Online Testbed for Evaluating Vulnerability of Deep Learning Based Power Grid Load Forecasters

Himanshu Neema, Peter Volgyesi, Xenofon Koutsoukos
Vanderbilt University
Nashville, TN 37235

Thomas Roth, Cuong Nguyen
National Institute of Standards and Technology
Gaithersburg, MD 20899

***Abstract*—Modern electric grids that integrate smart grid technologies require different approaches to grid operations. There has been a shift towards increased reliance on distributed sensors to monitor bidirectional power flows and machine learning based load forecasting methods (e.g., using deep learning). These methods are fairly accurate under normal circumstances, but become highly vulnerable to stealthy adversarial attacks that could be deployed on the load forecasters. This paper provides a novel model-based Testbed for Simulation-based Evaluation of Resilience (*TeSER*) that enables evaluating deep learning based load forecasters against stealthy adversarial attacks. The testbed leverages three existing technologies, viz. *DeepForge*: for designing neural networks and machine learning pipelines, *GridLAB-D*: for electric grid distribution system simulation, and *WebGME*: for creating web-based collaborative metamodeling environments. The testbed architecture is described, and a case study to demonstrate its capabilities for evaluating load forecasters is provided.**

***Index Terms*—power grid, load forecasting, machine learning, security, resilience, adversarial attacks, model-based testbed**

I. Introduction

This work has been motivated by our NSA Science of Security Lablet research efforts to create executable simulation models and repeatable experiments for evaluating potential vulnerabilities and successful resilient strategies for complex Cyber-Physical Systems. To address these needs we developed a web-based, cloud-hosted design environment and integrated state-of-the-art simulation engines for multiple CPS domains (highway and railway transportation, power distribution). This paper focuses on the power grid domain of our multi-model testbed.

In electrical grids, the power generation is typically conducted on-demand, which requires utilities to continuously forecast the grid loads [1]. The loads are estimated for the long-term (LT) (i.e., more than a year), medium-term (MT) (i.e., a month to a year), and short-term (ST) (i.e., an hour to a week). LT forecasts are used for planning the necessary generation, transmission, and distribution equipment. MT forecasts are used for adjusting the LT plans. ST forecasts are used for real-time grid operations and operating both the grid and power markets in a safe, secure, and reliable manner.

ST and MT forecasting have become challenging due to the dynamic and distributed nature of modern electrical grids. In the traditional grid, power is centrally generated, and the power flow is unidirectional from generation to transmission to distribution network. Smart grid technologies enable the integration of distributed energy resources (DERs) that provide local sources and introduce bidirectional power flow into the system. Additionally, most DERs are variable sources such as wind turbine and solar photovoltaic (PV) that are not always available due to changing weather conditions such as storms. Because of capability for bidirectional power flow and the variable nature of DER, grid operations need to evolve from a deterministic to a stochastic model.

Smart grid with DER integration have enabled consumers to generate power locally and provide excess generated power back to the electric grid for a financial benefit. In addition, the potential dynamic pricing of electricity necessitates the use of transactive controllers and smart demand management to get the best pricing of power (e.g., move consumption to off-peak hours). This stochastic environment has made load forecasting significantly challenging where traditional demand-supply and failure modeling is no longer suitable.

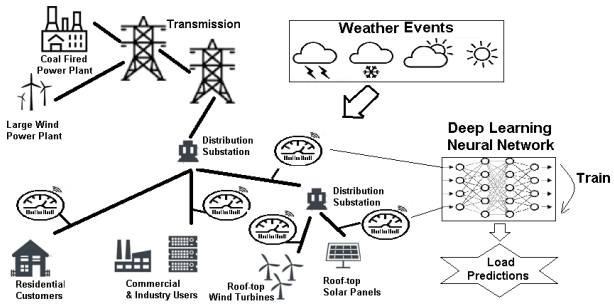


Figure 1: Deep Learning Based Load Forecasting in a Dynamic Power Grid with Distributed Energy Resources

