Paper SAS3278-2019

How to Use Deep Learning with Your Internet of Things (IoT) Digital Twin

Brad Klenz, SAS Institute Inc.

ABSTRACT

With the Internet of Things (IoT), a digital twin is created to have a virtual representation of a remote device or system. The digital twin shows you the device's operating condition, no matter where it is physically located. IoT devices have a number of sensors installed on them, as well as sensors for the environment around them. Analytics can bring this sensor data together to create a true real-time digital twin. A previous paper showed how streaming analytics are used for device state estimation and anomaly detection. This paper explains how deep learning can be added to your digital twin for more understanding. Image and video analytics are used to capture operating conditions that are missed by regular sensors. Recurrent neural networks (RNN) add temporal data analysis and pattern detection in real-time data streams that are prevalent in digital twins. With these deep learning capabilities, your digital twin provides a new level of insight for your remote devices.

INTRODUCTION

Deep learning is becoming more prevalent, with some typical use cases emerging. Natural language processing (NLP) is used with personal assistants and chatbots. Computer vision is used for object recognition in autonomous driving and detecting tumors on medical images. Facial recognition is used for access control. Reinforcement learning is being researched in game applications.

Now research is emerging on industrial IoT applications that will help augment existing applications of digital twins. Digital twins are a virtual representation of a physical asset or device, frequently in a remote location.[1] With IoT, data is collected from sensors on a device, on neighboring devices, the environment around a device, and whatever interacts with the device. The speed is real time, and connectivity allows us to span distances instantly in many cases. Advances in streaming analytics now enable us to process this real-time data using machine learning and artificial intelligence.

To add more value to your digital twin, deep learning can be added for specific use cases. These use cases are fairly targeted based on typical sources of data found in digital twin applications. Further in the future, artificial general intelligence (AGI) will be able to use the entirety of data from your digital twin for more general AI application.

Here are some of the deep learning techniques shown in this paper:

- Computer vision – Using images and video to provide insight not easily available with existing sensors. Cameras can sometimes be installed much more easily than other sensors. Cameras can also be retrofitted to existing assets passively, where adding sensors can more invasive.
- Recurrent neural networks (RNN) – Sensor data frequently captures measurement over time, and we are looking for issues that develop over time. RNNs can provide complex pattern recognition as well as specialized forecasting.

- Reinforcement learning (RL) – Your digital twin is frequently a twin of a physical asset that is being controlled for optimal operation. The output of the physical asset is typically captured. Using RL, we can learn how the controls of the asset can be set to learn how to achieve optimal output.

REAL-TIME APPLICATION OF DEEP LEARNING IN YOUR DIGITAL TWIN

Deep learning is very compute intensive. The deep learning models are trained on large databases and are almost always done offline. It's not unusual to take hours or days to train a model. Once the model is trained, the application of the model through inferencing is less compute intensive, but still requires more compute resources than is typical for digital twin applications. For some applications, near real-time or slightly delayed results are sufficient. For example, in the computer vision defect detection described below, it might be acceptable to hold a production batch while the defect detection is performed. In other cases, real-time inferencing is needed. Inferencing can be done in the cloud or data center where sufficient resources are readily available. For edge inferencing, edge gateways are now becoming available with sufficient compute power, but you must plan for this specialized need.[2,3]

COMPUTER VISION

Popular uses of computer vision include facial recognition and object detection. To see where computer vision can help a digital twin, look for applications that would require visual inspection. Here are some examples:

- Defect detection in semiconductor manufacturing – In semiconductor manufacturing of wafers and dies, many tests cannot be run until the packaging phase. With computer vision it is practical to take images of the wafer earlier in the production process and inspect it for issues. The inspection can be used to find defects and determine the number and location. This will allow an earlier determination of final yield from the wafer. With previous labeled images of diagnosed defects, it is also possible to classify the defect types using computer vision. This will help augment a larger root cause analysis for process improvement.

