RESEARCH ARTICLE | FEBRUARY 15 2024

Predicting wind farm wake losses with deep convolutional hierarchical encoder–decoder neural networks

David A. Romero ; Saeede Hasanpoor ; Enrico G. A. Antonini ; Cristina H. Amon

APL Mach. Learn. 2, 016111 (2024)
https://doi.org/10.1063/5.0168973

Predicting wind farm wake losses with deep convolutional hierarchical encoder–decoder neural networks

Cite as: APL Mach. Learn. 2, 016111 (2024); doi: 10.1063/5.0168973
Submitted: 21 July 2023 • Accepted: 22 January 2024 •
Published Online: 15 February 2024

View Online Export Citation CrossMark

David A. Romero,[1] Saeede Hasanpoor,[1] Enrico G. A. Antonini[2] and Cristina H. Amon[1,a)]

AFFILIATIONS

[1] Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario M5S 3G8, Canada
[2] RFF-CMCC European Institute on Economics and the Environment (EIEE), Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC), Milan, Lombardy, Italy

[a)] **Author to whom correspondence should be addressed:** cristina.amon@utoronto.ca

ABSTRACT

Wind turbine wakes are the most significant factor affecting wind farm performance, decreasing energy production and increasing fatigue loads in downstream turbines. Wind farm turbine layouts are designed to minimize wake interactions using a suite of predictive models, including analytical wake models and computational fluid dynamics simulations (CFD). CFD simulations of wind farms are time-consuming and computationally expensive, which hinder their use in optimization studies that require hundreds of simulations to converge to an optimal turbine layout. In this work, we propose DeepWFLO, a deep convolutional hierarchical encoder–decoder neural network architecture, as an image-to-image surrogate model for predicting the wind velocity field for Wind Farm Layout Optimization (WFLO). We generate a dataset composed of image representations of the turbine layout and undisturbed flow field in the wind farm, as well as images of the corresponding wind velocity field, including wake effects generated with both analytical models and CFD simulations. The proposed DeepWFLO architecture is then trained and optimized through supervised learning with an application-tailored loss function that considers prediction errors in both wind velocity and energy production. Results on a commonly used test case show median velocity errors of 1.0%–8.0% for DeepWFLO networks trained with analytical and CFD data, respectively. We also propose a model-fusion strategy that uses analytical wake models to generate an additional input channel for the network, resulting in median velocity errors below 1.8%. Spearman rank correlations between predictions and data, which evidence the suitability of DeepWFLO for optimization purposes, range between 92.3% and 99.9%.

I. INTRODUCTION

Wind energy, which currently meets ~6% of global energy demand, is expected to significantly grow over the next two decades, reaching an estimated 33% share by 2050.[1] This expected growth highlights the need for systematic research efforts to improve wind energy efficiency, reduce its costs, mitigate environmental impacts, and increase equipment and infrastructure lifetime, among others.

The most significant factor affecting wind farm performance is the aerodynamic interaction between turbines due to wakes that develop, propagate, and expand downstream of each turbine. Given the expected expansion of wind energy generation capacity, combined wake effects from both individual wind farms and wind farm clusters may affect transport mechanisms in the atmospheric boundary layer (ABL) and, in turn, reduce overall energy generation.[2] In this context, improved understanding and multi-scale modeling of turbine wakes will be necessary to maximize the energy extraction and minimize environmental impact of large-scale exploitation of wind energy resources.

At present, wind farms are designed using a portfolio of wake models with different levels of fidelity, including analytical wake models and 3D turbulent Computational Fluid Dynamics (CFD) simulations for the wind farm[3] and the ABL.[2] Leveraging these models, the location, layout, and size of the farm can be designed to maximize energy extraction,[4,5] minimize infrastructure costs and land use,[6–8] and even reduce noise levels.[9] Wake models are also used for turbine-level and farm-level real-time control,[10–13] which has a significant impact on both energy generation and turbine lifetime.[14]

Computational Fluid Dynamics (CFD) is currently the most accurate method for predicting energy production in wind farms, especially in complex terrains.[15] However, full-scale, three-dimensional, turbulent flow simulations of wind farms and wind farm clusters are time-consuming and computationally expensive, typically requiring days of simulation time even in parallel computing environments[16] for a single simulation under nominal conditions. Computational costs are further increased when calculating the annual energy production (AEP) of wind farms, which requires many simulation runs under different boundary conditions corresponding to the site-specific statistical distribution of wind velocity. To overcome the computational cost of wind farm design and optimization, a suite of models with varying levels of fidelity,[17,18] such as physics-driven reduced order models[19,20] and data-driven surrogate models,[21,22] are used.

Machine Learning (ML) methods have been extensively used in the context of surrogate-based design optimization (SBDO)[23] to reduce the use of computationally expensive CFD simulations in parametric studies, design exploration, optimization, and sensitivity analysis.[24] This results in an overall reduction in the time and resources required for the design and optimization of complex systems. Among a variety of ML methods, state-of-the-art neural networks have recently received the most attention due to advances in architectures and associated training methods. In particular, neural networks have shown great potential in physics-based applications, i.e., predicting spatiotemporal physical fields (e.g., temperature, pressure, flow velocity, etc.) and derived quantities (e.g., shear stress, drag/lift forces, heat flux, etc.), based on low-resolution measurement data from a system of interest or a physical prototype, high-resolution data from computer simulations of a virtual prototype, or both.

Two main approaches have been used in the literature to leverage computer simulation data for training neural networks for physics-based applications. In the first approach, referred to in the literature as Physics-Informed Neural Networks (PINNs), spatiotemporal coordinates and boundary conditions are used as inputs to traditional feed-forward neural network architectures.[25,26] The networks are then trained to predict physical fields of interest by minimizing a modified loss function that may combine prediction errors with respect to training data and boundary conditions and residuals resulting from substituting the network predictions into the partial differential equations that govern the behavior of the system. To date, PINNs have been successfully demonstrated on canonical 1D and 2D boundary value problems,[27,28] external laminar fluid flow around arbitrary objects[29] and, with limited success, turbulent fluid flow.[30,31] The second approach uses image representations of the field variables and leverages neural network architectures commonly used in computer vision tasks. One example of this approach is Image-to-Image (Im2Im) regression, in which both the inputs and outputs of the neural network are images. For instance, an encoder–decoder neural network architecture was used[32,33] to generate image representations of the two components of the velocity field within a 2D heterogeneous porous media, using an image of the porosity field as an input to the network. Results showed that the Im2Im regression exhibited better prediction accuracy than PINNs. Notably, the authors obtained accurate predictions of the pressure field by integrating Darcy's Law based on the predicted velocity fields, even though images of the pressure fields were not used to train the network.

There have been several studies leveraging neural networks for wake modeling in wind farms. Ti *et al.*[34] used a set of 2000 single-layer, ten-neuron artificial neural networks, each network trained independently and in parallel to predict the velocity deficit and add turbulence intensity at a specific location downstream of a wind turbine, using as inputs the inflow velocity and turbulence intensity at hub height. Their results showed good agreement with both simulation and measurement data. An alternative approach focusing on dynamic wake models was proposed by Zhang and Zhao,[35] who used a long-short term memory (LSTM) recurrent neural network to predict the evolution of a reduced-order representation of the flow field created with proper orthogonal decomposition, using inflow velocity and the turbine yaw angle as inputs to the model. Pawar *et al.*[36] also used reduced order models to decrease the dimensionality of the flow field data, which in their case was generated exclusively with analytical wake models. In their approach, a neural network is used to predict the principal components of the reduced-order flow field, using as input the incoming velocity, turbulence intensity, and yaw angle. To handle data with different levels of fidelity, the authors layered two of their networks, the first mapping inputs to the principal components of the low-fidelity data and the second network to map these components to the principal components of the high-fidelity data. In another example,[37] a generative adversarial network (GAN) was trained to predict images of the 2D flow field around a single turbine under mean turbulent flow. The GAN used as inputs the turbine yaw angle and the incoming streamwise velocities at 32 uniformly distributed points upstream of the turbine and was trained with CFD-generated data to predict both streamwise and spanwise velocity fields. Results showed GANs could accurately predict both velocity components under a variety of yaw conditions.

In this paper, we propose a methodology for fast, accurate prediction of the flow field in wind farms for WFLO studies using an Im2Im regression paradigm. Specifically, we propose and optimize a deep convolutional hierarchical encoder–decoder neural network architecture, named DeepWFLO hereafter, that uses images of the turbine layout and undisturbed flow field as inputs and generates an image of the resulting flow field, taking into account wake generation, propagation, and interactions between multiple wakes. Therefore, the proposed approach can naturally handle wind farms with arbitrary layout arrangements, in contrast with previous work that focused on predicting the flow field behind a single turbine. Furthermore, we characterize how modeling decisions regarding spatial resolution, velocity encoding, and training loss function affect the performance of the proposed network architecture. Finally, we evaluate the effectiveness of the proposed architecture for the hierarchical integration of wind farm models of varying fidelity, combining closed-form analytical wake model predictions and CFD data through a variety of transfer learning strategies.

