Hardening Deep Neural Networks in Condition Monitoring Systems against Adversarial Example Attacks

Felix Specht and Jens Otto

¹ Fraunhofer IOSB-INA, Lemgo 32657, Germany

² <https://www.cybersecurity-owl.de/>

³ {felix.specht,jens.otto}@iosb-ina.fraunhofer.de

Abstract. Condition monitoring systems based on deep neural networks are used for system failure detection in cyber-physical production systems. However, deep neural networks are vulnerable to attacks with adversarial examples. Adversarial examples are manipulated inputs, e.g. sensor signals, are able to mislead a deep neural network into misclassification. A consequence of such an attack may be the manipulation of the physical production process of a cyber-physical production system without being recognized by the condition monitoring system. This can result in a serious threat for production systems and employees. This work introduces an approach named CyberProtect to prevent misclassification caused by adversarial example attacks. The approach generates adversarial examples for retraining a deep neural network which results in a hardened variant of the deep neural network. The hardened deep neural network sustains a significant better classification rate (82% compared to 20%) while under attack with adversarial examples, as shown by empirical results.

1 Introduction

Cyber-physical production systems (CPPS) consist of hardware and software components controlling physical processes. They are in the focus of initiatives such as Germany's Industrie 4.0 or the US Industrial Internet Consortium. CPPS adapt efficiently to new products or product variants without extensive manual engineering effort [1–3].

Figure 1 a) shows an example of a CPPS where material is moved and processed between a storage module, a conveyer module, a heating module and a pick-and-place module.

Condition monitoring systems (CMS) can be utilized to detect system failures of CPPS (cf. figure 1 b), e.g. a broken heating module. Therefore, process data from modules are analyzed with machine learning algorithms such as deep neural networks (DNN) [4]. DNNs enable automatic generation of mathematical models representing the normal behavior of a CPPS. The normal behavior is learned using historical process data from production modules. As a representation of

the learned normal behavior, the model is compared with the actual CPPS state to classify its condition as normal behavior or anomaly. DNNs have successfully been used to model physical manufacturing processes [5, 6].

However, DNNs are vulnerable to adversarial example attacks [7]. Exploiting such attacks may result in a manipulation of the physical production process, which can cause enormous damage to facilities, production systems and employees [8].

Fig. 1. Adversarial example attack against a condition monitoring system (b) results in misclassification of the observed cyber-physical production system (a).

An adversarial example (AE) is a specially manipulated input with the ability to mislead a DNN into misclassification [7]. It is generated from an undistorted original input by intentionally applying worst-case perturbations [9]. This results in an adversarial input being almost identical to the original one.

Fig. 1 shows an example of a condition monitored CPPS, where an adversary gained access to the production system. One objective of the adversary may be the manipulation of the production process without triggering an alert by the CMS (false-negatives). Another objective may be the triggering of false alerts (false-positives).

A successful attack may be achieved by the following steps: (i) process data from production modules is collected, (ii) collected process data is used to generate AEs, (iii) AEs are exploited to manipulate the physical process.

A false-positive classification triggers an anomaly alarm by the CMS, although the production process was actually correct. This may lead to unscheduled maintenance, which temporarily stops production. Likewise, the confidence in the CMS may be reduced. Furthermore, a false-negative classification may result in e.g. damaged products. A long-term operation in an insecure system

state may even result in severe damage to the production system and pose a threat to employees [8].

Our contribution is the introduction of an approach to prevent misclassification caused by adversarial example attacks on deep neural network based condition monitoring systems, which detect system failures of cyber-physical production systems. Empirical results show that our approach results in a hardened deep neural network with a significant less misclassification rate despite being attacked.

2 Related work

Szegedy et al. introduce AEs as an anomalous property of a DNN [7]. A DNN can be formalized as a mathematical model $F(x, \theta) = Y$. The DNN decides whether a given input $x \in \mathbb{R}^n$ belongs to a learned class $Y \in \mathbb{R}^m$, using a set of internal parameters θ .

An AE x' is defined as an incorrectly classified input, which deviates minimally from the correctly classified original x . As shown in definition 1, an AE is generated by applying perturbations Δ_x to the original input x , where Δ_x is kept as small as possible.

$$\begin{aligned} \text{find } x' &= x + \Delta_x \\ \text{s.t. } F(x', \theta) &= Y', \quad Y' \neq Y \end{aligned} \tag{1}$$

A quality criterion for AEs is inconspicuousness and a minimal deviation from its original. E.g. in image classification an objective of AEs is the perturbation to the original, imperceptible to the human eye. For quantification of this property the three distance metrics L_0 , L_2 and L_∞ are commonly used in literature [10]. The L_0 metric corresponds to the number of input signals that have been altered (e.g. pixels). The L_2 metric measures the standard Euclidean (root-mean-square) distance between x and x' . At last, the L_∞ metric measures the maximum change for any of the input signals.

