected networks on graphs. arXiv, arXiv:1312.6203, preprint: not peer reviewed.

55 Ha W, Singh C, Lanusse F, Upadhyayula S, Yu B. 2021. Adaptive wavelet distillation from neural networks through interpretations. *Adv Neural Inf Process Syst*. 34.

56 Xu ZQJ, Zhang Y, Luo T, Xiao Y, Ma Z. 2019. Frequency principle: fourier analysis sheds light on deep neural networks. arXiv, arXiv:1901.06523, preprint: not peer reviewed.

57 Lampinen AK, Ganguli S. 2018. An analytic theory of generalization dynamics and transfer learning in deep linear networks. arXiv, arXiv:1809.10374, preprint: not peer reviewed.

58 Kalan MM, Fabian Z, Avestimehr S, Soltanolkotabi M. 2020. Minimax lower bounds for transfer learning with linear and one-hidden layer neural networks. *Adv Neural Inf Process Syst*. 33: 1959–1969.

59 Wu X, Manton JH, Aickelin U, Zhu J. 2022. An information-theoretic analysis for transfer learning: error bounds and applications. arXiv, arXiv:2207.05377, preprint: not peer reviewed.

60 Beucler T, *et al.* 2021. Climate-invariant machine learning. arXiv, arXiv:2112.08440, preprint: not peer reviewed.

61 Kashinath K, *et al.* 2021. Physics-informed machine learning: case studies for weather and climate modelling. *Philos Trans R Soc A*. 379(2194):20200093.

62 Erichson NB, *et al.* 2022. Noisymix: boosting robustness by combining data augmentations, stability training, and noise injections. arXiv, arXiv:2202.01263, preprint: not peer reviewed.

63 Salman H, Ilyas A, Engstrom L, Kapoor A, Madry A. 2020. Do adversarially robust imagenet models transfer better? *Adv Neural Inf Process Syst*. 33:3533–3545.

64 Utrera F, Kravitz E, Erichson NB, Khanna R, Mahoney MW. 2020. Adversarially-trained deep nets transfer better: illustration on image classification. arXiv, arXiv:2007.05869, preprint: not peer reviewed.

65 Sagaut P. 2006. *Large eddy simulation for incompressible flows: an introduction*. New York: Springer Science & Business Media.

66 Zanna L, Bolton T. 2020. Data-driven equation discovery of ocean mesoscale closures. *Geophys Res Lett*. 47(17):e2020GL088376.

67 Mathieu M, Henaff M, LeCun Y. 2013. Fast training of convolutional networks through FFTs. arXiv, arXiv:1312.5851, preprint: not peer reviewed.

68 Yao Z, Gholami A, Keutzer K, Mahoney MW. 2020. Pyhessian: neural networks through the lens of the hessian. In: *2020 IEEE International Conference on Big Data (big data)*. IEEE. p. 581–590.

69 Frezat H, Le Sommer J, Fablet R, Balarac G, Lguensat R. 2022. A posteriori learning for quasi-geostrophic turbulence parametrization. *J Adv Model Earth Syst*. 14:e2022MS003124.