To address the above challenges, smart grid deploys a large number of smart sensors that collect power flow readings at various points in the network. The measurements from these sensors can be used by deep learning based load forecasting systems to estimate the expected system loads. Figure 1 illustrates the variety of factors affecting ST load forecasting and how a deep learning system can be used to predict future loads. The sensor readings recorded by smart meters (generated by real system or a simulation) are stored in a time series database, which is used by a neural network to learn expected load on the system. Deep learning based forecasting is fairly accurate under normal circumstances, but, due to its complexity, becomes highly vulnerable to stealthy adversarial attacks. These

attacks [2] subtly modify some of the sensor readings (by intercepting and forwarding modified data) such that not only the attacks remain undetectable by anomaly detection software, but also the accuracy of the load predictors is significantly reduced. As the adversarial attacks are not easily detected, they can result in cumulative disruptions leading to significant damage and loss before the system could be reverted to fallback methods.

Adversarial machine learning has been studied in many different categorization and estimation problems [2] [3]. Investigation of its use in cyber-physical systems (CPS) is recent, but is quickly emerging as an active research field [4]. The cyber-physical nature of smart grid makes it possible to attack specific cyber components such that the resulting failures cascade to a much wider region of the grid [5]. Further, attackers can exploit the growing networking capability of monitoring and control equipment to remotely attack specific components. Because the electric grid is one of the national critical infrastructures, it is crucial to study the vulnerability of different deep learning based forecasting methods earlier in the design cycle to avoid large-scale failures in a deployed system.

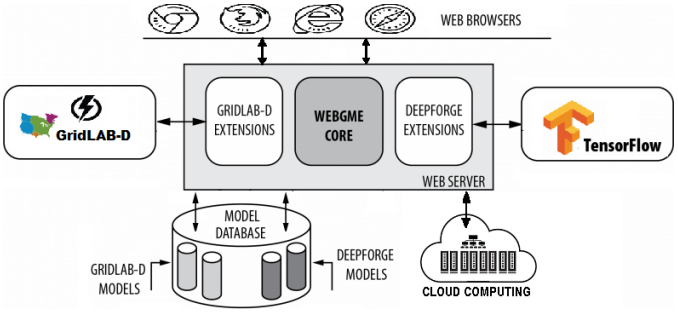


Figure 2: TeSER Testbed Architecture

In this paper, we describe a model-based Testbed for Simulation-based Evaluation of Resilience *(TeSER)* [6] that can analyze the resilience of load forecasters in the presence of stealthy adversarial attacks. As shown in Figure 2, the modeling front-end is built using the open-source Web-based Generic Modeling Environment (*WebGME*) [7] that supports creating web-based collaborative metamodeling environments. WebGME's plugin architecture enables interpretation of models and generation of system artifacts (e.g., source code, configuration files, and others), which can be used to execute experiments on a compute platform (e.g., desktop, server, or cloud). The stealthy adversarial attacks in this research were modeled using *DeepForge*, an open-source web-based environment for deep learning that enables collaborative modeling of neural networks and machine learning pipelines for reproducible deep learning experiments [8]. The testbed also utilizes an open-source power distribution system simulator, *GridLAB-D* [9], for simulating the electric grid and generating power flow data from smart sensors. In addition, the web-server contains an integrated *model database* that stores models of the electric grid, neural networks, and machine learning pipelines. TeSER executes the experiments on a connected *cloud computing* platform that enables the large-scale computations required by these experiments. The experiment results are collected and presented to the user as both raw data and digestible plots for analysis. The user is also provided with a full record of machine learning training iterations and console logs of the machine learning pipelines. We argue that by enabling earlier detection of the vulnerabilities in deep learning based load forecasting systems, our framework can help to both minimize the associated cost and make these systems more effective.

In the rest of the paper: Section II provides the motivation for analyzing vulnerabilities of load forecasters; Section III describes the core architectural components of TeSER; Section IV gives a case study to demonstrate TeSER's capabilities; and Section V concludes the paper and highlights future work. Note that a more detailed experiment—analyzing various attacker-defender games—is given in [10], this paper focuses on the architectural aspects of the testbed.

