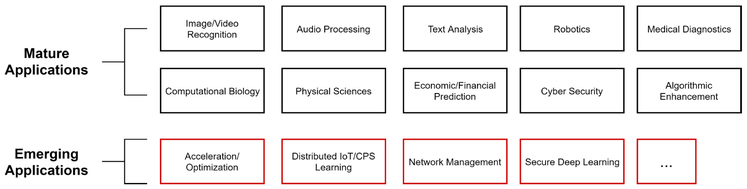
Case Study of Building Machine Learning for Cyber-Physical Systems

William Grant Hatcher, Fan Liang, Jarrett Booz, Josh McGiff, Chao Lu and Wei Yu
Cyber-Physical Networked System and Security Research Laboratory
Department of Computer and Information Sciences, Towson University
Web: http://wp.towson.edu/wyu Email: wyu@towson.edu
Acknowledgement: NSF CAREER Award - CNS-1350145

Overview

- Cyber-physical Systems (CPS) integrate computing, network communication, and control to facilitate smart-world systems.
- The interconnection of sensing and actuating devices in the Internet of Things (IoT) creates new uncertainties and countermeasures must be developed.
- Deep Learning is an emerging tool with the power to conduct data analysis to address uncertainties.
- In this research, we have conducted a survey of deep learning platforms and applications and applied deep learning to handle typical CPS functions, including monitoring and control (energy demand prediction) and security (malware detection and prediction), developing appropriate system frameworks and evaluating a variety of learning mechanisms and relevant technologies.



Research Focus

- Develop learning methodologies for classification and prediction for systems of constrained devices, including model training and evaluation procedures.
  - Mechanisms include shallow learning, deep neural networks, multimodal learning, time-series forecasting, parameter optimization, and machine learning software infrastructures.
- Apply learning methodologies to typical CPS smart-world systems such as smart grid, smart home, smart city, etc. in practical experiments.
  - Evaluate monitoring and control mechanisms and security concerns in smart grid and smart home scenarios as case studies, with consideration for power usage efficiency and prediction, as well as vulnerabilities of smart mobile devices and operating systems.

Our Contributions

- Designing, implementing, and evaluating an online deep learning strategy to predict energy consumption, deploying the training process to the network edge.
- Developing an approach to optimize deep learning for smart meter power consumption forecasting via feature selection and model design.
- Optimizing deep learning training through model, hyperparameter, and learning backend selection, and constructing a framework for remote cloud-based training.
- Evaluating multimodal data strategies for deep learning-based malware detection.

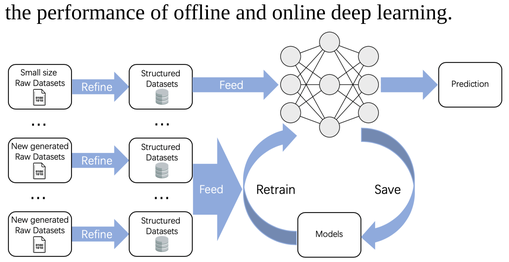
Online Learning for Energy Usage Prediction

Step 1. Implement Long-Short Term Memory (LSTM) recurrent neural network model and evaluate accuracy, adjusting the input features of datasets and hyperparameters.

Step 2. Develop an online training model, using a data subset for training, saving the model when it meets the desired accuracy requirement.

Step 3. Retrain the saved model after a time interval using newly generated data.

Step 4. Compare the performance of offline and online deep learning.



Online Training Workflow

Tools and Testbed

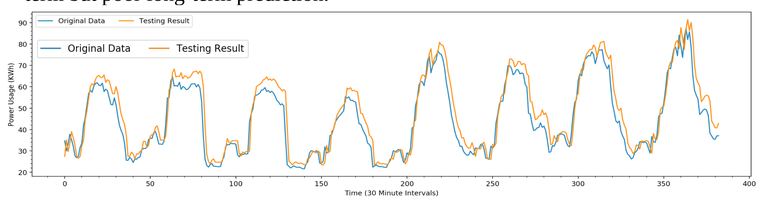
- Keras: a high-level python library that serves as API to support a variety of machine learning libraries.
- TensorFlow: an open source software library for high-performance datfaflow computation.