Several approaches for generating AEs have been proposed, such as box-constrained limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS), Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) [7,9,11]. By leveraging AE generating approaches, both Black-Box attacks and White-Box attacks against machine learning classifiers can be performed [12].

Several defensive strategies have been proposed to harden a DNN model against AE attacks, such as Adversarial Training, Defensive Distillation, feature squeezing and PGD based Adversarial Training [7,11,13,14].

This work transfers AE generation and prevention into the field of industrial automation, in contrast to presented approaches considering image processing mainly. Our approach adapts the FGSM approach [9] for AE generation and the adversarial training [7] for preventing misclassification. Both, FGSM and adversarial training are suitable for the CPPS requirement of rapid adaptability.

3 Solution

The objective of the CyberProtect approach is prevention of misclassification caused by AE attacks and is achieved by the following steps: (i) A DNN is exclusively trained on process data $\mathcal{P}$ in an initial training phase. (ii) The training phase is extended by an additional retraining phase, where $\mathcal{P}$ is used to generate an AE $\mathcal{P}'$ as described in the previous section. (iii) Both, the original $\mathcal{P}$ and the manipulated $\mathcal{P}'$ serve as input to the DNN. Generating the required AEs can be formalized as follows.

Process data of a CPPS is defined as $\mathcal{P} = (p_0, \dots, p_m)$, where $p_i \in [0, 1]$ for $i \in \{0, \dots, m\}$ is a sensor or actor value. $\mathcal{P}$ is input to the CMS utilizing a DNN. As described in section 2, a DNN is a mathematical model $F(\mathcal{P}, \theta) = Y$, where $Y \in \{0, 1\}$ is the predicted class corresponding to normal or anomaly behavior and θ is a set of parameters. The adversarial objective is reached by solving the following search problem:

$$\text{find } \mathcal{P}' = \mathcal{P} + \Delta \quad (2a)$$

$$\text{s.t. } F(\mathcal{P}, \theta) = Y \quad (2b)$$

$$F(\mathcal{P} + \Delta, \theta) = Y' \neq Y \quad (2c)$$

$$\sqrt{\sum_{i=0}^m |\delta_i|^2} \leq \epsilon, \epsilon > 0 \quad (2d)$$

A perturbation $\Delta = (\delta_0, \dots, \delta_m)$, where $\delta_i \in [-1, 1]$ for $i \in \{0, \dots, m\}$, is added to an original $\mathcal{P}$ to generate an adversarial example $\mathcal{P}' = (p_0 + \delta_0, \dots, p_m + \delta_m)$. Due to constraint 2c, $\mathcal{P}'$ is not predicted as the original class 2b. The constraint 2d increases the inconspicuousness of $\mathcal{P}'$ by limiting the euclidean distance between $\mathcal{P}'$ and $\mathcal{P}$ to an upper bound.

By exploiting adversarial examples, the adversary can manipulate the production process unrecognized within the specified constraints. This leads to false-positive or false-negative results in classification by the condition monitoring system.

3.1 Generation of Adversarial Examples Algorithm

This approach generates AEs by using the Fast Gradient Sign Method (FGSM) [9]. Due to its low computational costs, FGSM is suitable for the CPPS requirement of rapid adaptability.

Generation of AEs is formally described by algorithm 1. The algorithm requires inputs $\mathcal{P}, F(\mathcal{P}, \theta), Y, \epsilon, s$, where $\mathcal{P}$ describes process data, $F(\mathcal{P}, \theta)$ describes the trained DNN, Y is the original class label, ϵ is a threshold parameter and s is a precision parameter.

