Advancing Wind Energy Forecasting: Integrating Physics-Informed Neural Networks with Timestep Dynamics for Improved Turbine Energy Prediction

Jessica Lin

Advised by Professor Ryan Adams

Abstract

This project addresses the urgent need for improved wind power forecasting (WPF) in the context of escalating demand for sustainable energy. We tested the extent to which physics-informed neural networks (PINNs) and simple time-dependent neural networks could accurately predict wind turbine energy output when used in isolation as well as in conjunction with each other. We constructed four neural networks to test these features and found that incorporating timestep data had extremely high performance accuracy, and any model that incorporated PINNs performed more poorly than the control model. With this work, we added to the replicability of PINNs in WPF and contributed valuable insights to the application of PINNs and hybrid ML approaches in WPF by using novel methods to improve the prediction accuracy of wind turbine power.

1. Introduction / Motivation

As the demand for cleaner sources of energy rises, research in wind turbine energy prediction has become more relevant than ever. Wind energy is an optimal choice for renewable energy, not only due to its ability to offset CO_2 emissions and decrease water usage, but also because it is an

inexhaustible, emissions-free resource. The Global Wind Energy Council (GWEC) expects 680 GW of wind capacity to be added globally between 2023 and 2027 [1]. Due to the high projected demand for wind energy, it is crucial to identify how much energy windmill farms can produce to satisfy the rapidly growing economical and physical demand for electricity.

Many factors affect the amount of energy a given wind turbine can produce at a certain time, such as wind speed, wind direction, and internal factors like turbine power capacity. The wind energy industry has seen a growing need for better accuracy on how they can predict turbine power output from these variables; this increasingly relevant field of research is known as wind power forecasting (WPF). On the short-term scale, better prediction accuracy in WPF is needed for energy mix allocation in cities that use wind energy among other sources of electricity. On the long-term scale, as wind turbine capacity grows, regions that are looking to incorporate wind energy into their electricity grids need to know how much energy turbines could generate if placed in that area. Additionally, institutions can be reluctant to incorporate wind energy into the grid unless they are reassured of its reliability in power generation. Therefore, improving WPF is essential to providing reliable clean energy. However, wind turbine power is highly unpredictable to forecast using traditional time-series analysis methods, because these weather variables have a non-linear, irregular relationship with energy output [1]. Therefore, machine learning presents as an excellent opportunity to better analyze energy output from wind turbines. In the last decade, researchers have taken to applying a variety of machine learning approaches to WPF. In a literature review conducted by Jørgensen and Shaker (2020), researchers concluded that the most common machine learning method for this context was neural networks [2].

We propose that a physics-informed neural network (PINN) or a hybrid PINN model has the potential to perform well on offshore WPF, as PINNs have successfully been used in many

applications across material sciences and geophysics [3]. PINNs minimize data loss by taking natural laws of physics into account as a sort of regularization term. By incorporating known differential equations that explain natural relationships into the loss equations of neural networks, PINNs are able to produce strong generalization, whereas traditional neural networks may overfit to the training data if there is not enough data available [3]. This suggests that PINNs are a great candidate for WPF, where the data is highly irregular depending on where the turbine is placed or the seasonal data that's fed into the model. Despite their high potential to the field of WPF, there is currently a gap in the literature pairing the PINN method with wind prediction. in fact, PINNs have only been applied to the context of wind power prediction once, in a 2023 study conducted by Gijón et al. that predicted wind turbine power as a function of mechanical properties of the turbine [4]. In this paper, we set out to determine whether PINNS can be feasibly applied to this context. Second, we aim to explore whether incorporating the timestep nature of turbine data in a simple way can improve prediction accuracy, tested both alone and in conjunction with the PINN method.

2. Related Work

In recent years, much of the research has pivoted towards using neural networks as well as hybrid neural network models, as they have been shown to perform better than ensemble learning models or regressions. Jørgensen and Shaker conducted an analysis on the frequency of ML methods in the wind energy prediction field, and they found that neural networks have been the most popularly discussed approach for over ten years and have been dominating the field in the past decade, followed in popularity by support vector machines (SVM) [2]. In a literature review comparing ML methods for WPF, Buturache and Stancu concluded that out of the four most prominent ML methods in the field, artificial neural networks (ANN) achieve the best

prediction accuracy [5]. This conclusion supports the notion that pursuing some sort of neural network model or hybrid neural network model has high potential for improving performance.

There has been an abundance of neural network hybrid models used in wind power prediction, although there has not yet been a hybrid approach that incorporates PINNs, as they are relatively new to the field of WPF. Ma and Zhai used a unique methodology of a dual-step prediction method which seems promising, including a feed forward artificial neural network (FFAN) in their hybrid model [6]. First, they used weather variables to estimate the wind speed at the location of the wind farm. Then, they fit the actual wind speed at the farm with real power generation from the farm; in other words, they fit how to predict the real power from real speed. Lastly, they combined these two steps by feeding the second model the predicted speeds from the first model to get the predicted generated power. Using this method, they were able to achieve 0.82% normalized mean absolute error (NMAE). This method improved forecast skill compared to previous hybrid FFAN models as well as a standalone neural network model and was able to successfully predict 24H forecasts [6]. This kind of methodology provides insight as to how we can compose a model that is best tailored to wind patterns.

