By Spyros Chatzivasileiadis, Andreas Venzke, Jochen Stiasny, and Georgios Misyris

Machine Learning in Power Systems

Is It Time to Trust It?

WE EXPERIENCE THE POWER OF MACHINE LEARNING (ML) in our everyday lives—be it picture and speech recognition, customized suggestions by virtual assistants, or just unlocking our phones. Its underlying mathematical principles have been applied since the middle of the last century in what is known as *statistical learning*. However, the enormous increase in computational power, even in devices as small as a smartphone, has enabled significant advances and wide adoption of ML in nearly every part of our lives and the scientific world.

Still, despite the large body of academic literature on ML approaches for power system operation during the past 30 years, there is only an extremely limited set of approaches that has found application in practice. From them, the vast majority is related to load forecasting, such as the Artificial Neural Net

Digital Object Identifier 10.1109/MPE.2022.3150810
Date of current version: 19 April 2022

Short Term Load Forecaster, a tool developed by the Electric Power Research Institute in the United States; and a few approaches have been related to decision trees for security assessment. The main reason for the slow adoption of ML in the power industry is that most ML approaches, such as neural networks and others, are considered a black box.

Who would trust a black box to avoid a blackout or to find an optimal operating point that will not violate any line limits? Who would trust a black box for any safety-critical application, e.g., in energy, health care, or automobiles? The risk is just too high.

What does it take for ML to get adopted? For insight into the adoption of novel methods, not only related to ML, consider the following two aspects:

1) To what extent does the industry already have solutions to a problem?
2) What is the risk associated with using a novel method instead of a conventional one?

A novel method will only be seriously considered if it falls below a risk threshold. If it satisfies the risk requirements, its better performance, such as higher accuracy or computation speed, can lead to adoption. Figure 1 presents an illustration of these considerations.

ML, and artificial intelligence (AI) in general, has been particularly successful in areas where there are little to no established solution methods. AlphaGo is an AI program developed by DeepMind Technologies that harnessed the complexity of the game Go and repeatedly beat the human world champion and other software programs. By itself, this feat is impressive if we consider that Go is an ancient Chinese game with an astronomical number of possible combinations of player moves (2.1×10^{170}), vastly greater than the number of known atoms in the known, observable universe (1×10^{80}). AlphaFold is an AI program that predicted, better than any other program, protein structures that were impossible to predict for the past 50 years. Both AlphaGo and AlphaFold yielded results never before possible. Because the risk of catastrophe associated with their task was nonexistent, their adoption was extremely rapid. On the other hand, self-driving cars, which heavily use ML algorithms to detect surroundings, are also attempting to solve an unsolved problem, but in several cases, and despite significant progress, we are still above the risk threshold when it comes to their widespread deployment without the involvement of a human driver.

Power systems are different. We have spent the past 100 years trying to understand how electricity flows along power lines and what happens in the voltage and current right after a disturbance. And we have managed that quite well. The models, despite being computationally intensive, can predict well how voltage and current behave in many operating conditions. But compared to a few decades ago, power systems have become significantly more complex. The connection of millions of power electronic devices leads to rapidly evolving phenomena, requiring the inclusion of more complex models and a faster decision-making process. Distributed renewable generation and electric vehicles add a lot of uncertainty and create thousands of new injection points that continuously change the balance between energy supply and demand. This all means one thing: if a system operator wants to make sure that no blackout happens (a procedure called *security assessment*), he or she needs to run not only more complex models but also for a lot more scenarios at faster paces than before. A security assessment that once took a few hours to carry out would now need a few days, which is unacceptable for meeting real-time grid-operational requirements.

ML methods, on the other hand, can continuously learn and adapt to their environment and are extremely fast when computing an output. Such practices have been shown to outperform conventional techniques, e.g., predictive maintenance of transmission lines and transformers, or smart charging of electric vehicles. ML approaches are gradually being adopted for such applications because of their low risk to power system operation. However, handling the sheer complexity of power system operation procedures to avoid blackouts (security assessment) or determine an optimal operating point without violating any single operational constraint is still too "safety critical" to accept an ML-based solution. This, despite early promising results reported in the research literature that ML can help. Power system operators find it difficult to trust methods they do not understand, and which have thus far provided no performance guarantees.

Researchers have recently been working to address this skepticism and remove barriers, allowing ML to enter power system applications, exploiting its benefits. In the first transition period of using ML tools for power system safety-critical operations, the authors view ML algorithms as "assistants" to established procedures in the form of a

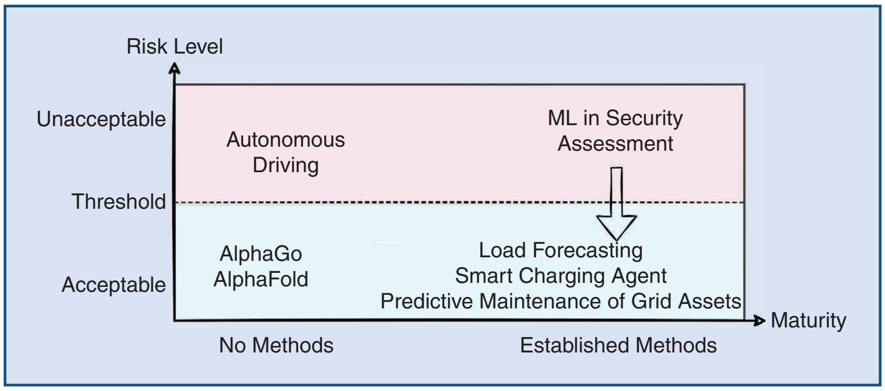


figure 1. The adoption of novel methods (blue area) in light of the existence or nonexistence of alternative solution techniques, and the risks the novel processes pose.

decision-support tool. The ultimate responsibility for the final decision remains with the operator. ML algorithms can be extremely fast to estimate, for example, whether operating points are safe, as ML methods can assess thousands of scenarios at the same time, whereas conventional tools can assess only a few. This allows the system operator to screen an extremely large number of scenarios and determine the few critical ones in a very short period of time. Following this step, the operator can then use conventional techniques to study, in more detail, those few scenarios and determine what drives them to instability and when.