II. PROBLEM DEFINITION

In the remainder of this paper, we will focus on predicting the flow field within a wind farm, including the effect of turbine wakes, for the purpose of calculating the AEP to support wind farm design optimization and control. Namely, given

(a) a set of turbine locations, represented as a grayscale image of a 2D top-view of turbine locations on a rectangular wind farm terrain and
(b) the undisturbed (free stream) wind velocity within the wind farm terrain, encoded as a grayscale (one channel) or color (three channel) image,

the goal is to generate an image representing the streamwise velocity field within the wind farm, encoded as a grayscale or color image with a suitable mapping between wind velocity and image pixel intensities, and to calculate the AEP of the farm from that image.

Although our objective is to use this methodology in 3D applications for the design, optimization, and control of large scale wind farms in complex terrains, in this paper, we focus, without loss of generality, on evaluating the feasibility and performance of the proposed methodology based on flow predictions from analytical, closed-form wake models and 2D turbulent CFD simulations. In addition, we assume the wind direction is from left to right since any other wind direction can be modeled by rotating the layout (or its image representation) accordingly.

III. METHODS

A. Flow field predictions

The basis of this data-driven approach to the prediction of the flow field and power output in wind farms is a comprehensive dataset of 2D, turbulent flow simulations of wind farms with multiple turbines and under multiple incoming wind velocities. For the purpose of feasibility testing and problem characterization, however, we also use 2D analytical models for wind farm wakes to generate the required data at a lower computational cost.

Among several analytical wake models proposed in the literature, e.g., Refs. 20 and 38, in this work, we build on the Jensen[39] and Gaussian[19] wake models. Although the Jensen model is generally considered to be the least accurate of the analytical wake models, our rational for using it in this work is threefold. First, our focus in this work is to generate a dataset that captures the main features of the phenomena of interest, namely wake propagation and wake interactions. Such a dataset, even if it has low fidelity, allows us to assess the feasibility of using the proposed encoder–decoder networks to capture this behavior accurately. Second, having a low-fidelity wake model allows us to assess the performance of transfer learning strategies to provide a framework for integrating predicting models with different levels of fidelity, in this case, using the lowest-fidelity analytical wake model and high fidelity CFD simulations. Finally, using the Jensen model allows for easier comparison with previous work in the wind farm layout optimization (WFLO) literature, which has primarily relied on the Jensen model.

In the remainder of this section, we briefly describe the models used for flow field prediction. Detailed descriptions of these models can be found in the cited literature and, for the CFD model, in the supplementary material.

1. Jensen wake model

Jensen's model assumes that the wake region expands linearly in the downstream direction and that the velocity profile inside the wake in the spanwise direction is uniform and transitions sharply to the freestream wind velocity at the wake boundary; this has been typically referred to as a "top hat" profile. The Jensen model uses the Betz Law to relate wind velocity before and after the turbine rotor. Under the assumption of turbines operating at maximum power coefficient, the wind velocity at any point downstream of the turbine can be calculated as

$$u = u_0\left(1 - \frac{2}{3}\left(\frac{r_r}{r_r + \alpha x}\right)^2\right), \tag{1}$$

where α is the wake decay constant, calculated as

$$\alpha = \frac{0.5}{\log \frac{z}{z_0}}, \tag{2}$$

where z is the hub height and z_0 is the roughness length of the terrain. For any points inside the wind farm domain that are under the effect of multiple wakes, the combined effect of the wakes is calculated as the sum of squared velocity deficits,[40] namely

$$u(x) = u_0\left[1 - \sqrt{\sum_{i=1}^{n}\left(1 - \frac{u_i(x)}{u_0}\right)^2}\right]. \tag{3}$$

2. Gaussian wake model

The Gaussian wake model assumes that, at any downstream position inside the wake of a turbine, the wind speed deficit profile is Gaussian-shaped. Therefore, the wind speed deficit becomes a function of both axial and radial positions with respect to the turbine. By assuming self-similarity in the wake, the normalized wind speed deficit can be expressed as

$$\frac{\Delta U}{U_\infty} = C(x)\exp\frac{r^2}{2\sigma^2}, \tag{4}$$

or alternatively,

$$U_w(x,r) = U_\infty\left(1 - C(x)\exp\frac{r^2}{2\sigma^2}\right), \tag{5}$$

where σ is akin to a standard deviation, i.e., a measure of the spread of the Gaussian-shaped profile, $U_w(x,r)$ is the wake velocity, $C(x)$ is the maximum normalized velocity which occurs at the center of the wake, and (x,r) are streamwise and radial coordinates relative to the turbine hub and centerline, respectively.

By neglecting viscous and pressure terms in the mass and momentum equation and using $T = 0.5C_T\rho A_0 U_\infty^2$ to calculate the total thrust force acting on the turbine, with A_0 being the swept area by the turbine blades, and C_T the thrust coefficient for which the power is maximized ($C_T = 8/9$), the following equation can be obtained for the normalized velocity deficit:

$$\Delta U/U_\infty = \left(1 - \sqrt{1 - \frac{C_T}{8(k^* x/d_0 + \varepsilon)^2}}\right) \times \exp\left(-\frac{1}{2(k^* x/d_0 + \varepsilon)^2}\left[\left(\frac{z - z_h}{d_0}\right)^2 + \left(\frac{y}{d_0}\right)^2\right]\right), \tag{6}$$

where $k^* = \frac{\partial \sigma}{\partial x}$ is the growth rate of the wake spread and was found to range from 0.023 to 0.055, based on the calibration of this model with respect to LES results obtained using surface roughness values in the range 0.00005–0.5 m. In Eq. (6), $\varepsilon = \sigma(x = 0)$ is the wake spread immediately after the turbine and can be calculated as

$$\varepsilon = 0.25\sqrt{\beta}, \tag{7}$$

where

$$\beta = \frac{1}{2}\frac{1 + \sqrt{1 - C_T}}{\sqrt{1 - C_T}}. \tag{8}$$

3. Computational fluid dynamics model

The governing equations of our CFD simulations are the Reynolds-averaged Navier–Stokes equations for incompressible, steady, turbulent flows, where the transport equations of turbulence quantities are closed using the shear-stress transport (SST) $\kappa - \omega$ turbulence model. In these simulations, the wind turbines are modeled using actuator disks, and flow conditions (pressure and velocity) are specified on the four boundaries of the 2D domain. OpenFOAM is used to solve the model equations on a high performance computing cluster, with each simulation taking ~30 min on a single 2.1 GHz Intel Xeon processor with 24 cores. The testing and validation of our CFD model were conducted in previous studies with geometric, operating, and atmospheric conditions consistent with those used in the present work.[42,42] Further details of the CFD model can be found in the supplementary material.

4. Annual energy production (AEP)

Based on the velocity field calculated with any of the models described earlier, the annual energy production (in kW h) of the wind farm can be approximated as

$$AEP = 8766\sum_{i=1}^{k}\sum_{d\in D}\frac{1}{3}u_{id}^3 p_d, \tag{9}$$

where i is an index over the number of turbines, d is an index over the magnitude and direction of wind velocity states, p_d is the probability of each wind state, 8766 is the number of hours in a year, and u_{id} is the effective velocity at the turbine location in ms^{-1}. This effective velocity is directly extracted from the calculated flow field in the case of analytical or computational wake models. In contrast, when using the proposed image-based approach, we extract the effective velocities at the turbine sites from the flow field image using an image mask corresponding to the turbine locations. The resulting masked image contains pixel intensities equal to zero at all locations except where turbines are located, wherein the pixel level intensities correspond to the normalized flow velocity. These pixel-level intensities are denormalized and used with Eq. (10) to calculate the AEP.

B. DeepWFLO: Image-to-image regression with encoder–decoder neural networks

To enable fast, accurate, real-time prediction of wind flow in large-scale wind farms, we propose DeepWFLO, a deep convolutional hierarchical encoder–decoder neural network to predict the streamwise velocity field inside wind farms with arbitrary turbine layouts using an Im2Im approach. Due to its encoder–decoder architecture, DeepWFLO conveniently combines the tasks of nonlinear, multi-scale encoding of inputs into a lower-dimensional feature space and surrogate modeling from that space to the outputs into a single task of supervised learning. In what follows, we assume that the system of partial differential equations (PDEs) governing fluid flow in the wind farm is solved over a discrete set of spatial locations $\mathscr{X} = \{\mathbf{x}_i, \ldots, \mathbf{x}_k\}$, $\mathscr{X} \in \mathbb{R}^{d_x}$, where $\mathbf{x}_i$ is a vector of spatial coordinates of dimension d_x and k is the number of discrete grid locations.