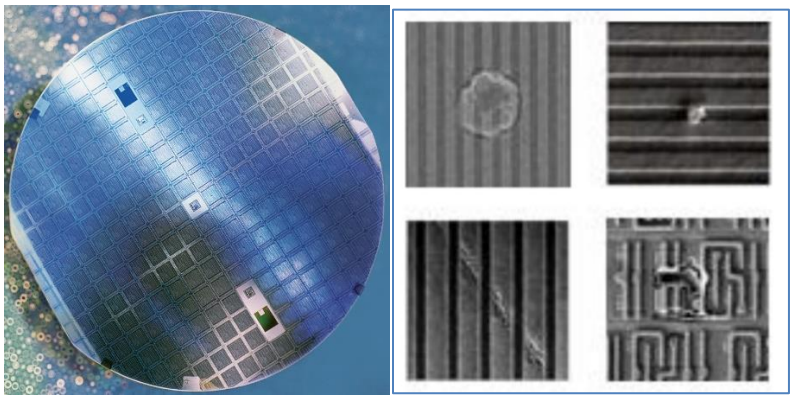

Figure 1 Silicon Wafer and Wafer Defects

- Defect detection in discrete parts – With visual inspection of discrete parts, you can more easily catch a number of production defects. These are various issues in production quality. For example, in aerospace and automotive parts production, you can determine incomplete finishes or casting issues.

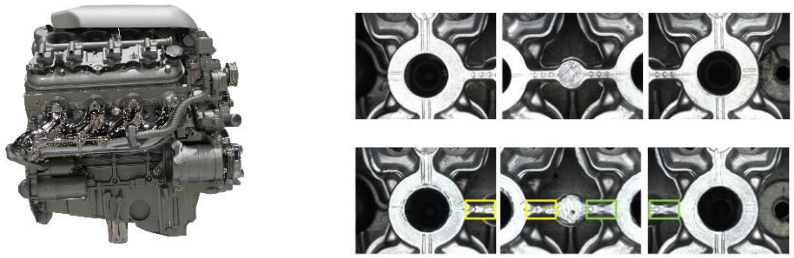

Figure 2 Automotive Engine and Casting Defects

- Infrared patterns for heat buildup in power substations – Using specialized cameras, you can capture images for different spectrums. Infrared cameras can capture a more complete picture of temperature deviations and patterns than what would be possible with individual temperature sensors. This allows for new applications such as monitoring power substations for components getting ready to fail.

IMPLEMENTING A COMPUTER VISION MODEL

The process for implementing a computer vision model is as follows:

- If possible, fix the camera to a stable mount point so that all images will be taken from the same angle and with the same proportions. This vastly simplifies the model training as compared to general object recognition models, which must capture objects from many angles. The fixed camera location also simplifies the process of determining the location of defects on the piece.
- Another option is to initially create a model that finds easily identified features on the piece. For the power substation example, you could have general instructions on how to point the camera at a transformer in the substation. An object recognition model could identify the bushings on the top of the transformer. This would provide reference points to scale the images with images captured at similar angles. This is similar to how facial recognition models determine the various key points on a face.[4,5]
- In the case where issues develop or occur over time, a video can create a large number of images, both of known good cases and defect cases.
- You will use the images to create a classification model using Convolutional neural networks (CNN). Depending on how well labeled your data is, you can create models of various complexity.
    - If you primarily have a collection of known good images, you can create a binary classification model that identifies images with a high likelihood of known good or suspected anomaly images. The power transformer is an example of this.
    - If you have images that have been labeled with known defect types, you can create a more complex classification model that identifies the various defects. The discrete parts are an example of this. There might be previous images labeled with an incorrect bearing insertion, and other images labeled with incorrect part milling.
    - If you have good location identification, you can also break down the images and find the locations of portions of the image with defects. The

semiconductor wafer is an example here. This would allow you to quantify the expected yield based on the proportion of the wafer with defects.