The approach is similar when we apply ML, and AI in general, for optimization. Neural networks can determine a very good estimate for an optimal operating point. We can then assess, with conventional tools, whether the point violates any constraints. This way of approaching AI can drive wider adoption of AI tools for system operation, at least in the first transition period. But even before this step, the AI tools need to build trust. If operators and other users do not trust them, they will not even be used for screening.

The rigorous methods this article describes aim to deliver the guarantees (through performance certificates) that will allow users to trust AI tools for safety-critical operations. Besides the benefits these certificates give us in terms of trusting AI tools and determining the worst-case performance, they can help provide better insights into how the resulting AI model can be improved. The techniques discussed in this article fall into the broader spectrum of transparent AI (the results of the AI process are explainable) and interpretable AI (the accuracy of a model to associate a cause to an effect), initiatives recently launched in the AI community to bring a deeper understanding of the inner workings of ML algorithms.

Several algorithms under this umbrella deliver rigorous approaches to understand how different features of the problem at hand affect the performance of the applied neural network. The algorithms we discuss go a step further and deliver performance guarantees. To enhance the performance of AI tools, this article also presents ways to take advantage of available power system models developed for decades based on first principles. Through that, neural network applications can better avoid overlooking critical cases or giving predictions that are largely at odds with governing physics. In short, these possibilities are enabled by the following two approaches:

1) *Verification*: This approach delivers certificates on how the neural network will behave, i.e., what its output will be for all possible inputs. Thus far, researchers have been assessing the neural network performance purely statistically, naïvely hoping that testing an application on a large set of random samples can accurately capture the behavior of the neural network model over the whole input space.
2) *Physics-informed neural networks*: These networks include the governing physical equations inside a neural network training procedure. Instead of generating sets of relevant data in a process to train the model, which can be computationally intensive, integrate a physics-based model into the training process and let the neural network learn from it.

Before diving into these approaches in the next sections, we first introduce the type of ML tools that are the focus of this article: neural networks and their training procedure. In the rest of this article, we use the terms *AI* and *ML* interchangeably; in reality, *AI* is a more general term that includes all ML methods and tools.

How Do Neural Networks Work?

Neural networks belong to the most promising group of AI tools, proving their ability to approximate any function (universal approximator). They have received considerable attention in recent years. The two main categories are neural networks for classification and regression tasks.

Classification neural networks are used in a variety of applications. In self-driving cars, for example, neural networks continuously receive pictures of their surroundings from car cameras and answer simple questions such as, "Is this a red traffic light?" A classification neural network shall return a yes or a no, and even if it is not quite sure, it must choose between the two. For power systems, such a classification task can be "Will this combination of generator set points and loads—called an *operating point*—lead to a blackout or not?" We can ask this question for many possible operating points (see the "Possible Inputs" bubble in Figure 2), and through the neural network, we can obtain an answer for each point extremely fast. We can label the resulting regions "safe" (blue dot) or "not safe" (red dots), as shown in the "Predicted Classifications" bubble in Figure 2.

If we want to assess the predictions of our neural network, we obviously cannot check for every single operating point as this requires immense computing resources. Instead, we sample a fraction of operating points and perform a conventional security assessment (e.g., run power flow, time-domain simulations, or an eigenvalue analysis) and associate a true label with each point, such as truly safe or truly unsafe. A good neural network will match (i.e., predict) these true labels as closely as possible.

Instead of a binary response (yes/no), regression neural networks use almost the same training process as the classification neural networks but yield a continuous function value. These networks are widely used, for example, to forecast load demand or predict the power output of wind turbines and solar photovoltaics based on weather conditions, including the movement of clouds, temperature variations, and others. In a power system operation context, a regression neural network can predict a cost-optimal operating point or how the value of frequency evolves after a line outage. As presented in Figure 2, in the case of frequency evolution, a neural network predicts a continuous function of the input to output variables over time (dashed orange line) that approximates the "true" trajectory (dashed blue line). A good neural network will do so accurately.

Classification and regression neural networks have the same structure (see the right side of Figure 2). Neurons are organized in layers, with each layer connected in series with each other. Each neuron contains a nonlinear activation function. The neurons in each layer are connected through linear functions with some or all of the neurons of the previous layer and the next layer. In this simple structure, the linear functions contain parameters we can adjust, whereas the nonlinear activation function in each neuron remains unchanged.

In many neural network implementations, neurons act as a switch, either allowing the signal to pass or restricting it from doing so. Because of these nonlinear activation functions, neural networks can represent very complex, and sometimes not even explainable, nonlinear processes. The magnification of a single neuron in Figure 2 shows one option for how such a nonlinear function can look, but there are many more. With the appropriate selection of linear