DeepWFLO considers as inputs (1) the source term $f(\mathbf{x})/\rho$ of the Navier–Stokes PDE that models the interaction between the turbines and the flow field and (2) the undisturbed wind velocity $\mathbf{u}_\infty(\mathbf{x})$, representing the flow field in the wind farm terrain

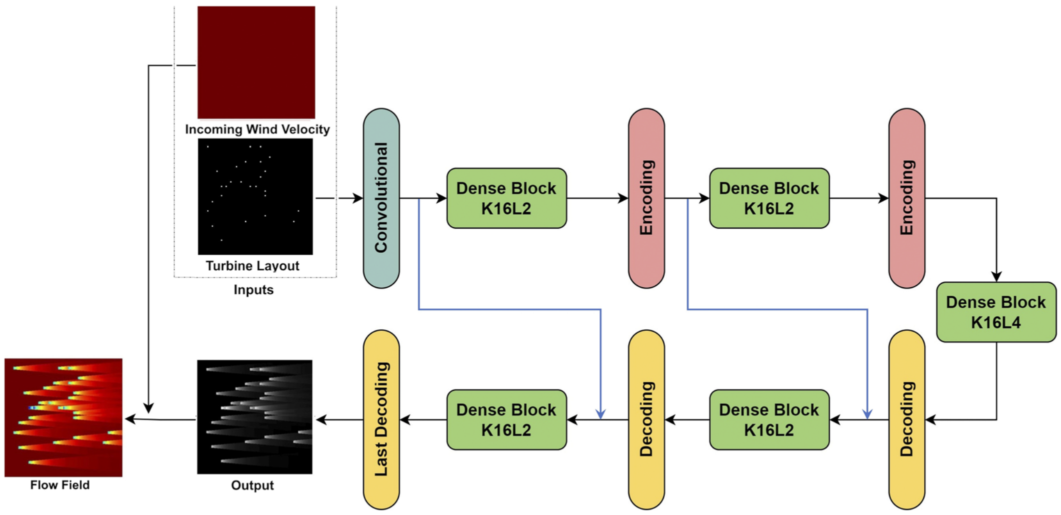
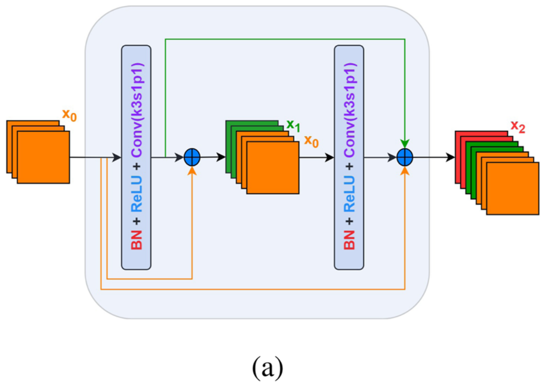


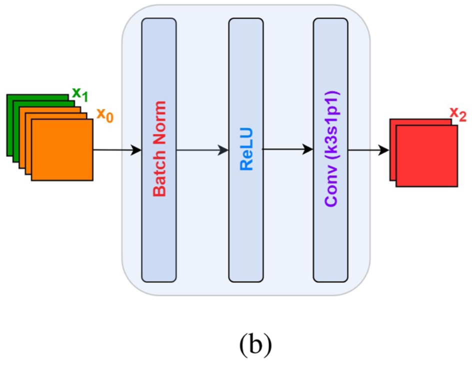
FIG. 1. Sample DeepWFLO architecture, used as the baseline model in this paper. Blue arrows indicate hierarchical connections, which are absent in DenseED networks.

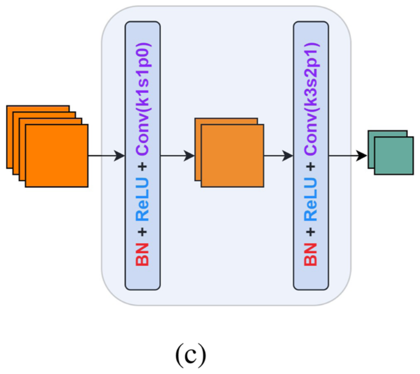
with the turbines absent. Without loss of generality, we assume that all quantities of interest are represented as computer images of size $H \times W$, i.e., over a regular 2D grid with $k = H \times W$ pixels, noting that CFD simulation results on unstructured meshes are routinely mapped to regular grids for display and analysis on computer screens. Hereafter, the DeepWFLO inputs will be treated as different image *channels*, using terminology commonly used to refer to the red, green, and blue (RGB) channels used to encode digital color images. Similarly, the output of the model is the resulting streamwise velocity field $\mathbf{u}(\mathbf{x})$, including turbine wake effects; thus, the output of the neural network will be an image with one channel. Natural extensions of this formulation to multiple outputs, e.g., predicting the spanwise velocity and pressure fields or predicting 3D fields, will be considered in a future publication.

Figure 1 shows an instance of the proposed DeepWFLO architecture, which will be used as a baseline model for this paper. The input, an image representation of a turbine layout, passes through an initial convolution layer that generates several feature maps with the same size as the input image. These feature maps are then fed through an alternating sequence of dense blocks that generate nonlinear feature maps and transition (encoding or decoding) blocks that change the size of the feature maps. Dense blocks, shown in Fig. 2(a), consist of a sequence of L sub-blocks. Each sub-block [Fig. 2(b)] is a fixed sequence of batch normalization (BN), ReLU, and convolution (Conv) operations that preserve the size of the feature maps and are represented by a single rectangular box in Figs. 2(c) and 2(d). Within each dense block, each sub-block uses as input a concatenation of the output of all previous sub-blocks with the same image size, i.e., all sub-blocks within the same block. This results in a linear increase in the number of feature maps between any two consecutive sub-blocks, with the rate of increase K referred to as the *growth rate*. Dense blocks are connected with each other through transition blocks that change the size of the feature map by 0.5X and 2X in the encoding [Fig. 2(c)] and decoding [Fig. 2(d)] paths, respectively. Comparing Figs. 2(c) and 2(d), it can be noted that the only difference between the architecture of the encoding and decoding blocks is the use of transposed convolution (ConvT) operations in the decoding paths, which implement the required upsampling of the feature maps.

The proposed DeepWFLO architecture is based on the DenseED architecture[32] and, therefore, also combines innovations introduced in image segmentation and Im2Im regression tasks over the last decade. First, DeepWFLO implements the DenseNet architecture,[43] in which each layer receives as input all the feature maps of the previous layers that share the same image size. This dense connectivity pattern improves information flow, explicitly reuses feature maps, and addresses the vanishing gradient issues that plague strictly sequential deep networks. Second, DeepWFLO is fully convolutional, doing away with fully connected layers of ReLU units and instead using convolution and upsampling layers to generate output images of the same resolution as the input images. Finally, DeepWFLO implements symmetric downsampling and upsampling







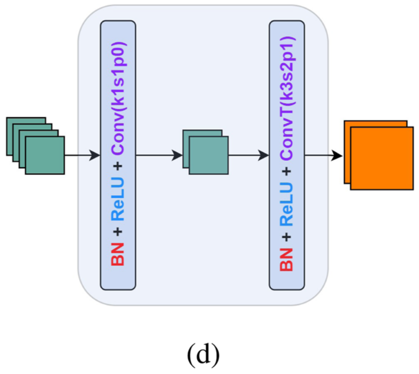


FIG. 2. Details of the inner blocks of the baseline architecture. (a) Dense block. (b) Sub-block. (c) Encoding block. (d) Decoding block.

paths to directly generate predictions with the same image size as the inputs, similar to U-Nets.[44] In contrast to U-Nets, however, DeepWFLO uses strided convolutions in the decoding and encoding blocks instead of the max pool operations used in previous work.[44]

In contrast with the DenseED architecture, however, DeepWFLO includes hierarchical connections between the encoding and decoding paths, shown as blue lines in Fig. 2. These connections were absent in the DenseED architecture due to the weak correspondence between the input (permeability) and output (flow velocities and pressures) fields in that application (flow in porous media). In the WFLO application, on the other hand, there is a stronger correspondence between the input (turbine locations, each a source/sink term in a parabolic PDE) and output (streamwise velocity) fields due to the convection-dominated nature of the parabolic PDE, the strength of the source/sink terms, and the large-scale nature of the wake structures. Therefore, including these hierarchical connections effectively ensures that this strong correspondence between inputs and outputs is preserved and exploited at every scale, allowing both global and local features to influence the network output. Second, we changed the size of the convolution filters in the dense blocks, increasing their receptive fields beyond the 3×3 size that was used in DenseEDs.[32] This modification is supported by the fact that wake effects extend for large distances in the domain, and small convolution filters would not be able to propagate these effects beyond a small region around any given turbine at any given convolution layer. We present a performance comparison between the original DenseED and DeepWFLO architectures in Sec. V D.