II. Motivation

Smart grid is a complex example of CPS [5] where the physical and computational components interact in specific ways to determine overall system functionality. For effective smart grid operations, a range of sensors are used to monitor aspects of the grid such as power flows at different locations, line continuity, equipment and device failures, actuator positions, and thermal characteristics. Load forecasting is essential for proper planning of the electrical system to determine the power equipment required and their arrangement, minimize system overloads, reduce power losses, manage operations effectively, and maintain the balance of supply and demand [1] [11].

Owing to bidirectional power flow and increases in DER integration, the smart grid has become highly dynamic, which directly affects the accuracy of load forecasts. The system dynamics are further affected by factors such as the variations in weather conditions and energy usage at different times of the day, price fluctuations in power markers, and a deeply integrated mix of residential, commercial, and industrial loads. Traditional methods of load forecasting use deterministic approaches (e.g., expert surveys and scenario-based assessment) and quantitative methods (e.g., time series analysis, smoothing averages and trend projections, least square estimates, and regression analysis). Since these methods tend to fit load expectations into a trending model, they do not work for highly dynamic variations in smart grid network topology (e.g., bidirectional power flow) and power supply and demand (e.g., DERs and time of use rate). Machine learning techniques can effectively handle these cases—where continuously updating the model is neither plausible nor pragmatic. The deep learning methods can be used for predicting loads in a more reliable manner while minimizing cost.

Deep learning methods, however, suffer from the *black box* problem, where there is no direct relation between inputs and outputs. Moreover, the electric grid has become more connected as the interactions between sensors, actuators, and controllers are largely enabled through a cyber communication network. This makes smart grids that use deep learning methods for

load forecasting vulnerable to cyber-attacks and cascading failures. There are several examples where significant damage to the electric grid has occurred [12]. Therefore, the smart grid must be secured from potential cyber-attacks on deep learning based load forecasting methods. Stealthy adversarial attacks can intercept inputs of deep learning systems and subtly modify them to impact the system without being detected by anomaly detectors. These types of attacks on deep learning methods have been researched in the past, but as the deep learning methods are only recently being used in a CPS context, a testbed to enable such investigations is needed.

This paper describes a novel web-based and cloud-deployed testbed that can evaluate the vulnerability of deep learning models, that rely on a network of sensors for their inputs, to stealthy adversarial attacks. Specifically, the testbed is applied to load forecasters based on deep learning that use power flow readings from smart meters as their input. TeSER uses GridLAB-D for simulating an electric distribution system, and DeepForge for designing adversarial attack models and anomaly detectors. To avoid detection, the adversarial attacks are limited to modify the sensor readings within a given lower and upper threshold [13]. The testbed enables modeling of attacker-defender interactions as a *Stackelberg* game [4], where the defender uses a random subset of sensor readings and a neural network for load prediction, and then the attacker modifies some of those sensor readings within a configurable lower and upper bound. In addition, several load forecasters can be designed for introducing uncertainties that defend the load forecasting system against stealthy adversarial attacks.

III. TeSER Components & Features

The goal of TeSER is to provide a collaborative design and experimentation environment for evaluating the security and resilience of CPS amidst various cyber attack and defense strategies and the impact of these strategies on the physical infrastructure. The testbed leverages open-source technologies to build design tools that are reusable and configurable. It is web-based, cloud deployed, and supports real-time collaboration among researchers and analysts on models and experiments. In addition, it stores all input data, model parameters, and simulation results in the models, and version controls the models for experiment repeatability and provenance. This section provides an architectural overview of TeSER's core components and key features (as summarized in Figure 3).