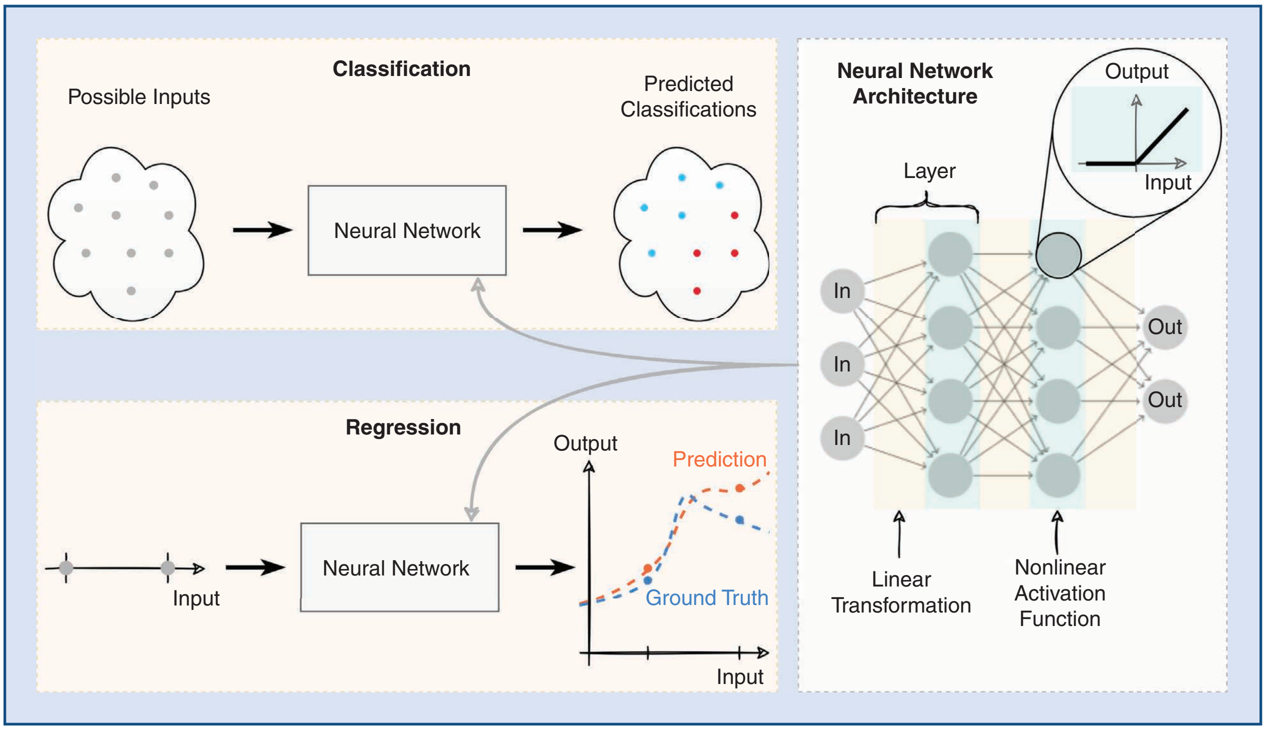


figure 2. The basic structure of a classification and regression task, and the fundamental architecture of a simple neural network that forms the core of either task.

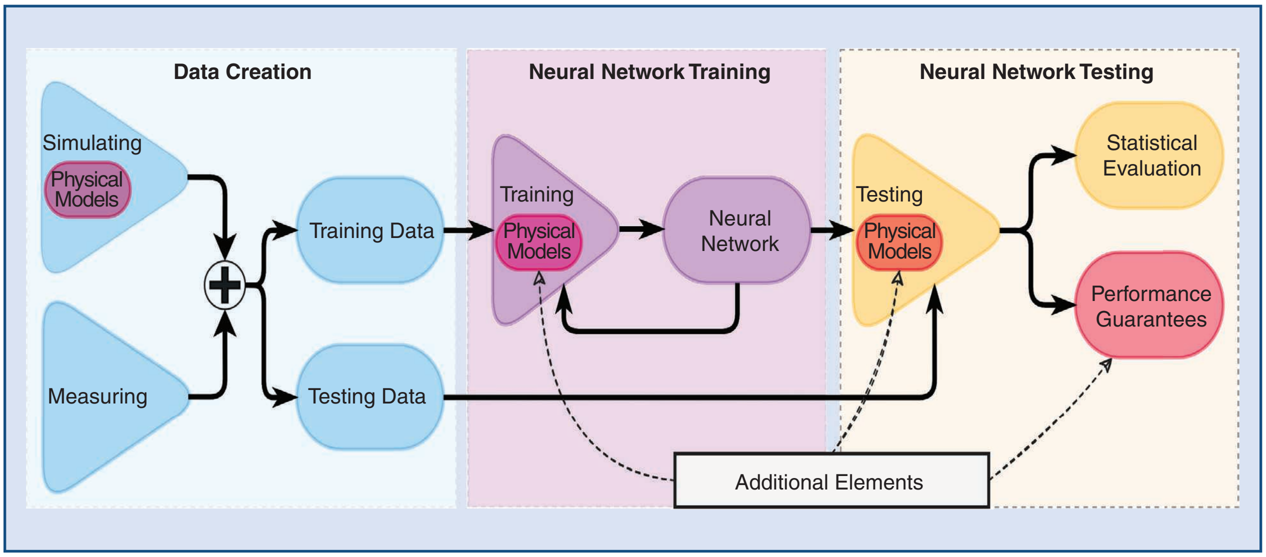


figure 3. The classical ML procedure, and the new elements for improving the procedure.

function parameters and neural network size, in theory, the network has the power to approximate any possible function or process.

The process of determining these parameters is commonly referred to as the *training* of a neural network. It is a part of a general ML algorithm, together with the creation of the training data set beforehand and the testing carried out afterward. Figure 3 illustrates this procedure and highlights where new elements of neural network verification and physics-informed neural networks enter the process.

Before neural network training, we start by putting together a data set with both actual measurements and simulation results (if the measurements are not enough). This data set is usually a representative scenario, yet it is a small fraction of the possible scenarios that could occur. It contains both the inputs and the outputs, e.g., what is the operating point and whether it is "safe" or "unsafe?" The goal is that through this data set, the neural network learns the relationship between inputs and outputs so that when training is over, it can correctly estimate the output for any unseen input not included in the training data set. To assess how well the neural network predicts that output, we use a separate test data set to statistically determine the neural network's performance. The test data set is often a smaller part of the one generated for training but kept aside to use only for testing.

The next step in the ML algorithm is to determine the neural network's parameters so that predicted outputs match the true ones from the training data set as closely as possible. The procedure is an iterative optimization algorithm, where we pass batches of data several times through the neural network until we arrive at a good match between network outputs and those in the training set. When we are satisfied with the performance on the training data set, or we reach a predetermined time limit, we fix the parameters.