There has been little work done with PINN applications for WPF because the field is so new. The concept of PINNs as a new class of data-driven solvers was first introduced by Raissi et al. in 2017 [7], and while there is currently only one study on PINNs in WPF, PINNs have been used in adjacent applications within the field of wind energy. Zhang and Zhao used PINNs to predict spatiotemporal wind speeds with the aim of beating current measurement methods of Light Detection and Ranging (LIDAR) sensor technology, which can only give sparse point measurements of wind speed [8]. They combined LIDAR measurements with the Navier-Stokes equations, which describe atmospheric flows, to predict spatiotemporal wind field measurements

and provide new ways to monitor wind farm performance. The authors acknowledge a limitation of their research in the sense that Navier-Stokes equations are 2D, where the flow structures in question are 3D. To the authors' knowledge, this is the first paper of its kind to combine physics laws with deep learning in wind pattern mapping, and because neural networks have shown to have the highest performance in the field of wind energy, it would be interesting to extend this approach to wind power prediction rather than wind speed. Alternatively, a dual-step approach such as the one proposed by Ma and Zhai [6] also shows promise, where PINNs are used for the first step of predicting wind speed from weather variables, and then this output is fed into the second step to predict generated power.

Li and Zhang used PINNs in another WPF-adjacent context to predict when fatigue damage would be most likely to occur for wind turbines, as strong cyclic wind and wave loads tend to damage turbines over time [9]. They proposed a deep residual recurrent neural network (DR-RNN) as a physics-informed deep learning model, where their physical constraint for the data loss was expressing the excitation force vector as:

$$f = M\ddot{\mathrm{y}} + \mathrm{C}\dot{\mathrm{y}} + \mathrm{ky} \tag{1}$$

where M is the mass matrix, C is the damping matrix, k is the stiffness matrix, and y is the structural displacement vector. Comparing its performance to a purely data-driven LSTM model, they found that while their DR-RNN succeeded in capturing the general trend of the response, it did not perform well on the high-frequency oscillating response prediction. Their DR-RNN also demonstrated a strong capability in long-term response prediction even if it was trained for a short duration [9]. This study supports the growing idea that PINNs are not just intriguing but also show substantial promise as a tool in the field of wind energy prediction.

The work most relevant to our proposal comes from the only study applying PINNs to WPF: the 2023 work of Gijón et al., in which they predicted the generated power of turbines using PINNs. Their model was trained using Supervisory Control and Data Acquisition (SCADA) data from four turbines from 'La Haute Borne' wind farm, an onshore farm in France. In their neural networks, Gijón et al. used the factors wind velocity, pitch angle, torque, and rotor angular velocity. For their physics-informed loss terms, they used two approaches, incorporating two equations for wind output, where both predict wind power. The first is from a mechanical standpoint, where wind power is expressed as:

$$P = gT\omega \tag{2}$$

where g is gear ratio, T is rotor torque, and ω is angular velocity. The second equation takes the natural laws wind kinetic energy into account and is more commonly of interest in the field of WPF, expressing wind power as:

$$P = \frac{1}{2} C_p \rho A v^3 \tag{3}$$

where C_p is the power coefficient, ρ is the air density, A is the swept area by the blades of the turbine (also expressed as $A = \pi r^2$, where r is the rotor length), and v is the wind velocity. It's important to note that wind power is a cubic function of wind velocity. Among the data available to them, they focused on the following variables: wind velocity, pitch angle, rotor speed, torque, and power, represented as (v, β, ω, T, P) where β is the pitch angle. They calculated C_p straightforwardly from the generated power and wind velocity. Gijón et al. performed three experiments, optimizing for variables P, C_p, and T, where P was expressed from a mechanical

standpoint ($gT\omega$), and C_p, and T were expressed by manipulating the cubic velocity function. They showed that the combination of a physical formula to predict the power coefficient and a neural network was more effective than using the physical formula alone, and they achieved a MAPE less than 4% and a MAE below 16 kW of power. They acknowledge that although this family of models seems promising, its robustness against different turbines would need further investigation. Another limitation of their research is that they separate wind power prediction into two approaches; Using the two equations outlined, they create two PINNs to predict wind power as a function of purely mechanical factors, and secondly, as a function of natural factors like air density. However, Gijón et al. fail to take advantage of the fact that SCADA data is conveniently organized by timestep, and the relative temporal aspect of the data can improve prediction accuracy. Previous research on time-dependent neural networks has shown great success in WPF, yet there remains a gap in the literature on a methodology that combines the two concepts, partially due to the novelty of PINNs to the field.

Many types of time-dependent neural networks exist that predict turbine power, but the most popular in WPF are Long Short-Term Memory (LSTM), Paula et al. found that LSTM models outperformed Extreme Gradient Boosting and Random Forest models when tested on data from a Brazilian wind farm, as the LSTM model achieved an MAPE of 4.55% [10]. In another study that focused entirely on comparing temporal deep learning networks, researchers Lin et al. found that out of the Temporal Convolutional Network (TCN), recurrent neural network (RNN), and gated recurrence unit (GRU) models, TCN performed the best and achieved a 5.13% MAPE for 72h prediction, whereas Lin et al. claim that traditional ML models are only able to achieve 10-17% MAPE for long-term prediction [11]. Gong et al. used a hybrid TCN model to predict wind power with the addition of a novel AdaBelief optimizer. They compared this hybrid model to

other timestep models like LSTM and GRU and concluded that the AdaBelief Optimizer boosted accuracy. However, the focus of their results appears to be on the effects of the optimizer rather than how the timestep models perform in relation to each other. Whereas this paper lacks a controlled focus on temporal models, this approach seeks to discern how using temporal data can improve prediction accuracy alone as well as when used in conjunction with other methods (specifically PINNs).

In this paper, we focus on how to predict power through the mechanical equation for power (Equation 2) to discern whether PINNs are applicable and successful in the context of WPF, specifically when predicting power as a function of torque and angular velocity (Equation 2). We will then explore how the addition of the timescale nature of the data will affect PINN performance, and lastly how a hybrid approach combining the temporal aspect of the data with the PINN will perform. With this work, we hope to resolve the scarcity of PINN applications in WPF and to explore how using simpler temporal models can potentially improve performance accuracy.