- When you have trained the model, you can determine at what latency you can infer and test new images being captured. Determine if you need to stream image-by-image and get immediate results. Alternatively, you might be able to capture a batch of images and process in batch. Also determine if the inferencing can be done in the cloud or server, or if an edge gateway is needed.
    - The power transformer example might be a good candidate for edge inferencing. Since the transformers are located in remote locations with reduced or sporadic network bandwidth, an edge device could process the images at the substation. In addition, only rarely will there be an issue requiring attention. Processing at the edge would eliminate the need for a large amount of image transfer for rare events.
    - The discrete parts application would be a good example of real-time inferencing in the cloud or server. The factory installation will allow high quality network connectivity. The low latency would allow defective parts to be identified immediately and be removed before being used in subsequent assemblies.
    - The semiconductor example is a batch process that would fit well with batch inferencing. Since each step in the process is costly, it would be worth the time needed to hold the batch until it could be verified in full detail.

Computer vision is a technique that will have many applications for digital twins.[6,7,8]

RECURRENT NEURAL NETWORKS

Recurrent neural networks (RNNs) are a special class of deep learning neural networks designed for sequence or temporal data. An area where RNNs are popular is for natural language processing. In this application, large text and document databases are modeled to discover common word sequences in context. The model can then be used to generate new messages for the appropriate context. This is seen in chatbot applications. Within IoT and digital twins, there are many examples of such sequence and temporal data. Many sensors are collecting data over time. The sequence or pattern of the measurements over time can be used to understand interesting characteristics of the digital twin asset. One example is measuring energy circuits in a smart building or power grid. The pattern of the energy use on a circuit can capture the start or end of an asset operation, such as a motor start, which signals an operation change in the digital twin asset. Another use of RNNs is for forecasting unusual time series data. An example is forecasting the energy output from a solar farm.

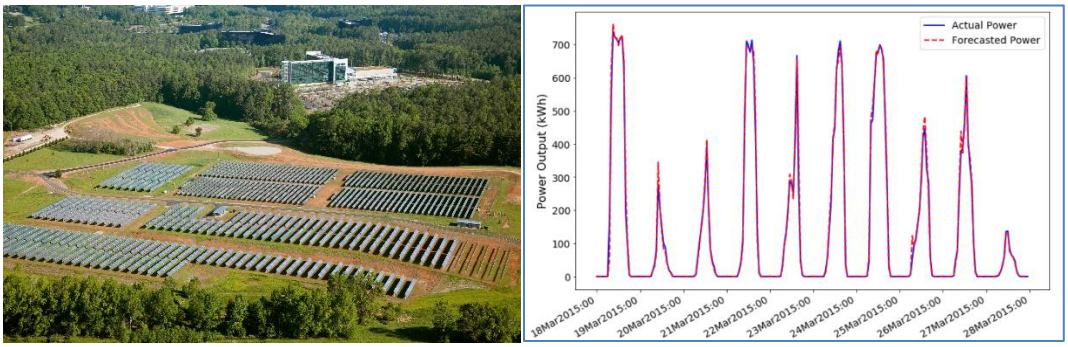

Figure 3 Solar Farm and Power Output Chart

In this case, there is a cyclical component that could be forecasted using traditional methods, but there is a less well modeled component of weather and cloud cover. With the large amount of data available from the solar farm and nearby solar farms, a deep learning RNN can capture the more sporadic aspects of the energy output.[9]

The process for training an RNN is different if you are working with sequence data versus working with temporal data.