After training comes testing. Depending on how accurate the neural network's predictions are on a random set of previously unseen test samples, we conclude how "good" the network is compared to others. But what happens if the samples do not reflect reality? What if the neural network achieves 99% accuracy but the test set fails to include critical cases or even some high-risk outliers? How can an operator trust this performance index? We address this in the next section by describing methods that do not depend on the test data set to determine the network performance.

The creation of a neural network is purely based on data and statistical learning. One could even say that training a neural network is an advanced form of nonlinear regression. Physical models have thus far not been involved in training. In traditional ML algorithms, a physical model influences only the data-generation stage, either because data were generated through simulations using first-principles models or the data were collected from the actual physical process. The physical models, in this case, are represented in a discrete form through data points but not in equations. In the following sections, we explain how we adjust this process to add the physical models inside the neural network training in the form of continuous equations. But first, we focus on how to obtain rigorous performance guarantees that are not based on statistical indices and are not dependent on the quality of the test set.

Guarantees of Neural Network Behavior

Thus far, conventional methods evaluate how well a neural network performs by measuring its performance on a test set (e.g., accuracy and other statistical indices). However, this process is purely statistical and cannot tell us with certainty what the prediction will be for any points not included in this set, which is crucial for any safety-critical applications. The goal is to develop techniques that evaluate whole continuous regions instead of just discrete points when testing a neural network. This allows us to consider any possible point within an input region and guarantee the neural network performance for any point in this region. We achieve that using optimization methods. As it turns out, we can rewrite the equations that define a neural network such that they can be incorporated into an optimization problem. Due to this reformulation, we can now analyze a neural network in a completely new way.

Methods such as the ones mentioned in this article can provide system operators with guarantees about how a neural network will behave for entire regions of power system operation. At the same time, they eliminate the dependency on the quality of the test data set. Instead of having to sample a potentially extremely large number of test data to cover all possible scenarios (and still not extract any guarantee or provide any certainty), we can now solve a single optimization problem for one continuous part of the input region.

To illustrate this framework, we examine the following two vital questions, one each for a classification and regression task:

1) How large is the continuous input region for which the neural network classification remains the same? The answer provides a guarantee that any input in this region will be classified to a specific known class. Let us consider a power system. Such a guarantee can state that the all operating points where generator one is between 0 and 200 MW, generator two is between 0 and 100 MW, and the load is between 0 and 300 MW will be classified as "safe" by the neural network.
2) What is the largest prediction error of the neural network across a continuous input region (regression)? Considering power systems, assume a simple example: we have two generators serving a load over two transmission lines; we train a neural network to output the combination of generator set points that result in the minimum cost 1) for a load that varies between 10 and 300 MW and 2) without violating the transmission capacity of any line. A worst-case guarantee

can indicate that across all combinations of generator set points (i.e., one combination for 10 MW, one for 10.1 MW, and so on), there is a 5% maximum violation of the line transmission limit.

The methods we describe allow for such an assessment. If the performance is unsatisfactory, we can take a step back, retrain the neural network on more data points, and re-evaluate the performance. This improves the accuracy and robustness of the prediction, ultimately delivering better guarantees and enabling the adoption of such techniques.

Secure or Not Secure? Guaranteeing How Neural Networks Classify

Let us consider a guiding example: we want to determine whether a specific operating point of a given power system is safe under a combination of security criteria (e.g., N-1 criterion, small-signal stability, and so on). Instead of using conventional methods (e.g., power flows and eigenvalue analyses), we train a neural network to assess whether the operating point is secure or not secure in a fraction of the time (usually 100 to 1,000 times faster). As inputs, we provide the active power set points of generators and active and reactive power consumption of loads. The boundary where the classification changes from secure to not secure is called the *security boundary*. The correct prediction of that boundary is crucial because the neural network could misclassify a "not secure" operating point as "secure," posing a risk to power system operation and potentially leading to a blackout.

Using this guiding example, Figure 4(a) and (b) shows a comparison of the standard procedure used to evaluate the performance and the proposed methodology, respectively. The conventional procedure relies on a test data set of probable power system operating scenarios. For this test data set, the neural network classification is compared to the ground truth, assessing classification accuracy.