3. Approach

To test the effects of adding a physics constraint and temporal dependency on model accuracy, we constructed a control neural network and built each feature we wanted to test upon it. All code to construct these models are available at the repository https://github.com/jl0274/windturbinemodels [12], and because the files are too large to upload to GitHub, all datasets used for this work are available in this Google Drive folder https://drive.google.com/drive/folders/1xE_XN5PlowJKq7WRt_khAWRmARizjEKO?usp=drive_link [13]. The original datasets can also be accessed at this website: https://zenodo.org/records/8213270 [14, p. 1]. First, we constructed a control neural network with no special features; we optimized for the hyperparameters of this control, then proceeded to

use these hyperparameters for all following models. We then constructed a PINN with the added custom loss term based on Equation 2 in the total loss equation. Next, we constructed a neural network that takes in previous timesteps of data as input to predict the immediate future, which we refer to in this paper as our "timestep model" or "timestep NN." This model also had the same architecture as the control network. Lastly, we constructed a hybrid PINN-timestep neural network, which has the same architecture as the control with the addition of the physics constraint as well as the reshaping of the data that allows for the previous timesteps to feed in as input, and the predicted variable is the immediate future.

While the research conducted by Gijón et al. [4] also used the same physics constraint for their PINN model, they only included five variables; our approach takes in 16 independent variables. Taking in more variables as input makes our model thorough and more diverse, hopefully stabilizing its prediction accuracy; at the same time, using variables that are widely available across most SCADA datasets still allows for this work to be replicated on other datasets.

Our approach adds to the replicability of PINNs in the field of WPF. We noticed that there currently only exists one paper that directly predicts turbine power [4], which concluded that the PINN they constructed performed on par with the control neural network that they constructed. By using the same physics equation for the constraint, we aim to answer the question of whether PINNs are feasible and useful for predicting wind turbine power. On the other side of the literature, although many time-dependent models have shown success with predicting turbine power, the main drawback is that these models are costly to train and execute, which is why we offer a simple approach with a feedforward network, where the timesteps are incorporated by reshaping the data so that the previous k rows of input feed into the model to predict power in the

immediate future timestep. Details are explained in the Implementation section, but in short, this type of structure allows for faster training time and runtime compared to more complex, formalized time-dependent models such as LSTM. We aim to use this approach to explore whether such a simple model can achieve the same level of accuracy while reducing training and implementation costs.

This approach is also one of the first to explore the interaction between timestep dependency and physics constraints; while previous literature is saturated with time-dependent models [10], [11] and partially saturated with PINNs, there has been no attempt to bridge the gap in the literature between these two types of models. Therefore, we are interested in exploring the effects of these approaches when combined. Our methodology is designed to answer the question of whether a hybrid PINN-timestep model can improve performance compared to a PINN or timestep model alone.

4. Implementation

4.1 Dataset

It has been established within the last decade that ML methods cannot be generalized on a training set broader than turbines from the same wind farm. In Kariniotakis et al., the authors concluded in a review of offshore wind prediction that WPF cannot be “plug-and-play” since it is always site-dependent [15]. Therefore, we will be conducting a case study on the Björkö Wind Turbine located on the island of Björkö in Sweden [16]. Developed by Modvion for the Chalmers Institute of Technology, this project consists of a single 30m tall turbine with 3 blades and a rated power of 45 kW. This publicly available dataset contains SCADA data spanning from July 5th 2022, to June 9th 2023. The data is originally sampled at 100 Hz and contains meteorological measurements at 30m as well as mechanical measurements of the turbine parts.

Note that the geartrain ratio is a constant property of the turbine, so it was not necessary to include this constant in our data nor in our calculations.

In total, we reduced our dataset from the original 69 variables to the following 16 variables: rotor speed (equivalent to angular velocity), generator temperature, air temperature, yaw position, wind direction, air humidity, pitch angle of each of the three blades, rotor shaft torque, wind direction, wind speed, nacelle wind direction, and nacelle wind speed. Our predicted variable power was taken from the variable labeled "Maximum Power Estimate" in the dataset's metadata. We also reduced the frequency at which the data is being taken from 100 Hz to 1 Hz by averaging every 100 rows, as we deemed upon visual inspection of the predicted variable power that taking the data once per second is sufficient for how much power fluctuates. Lastly, we visually inspected the spread of the power variable and took 100,000 rows where the spread was not too unbalanced:

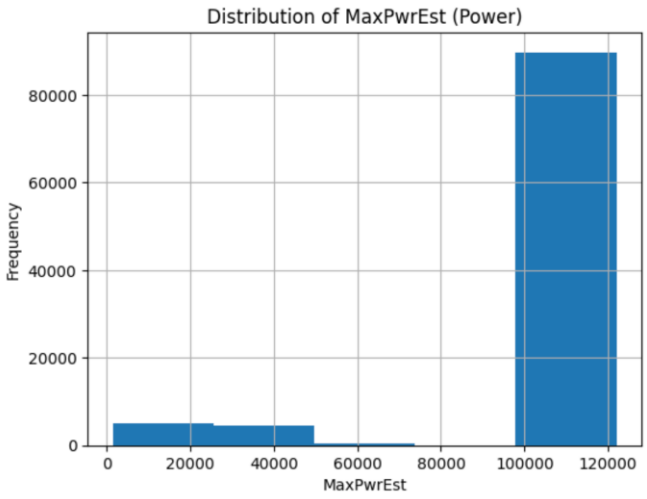

Figure 1: Histogram of power spread over the interval of data taken.