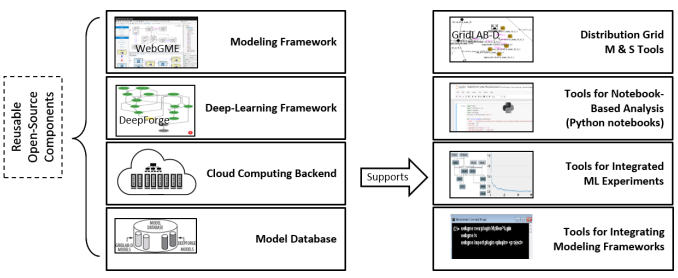


Figure 3: TeSER Core Components and Features

A. Modeling Framework

WebGME allows both the creation of rich, domain-specific modeling languages (DSMLs) and the use of those DSMLs to create domain models. It supports creating plugins that interpret the domain models, generate related system artifacts (e.g., source-code, scripts, configuration files), execute code on integrated compute platforms (e.g., cloud), collect experiment results, and display them to the users as digestible charts/plots and downloadable artifacts. For the modeling and experimentation front-end, WebGME's decorators and visualizers enable custom visualization of models, and the integrated console logging and feedback notifications allow users to get insights into long-running applications in the backend. A key aspect of WebGME is that it is web-based, highly scalable, and supports real-time collaboration among researchers and analysts who can edit the system and experiment models simultaneously from different locations using various web-browsers. WebGME stores all models in a MongoDB database and provides full version control and change tracking. Once designers create system models, analysts can use them to experiment with different designs. TeSER leverages WebGME for power distribution grid and deep learning models, but other CPS domains, such as transportation and healthcare, can also be supported using WebGME's extensible architecture.

B. Deep Learning Framework

DeepForge is built using WebGME and supports rapid development of neural-network and machine learning (ML) models. Modeling in DeepForge uses four main concepts, viz. *Operations*: atomic functions that accept named inputs and generate named outputs; *Pipelines*: specific ML activities such as training, data processing, and predicting; *Executions*: run-time instances of pipelines; and *Jobs*: run-time instances of operations along with associated execution metadata. As shown in Figure 4, the left side is a popular neural network model of a long-short term memory (LSTM) autoencoder for time series forecasting. The right side shows some of the reusable operations for creating ML pipelines (e.g., *GetLocalPredictor* implements a prediction routine and *PlotOperation* plots the time series data). A load forecasting processing pipeline is shown in the middle. Note, that the contents of some compound layers—most notably the LSTM blocks—are not modeled in DeepForge but directly mapped to classes in the Keras library.

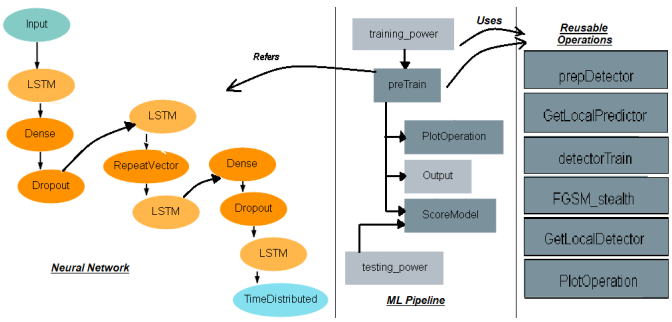


Figure 4: Deep Learning Framework

DeepForge allows users to add their own reusable operations in the library, and reuse pipeline models for creating variant or derived models. The neural network models are stored in a separate library for reuse. A neural network model can be used in multiple pipelines, and a single pipeline could use multiple neural network models. The integrated cloud backend is capable of executing several pipelines in parallel. DeepForge supports detailed views into pipeline execution progress by providing console logs that show the training iterations and by displaying integrated plots about training, testing, and prediction results. Another important feature of DeepForge is that it continuously aligns the Python code with corresponding pipeline models. The user can edit either the Python code or the pipeline model independently and DeepForge automatically synchronizes the other. In addition, all test data and models are stored in the connected artifacts store. With these comprehensive features and integrated experimentation support tools and compute infrastructure, DeepForge provides a powerful web-based environment for deep learning that TeSER leverages for creating its deep learning framework.