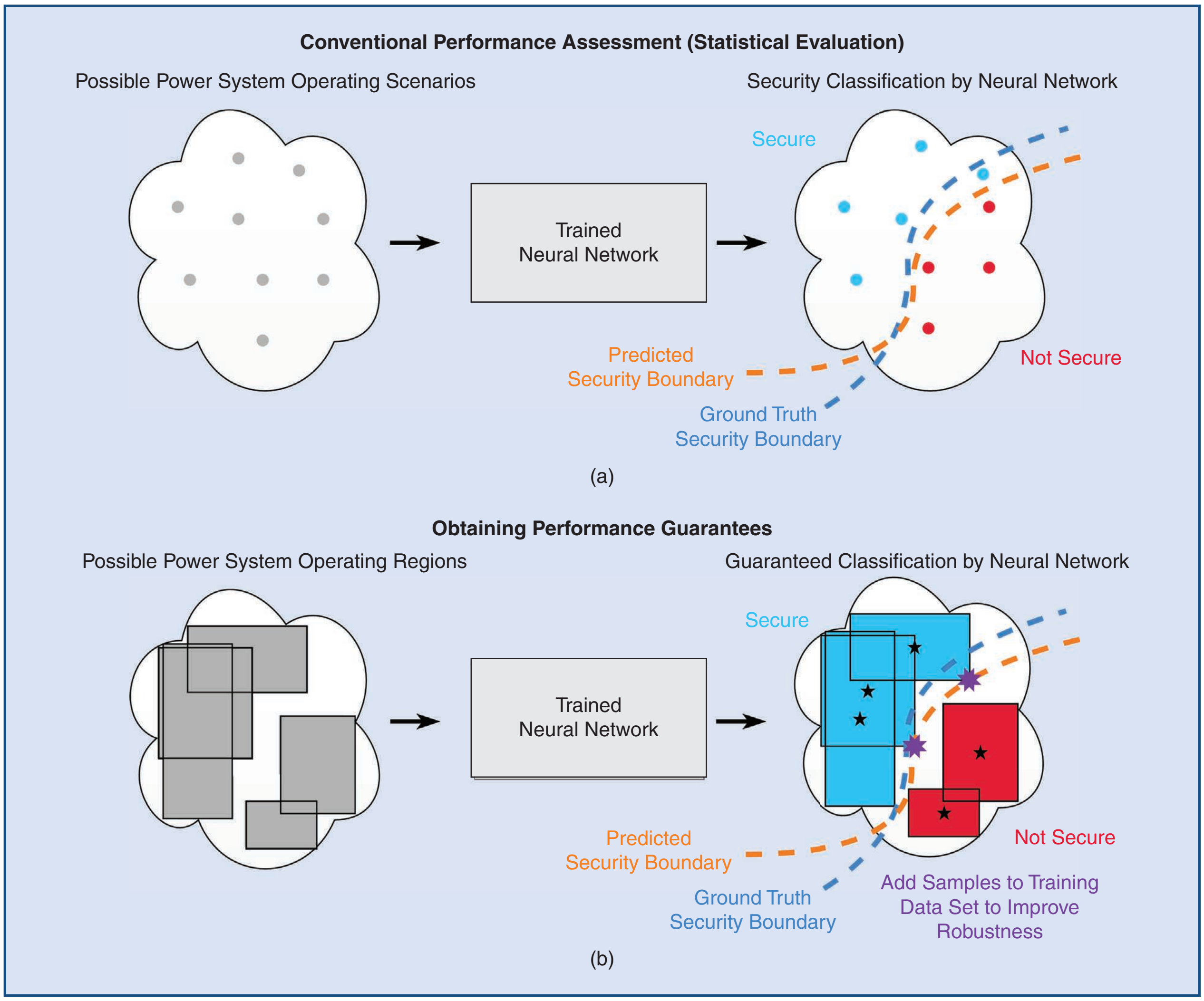

Conventional Performance Assessment (Statistical Evaluation)
Possible Power System Operating Scenarios
Security Classification by Neural Network
Trained Neural Network
Secure
Predicted Security Boundary
Not Secure
Ground Truth Security Boundary
(a)
Obtaining Performance Guarantees
Possible Power System Operating Regions
Guaranteed Classification by Neural Network
Trained Neural Network
Secure
Predicted Security Boundary
Not Secure
Ground Truth Security Boundary
Add Samples to Training Data Set to Improve Robustness
(b)

figure 4. A comparison of the standard evaluation of the neural network performance with the new proposed methodology. (a) The conventional procedures use a test set, and (b) the proposed techniques obtain performance guarantees for continuous input regions.

The drawback of this approach is that it requires many samples to build confidence in the correctness of the network prediction, including prediction of the security boundary. As illustrated in Figure 4, using only a small number of samples could suggest that the boundary is predicted accurately, although there may still be an undiscovered mismatch. Related to this is that even for a larger number of samples, an effective sampling strategy is difficult to devise along the security boundary, especially if many power system states are considered. A second shortcoming of the traditional evaluation is that there are no guarantees the neural network classification will not falsely change between discrete samples.

Instead, we apply a new method that reformulates the neural network to an optimization problem. By solving it, we can certify how the neural network classifies entire input regions (grey rectangles), i.e., as secure or not secure [Figure 4(b)]. The procedure works as follows:

- ✔ First we take a discrete operating point. This could be a random sample from the training set or any other operating point for which we know it is secure (the process is very similar for points that are not secure).
- ✔ Then we solve an optimization problem to compute the closest input for which the neural network prediction changes from secure to not secure.

The distance between the reference point and this input determines the size of the region around the reference operating point where the classification is guaranteed not to change. In other words, the result of this optimization supplies us with the guarantee that all the points inside this region will be classified by the neural network as secure. This is represented as the colored rectangles in Figure 4 (i.e., each rectangle corresponds to a different region we certify). Using that, the operator can start trusting the neural network as they can now anticipate how it will behave for any possible point in the certified regions.

The neural network is no longer a black box. We can repeat this process for many reference points until we obtain several regions with certified mapping and unveil the neural network behavior across the input domain to system operators.

This procedure can also be used to evaluate the neural network robustness illustrated in Figure 4. For the point we determined through our optimization, we can also compute a ground-truth classification and assess whether the neural network predicts the security boundary correctly.

Thus far, existing approaches could only evaluate misclassification through random sampling. This is the first time that a systematic procedure can measure how well a neural network predicts a power system's security boundary. If we are not satisfied with the network performance, we can add the misclassified points to the training data set with the correct ground-truth labels and retrain the neural network to improve its robustness. This is a systematic and well-defined procedure that can be repeated until we reach a desired level of robustness. The moment the operator has the performance certificates and is satisfied with the neural network performance is when we will see neural networks applied in safety-critical power system applications, such as security assessment.

Worst-Case Guarantees: What Is the Worst Neural Network Prediction Error?

Let us consider a guiding example where a neural network acts as an optimal power flow algorithm. It predicts how much controllable generators should produce for a given load situation so that 1) the production cost is as inexpensive as possible and 2) there is no violation of system constraints, such as transmission line loading limits. Existing optimal power flow algorithms can perform this task quite well. However, the optimal power flow problem in its full nonlinear form is still a major challenge to solve in a reasonable time. Well-trained neural networks can determine a solution up to 1,000 times faster and consider constraints that are impossible to include directly in conventional optimization methods (e.g., differential equations representing stability constraints). Now imagine that we need to run thousands of scenarios through an optimal power flow algorithm, considering different load profiles and highly varying injections from renewables, and determine the cost-optimal and safe generation dispatch for all of them. That is when neural networks can be quite helpful.