The final shape of our working X dataset was 100,000 rows by 16 columns, and the final shape of our working y dataset was 100,000 rows by 1 column. Note that for each timestep model, we modified the shape of the working dataset in order to incorporate the time-dependent

aspect of the data; exact specifications of the reshaping are explained in the Methodology section.

4.2 METHODOLOGY

We used the Python Tensorflow and Keras libraries to construct our neural networks, and the Pandas and Numpy libraries to structure our data. We used Python's matplotlib libraries to plot our results. For each network, we split the data into a train/test set ratio of 70:30, and for each training and test set, we normalized the data by subtracting the mean then dividing by the standard deviation for each variable. This method of normalization matches that of Gijón et al., as their methodology showed high promise with the results of their PINN.

In our first step which consists of constructing the control neural network, we used the Keras Tuner Hyperband algorithm [17] in order to optimize the following hyperparameters: number of units per layer, number of layers, activation function, and learning rate. This is the same approach as Gijón et al., and we also used the same testing options for each hyperparameter, which is outlined below:

Hyperparameters Optimized with Hyperband Algorithm	
	Values Tested
Number of units per layer	$\{ 8, 16, 32, 64, 128 \}$
Number of layers	$\{1, 2, 4\}$
Activation function	{'relu', 'tanh'}
Learning rate	$\{0.01, 0.001, 0.0001\}$

We tuned these hyperparameters on 10% of our training set data (7% of the total dataset) due to the high training costs for the tuner model. Our resulting optimized model had 64 units, 4 layers, used the reLU activation function and learning rate of 0.001. We used this architecture for all following models to eliminate any confounding variables and test the isolated effect of each feature on performance accuracy. We trained our model over 75 epochs and saw that the loss

stabilized after this period. We found that our loss value on the test set was adequate to proceed with using the same architecture for the other models (MAPE $= 8.780\%$), and as an additional sanity check, we also constructed a simple linear regression on the data to confirm that its accuracy was weaker than our control neural network (MAPE $= 32.153\%$) before proceeding with using the control's hyperparameters for the following models.

Next, our PINN was constructed with the same architecture as the control but with the addition of a custom loss function, which is defined by expressing power as a function of torque (T) and angular velocity (ω) (Equation 2):

$$L_{total} = L_{data} + L_{phys}$$
$$L_{data} = \frac{1}{N_{data}} \sum_{i=1}^{N} |NN_i - P_i|$$
$$L_{phys} = \frac{1}{N_{phys}} \sum_{i=1}^{N} |NN_i - T_i \omega_i| \quad (4)$$

We defined the loss equation like Gijón et al. did to provide the most direct way of forecasting power generation from wind turbine farms, compared to other PINN methods that failed to address the open-ended problem of predicting wind output as a function of other input variables. However, when running the PINN initially, the model failed to converge at all, and upon further examination of the data, we saw that the loss function's predictions for power were inversely correlated with the true power values (visualized in Figure 5), so we added a multiplicative factor and constant to the formula in the form of $a \cdot f(T, \omega) + b$, where $f(T, \omega)$ is the function that calculates power (Equation 2). With this modification, we were able to construct a model that converged over 20 epochs.

Next, we constructed the timestep model, where the only change between this and the control model was that we reshaped the input and output data to match the specifications shown in

Figure 2 below. Formalizing the structure of our timestep model, we reshaped the data for certain values of k such that at any timestep t, the input which originally had NUM_FEATURES columns were reshaped to a DataFrame with NUM_FEATURES $\times k$ columns (one for every variable being taken at each of the k timesteps).

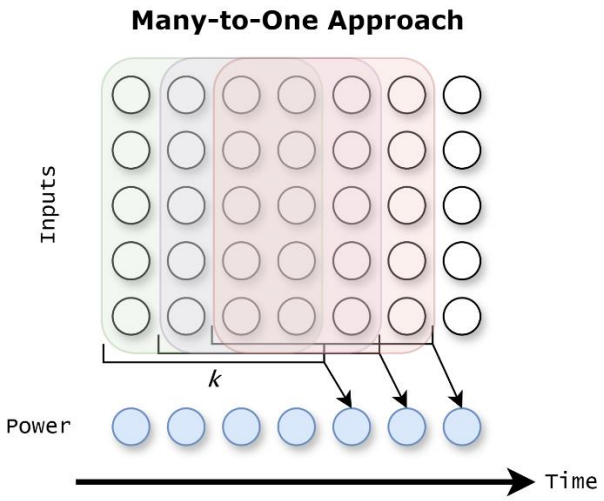

Figure 2: The Many-to-One approach, a simpler approach that allows us to take in any arbitrary interval of timesteps of size k *and feed it as input to predict power in the immediate future. In this figure, we use the previous* k = 4 *steps to predict future power.*

We also coded the reshaping function to ensure that only consecutive time sequences would be considered. We tested values $k \in \{5, 10, 100\}$, where the units of k are the number of seconds in advance that we consider. We originally planned to test k up to a magnitude of 10^3, but there were not enough consecutive time sequences to allow for such large values of k.

Finally, we constructed a hybrid PINN-timestep model; its architecture and loss function remain the same as the standalone PINN, but the only difference is that for the model's input, we reshaped the data for each value of $k \in \{5, 10, 100\}$ and fed it into the PINN. With this model, we sought to determine whether using timestep data in conjunction with PINNs could improve the performance accuracy of the model compared to using any one feature alone.