The process for training the RNN with sequence data is as follows:

- Break the data into segments of sequential measurements. The length of the segment is determined by the time interval of the data and the expected duration of the precursor to an event. For the energy circuit example in smart buildings, the data is collected at 5-second intervals, and we use the previous minute of data.
- Create a target variable for the events of interest and use it to label the sequences where the event occurs. For our example, we are using motor starts and identifying weak motor starts indicating capacitor failure.
- Train the RNN. Note that RNN training has a feature for bidirectional model fitting. That is useful for natural language processing applications where words can be in a different sequence, but still indicate the same event, like a positive product review. Since our measurement data is always moving forward in time, we do not need to use bidirectional model fitting.
- The trained model can then be deployed for inferencing. In most cases, the model inferencing function will be sufficiently fast to be used on the real-time measurement stream, either in the cloud, server, or edge device.

The second type of RNN is used to forecast. The example in this case is to forecast the energy output of a solar farm for short time periods in the future (1 hour). The key in this case is to create a set of lagged variables for the predictors and the response variable. The response variable is the energy produced. The steps for training this RNN are as follows:

- Take the historical input database and create lagged variables for the predictors and response variable. The number of lags is determined by the time interval of the measurement data and the expected correlation of previous measurements on the forecast time horizon. For the solar farm example, we are producing 1-hour-ahead forecasts, and the data over the last few hours is sufficient to capture the primary effects for the forecast. Note that there are a large variety of conditions possible throughout the year and previous observed weather, even though the forecast horizon is fairly short. Since we have a large amount of historical data of the various conditions, the use of an RNN is appropriate for this problem.
- When creating the lags, you should evaluate your data for missing values and consistent time intervals. The ideal case is data collected at consistent intervals with few missing values. If the data has a large number of missing values or inconsistent collection intervals, you can use the TIMEDATA procedure to improve the data for training.
- Since training and evaluating the RNN model is dependent on the sequence, partitioning the data requires more care than typical random partitioning. In this case, we need to preserve the sequence of the data for use in the model creation steps (training, validation, test). The easiest way to do this is to partition the data based on the time variable. Use the earliest historical data for the training data set. Then use the next time partition for the validation data set. Finally, use the most recent data for the testing data set. This is sufficient if the performance of the asset has been consistent over the historical data sample. If there have been periods of

degraded performance, it is best to eliminate that data from the data sets used to create the model.

- Train the RNN. Note that some predictors might be estimates that you can capture from other sources. A major factor in solar farm energy forecasting is weather conditions such as cloud cover. In this case, you will want to include these predictors from a source that can provide actual and estimated values.
- For the solar farm example, we are doing short term forecasting to be used in energy load and generation balancing. Longer term demand planning is done through a separate, more traditional, forecasting process.
- You can use RNNs for one-step-ahead forecasting, where the forecast interval matches, or is less than, the desired forecast interval. This will yield the most accurate forecast. In some cases, you might need a multistep forecast to forecast future time periods based on the near term forecast estimates. These forecasts are typically less accurate but can be tested to determine if they have sufficient accuracy.

RNNs are a valuable deep learning technique for IoT digital twins.[10]

REINFORCEMENT LEARNING

Reinforcement learning (RL) is a subfield of machine learning and deals with sequential decision-making in a stochastic environment. In any RL problem, there is at least one agent and an environment. The agent observes the state of the environment and takes and executes a decision. Environment returns a reward and a new state in response to the action. With the new state, the agent takes and executes another action, environment returns reward and new state, and this procedure continues iteratively. RL algorithms are designed to train an agent through this interaction with the environment, and the goal is maximizing the summation of rewards.

RL has recently got a lot of attention due to its successes in computer games and robotic applications.[11,12] Beside the simple RL applications, there are still few real-world applications of RL to increase efficiency. We studied and did some research to extend an RL algorithm for controlling the heating, ventilation, and air conditioning (HVAC) systems. HVAC includes all the components that are supposed to maintain a certain comfort level in the building. Buildings consume 30% to 40% of all consumed energy in the world, so that any improvement could result in a huge saving in energy consumption and CO2 release.[13] Advances of the new technologies in recent years have improved the efficiency of most components in the HVAC systems. Nevertheless, still there are several directions to reduce the energy consumption by controlling different decisions on these systems.