IV. TEST CASE

We evaluate the performance of the proposed DeepWFLO architecture with a test case that has been commonly used in the literature for wind farm layout optimization methods, e.g., Refs. 5, 9, and 45–48, to ensure consistency and comparability with previous work.

The selected test case consists of a wind farm in a flat rectangular terrain with dimensions of 2.0×2.0 km^2. In this terrain, a set of identical turbines is placed, with the number of turbines typically ranging from 10 to 40. Turbines are randomly placed within the wind farm terrain while enforcing a minimum inter-turbine distance constraint of $2D$, where D is the diameter of the turbine rotors. Turbine and terrain parameters are included in Table I and are consistent with previous work. The wind resource for this test case is considered to be a spatially uniform and constant wind velocity coming from the West. We considered wind velocity values in the range of 5.0–15.0 m/s.

TABLE I. Turbine and wind farm characteristics for the selected test case.

Parameter	Value
Turbine diameter (D)	80 m
Turbine hub height (H)	60 m
Thrust coefficient (C_t)	0.88
Turbine power curve	$0.3u^3$ kW
Wind farm terrain size $(W \times L)$	2.0×2.0 km
Terrain roughness length (z_0)	0.3 m

This test case was used to generate the dataset used in this work. Specifically, a total of 1024 training samples were generated, each consisting of a random turbine layout image with a random number of turbines in the range of 5–30 and its corresponding flow field image predicted with the Jensen model. A subset of 128 of these random layouts was also evaluated with the CFD model and used to train the models described in Sec. V F. To ensure the adequacy of this dataset for our application, we quantified the amount/degree of wake interactions present in the dataset. Among the 1024 images, wake regions covered ~42.4% of their pixels (range 14.66%–61.97%); this corresponded to ~45.85% of the turbines being in the wake of at least one upstream turbine (range 7.14%–70.83%).

V. RESULTS AND DISCUSSION

The results presented below explore the feasibility of the proposed DeepWFLO architecture and characterize its performance based on data generated with the Jensen analytical wake model, unless otherwise noted.

A. Spatial resolution and velocity encoding errors

To explore alternative problem representations and estimate the performance limits of the proposed methodology, we quantified the effects of spatial resolution and velocity encoding on prediction accuracy. We generated 1024 turbine layouts by randomizing the number of turbines in the layout, their continuous-variable coordinates, and the incoming wind velocity within the wind farm. For each turbine layout, we predict the wind velocity at the spatial location corresponding to each image pixel using the Jensen model and encode this velocity value as an image intensity. The resulting image is saved to a file, reloaded, and the image intensities are mapped back to real-valued wind velocity at the turbine locations. Note that, as these steps do not involve any neural network model training, any differences in wind velocity values are solely caused by encoding and decoding operations.

Results from these tests (included in the supplementary material) show that velocity errors caused by spatial encoding are ~4% for continuous-variable turbine coordinates and decrease to 0.1% when turbine coordinates are defined as discrete variables and restricted to pixel corners. Velocity encoding errors are in the range of 0.1%–0.25% when velocity values are encoded into pixel intensities as integers. These errors can be reduced to 0.06% if the velocity field is normalized by the incoming wind velocity prior to encoding, and further reduced to 0.05% when velocity values are encoded into pixel intensities as floats. Based on these results, in the remainder of the paper, we restrict turbine positions to pixel corners and encode normalized velocity values as floats.

B. Loss functions

The definition of appropriate loss functions is a critical but often overlooked step in successful surrogate modeling efforts. This is particularly true in applications of deep neural networks to engineering systems whose behavior is governed by physical laws. The standard practice for encoder/decoder networks is to minimize an MSE of pixel-level differences between the decoded and target images. In this application, pixel intensity values represent velocity, so this, in effect, would calculate the MSE of predicted velocity values. However, an important application of wind velocity predictions is the calculation of AEP. Hence, we formulated an alternative loss function that included the error in AEP as predicted from the decoded image with respect to the target AEP predicted by either the analytical or CFD models used in this work.

Specifically, we added an additional error term to the training loss L, representing the error in predicting power from the discretized flow field, creating a weighted combination of velocity and AEP errors. Then, we trained a set of DeepWFLO networks using the baseline architecture (Fig. 1), varying the relative weighting between the two error terms,

$$L = \omega_{AEP}\mathrm{MSE}_{AEP} + (1.0 - \omega_{AEP})\mathrm{MSE}_{U}, \tag{10}$$

where $\omega_{AEP} \in [0, 1]$. Note that both MSE_{AEP} and MSE_{U} are normalized independently before the calculation of the loss function using a rolling estimate of their maximum value over the preceding five epochs.

Figure 3 shows the resulting velocity and power errors from network predictions evaluated over the testing dataset when the networks are trained with different sets of weights for each term. The horizontal axis of this figure shows the relative weight of the power term (ω_{AEP}), and the two vertical axes show the resulting errors in wind speed and AEP. Including the AEP error as part of the loss function with $\omega_{AEP} \in [0.0, 0.8]$ improves the accuracy of the network when predicting AEP, reducing AEP errors from 2.7% to 1% without increasing the prediction error for wind velocity. For values of $\omega_{AEP} > 0.8$, it can be observed that the velocity error increases as the AEP error decreases further. We hypothesize that this is due to the fact that when the loss function considers only velocity errors, it gives equal weight to predictions in all areas of the wind farm. Therefore, the network captures wake patterns comprehensively, resulting in accurate predictions for velocity and, consequently, for AEP estimations based on the predicted velocity. However, adding the AEP error to the loss function increases the importance of accurate prediction of wind velocity in areas of the image immediately

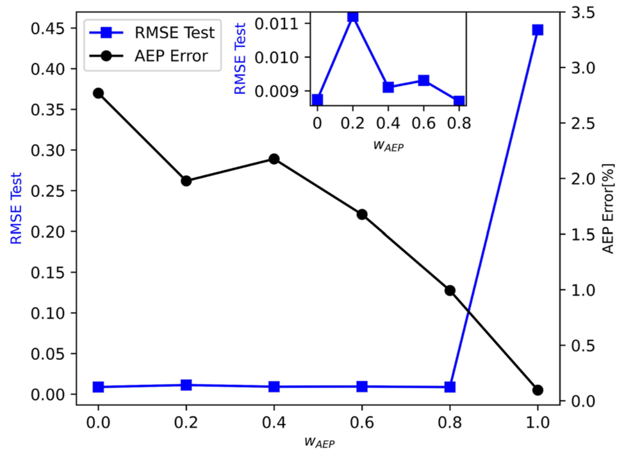


FIG. 3. Errors for different loss function weights over the test data.

surrounding the turbine locations, thus resulting in improved AEP predictions without an adverse impact on velocity errors. Therefore, this loss function, with $\omega_{AEP} = 0.8$, is used for the remainder of the paper.

C. Neural architecture search

To fine-tune the DeepWFLO architecture, we conducted a formal network architecture search using NSGA-Net,[49] a multi-objective genetic algorithm for architecture optimization. A detailed description of the architecture search process and the resulting Pareto set of network architectures are included in the supplementary material. Briefly, we considered as optimization variables the number of encoding/decoding blocks, the number of dense

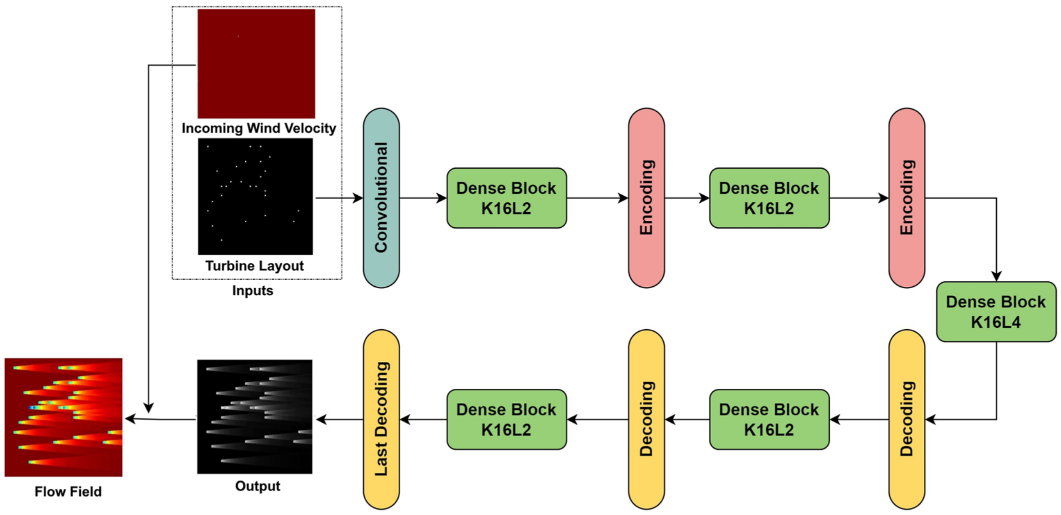


FIG. 4. Selected optimal DeepWFLO architecture.

blocks, and the internal architecture of each of these blocks. As optimization objectives, we chose to maximize the prediction accuracy (RMSE) of the resulting network while minimizing network size.