5. EVALUATION

5.1 Metrics

For each model, we consider the following metrics: loss over epochs, mean squared error (MSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and R-squared value. We chose these metrics because they're in line with what the majority of the WPF research field uses [2], [5], and we chose this breadth of metrics in order to give a more comprehensive, diverse outlook on how each model performed. We visualized the loss over epochs to illustrate if or when each model converges. Note that MAPE and R-squared may be the most interpretable quantified metrics though, as MSE and MAE are in terms of watts, but MAPE is proportional to the true value.

5.2 Results

5.2.1 Control Neural Network

We saw that the loss for our control neural network converged and stabilized nicely at approximately 75 epochs:

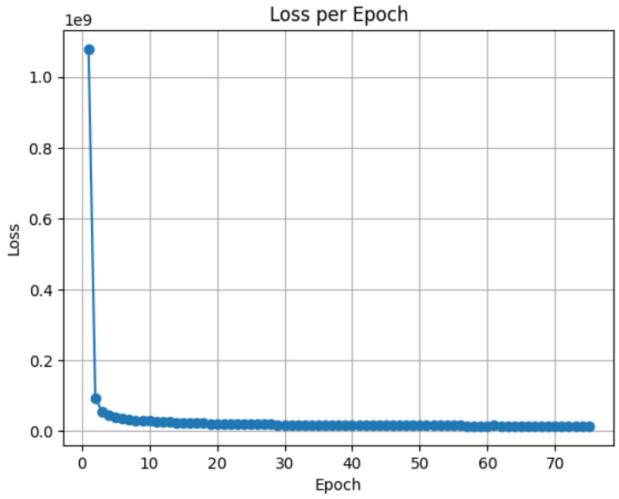

Figure 3: Control NN's loss over 75 epochs.

The control model's quantifiable metrics are as follows:

Control NN Metrics (on Test Set)	
MSE ($Watts^2$)	14048948.081
MAE (Watts)	821.062

MAPE (%)	8.780
R-Squared	0.984

5.2.2 Standalone PINN

Our original PINN model's loss failed to converge over the tested 75 epochs and yielded an R-squared value of -11.988. The first 20 epochs are shown below:

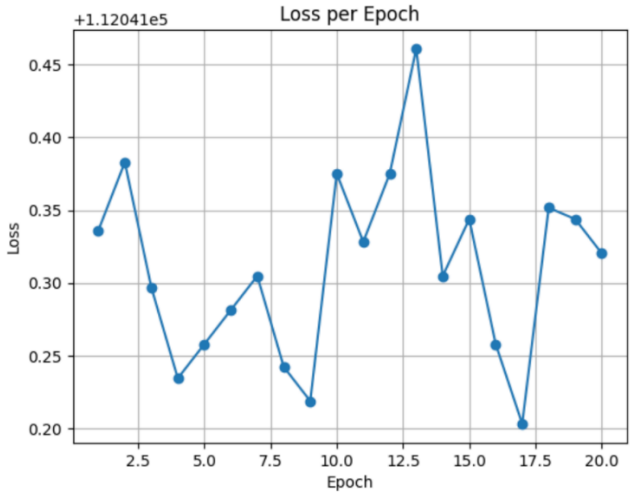

Figure 4: Original PINN's loss over 20 epochs fails to converge.

Upon further visual inspection of the data, we found that the predicted values for power (coming from $P = gT\omega$) were inversely correlated with the true values for power:

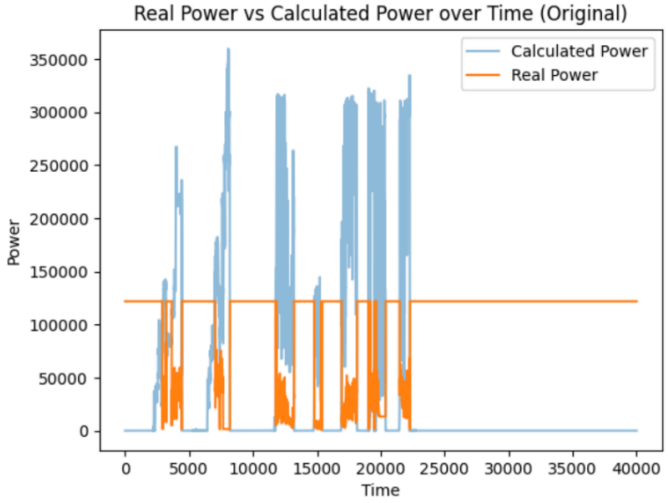

Figure 5: Real power and calculated power using Equation 2 appear to be inversely correlated.

We speculate that the discrepancy in scale may have arisen from a difference in units between the data recorded and the true interactions between the equation for power. However, we are uncertain as to why calculated power is negatively correlated with real power. Further explanation of this occurrence is elaborated on in the Limitations section. To correct for this discrepancy between calculated and real power, we multiplied the calculations for power by -0.4 and added a constant of 122020 to make the baseline of each line the same. The multiplicative factor and additive constant were visually estimated. Our equation then became:

$$P = -0.4(T \times \omega) + 122020$$

With the modified equation, the calculated power became a much more realistic approximation for real power:

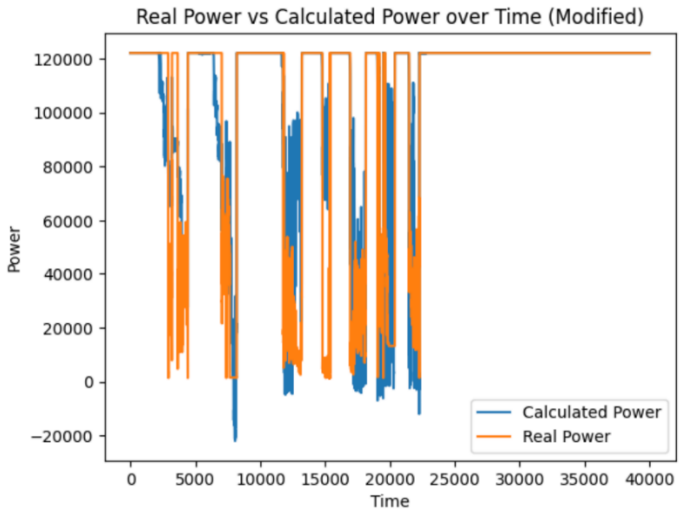

Figure 6: Modified real power versus calculated power.