C. Distribution Grid Modeling & Simulation Tools

TeSER leverages a web-based platform from prior work [14] and WebGME for building its collaborative tools to support evaluation of distribution grids with integrated DER and their resilience against cyber and physical attacks. As shown in Figure 5, first a user either creates a new grid model or imports an existing GridLAB-D file and updates it as needed. Next, it provides *player* files (timestamped values of grid objects as input) and *recorder* files (specification of object values to collect as output). Finally, the *weather* files, if needed, can be supplied. The plugins are used to interpret the models and configurations to generate artifacts that are sent to a *Simulation Driver and Event Manager* module, which orchestrates the power grid simulation in the cloud accordingly, and gathers feedback from the executing experiments. The generated statistics specified in the recorder files are collected and returned to the user when the simulation completes. Continuous feedback is given while the simulation is running. TeSER uses this tool for simulating power distribution grid and generating smart meter recordings as input data for the deep learning models.

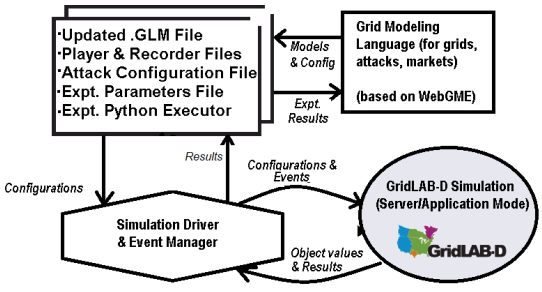


Figure 5: Distribution Grid Modeling & Simulation

D. Cloud Computing Backend

TeSER currently supports ML experiments in the power grid domain. The distribution grid simulation can take a long time for large grid models (e.g., with 1000+ nodes) or when multiple simulations are needed for different training inputs. The deep learning pipelines can also be computationally expensive, but multiple ML pipelines can be executed in parallel for faster responses. For these reasons, it is necessary to integrate a scalable and powerful compute platform. TeSER supports an integrated cloud computing backend that is hosted at Vanderbilt. The cloud also provides capability to store large datasets for ML pipelines and results of previous executions of pipelines. Users can login and inspect all of the executions ran previously (which could have taken hours or even days) including the console logs of all iterations, generated result files, and plots. Further, the user can investigate the execution results step-by-step and even re-execute any step (i.e., job).

E. Model Database

For web-based ML experimentation environments, it is crucial that models are version-controlled and accessible directly from the modeling frameworks. TeSER relies on the WebGME supported MongoDB object-oriented database. However, collaboration is highly challenging for large models (e.g., 1000+ node grid) as every small change in the interconnected graphical model can amount to a large update to be broadcasted. WebGME solves this issue by utilizing a Git-like commit architecture for model changes and only broadcasting deltas to all the modelers. The commits also enable fine-grained change tracking and allow relatively easier merging of models when a conflict arises. The WebGME model databse is used in TeSER to store not only the models of power grid, neural networks, machine learning pipelines, but also to store data and result artifacts. This greatly enhances the reproducibility and provenance of experiments. This database can also be queried using WebGME command-line tools, which enables automated experiments as well as testing.

F. Tools for Notebook-Based Analysis

CPS contain many interconnected physical and cyber components and their experimentation generates a large amount of data, which requires a rich framework for automated analysis. In the machine learning communities, Jupyter Notebooks are popular as they have integrated Python interpreter and analysis tools. TeSER has integrated Jupyter Notebooks in the WebGME front-end to facilitate analysis of data through Python scripts. As mentioned in Section III-B, DeepForge keeps the pipeline models and their corresponding Python code side-by-side and synchronized. The integration with Jupyter Notebooks allows embedding those Python scripts in Notebook cells and testing the corresponding pipeline models in an automated manner. For Notebook-based analysis, the user generates Jupyter Notebook files by executing TeSER plugins on a loaded model (or WebGME's Python libraries could be used to query the model from a user-created Notebook). TeSER also runs a Jupyter Notebook server besides the WebGME server and accesses

it using a WebGME visualizer. The Notebook server enables users to develop scripts and algorithms for analyzing experiment results. As the Notebook server is accessed using WebGME visualizer, that can also update the model based on analysis results received from the Notebook server. Note that the notebook-based approach is optional for more advanced post-simulation analysis tasks.