The optimal DeepWFLO architecture, selected from among the Pareto set, is shown in Fig. 4, which has the lowest RMSE over a dataset not used for training. This architecture contains dense blocks with 7 × 7 convolutions, stride 1, and padding as necessary to preserve image sizes within a given block. This architecture was trained on a desktop system with 16 GB RAM and a NVIDIA Quadro RTX5000 GPU with 16 GB or VRAM. On this system, training the DeepWFLO model on 256 images took 2008.96 s. The inference time for the trained model was 0.0133 s/image.

D. Dataset size

Figure 5 compares the learning curves for the optimal DeepWFLO architecture and the baseline DenseED-c8 architecture[32] for different image resolutions. Learning curves typically show the behavior of a performance metric as a function of the size of the dataset. It can be observed that the optimal architecture exhibits lower velocity and AEP errors than the baseline DenseED-c8 architecture. Importantly, the optimal DeepWFLO architecture is less sensitive to the image resolution, regardless of the number of images used for training, as evidenced by a lower dispersion between curves corresponding to different image resolutions.

Regarding the effect of dataset size, it can be observed that the optimal DeepWFLO architecture is 2X to 4X more data-efficient, reaching velocity errors under 0.2% and AEP errors under 5% with only 64 training images, an error level that required at least 256 images for DenseED-c8. In addition, the decreasing slope of the learning curves is evidence of the diminishing returns of larger training dataset sizes. Overall, the combination of low velocity and AEP errors and low learning curve slopes provides evidence that the proposed DeepWFLO architecture has sufficient capacity to capture the underlying behavior of the velocity field and that it can do so with a dataset containing between 256 and 512 images of the flow field in a wind farm with arbitrary turbine layouts. It is important to note that the number of estimable parameters in the proposed DeepWFLO architecture is only weakly dependent on the image resolution, given the convolutional nature of the network and the absence of fully connected layers of ReLU units. Specifically, the number of parameters depends on the image resolution only through the number of encoding/decoding blocks and the desired size of the final encoded image, i.e., the size of the bottleneck latent space.

E. Power prediction

The results presented so far have been based on comparisons of accuracy expressed in terms of average errors for wind velocity and AEP predictions. In this section, we show in more detail the prediction errors for AEP, which are of particular importance for the application of the DeepWFLO architecture in wind farm layout optimization.

Figure 6 compares AEP values predicted by DeepWFLO models with those predicted by the Jensen [Figs. 6(a) and 6(b)] or Gaussian [Figs. 6(c) and 6(d)] analytical wake models used to generate the data upon which each DeepWFLO model was trained. Guided by the results of the previous sections, both DeepWFLO models were trained with 256 images with a resolution of 400 px × 400 px. For performance evaluation, a total of 1.000 random layouts not included in the training dataset were used. It can be observed that the median AEP error of the model is just below 1.0%. Importantly, the Spearman correlation between the reference and predicted AEP values, which is based on rank-ordered AEP data, is 99.9% in both cases. In other words, the trained DeepWFLO model is capable of accurately ranking layouts according to their AEP ~99.9% of the time. Overall, these results provide evidence of the ability of the proposed architecture to accurately capture wake patterns, both idealized (linear expansion, top-hat wake profile) and more realistic (non-linear expansion, Gaussian wake profile).

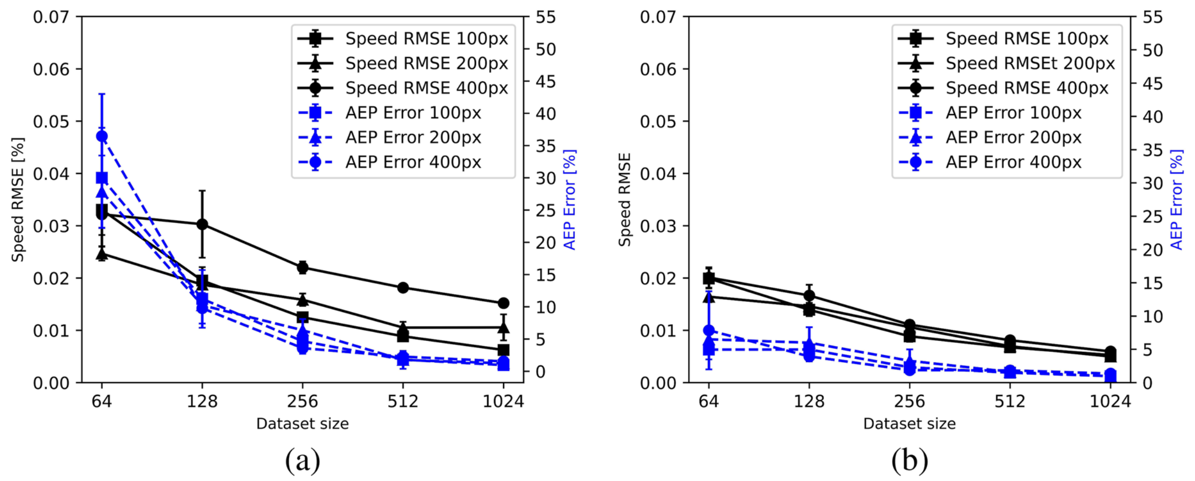


FIG. 5. Effect of dataset size and image resolution on wind velocity and AEP errors for baseline and optimal architectures. (a) Velocity and AEP errors, DenseED-c8 architecture.[32] (b) Velocity and AEP errors, optimal DeepWFLO architecture (Sec. V C).

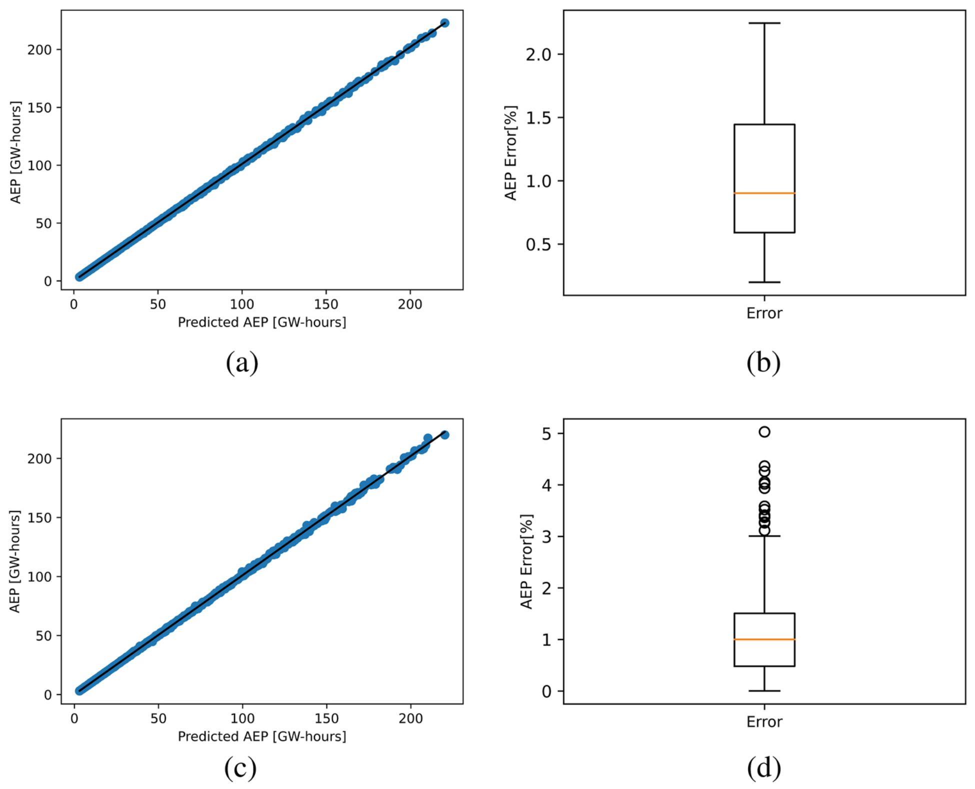


FIG. 6. Comparison of AEP predictions for 1000 arbitrary turbine layouts not used for training. DeepWFLO model trained on Jensen wake data vs Jensen model; (c) and (d) DeepWFLO model trained on Gaussian wake data vs Gaussian model. (a) Predicted vs reference, Jensen data. (b) AEP errors, Jensen data. (c) Predicted vs reference, Gaussian data (d) AEP errors, Gaussian data.