This modification to the loss function resulted in a more accurate model with a loss that stabilized over 20 epochs, shown below:

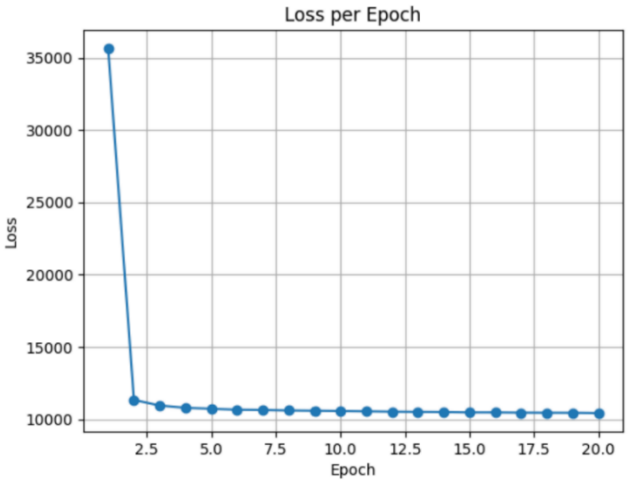

Figure 7: Loss converges after altering the loss function for the PINN.

However, although our quantifiable evaluation metrics show higher accuracy than the original PINN, we still do not see results that are comparable to that of the control NN:

PINN Metrics (on Test Set)	
MSE (Watts^2)	676693355.802
MAE (Watts)	8340.559
MAPE (%)	99.598
R-Squared	0.227

Overall, it's interesting that the loss converges, but the R-squared value shows a weak correlation between predicted and true values for power, and the MAPE also clearly indicates that the model is not accurate enough for reliable turbine power prediction. This suggests that the PINN is not a feasible method for turbine power prediction using Equation 2.

5.2.3 Timestep Neural Network

Our timestep model performed extremely well. The loss converged after 75 epochs for each value of k, shown below:

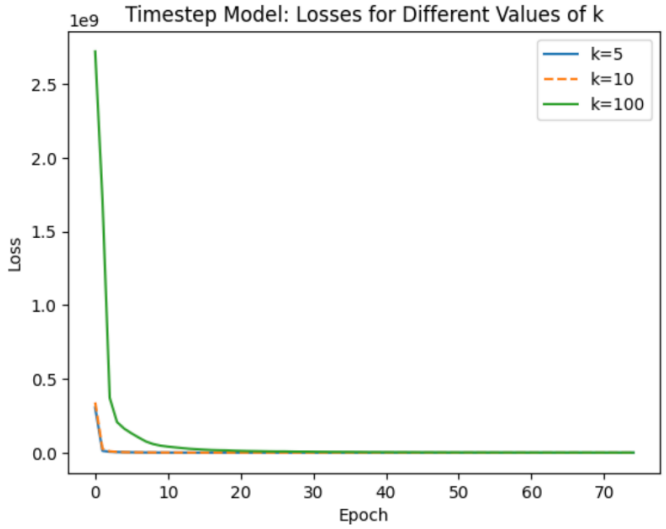

Figure 8: Loss for each iteration of the timestep model converges nicely.

We are unsure as to why the initial loss value is much higher when $k = 100$, although all losses for each value of k stabilizes nicely. We can more closely examine how accurate each rendition of the timestep model is with our quantitative evaluation metrics:

Timestep NN Metrics (on Test Set)			
	$k = 5$	$k = 10$	$k = 100$
MSE (Watts^2)	760783.677	523131.627	8637904.697
MAE (Watts)	68759.231	52960.260	177825.808
MAPE (%)	3.443	3.269	13.831
R-Squared	0.999	0.999	0.990

Overall, we can see that the timestep model performs better than the control NN as well as the PINN for all values of k tested. Notably, the renditions where $k = 5, 10$ perform best, with little discernible difference between them, and the rendition where $k = 100$ performs slightly worse, although still more than adequate for reliable WPF.

5.2.4 Hybrid PINN-Timestep Neural Network

Following gathering the results from both the standalone PINN and the timestep model, we hypothesized that using timestep data in conjunction with PINNs in the hybrid model would improve the performance of the PINN; the results refute this hypothesis, as although the loss

decreases over time and converges for each value of k, the overall accuracy is even poorer than the standalone PINN model.

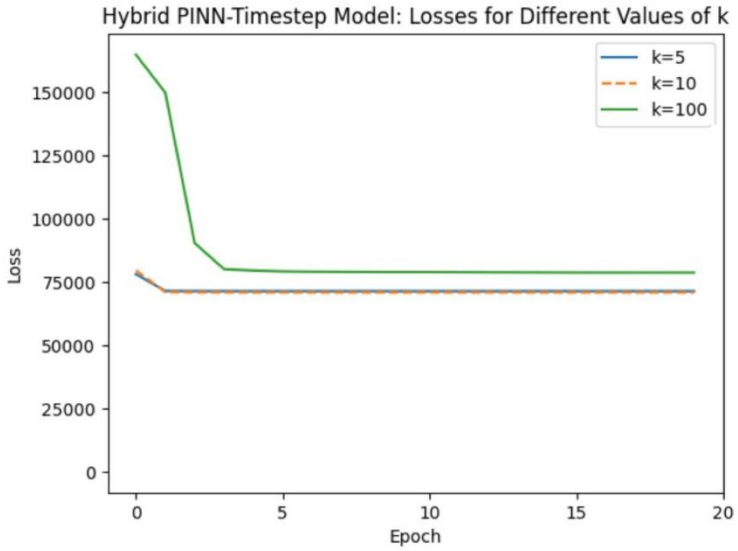

Figure 9: Losses for each rendition of the hybrid model converges but still performs poorly.