G. Tools for Integrated ML Experiments

Experimentation with complex CPS not only requires parametric variations and cyber defense and attack combinations (i.e., design of experiments), but also an integrated deep-learning framework for analyzing ML algorithms. TeSER currently supports the power grid domain. It integrates GridLAB-D simulator for power distribution simulation and frameworks, such as Keras and TensorFlow, for creating deep-learning models. At present, the data from grid simulation is fed into deep learning models manually. However, we are working on integrating the two modeling environments into a single framework. This will support automated workflows of power grid simulation, generation of simulation data and its feed into corresponding ML pipelines. The pipeline execution results could also be fed back to update the grid model and close the loop. Figure 6 shows the overall architecture for integrated ML experiments. Here, the training operation in the ML pipeline refers to the corresponding grid model. The DeepForge code generator converts the pipeline model into an executable job script and the GridLAB-D Extension code generator generates templates to configure the simulation driver. We are developing an API for orchestrating the integrated ML experiments, which can execute grid simulations using an integrated simulation driver (like in III-C) and run the ML pipelines using the simulation data generated from grid simulations.

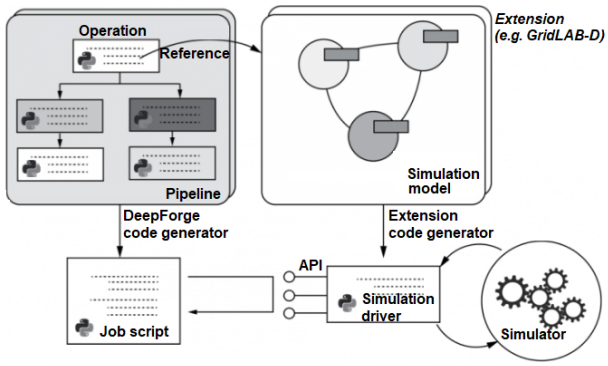


Figure 6: Integrated ML Experimentation in TeSER

H. Tools for Integrating Modeling Frameworks

Integrating modeling frameworks is needed in TeSER for connecting the power grid simulations and deep learning pipeline executions in a tight loop. We leverage WebGME's command-line interface for merging the corresponding DSML and creating an integrated modeling environment. In addition, we also leverage DeepForge's extension mechanism for integrating other modeling languages. The currently available support for creating Keras library based neural network models is an example of how DeepForge's extension mechanism works.

IV. Case Study

The collaborative model-based approach provided by TeSER enables rapid prototyping and experimentation with various neural network architectures and data processing, training and evaluation pipelines. Furthermore, tight integration with the CPS simulation tools—such as GridLAB-D—simplifies the process and shortens the time for input data generation for such models. Below we aim to illustrate how one can use the testbed to easily build and compare various ML-based predictors for load estimation in power distribution networks, and how this process is supported by the web-based design environment. For specific detailed analysis of prediction accuracy, its effects on load forecasting and of various attack and defend strategies, please refer to [10] [4].

A dataset has been generated with a realistic GridLAB-D model with 109 commercial and residential loads, where each endpoint reports its power usage on an hourly basis over a period of 90 days. Based on these reports the total power consumption in the distribution network is calculated. The case study aims to create and evaluate alternative regression models, which can predict the network-level (total) load for the next hour based on the previous 24 hours of data. The dataset was split into training and testing parts with 81 and 9 days of measurement data respectively.

We built a single generic data processing pipeline to train and evaluate the alternative ML models (see Figure 7). The testbed allows to (re)assign different ML architectures to the data processing pipeline. The current workflow loads the simulation results, implements the train/test split, drives the training (for a given number of epochs) and evaluates the model on the test data. It generates plot data for the training loss and summary statistics on the test results.