F. CFD data

After demonstrating the feasibility and performance of the proposed DeepWFLO architecture for the prediction of wind farm wakes based on data generated with analytical wake models, in this section, we use data generated with CFD simulations, as described in Sec. III A 3. A total of 128 CFD simulations were conducted, and the resulting images were split into training (80%), validation (10%), and testing (10%) datasets.

Several strategies were used for creating these models, leveraging results from Sec. V E, with the goal of defining the most data-efficient strategy to create accurate models based on the smallest possible CFD dataset. First, in what we refer to as the *CFD Only* strategy, we use the optimal network architecture from Sec. V C and train it with CFD data following best practices for data splitting, weight initialization, normalization, etc. Second, we use a *Warm Start* strategy in which we use the optimal weights from the network models trained with analytical wake predictions (Jensen or Gaussian) as the initial weights for training the network based on CFD data. Third, we use a *Transfer Learning* strategy in which, starting from the optimal network based training with analytical wake images, only the first encoding layer and last decoding layer are trained with the CFD data, the rest of the layers being frozen. Finally, a *Model Fusion* strategy is implemented by adding an additional input: a network that is trained with CFD data. The additional input is again generated from either Jensen or Gaussian wake models. For all strategies, training hyperparameters are kept the same as used for the results in the previous sections. All strategies exhibited similar computational costs, with model-fusion showing the largest cost, taking 1188.51 s for training on 104 images from CFD results on the same hardware mentioned in Sec. V C. Inference time remained at 0.0133 s/image since the same optimal DeepWFLO architecture was used for all results in this section.

Figure 7 provides a visual representation of the speed and AEP error distributions over the CFD test dataset for the four strategies described earlier, with either Jensen [Figs. 7(a) and 7(b)] or Gaussian [Figs. 7(c) and 7(d)] wake data used to initialize the model (Warm Start, Weight Transfer) or as an additional input to the network (Model Fusion). It can be observed that prediction errors for these CFD-trained networks are low, with median errors in the ranges of 1.8%–8.1% for speed and 3.0%–13.0% for AEP. Notably, test errors are generally larger for these networks than for those trained exclusively with flow field images generated with analytical wake models (ref. previous sections). This may have been expected since the CFD-predicted wake patterns (Fig. 8) are more complex than those predicted by Jensen or Gaussian models; e.g., acceleration effects due to flow blockage are not captured in the standard

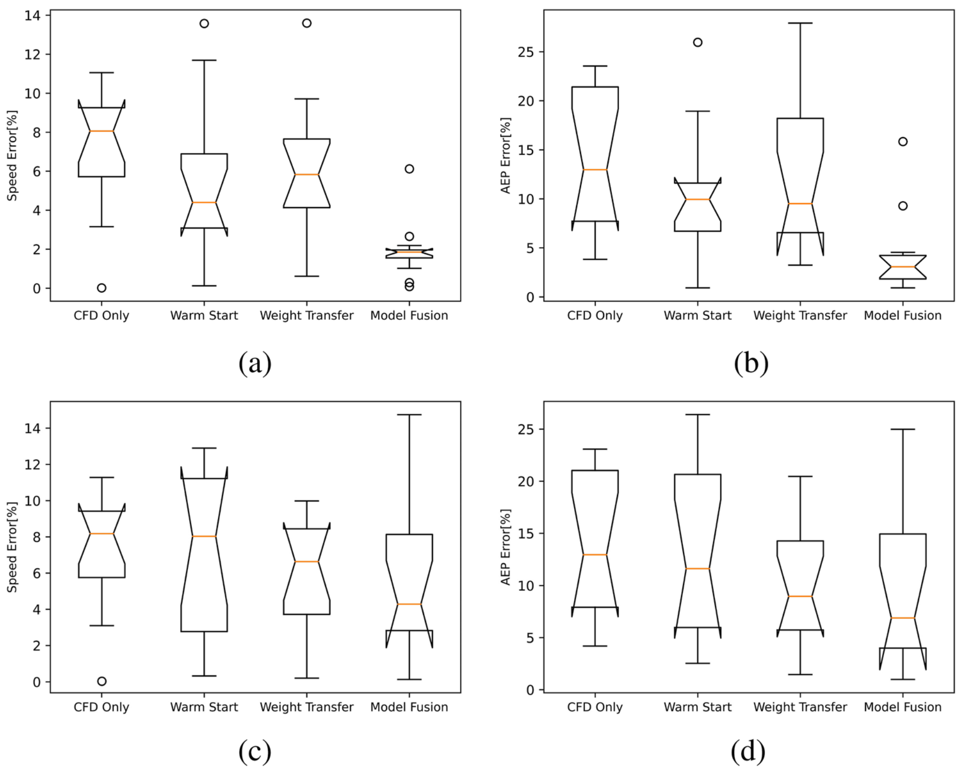


FIG. 7. Comparison of strategies for training with CFD data. (a) Speed errors, CFD + Jensen data. (b) AEP errors, CFD + Jensen data. (c) Speed errors, CFD + Gaussian data. (d) AEP errors, CFD + Gaussian data.

Jensen or Gaussian wake models but are clearly visible in the CFD results.

Confidence intervals (95%) for the medians of the error distributions, represented by boxplot notches, are also shown in Fig. 7. Comparing these confidence intervals, it can be observed that there is a statistically significant difference between the median errors (1.8% for velocity and 3.0% for AEP) of the *Model Fusion* strategy based on Jensen wake data and all other strategies based on the same data, as evidenced by the non-overlapping boxplot notches. We also calculated the Spearman (rank) correlations between predicted (DeepWFLO) and reference (CFD) values, resulting in a 95.1% rank correlation for the *Model Fusion* strategy based on Jensen wake data, compared with 88.1%–90.9% for the other transfer learning strategies based on the same data. For the *Model Fusion* strategy based on

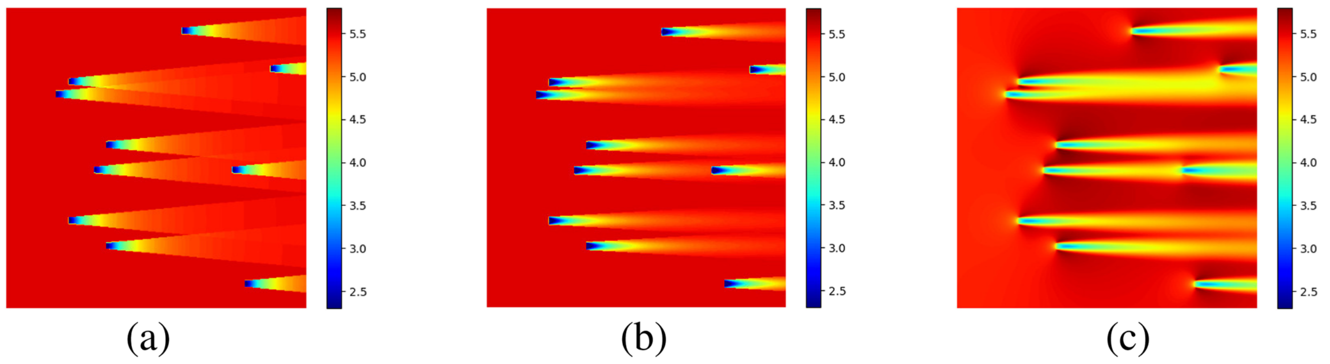


FIG. 8. Sample streamwise velocity field predictions for the W-E wind direction assumed in this paper. Note that predictions for any other wind direction can be generated by rotating the turbine layout accordingly. (a) Jensen wake model. (b) Gaussian wake model. (c) CFD.

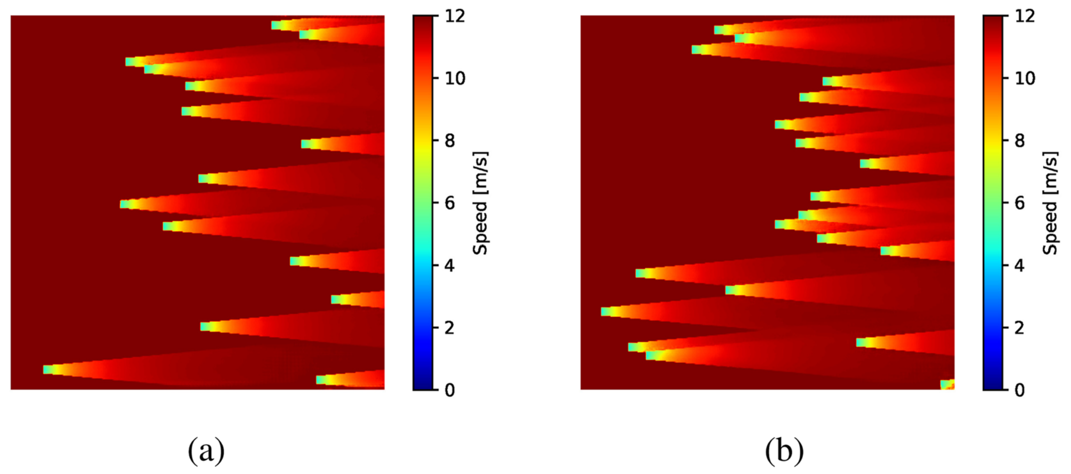


FIG. 9. Optimized turbine layouts using the DeepWFLO model based on Jensen and CFD data with the Model Fusion strategy. (a) n_T = 15, AEP = 67.97 GW h, η = 99.7%. (b) n_T = 20, AEP = 89.57 GW h, η = 98.6%.