There are two general categories of HVAC system, single and multizone. The single zone problem refers to an area that uses an HVAC system (for example, a heater or an AC system that is installed in an office), where the main control decision is the temperature set-point or just the binary action of turning the device on or off. In multizone systems, a central HVAC system supports several zones (for example, offices, hallways, conference rooms), and with a given set-point for each zone, the system needs to maintain a certain comfort level in each zone, while there are many possible control decisions.

We considered a multizone system and selected the amount of air flow as the main control decision. Using the obtained data from a building at SAS in Cary, NC, we trained an environment and used it to train an RL algorithm when there are 10 zones in the system with a set-point of 72 with ±3 allowance. Figure 4 shows the results of 50 cases with different initial temperatures. The upper figure is the temperature and the lower figure is taken actions over 150 minutes, in which every three minutes a decision is taken. We

compared this result to the commonly used rule-based algorithm (in which the system is turned on/off at 69/75) and RL obtained 47% improvement on combination of obtained comfort and energy consumption.

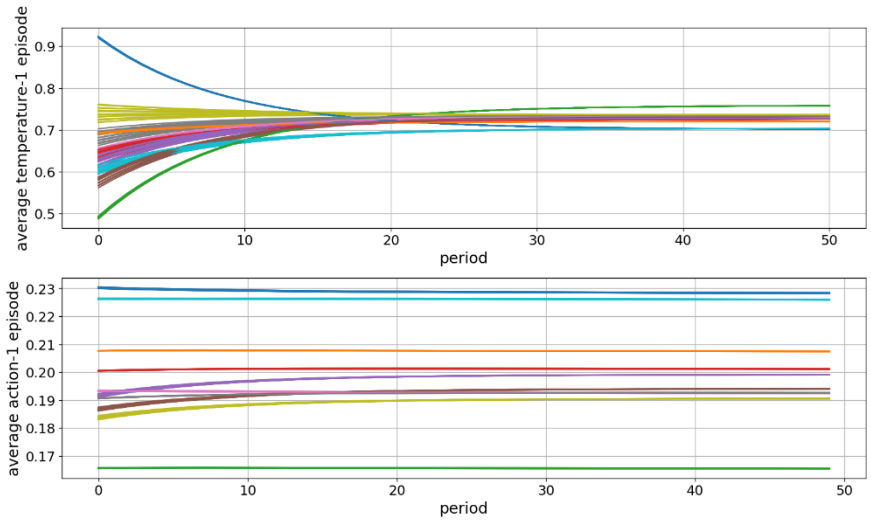

Figure 4 The Average Temperature and Average Action of the RL Algorithm

HYPERPARAMETER TUNING

For all deep learning methods, hyperparameter tuning is an important step. Hyperparameter settings are often dependent on the domain knowledge of the application. Research into the specific application can yield a set of parameter settings to be tested. In some cases, a set of parameter settings has been established as best practices. In other cases, research will be needed to determine the best settings.

One feature in SAS® Visual Data Mining and Machine Learning is hyperparameter autotune. This feature will take a range of potential parameter settings and perform an optimal search for the best performing settings. This will greatly help cases where research is needed on the parameter settings.[14,15]

CONCLUSION

Use of deep learning is growing into new application areas, and IoT digital twins are an application that will benefit from this growth. The applications for deep learning are different for digital twins than the typical applications seen today, like autonomous driving, facial recognition, and natural language processing. Instead, computer vision will be used to perform visual inspection for defect detection and identification. Recurrent neural networks will be used on the time series data streams that are prevalent in IoT to find complex pattern sequences and forecast where effects of environmental variables can be discovered with the large amount of data in IoT. Reinforcement learning will find many applications in IoT as assets in rich environments can be controlled by the algorithm, and results can be measured to determine the benefits achieved.