We have to keep in mind, however, that a neural network does not necessarily predict a solution that respects all system constraints and is also the least expensive option. Its strength is its extremely fast computation, not necessarily its accuracy. When testing the network, we want to ensure that predictions 1) do not significantly violate constraints (e.g., do not breach more than 1% of the maximum limits) and 2) remain close to the optimal solution from a cost perspective (e.g., they determine a solution that does not cost more than 1% of the optimal cost).

Similar to classification neural networks, the standard approach used to assess network performance after training is to take all samples of the test data set, compare the network prediction to the physical ground truth, and compute whether and to what extent constraint violations occur for specific samples (see Figure 5(a)]. As an additional metric, we can also calculate the Euclidean distance of the neural network prediction to the ground-truth optimal point for these points to calculate the "optimality" of the prediction. Reviewing all test data set samples, we can then easily find the maximum of these violations and maximum "optimality distance." However, this process does not give any guarantee that the worst prediction that can occur was found. With conventional neural network assessment methods, the only option for improving this is to evaluate the metrics on even more data samples. Beyond a point, the procedure becomes computationally expensive and still does not yield certainty that an even worse prediction has not been missed.

A recently introduced method, however, bypasses all these barriers as it eliminates the need to evaluate discrete test samples. The trained neural network is reformulated to a set of linear equations with continuous and binary variables to set up an optimization problem. Inside the optimization problem, we also include the physical power system modeling equations. In essence, these act as a representation of the ground truth. Note that this procedure is applied only to extract worst-case guarantees. As soon as the user (e.g., power system operator, trader, and so on) is satisfied with the neural network performance, the trained neural network can be deployed for real application.

The goal of this method is to find the maximum constraint violation, which can now run across the whole continuous input region [Figure 5(b)]. If we find that the maximum constraint violation is zero, we can then certify that for the entire input region, the neural network predictions will never violate the constraints. If the maximum constraint violation is not zero, we then obtain worst-case guarantees, i.e., we determine the maximum violation that the neural network prediction can result in for any possible input it can receive. This then becomes a powerful tool that can build the trust lacking from grid operators. When considered from a risk assessment perspective, these metrics can decide whether the network's performance is good enough or whether further training and evaluation are required to reach an acceptable performance.

Physics-Informed Neural Networks

The previous sections focused on how to provide guarantees about the performance of a trained neural network, building the trust of system operators and other users. In this section, we discuss how neural networks can take advantage of the decades-long development of physical power system models. Including this knowledge in neural network training yields significantly better performance with much fewer data. These are so-called physics-informed networks.

First let us look at a concrete example. Assume we want to determine the evolution of power system frequency under a disturbance. The models that describe this evolution often use differential equations that express how the system changes from one instance to the next (the simplest of which is a "swing equation" that many power engineers learn during their studies). To solve a system of differential equations, we usually employ computer software and numerical solvers which, in most cases, means that we track in small steps how the system state (e.g., the frequency) changes over time. The resulting trajectory can become a very complex function, possibly requiring significant computational resources. And the rapidly increasing integration of power electronic