Methodology

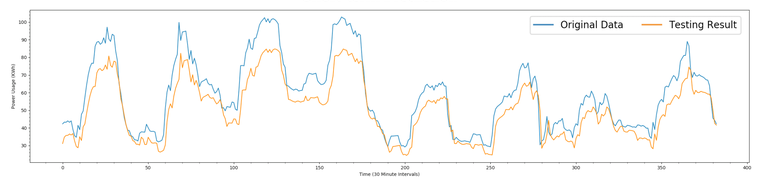
- Datasets Preparation
  - Our dataset consists of one year of smart meter data collected every half-hour.
  - We select the first four weeks (1,344 datapoints) as the training dataset, week five (366 data points) as the short-term testing set, and week seventy-three as the long-term testing set. We also compare the full year as the training set.
  - The dataset has four features (*Power Usage*, *Date*, *Time*, and *Day Type*), and *Temperature* has been collected and added to improve accuracy.
- Deep Learning Model
  - Use the Long-Short Term Memory (LSTM) neural network, which includes both work memory (wm) and long-term memory (ltm). The activation function is *tanh*.
- Evaluation
  - We evaluate both Mean Average Error (MAE) and Mean Square Error (MSE) to assess accuracy, as well as compare running times.

Evaluation Results

- Short-term prediction results via traditional non-online methods show good short-term but poor long-term prediction.



Short-term Prediction

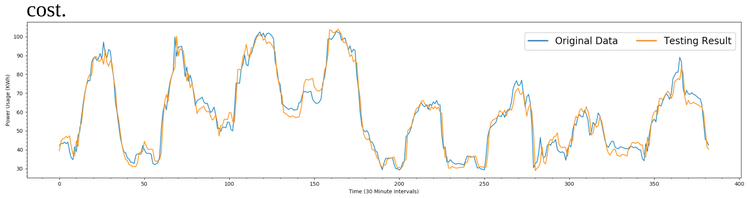


Long-term Prediction

- Our online retraining strategy shows that continuous updates significantly reduce the storage demands on constrained devices & training time.
- MAE (Mean Absolute Error) and MSE (Mean Squared Error) of online retraining closely matches those of training with the full year dataset with a greatly reduced time cost.

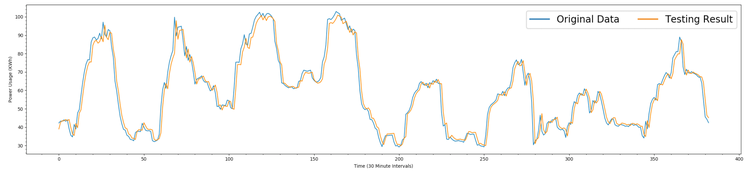
	MAE		*MSE*		
	Training	**Testing**	**Training**	**Testing**	**Average Training Time (s)**
Short-term	3.134	3.798	11.869	18.195	27.75
Long-term	3.433	10.934	11.012	43.553	
Full Year	1.393	2.015	6.328	12.072	231.37
Online	2.137	3.331	7.129	17.229	5.35

Table 1. Evaluation Results



Online Training Prediction

- Online training improves long-term prediction, and adding the temperature attribute later in the online training process further improves the prediction accuracy.



Online Training Prediction with More Feature

Android Malware Classification and Detection

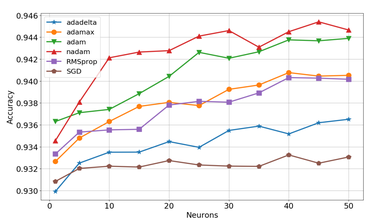
- An edge/cloud computing framework was designed to train dense neural networks offline. Strong computation resources allow for the comparison of model configurations optimized by grid search over hyperparameters.
- The framework was constructed using Keras and other learning libraries for Python, and conducted malware analysis via learning from Android application installation (APK) files.

Methodology

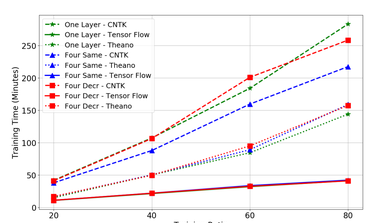
- Extracted Android Permissions and Hardware Feature data from some 58,884 APK packages (19,273 malicious, 38,941 benign) using Android Asset Packaging Tool (AAPT) via bash script.
- Tuned neural network hyperparameters in groups: Epochs, Batch Size, Number of Neurons, Optimizer, Dropout Rate, Weight Constraint, and Class Weight.
- Compiled multiple learning backends compatible with Keras for comparison.
- Designed five neural network shapes to compare performance of separate versus continuous input layers of Permissions and Hardware Features.

Evaluation Results

- Parameter tuning via grid search generally increases accuracy with more neurons. The best performance was 45 neurons with the Nadam optimizer.
- Accuracy for the best performers reached about 95%.



- Evaluation of deep learning backend libraries TensorFlow, Microsoft Cognitive Toolkit (CNTK), and Theano:
  - CNTK and Theano are much slower than TensorFlow.
  - CNTK and Theano can reach higher accuracies, but these results are marginal and unpredictable.



- Multimodal learning revealed that the inclusion of both Features and Permissions increases accuracy.
- Multimodal input shows marginal accuracy as well as minor improvements in training time.