Gaussian wake data, the median speed and AEP errors (4.0% and 5.9%) are still lower than other strategies based on the same data, though this difference in medians does not reach statistical significance. However, the Spearman correlation coefficient for this model fusion strategy was 92.3%, compared with 86.7% and 87.3% for the warm start and weight transfer strategies, respectively.

Notably, in all cases, the transfer learning strategies tested here resulted in lower median errors than training the DeepWFLO model exclusively on CFD data. Moreover, the rank correlation coefficients between predicted and reference (CFD) speed and AEP values were all above 88.0%, with the model fusion strategy based on Jensen wake data exhibiting the largest correlation coefficient, 95.1%. These results suggest that the proposed DeepWFLO architecture is suitable for predicting the wind velocity field inside a wind farm and provide evidence that an ensemble of wake models (e.g., Jensen and CFD) can result in low prediction errors and large rank correlations between predicted and reference values. Interestingly, the DeepWFLO network trained with the Model Fusion strategy achieved great accuracy while relying only on 128 CFD simulations (of which only 104 were used for training) and leveraging data generated with the lower-fidelity Jensen analytical model. In comparison with the *CFD Only* strategy, *Model Fusion* achieved approximately a 4X improvement in accuracy for the same dataset size. Therefore, it can be said that the model fusion strategy is 4X more data-efficient.

G. Optimization

As an illustration of the capabilities of the *Model Fusion* DeepWFLO model, we solved a wind farm layout optimization (WFLO) problem. In particular, we use the wind farm case described in Sec. IV and aim to optimally place n_T turbines ($n_T \in \{15, 20\}$) within the wind farm boundaries while enforcing a minimum inter-turbine distance constraint of $5D$, where D is the turbine diameter. Formally, let $\mathbf{x} = (x_1, y_1, \dots, x_{n_T}, y_{n_T})$ be a vector concatenating the 2D coordinate pairs (x_i, y_i) of all turbines, $i \in \{1, 2, \dots, n_T\}$. The WFLO problem is posed as

$$\begin{aligned} &\underset{\mathbf{x}}{\arg\max} && \mathrm{AEP}(\mathbf{x}) \\ &\text{subject to} && (x_i, y_i) \in [0, 2000] \times [0, 2000], \quad \forall\, i \in \{1, \dots, n_T\} \\ & && \min_{i<j} \|(x_i, y_i) - (x_j, y_j)\| \le 5D, \\ & && \forall\, i, j \in \{1, \dots, n_T\}, i < j. \end{aligned} \tag{11}$$

An $\mu + \lambda$ evolutionary algorithm (EA) was selected to solve this problem based on its demonstrated performance in the extensive literature on WFLO and our own previous experiences.[7,9] Since the evolutionary strategies do not handle optimization constraints directly, the objective function was modified to include inter-turbine distance constraints via the penalty method. Let $d_{\min}$ be the minimum distance between turbines in a given layout, then the WFLO problem is reformulated as

$$\begin{aligned} &\underset{\mathbf{x}}{\arg\max} && \mathrm{AEP}(\mathbf{x}) + C(5D - d_{\min}) \\ &\text{subject to} && (x_i, y_i) \in [0, 2000] \times [0, 2000], \quad \forall\, i \in \{1, \dots, n_T\}, \end{aligned} \tag{12}$$

where C is the penalty parameter. In this work, we set the penalty coefficient $C = 0.05 * \mathrm{AEP}_{ideal} * (1 - 5Dd_{\min}^{-1})$ to be a function of the maximum (ideal) AEP for the test case, i.e., which would be attained if there were no wake effects. This allows us to ensure that as the solution approaches the feasibility boundary, the size of the penalty is a small percentage of the objective function value.

Figures 9(a) and 9(b) show flow fields, AEP, and wind farm efficiency (η) corresponding to the optimal layouts for 15 and 20 turbines, respectively, as predicted by the most accurate DeepWFLO Model Fusion model (based on Jensen and CFD data). The wind farm efficiency is calculated as the ratio between the AEP and AEP_{ideal}, resulting in 99.7% and 98.6% for the cases with 15 and 20 turbines, respectively. These efficiencies show that turbines in the optimal layouts are not significantly exposed to wake effects in any of the cases.

VI. CONCLUSIONS

In this work, we presented DeepWFLO, a deep convolutional hierarchical encoder–decoder neural network architecture for Wind Farm Layout Optimization (WFLO). DeepWFLO generates accurate images of the wind velocity field within wind farms, using image representations of the turbine layout and undisturbed flow field as inputs. To train the DeepWFLO architecture, a loss function was proposed as a linear combination of mean-squared losses for the velocity field and the annual energy production (AEP). Results show that a relative weighting of 80%/20% of the AEP and velocity losses decreased AEP errors over the test set from 2.7% to 1% without increasing velocity errors. Using this loss function, the DeepWFLO architecture was optimized with a multi-objective genetic algorithm to minimize both velocity errors and network size. The resulting optimal DeepWFLO architecture was shown to be highly accurate when trained with data generated by analytical wake models (median velocity error below 1%) and by CFD simulations (median velocity error ~8%). Importantly, Spearman (rank) correlation coefficients between predicted and reference AEP were 99.9% and 88.1% for models trained with data from either analytical wake models or CFD data, respectively. This provides evidence for the suitability of DeepWFLO for surrogate-based optimization applications, which require accurate relative ranking of solutions.

Several transfer learning strategies were compared to further reduce velocity and AEP errors in networks trained with CFD data without increasing dataset size. The model-fusion strategy, in which an image of the wind velocity field generated with an analytical wake model is used as an additional input channel, outperformed the commonly used weight transfer and warm start strategies. Model fusion decreased velocity errors to 1.8% and increased the rank correlation between predicted and reference AEP to 95.1% for a DeepWFLO network trained with data from 104 CFD simulations.

Overall, the present work has demonstrated the feasibility of DeepWFLO and image-to-image regression in general to predict wind velocity and energy generation in wind farms based on a combination of analytical wake models and a small set of computationally expensive CFD simulations. We believe that Im2Im regression has great potential for the synergistic integration of neural network technologies and CFD simulations to support engineering design and optimization tasks in general, particularly in those applications in which (a) alternative design configurations can be intuitively represented as images (e.g., design of layouts, shapes, structural topologies, etc.), (b) system performance is evaluated based on functionals (e.g., integrals or convolutions) of spatiotemporal fields (e.g., flow velocity, pressure, temperature, density), and (c) system performance is highly dependent on local interactions that decay with distance and are translation-invariant.[50]

ACKNOWLEDGMENTS

The authors would like to acknowledge the Natural Sciences and Engineering Research Council of Canada (NSERC) for their financial support of this research.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

David A. Romero: Conceptualization (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Software (equal); Supervision (equal); Validation (equal); Writing – original draft (equal); Writing – review & editing (equal). **Saeede Hasanpoor**: Data curation (equal); Formal analysis (equal); Investigation (equal); Software (lead); Validation (equal); Visualization (equal); Writing – original draft (supporting). **Enrico Antonini**: Data curation (equal); Investigation (equal); Writing – original draft (supporting); Writing – review & editing (supporting). **Cristina H. Amon**: Conceptualization (equal); Project administration (lead); Supervision (lead); Writing – original draft (equal); Writing – review & editing (lead).

DATA AVAILABILITY

The data that support the findings of this study are openly available in GitHub at https://github.com/davidromerot/DeepWFLO.

REFERENCES

[1] IRENA, *World Energy Transitions Outlook 2022: 1.5 °C Pathway* (International Renewable Energy Agency, Abu Dhabi, 2022).

[2] E. G. A. Antonini and K. Caldeira, "Spatial constraints in large-scale expansion of wind power plants," Proc. Natl. Acad. Sci. U. S. A. **118**, e2103875118 (2021).

[3] E. G. A. Antonini, D. A. Romero, and C. H. Amon, "Improving CFD wind farm simulations incorporating wind direction uncertainty," Renewable Energy **133**, 1011–1023 (2019).