Our other metrics are as follows:

Hybrid NN Metrics (on Test Set)			
	$k = 5$	$k = 10$	$k = 100$
MSE (Watts^2)	1975701899.848	1703838604.255	1736113350.769
MAE (Watts)	3700553.650	3402290.844	3587035.690
MAPE (%)	553.199	507.743	509.363
R-Squared	-1.439	-1.128	-0.943

The negative R-squared values across all values of k indicate that each model performs much worse than simply predicting the mean power value every time. Contrary to the original hypothesis, incorporating timestep data into the PINN does not stabilize its performance and actually hinders it compared to the standalone PINN, despite the high performance accuracy of the timestep model on its own. We speculate that the negative R-squared value may have resulted from each effect (timestep and physics constraint) negatively interfering with each other, resulting in an overall less accurate model. We also speculated that this hybrid model's poor R-squared value may have resulted from extreme overfitting to the training data, but upon further

comparison of test to training loss for MAE and MSE, we saw that these metrics showed consistent results across both sets, with variations of approximately 1% observed among their values. This indicates that the performance of the model remained stable regardless of whether acting on the training versus test set, which provides evidence against the overfitting theory. Overall, we saw that the standalone timestep model performed best, while this hybrid model performed the worst, by all available metrics.

5.3 LIMITATIONS

There exist some limitations in this study related to the dataset as well as with the methodology. First, we believe that a more concrete formula is needed to fit the calculated power to real power, since we used visual inspection to adjust calculated power to generally fit over real power (see Figures 5 and 6). Additionally, the question remains unanswered as to why the values were negatively correlated in the first place. Although adding a multiplicative factor and constant could be standard practice with the way the loss function is designed, the fact that the multiplicative factor is negative is puzzling. With our limited knowledge in the field of wind turbine maintenance and SCADA data, we guessed that the negative correlation between calculated power and real power could be an issue with units, although the details of this interaction remain unclear. Perhaps individual differences in how SCADA data is maintained could also have led to differences in how calculated power vs real power is expressed. There also could have been an issue with the way the variable MaxPwrEst is derived. This dataset uses a variable called MaxPwrEst instead of Power like what most SCADA datasets use, and the metadata doesn't have more clarification to offer on how MaxPwrEst is calculated or differs from real power other than the variable description "Max Power Estimation: FFR regulator calculated variable" [16]. The fact that this variable is the maximum power estimation rather

than true power could also have interfered with the effectiveness of using the physics constraint to regularize the model. More expertise in the wind turbine maintenance field, and specifically with the creation of this dataset, would be helpful in solving this question.

We optimistically selected this dataset despite the uncertainty behind the MaxPwrEst variable because no other feasibly available dataset existed that measured both torque and kept track of the data with respect to time. Looking at adjacent work's datasets also proved to be unhelpful, as the dataset that Gijón et al. used has been taken off from public access, and the cleaned dataset that they posted on their GitHub repository has no timestamp values for each row of data. Therefore, we conclude that constructing PINNs that predict power with Equation 2 and use time-dependency is constrained by the limited types of data currently available in the field.

6. Conclusions and Future Work

In the past decade, wind energy has proven to be one of the most promising sources of clean energy, and applying ML to improve the prediction accuracy of wind turbine power output has become extremely relevant to increasing global confidence in wind energy. We designed our work with the goal of contributing to improving prediction accuracy in the field of WPF and to making wind energy more reliable as an energy source. We hypothesized that both PINNs and timestep-dependent neural networks have high potential to improve the field of WPF due to their successful applications in surrounding fields predicting wind patterns as well as predicting wind output directly [4], so we designed our work with the aim of answering the following questions. First, to what extent can PINNs successfully improve the prediction accuracy of a neural network model? Second, can we improve the performance of the PINN by creating a hybrid neural network model that takes a simple timestep-dependent data modification into account? To answer these questions, we designed a series of neural networks that tested both the physics constraint and timestep data in isolation, then in conjunction with each other.

Although our standalone PINN performed poorly, we found the timestep model to be extremely successful for all values of k tested, which shows promise for future work surrounding simpler timestep architectures. The best renditions of the timestep models occurred when $k = 5, 10$ and achieved MAPEs of 3.4% and 3.3% respectively. The hybrid model performed the worst out of all models constructed ($R^2 < 0$ for all values of k), suggesting that PINNs should not be used in conjunction with timestep data. The reason as to why this is the case is unclear, but further research on a more trustworthy dataset might provide a clearer picture of the factors at play. This table below synthesizes the most interpretable metrics used across all models:

Metrics for Each Model Using Best Value of k (on Test Set)				
	Control NN	PINN	Timestep NN ($k = 10$)	Hybrid PINN-Timestep Model ($k = 100$)
MAPE (%)	8.780	99.598	3.269	509.363
R-Squared	0.984	0.227	0.999	−0.943

Note that we used $k = 100$ as the best-performing timestep interval for the hybrid model because although the model performed similarly for values $k = 10$ and $k = 100$ based solely on MAPE, the R-squared value was less negative when $k = 100$, which is why we chose those results as our best rendition of that model.