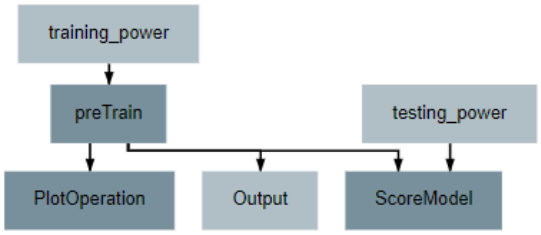
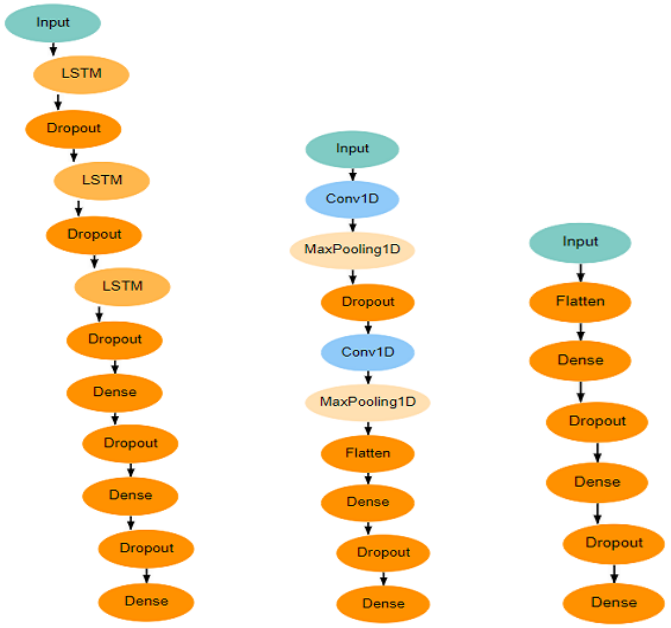


Figure 7: DeepForge generic pipeline model for training

As shown in Figure 8, we selected three popular architectures for time series forecasting with neural networks. First, a multilayer perceptron (MLP) model with two hidden layers (64-nodes each) is built. The second model is a deep convolutional network (CNN) with two sets of 1-dimensional filters sweeping on the time axis. Last, a more complex long short-term memory (LSTM) model is created (three LSTM layers with two fully connected layers). All three models use simple Rectified Linear Unit (*RelU*) activation functions and dropout layers for regularization. The training loss figure (Figure 9) of the three alternatives is captured from the testbed's web interface.



(a) LSTM model (b) CNN model (c) MLP model

Figure 8: Alternative neural network architectures for regression

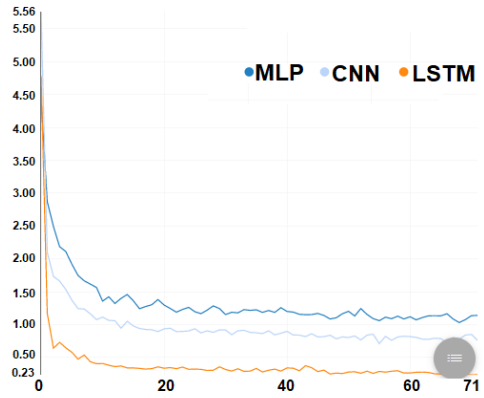


Figure 9: Training loss for MLP, CNN and LSTM

Evaluating the alternative predictors for the simulation data requires a single change in the pipeline–architecture assignment. The testbed takes care of input data generation, model versioning and tracing the analysis results back to the versioned model. This relatively simple experiment resulted in the prediction mean squared error (MSE) of 1.854 for MLP, 0.616 for CNN, and 0.106 for LSTM. As a baseline, the constant mean predictor has an MSE of 12.218. These results show the relative performances of the neural network models and agree with our assumptions on how 1-D convolutional and recurrent models can better learn and predict time series data.