NOTES

1. Klenz, Brad. 2018. “How to Use Streaming Analytics to Create a Real-Time Digital Twin.” *Proceedings of the SAS Global Forum 2018 Conference*. Cary, NC: SAS Institute Inc. Available: https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2004-2018.pdf
2. Williams, David. 2019. “NVIDIA Graphics Processing Units Accelerating SAS® Analytics.” *Proceedings of the SAS Global Forum 2019 Conference*. Cary, NC: SAS Institute Inc. Available: https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3618-2019.pdf
3. McGrath, Dylan. 2018. “New Architectures Bringing AI to the Edge.” Available: https://www.eetimes.com/document.asp?doc_id=1333920
4. Long, Xindian. 2018. “Understanding Object Detection in Deep Learning.” Available: https://blogs.sas.com/content/subconsciousmusings/2018/11/19/understanding-object-detection-in-deep-learning/
5. Long, Xindian. 2019. “Exploring Computer Vision in Deep Learning: Object Detection and Semantic Segmentation.” *Proceedings of the SAS Global Forum 2019 Conference*. Cary, NC: SAS Institute Inc. Available: https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3317-2019.pdf
6. Sethi, Saratendu. 2018. “Computer Vision Using SAS® Deep Learning.” Available: https://www.sas.com/offices/pdf/ax-2018-milan/sas-ax18-sethi.pdf
7. Gong, Julia. 2019. “Using Deep Learning for Tumor Segmentation in Medical Images.” Available: https://blogs.sas.com/content/subconsciousmusings/2019/02/15/using-deep-learning-for-tumor-segmentation-in-medical-images/
8. Gong, Julia. 2019. “Constructing the Front of the Computer Vision Pipeline.” Available: https://blogs.sas.com/content/subconsciousmusings/2019/02/28/constructing-the-front-of-the-computer-vision-pipeline/
9. Kahler, Susan. 2018. “Using Deep Learning to Forecast Solar Energy.” Available: https://blogs.sas.com/content/subconsciousmusings/2018/07/05/deep-learning-forecasts-solar-power/
10. Qi, Yui. 2018. “Recurrent Neural Networks: An Essential Tool for Machine Learning.” Available: https://blogs.sas.com/content/subconsciousmusings/2018/06/07/recurrent-neural-networks-an-essential-tool-for-machine-learning/
11. Hao, Karen. 2019. “The Rise of Reinforcement Learning” in “We Analyzed 16,625 Papers to Figure Out Where AI Is Headed Next.” Available: https://www.technologyreview.com/s/612768/we-analyzed-16625-papers-to-figure-out-where-ai-is-headed-next/
12. Burda, Yuri. 2018. “Reinforcement Learning with Prediction-Based Rewards.” Available: https://blog.openai.com/reinforcement-learning-with-prediction-based-rewards/
13. U.S. Department of Energy, 2008, “Energy Efficiency Trends in Residential and Commercial Buildings” Available: https://www1.eere.energy.gov/buildings/publications/pdfs/corporate/bt_stateindustry.pdf
14. Koch, Patrick, et al. 2018. “Autotune: A Derivative-Free Optimization Framework for Hyperparameter Tuning.” Available: https://www.kdd.org/kdd2018/accepted-papers/view/autotune-a-derivative-free-optimization-framework-for-hyperparameter-tuning
15. Koch, Patrick, Brett Wujek, and Oleg Golovidov. 2018. “Managing the Expense of Hyperparameter Autotuning.” *Proceedings of the SAS Global Forum 2018 Conference*. Cary, NC: SAS Institute Inc. Available: https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/1941-2018.pdf

ACKNOWLEDGMENTS

The author wishes to acknowledge Afshin Oroojlooy for the section on Reinforcement Learning. His work will be very beneficial to the advancement of digital twins.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Brad Klenz
SAS Institute Inc.
Brad.Klenz@sas.com