[4] S. Chowdhury, J. Zhang, A. Messac, and L. Castillo, "Unrestricted wind farm layout optimization (UWFLO): Investigating key factors influencing the maximum power generation," Renewable Energy **38**, 16–30 (2012).

[5] S. D. O. Turner, D. A. Romero, P. Y. Zhang, C. H. Amon, and T. C. Y. Chan, "A new mathematical programming approach to optimize wind farm layouts," Renewable Energy **63**, 674–680 (2014).

[6] S. Chowdhury, J. Zhang, W. Tong, and A. Messac, "Modeling the influence of land-shape on the energy production potential of a wind farm site," J. Energy Resour. Technol. **136**, 011203 (2014).

[7] S. Yamani Douzi Sorkhabi, D. A. Romero, G. K. Yan, M. D. Gu, J. Moran, M. Morgenroth, and C. H. Amon, "The impact of land use constraints in multi-objective energy-noise wind farm layout optimization," Renewable Energy **85**, 359–370 (2016).

[8] D. Guirguis, D. A. Romero, and C. H. Amon, "Gradient-based multidisciplinary design of wind farms with continuous-variable formulations," Appl. Energy **197**, 279–291 (2017).

[9] W. Yin Kwong, P. Yun Zhang, D. Romero, J. Moran, M. Morgenroth, and C. Amon, "Multi-objective wind farm layout optimization considering energy generation and noise propagation with NSGA-II," J. Mech. Des. **136**, 091010 (2014).

[10] F. Bürgel, R. Scholz, C. Kirches, and S. Stiller, "Potential of dynamic wind farm control by axial induction in the case of wind gusts," Wind Energy Science (submitted 2023).

[11] P. M. O. Gebraad and J. W. van Wingerden, "A control-oriented dynamic model for wakes in wind plants," J. Phys.: Conf. Ser. **524**, 012186 (2014).

[12] S. Kanev, "On the robustness of active wake control to wind turbine downtime," Energies **12**, 3152 (2019).

[13]S. Kanev, "Dynamic wake steering and its impact on wind farm power production and yaw actuator duty," Renewable Energy **146**, 9–15 (2020).

[14]H. Mendez Reyes, S. Kanev, B. Doekemeijer, and J.-W. van Wingerden, "Validation of a lookup-table approach to modeling turbine fatigue loads in wind farms under active wake control," Wind Energy Sci. **4**, 549–561 (2019).

[15]Y.-T. Wu and F. Porté-Agel, "Modeling turbine wakes and power losses within a wind farm using LES: An application to the horns rev offshore wind farm," Renewable Energy **75**, 945–955 (2015).

[16]M. de Oliveira, R. Puraca, and B. Carmo, "Blade-resolved numerical simulations of the NREL offshore 5 MW baseline wind turbine in full scale: A study of proper solver configuration and discretization strategies," Energy **254**, 124368 (2022).

[17]D. Seidl, G. Geraci, R. King, F. Menhorn, A. Glaws, and M. Eldred, "Multifidelity strategies for forward and inverse uncertainty quantification of wind energy applications," Tech. Rep. AIAA, 2020.

[18]J. Cao, C. M. Nyborg, J. Feng, K. S. Hansen, F. Bertagnolio, A. Fischer, T. Sørensen, and W. Z. Shen, "A new multi-fidelity flow-acoustics simulation framework for wind farm application," Renewable Sustainable Energy Rev. **156**, 111939 (2022).

[19]M. Bastankhah and F. Porté-Agel, "A new analytical model for wind-turbine wakes," Renewable Energy **70**, 116–123 (2014).

[20]H. Sun, X. Gao, and H. Yang, "Validations of three-dimensional wake models with the wind field measurements in complex terrain," Energy **189**, 116213 (2019).

[21]C. Zhang, G. Yuan, W. Niu, J. Tian, S. Jin, D. Zhuang, Z. Jiang, Y. Wang, B. Ren, S. L. Song, and D. Tao, "ClickTrain: Efficient and accurate end-to-end deep learning training via fine-grained architecture-preserving pruning," in *Proceedings of the ACM International Conference on Supercomputing (ICS '21)* (Association for Computing Machinery, New York, 2021), pp. 266–278.

[22]S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," Annu. Rev. Fluid Mech. **52**, 477 (2020).

[23]N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, "Surrogate-based analysis and optimization," Prog. Aerosp. Sci. **41**, 1–28 (2005).

[24]H. Liu, Y.-S. Ong, and J. Cai, "A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design," Struct. Multidiscip. Optim. **57**, 393–416 (2018).

[25]X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "NSFnets (Navier–Stokes flow nets): Physics-informed neural networks for the incompressible Navier–Stokes equations," J. Comput. Phys. **426**, 109951 (2021).

[26]M. Raissi, "Deep hidden physics models: Deep learning of nonlinear partial differential equations," J. Mach. Learn. Res. **19**(1), 932–955 (2018).

[27]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).

[28]M. Magill, F. Qureshi, and H. W. de Haan, "Neural networks trained to solve differential equations learn general representations," in *Advances in Neural Information Processing Systems*, 31 (Neural Information Processing Systems Foundation, 2018).

[29].N. Wandel, M. Weinmann, and R. Klein, "Teaching the incompressible Navier–Stokes equations to fast neural surrogate models in three dimensions," Phys. Fluids **33**(4), (2021).

[30]S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," Acta Mech. Sin. **37**, 1727–1738 (2021).

[31]A. S. Iskhakov, N. T. Dinh, and E. Chen, "Integration of neural networks with numerical solution of PDEs for closure models development," Phys. Lett. A **406**, 127456 (2021).

[32]Y. Zhu and N. Zabaras, "Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification," J. Comput. Phys. **366**, 415–447 (2018).

[33]Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data," J. Comput. Phys. **394**, 56–81 (2019).

[34]Z. Ti, X. W. Deng, and H. Yang, "Wake modeling of wind turbines using machine learning," Appl. Energy **257**, 114025 (2020).

[35]J. Zhang and X. Zhao, "A novel dynamic wind farm wake model based on deep learning," Appl. Energy **277**, 115552 (2020).

[36]S. Pawar, A. Sharma, G. Vijayakumar, C. J. Bay, S. Yellapantula, and O. San, "Towards multi-fidelity deep learning of wind turbine wakes," Renewable Energy **200**, 867–879 (2022).

[37]J. Zhang and X. Zhao, "Wind farm wake modeling based on deep convolutional conditional generative adversarial network," Energy **238**, 121747 (2022).

[38]T. Göçmen, P. van der Laan, P.-E. Réthoré, A. P. Diaz, G. C. Larsen, and S. Ott, "Wind turbine wake models developed at the technical university of Denmark: A review," Renewable Sustainable Energy Rev. **60**, 752–769 (2016).

[39]N. O. Jensen, A note on wind generator interaction, 2411, 1983.

[40]D. Li, J. Chang, G. Ma, C. Huo, and R. Li, "Optimized wake-superposition approach for multiturbine wind farms," Sci. Rep. **13**, 6672 (2023).

[41]E. G. A. Antonini, D. A. Romero, and C. H. Amon, "Analysis and modifications of turbulence models for wind turbine wake simulations in atmospheric boundary layers," J. Sol. Energy Eng. **140**, 031007 (2018).

[42]E. G. A. Antonini, D. A. Romero, and C. H. Amon, "Optimal design of wind farms in complex terrains using computational fluid dynamics and adjoint methods," Appl. Energy **261**, 114426 (2020).

[43]G. Huang, Z. Liu, L. Van Der Maaten, and K. Weinberger, "Densely connected convolutional networks," CVPR, 2017.

[44]O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science Vol. 9351, edited by N. Navab, J. Hornegger, W. Wells, and A. Frangi (Springer, Cham, 2015).

[45]G. Mosetti, C. Poloni, and B. Diviacco, "Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm," J. Wind Eng. Ind. Aerodyn. **51**, 105–116 (1994).

[46]P. Y. Zhang, D. A. Romero, J. C. Beck, and C. H. Amon, "Solving wind farm layout optimization with mixed integer programs and constraint programs," EURO J. Comput. Optim. **2**, 195–219 (2014).

[47]S. A. Grady, M. Y. Hussaini, and M. M. Abdullah, "Placement of wind turbines using genetic algorithms," Renewable Energy **30**, 259–270 (2005).

[48]A. Kusiak and Z. Song, "Design of wind farm layout for maximum wind energy capture," Renewable Energy **35**, 685–694 (2010).

[49]Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19)* (Association for Computing Machinery, New York, 2019), pp. 419–427.

[50]L. Ankile, M. Heggland, and K. Krange, "Deep convolutional neural networks," arXiv:2011.12960v1 [cs.CV] (2020).