Whether predicting power as the product of torque and angular velocity (Equation 2) is actually feasible for generating an accurate prediction model remains unclear, due to confounding factors with the dataset variable MaxPwrEst. So far, we have either seen that PINNs achieve performance that is on par with what typical neural networks can achieve, as seen in the work by Gijón et al., or performance that is poor as seen in this work [4]. Therefore, in terms of future work, we propose that performing the same methodology on a dataset that has a more transparently calculated power variable would be helpful in determining the feasibility of PINNs in WPF. A dataset with longer consecutive time sequences that allow one to test for larger values

of k would also be interesting to work with. With more funding and time, it would be more feasible to search for such datasets that have all the variables necessary for this type of work. It would also be interesting to explore how simpler timestep models such as the one in this work perform against more complex time-dependent models, such as LSTM or GRU. Simpler models that have the same accuracy as more complex ones would have the advantage of having lower training and computational costs, which could speed up and ease prediction which could be performed millions of times a day for energy grid mix allocation. Long-term WPF (defined in the field as 24H ahead or more) could also potentially be improved with the assistance of PINNs or simple timestep models, as long-term prediction is notoriously volatile. The addition of a regularization term such as Equation 2 might improve accuracy to long-term prediction models. Overall, this work explores novel prediction methods in the field of wind power forecasting and contributes towards guiding the literature towards more accurate and practical methods for predicting turbine power.

REFERENCES

[1] Global Wind Energy Council, "Global Wind Report 2023." Apr. 2023. [Online]. Available: https://gwec.net/wp-content/uploads/2023/04/GWEC-2023_interactive.pdf

[2] K. L. Jørgensen and H. R. Shaker, "Wind Power Forecasting Using Machine Learning: State of the Art, Trends and Challenges," in *2020 IEEE 8th International Conference on Smart Energy Grid Engineering (SEGE)*, Aug. 2020, pp. 44–50. doi: 10.1109/SEGE49949.2020.9181870.

[3] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nat. Rev. Phys.*, vol. 3, no. 6, Art. no. 6, Jun. 2021, doi: 10.1038/s42254-021-00314-5.

[4] A. Gijón, A. Pujana-Goitia, E. Perea, M. Molina-Solana, and J. Gómez-Romero, "Prediction of wind turbines power with physics-informed neural networks and evidential uncertainty quantification." arXiv, Jul. 27, 2023. Accessed: Nov. 04, 2023. [Online]. Available: http://arxiv.org/abs/2307.14675

[5] A.-N. Buturache and S. Stancu, "Wind Energy Prediction Using Machine Learning," *Low Carbon Econ.*, vol. 12, no. 01, pp. 1–21, 2021, doi: 10.4236/lce.2021.121001.

[6] Y.-J. Ma and M.-Y. Zhai, "A Dual-Step Integrated Machine Learning Model for 24h-Ahead Wind Energy Generation Prediction Based on Actual Measurement Data and Environmental Factors," *Appl. Sci.*, vol. 9, no. 10, Art. no. 10, Jan. 2019, doi: 10.3390/app9102125.

[7] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next," *J. Sci. Comput.*, vol. 92, no. 3, p. 88, Jul. 2022, doi: 10.1007/s10915-022-01939-z.

[8] J. Zhang and X. Zhao, "Spatiotemporal wind field prediction based on physics-informed deep learning and LIDAR measurements," *Appl. Energy*, vol. 288, p. 116641, Apr. 2021, doi: 10.1016/j.apenergy.2021.116641.

[9] X. Li and W. Zhang, "Physics-informed deep learning model in wind turbine response prediction," *Renew. Energy*, vol. 185, pp. 932–944, Feb. 2022, doi: 10.1016/j.renene.2021.12.058.

[10] M. Paula *et al.*, "Predicting Energy Generation in Large Wind Farms: A Data-Driven Study with Open Data and Machine Learning," *Inventions*, vol. 8, p. 126, Oct. 2023, doi: 10.3390/inventions8050126.

[11] W.-H. Lin, P. Wang, K.-M. Chao, H.-C. Lin, Z.-Y. Yang, and Y.-H. Lai, "Wind Power Forecasting with Deep Learning Networks: Time-Series Forecasting," *Appl. Sci.*, vol. 11, no. 21, Art. no. 21, Jan. 2021, doi: 10.3390/app112110335.

[12] "jl0274/windturbinemodels." Accessed: Apr. 21, 2024. [Online]. Available: https://github.com/jl0274/windturbinemodels

[13] "COS IW: Wind Turbines - Google Drive." Accessed: Apr. 21, 2024. [Online]. Available: https://drive.google.com/drive/u/1/folders/1xE_XN5PlowJKq7WRt_khAWRmARizjEKO

[14] S. Fogelström, H. Johansson, and O. Carlson, "Björkö Wind Turbine Version 1 (45kW) SCADA." Zenodo, Aug. 01, 2023. doi: 10.5281/zenodo.8213270.

[15] G. Kariniotakis, P. Pinson, N. Siebert, G. Giebel, and R. Barthelmie, "The state of the art in short term prediction of wind power - from an offshore perspective," presented at the SeaTech Week - Ocean Energy Conference ADEME-IFREMER, Oct. 2004. Accessed: Oct. 26, 2023. [Online]. Available: https://minesparis-psl.hal.science/hal-00529338

[16] “Björkö Wind Turbine Version 1 (45kW) high frequency Structural Health Monitoring (SHM) data”, doi: 10.5281/zenodo.8230330.
[17] K. Team, “Keras documentation: Hyperband Tuner.” Accessed: Apr. 16, 2024. [Online]. Available: https://keras.io/api/keras_tuner/tuners/hyperband/