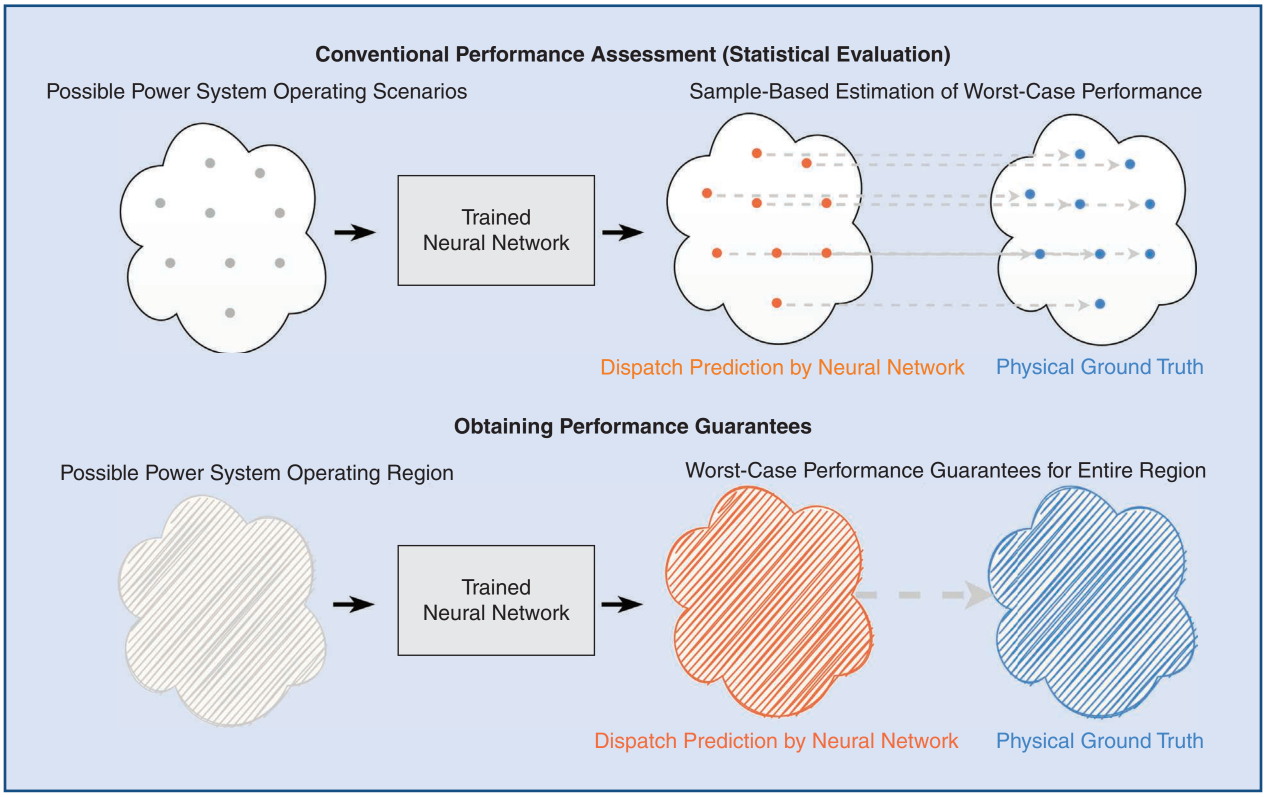

Conventional Performance Assessment (Statistical Evaluation)
Possible Power System Operating Scenarios
Sample-Based Estimation of Worst-Case Performance
Trained Neural Network
Dispatch Prediction by Neural Network
Physical Ground Truth
Obtaining Performance Guarantees
Possible Power System Operating Region
Worst-Case Performance Guarantees for Entire Region
Trained Neural Network
Dispatch Prediction by Neural Network
Physical Ground Truth

figure 5. A comparison of the (a) standard evaluation of neural network performance using a test set with (b) the proposed methodology for obtaining guarantees of neural network behavior. The proposed method can determine the largest deviation of the neural network prediction from the physical ground truth across the entire region (and not only for random discrete samples): this offers a rigorous, worst-case guarantee.

converters and renewables makes this computational effort significantly larger for the following two reasons:

1) Power electronic converters (e.g., from wind turbines, solar photovoltaics, batteries, high-voltage dc lines, electric vehicles, and so on) have much more complex models based on differential equations, with time constants that interfere with the electromagnetic transients of transmission lines.
2) Renewables add significant degrees of uncertainty with their fluctuating production, meaning that we need to assess a much larger set of possible operating points for each disturbance.

Neural networks could assist here by substituting traditional numerical solvers and providing a fast estimate of the actual solution (up to 1,000 times faster). This creates the following two major benefits:

1) A much faster estimate (within a few milliseconds) of what the frequency will be 2 or 5s after the disturbance, enabling the activation of necessary counteractions earlier.
2) The ability to assess up to 1,000 potentially critical disturbances in the same time frame that conventional software will calculate only a single disturbance. This allows us to screen a very large set of disturbances for many different operating points and select only the most critical ones, which we can then assess in more detail and with higher accuracy than with conventional software.

To train such a neural network with conventional methods, we need to provide data points (the black dots in Figure 6) and fit network parameters so they provide a prediction that matches the data points. As the training process progresses, the prediction function (orange line) becomes more sophisticated and fits the data points better. However, when comparing the prediction to the function that produced the data points (i.e., ground truth), which is governed by the model's differential equations, we see clear mismatches for all three examples. The prediction in Figure 6(c), in particular, highlights what is referred to as *overfitting*. The prediction matches all data points nearly perfectly but shows large approximation errors between the data points.

Now imagine that we want our neural network to predict a series of different trajectories that correspond to different disturbances and operating points. Poor fitting or overfitting becomes a major issue when checking whether the underfrequency or overfrequency thresholds have been violated. One way to achieve a better fit is to use more data points from more trajectories. However, this can be difficult if, for example, the data are obtained from measurement devices that do not sample the frequency to an adequate degree of granularity. It also can be computationally expensive if we need to generate more trajectories with simulation software to obtain more points.

Recent advances in neural network training, however, offer us a new way. We now can numerically compute the derivatives of neural network outputs with respect to inputs during the neural network process. If the frequency is a neural network output, we can calculate, for example, the derivative of the frequency over time inside the network training. This enables us to include differential equations, such as the swing equation, inside the neural network training to drive the training procedure to a neural network where the frequency (which is a neural network output) and its derivative validate the swing equation. This additional requirement leads to a much better approximation of the true functions.

Figure 7 (dashed black lines) shows the derivatives at selected points. Intuitively speaking, by adding physical equations inside the neural network training, the network prediction needs to cross a data point (black dot) and create a trajectory that fits the "shape" of its neighborhood.