V. Conclusions & Future Work

In this paper, we presented TeSER that is built using widely used open-source technologies. We presented its application in the power grid domain for evaluating deep learning based load forecasters. We described the testbed architecture and its core components and features and demonstrated it with a case study on load forecasters. TeSER uses a model-based approach and is web-based and cloud deployed. It provides strict version store for storing models (such as grid, neural network, and pipeline models), input datasets, and experiment results, thereby enabling experiments to be traceable and parameterizable.

We are working on integrating the power grid simulation and the deep learning framework into a single framework for end-to-end integrated workflows between them. This will support automated workflows of power grid simulation, generation of simulation data and its feed into corresponding ML pipelines. We are also extending our library of reusable and configurable cyber-attacks (modeled as ML pipelines), neural network models, and cyber-defense models of anomaly detectors that can mitigate the impact of stealthy adversarial attacks.

VI. Acknowledgments

This work is supported in part by the following grants: NSF #1521617, NIST #70NANB19H100, and NSA #H98230-18-D-0010. Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States. Certain commercial products are identified in order to adequately specify the procedure; this does not imply endorsement or recommendation by NIST, nor that such products are necessarily the best available for the purpose.

References

[1] Borges, Cruz E., et al., "Evaluating combined load forecasting in large power systems and smart grids," *IEEE Trans. on Industrial Informatics*, 2012, *doi*:10.1109/TII.2012.2219063.

[2] McDaniel, Patrick, et al., "Machine learning in adversarial settings," *IEEE Security & Privacy*, 2016, *doi*:10.1109/MSP.2016.51.

[3] Kurakin, Alexey, et al., "Adversarial machine learning at scale," 2016, *arXiv*:1611.01236v2.

[4] Ghafouri, Amin, et al., "Adversarial regression for detecting attacks in cyber-physical systems," 2018, *arXiv*:1804.11022v1.

[5] S. Soltan, M. Yannakakis, and G. Zussman, "React to cyber-physical attacks on power grids," *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 2, pp. 50–51, 2019, *doi*:10.1109/TNSE.2018.2837894.

[6] "Testbed for Simulation-based Evaluation of Resilience (TeSER)," February 2020, *URL*:http://lablet.webgme.org.

[7] Kecskés, Tamás, et al., "Bridging engineering and formal modeling: Webgme and formula integration." in *MODELS (Satellite Events)*, 2017, *Semanticscholar*:44617122.

[8] Broll, Brian, et al., "Deepforge: A scientific gateway for deep learning," *Gateways*, 2018, *doi*:10.6084/m9.figshare.7092272.v2.

[9] Chassin, David P., et al, "GridLAB-D: An open-source power systems modeling and simulation environment," in *IEEE PES T&D Conference and Exposition, 2008*, pp. 1–5, *doi*:10.1109/TDC.2008.4517260.

[10] Zhou, Xingyu, et al., "Evaluating Resilience of Grid Load Predictions under Stealthy Adversarial Attacks," *Resilience Week Symposium*, 2019, *doi*:10.1109/RWS47064.2019.8971816.

[11] N. Phuangpornpitak and W. Prommee, "A study of load demand forecasting models in electric power system operation and planning," *GMSARN Int. Journal*, 2016, *Semanticscholar*:52992311.

[12] H. He and J. Yan, "Cyber-physical attacks and defences in the smart grid: a survey," *IET Cyber-Physical Systems: Theory & Applications*, vol. 1, no. 1, pp. 13–27, 2016, *doi*:10.1049/iet-cps.2016.0019.

[13] Deka, Deepjyoti, et al., "Optimal data attacks on power grids: Leveraging detection & measurement jamming," in *2015 IEEE SmartGridComm*, pp. 392–397, *doi*:10.1109/SmartGridComm.2015.7436332.

[14] Neema, Himanshu, et al., "Web-Based Platform for Evaluation of Resilient and Transactive Smart-Grids," in *MSCPES 2019*. IEEE, 2019, pp. 1–6, *doi*:10.1109/MSCPES.2019.8738796.