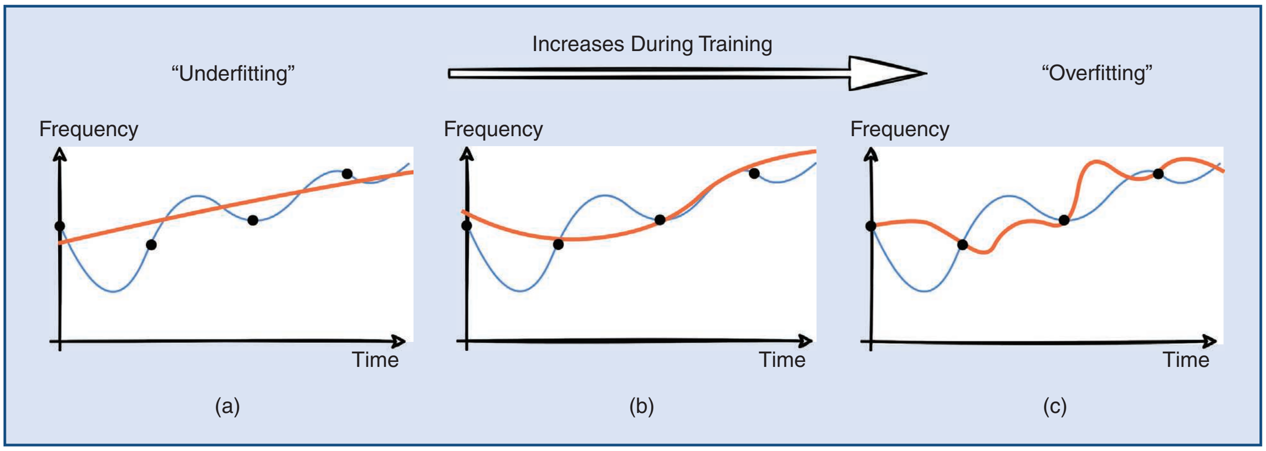
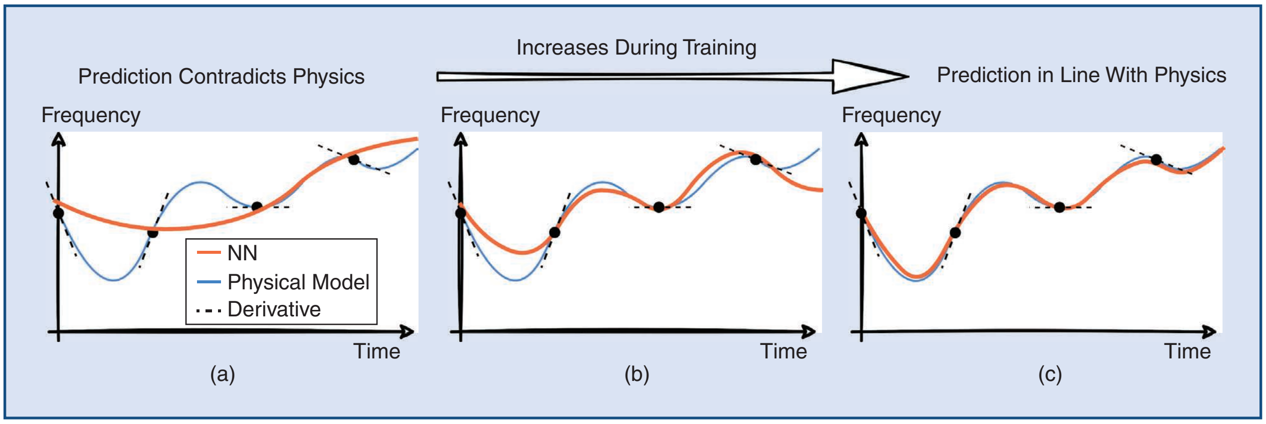


figure 6. The training process of a simple regression neural network, showing the evolution of network output during training (orange) versus ground truth (blue). (a) At the start of the training, the neural network output is "underfitting." (b) As the training progresses, the neural network output fits the data points better. (c) If the training continues, the neural network output is "overfitting."



Increases During Training
Prediction Contradicts Physics
Prediction in Line With Physics
Frequency
Frequency
Frequency
NN
Physical Model
Derivative
Time
Time
Time
(a)
(b)
(c)

figure 7. The training process for physics-informed neural networks, showing the evolution of neural network output during training (orange) versus ground truth (blue). NN: neural network. (a) At the start of training, the neural network output contradicts the physical model. (b) As the training progresses, the neural network output fits the physical model better. (c) After sufficient training, the neural network output is in line with the physical model.

Figure 7(b) and (c) show how the neural network adjusts its parameters so that the derivatives of the predicted trajectory can fit what the differential equations describing the system dictate.

Physics-informed neural networks may lead to much better predictions while requiring much fewer training data. This can remove significant barriers. Instead of creating large training databases by randomly sampling thousands of trajectories to train a network with sufficient accuracy, we can now directly include governing equations in the training process and let the neural network train until it minimizes the prediction error.

Where Does This Lead Us?

The two directions we presented, neural network verification and physics-informed neural networks, aim to build trust with grid operators, utilities, and other potential users and improve network performance while requiring much fewer data. They arguably remove the most important barrier when it comes to the safety-critical operations of energy systems: ML tools no longer need to be considered a black box; instead, they can be trusted. Such methods can also further help to develop systematic approaches that identify regions where the neural network demonstrates poor performance and then systematically improve it through retraining.

There are still many steps needed until ML tools are embraced inside the control room. For instance, scalable algorithms that can apply to very large neural networks and power grids need to be developed. Expertise in power systems, operations research, and ML is necessary to arrive at production-level algorithms. Once achieved, gaining the trust of system operators will be a deciding step toward the wider adoption of ML algorithms across energy systems. This includes safety-critical operations, where ML can drastically accelerate computation speed and enable the management of millions of controllable appliances and converter-connected devices.

For Further Reading

A. Venzke and S. Chatzivasileiadis, "Verification of neural network behaviour: Formal guarantees for power system applications," *IEEE Trans. Smart Grid*, vol. 12, no. 1, pp. 383–397, Jan. 2021, doi: 10.1109/TSG.2020.3009401.

A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, "Learning optimal power flow: Worst-case guarantees for neural networks," in *Proc. 2020 IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, pp. 1–7, doi: 10.1109/SmartGridComm47815.2020.9302963.

G. S. Misyris, A. Venzke, and S. Chatzivasileiadis, "Physics-informed neural networks for power systems," 2020, in *Proc. IEEE Power & Energy Society General Meeting*, 2020, pp. 1–5, doi: 10.1109/PESGM41954.2020.9282004.

L. Duchesne, E. Karangelos, and L. Wehenkel, "Recent developments in machine learning for energy systems reliability management," *Proc. IEEE*, vol. 108, no. 9, pp. 1656–1676, 2020, doi: 10.1109/JPROC.2020.2988715.

V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," 2017, arXiv:1711.07356.

M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving non-linear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019, doi: 10.1016/j.jcp.2018.10.045.

Biographies

Spyros Chatzivasileiadis is with the Technical University of Denmark, Kongens Lyngby, 2800, Denmark.

Andreas Venzke was previously with the Technical University of Denmark, Kongens Lyngby, 2800, Denmark.

Jochen Stiasny is with the Technical University of Denmark, Kongens Lyngby, 2800, Denmark.

Georgios Misyris was previously with the Technical University of Denmark, Kongens Lyngby, 2800, Denmark.

p&e