The algorithm performs the following steps: (1) A variable η is increased by the precision parameter s specifying the growth of η between iterations. (2) FGSM is used to generate a candidate for an AE $\mathcal{P}'$. FGSM requires $F(\mathcal{P}, \theta)$

Algorithm 1: GenAE: Generation of Adversarial Example $\mathcal{P}'$ from original Process Data $\mathcal{P}$

Input: Process data $\mathcal{P}$, DNN $F(\mathcal{P}, \theta)$, class label Y , threshold parameter ϵ , precision parameter s

Output: Adversarial example candidate $\mathcal{P}'$

```

1  $\eta \leftarrow 0$ 
  // while misclassification  $Y'$  not found and euclidean distance
  // threshold  $\epsilon$  not reached
2 while  $Y' = Y$  and  $\|\mathcal{P}' - \mathcal{P}\| \leq \epsilon$  do
  // increase perturbation
3    $\eta \leftarrow \eta + s$ 
  // use Fast Gradient Sign Method to generate an AE candidate
4    $\mathcal{P}' \leftarrow FGSM(F(\mathcal{P}, \theta), \eta)$ 
  // get class label
5    $Y' \leftarrow F(\mathcal{P}', \theta)$ 
6 return  $\mathcal{P}'$ 

```

and the variable η and extracts the gradients from a previous trained DNN $F(\mathcal{P}, \theta)$. (3) The class Y' is predicted by the trained DNN, where the generated candidate $\mathcal{P}'$ serves as input. These three steps are repeated until either the euclidean distance between the candidate $\mathcal{P}'$ and the original process data $\mathcal{P}$ exceeds the threshold parameter ϵ , or the new predicted class label Y' differs from the original class label Y . In the case of differing classes, a valid AE is found. The algorithm returns the last computed AE candidate $\mathcal{P}'$.

3.2 CyberProtect Algorithm

The CyberProtect algorithm 2 enables prevention of misclassification caused by AE attacks. The algorithm requires a $\mathcal{P}^n, Y^n, \theta, \epsilon, s$ as input. Input $\mathcal{P}^n$ describes n -dimensional historical process data $\mathcal{P}$, Y^n describes the corresponding n -dimensional class labels, θ describes a DNN configuration, ϵ and s are configuration parameters to use GenAE algorithm described above.

The algorithm executes the following steps: (1) A DNN $F(\mathcal{P}, \theta)$ is initialized with the configuration parameters θ (cf. line 1 function *initialize*) for DNN architecture, activation and cost function. (2) the DNN is trained with each entry $\mathcal{P}_i$ and Y_i (cf. line 2-3 function *train*) of the historical process data $\mathcal{P}^n$ and the corresponding class labels Y^n . (3) An empty set $\mathcal{P}'^n$ is defined (cf. line 4) after the first training phase. (4) The algorithm GenAE 1 is used to generate and store AEs to $\mathcal{P}'^n$ (cf. line 5-7) for each process data entry $\mathcal{P}_i$ resulting in an AE $\mathcal{P}'_i$. (5) A new DNN $\hat{F}(\mathcal{P}, \theta)$ is initialized with the configuration parameters θ (cf. line 8). (6) $\hat{F}(\mathcal{P}, \theta)$ is trained with each entry $\mathcal{P}_i$ and $\mathcal{P}'_i$ of both, the original process data $\mathcal{P}^n$ and the generated AEs $\mathcal{P}'^n$ (cf. line 9-11) using the same class label Y_i for $\mathcal{P}_i$ and $\mathcal{P}'_i$. The algorithm returns the new trained DNN $\hat{F}(\mathcal{P}, \theta)$ hardened against AE attacks.

Algorithm 2: CyberProtect

Input: Process data set $\mathcal{P}^n$, set of class labels Y^n , configuration parameter θ , threshold parameter ϵ , precision parameter s

Output: Hardened Deep Neural Network $\hat{F}(\mathcal{P}, \theta)$

```

1  $\hat{F}(\mathcal{P}, \theta) \leftarrow \text{initialize}(\theta)$ 
  // train DNN with process data set  $\mathcal{P}^n$ 
2 for  $i = 1$  to  $n$  do
3    $F(\mathcal{P}, \theta) \leftarrow \text{train}(\mathcal{P}_i, Y_i)$ 
4  $\mathcal{P}'^n \leftarrow \emptyset$ 
  // calculate AE set from process data
5 for  $i = 1$  to  $n$  do
6   // cf. section 3.1
7    $\mathcal{P}'_i \leftarrow \text{GenAE}(\mathcal{P}_i, F(\mathcal{P}, \theta), Y_i, \epsilon, s)$ 
8    $\mathcal{P}'^n \cup \{\mathcal{P}'_i\}$ 
9  $\hat{F}(\mathcal{P}, \theta) \leftarrow \text{initialize}(\theta)$ 
  // train DNN with process data set  $\mathcal{P}^n$  and AE set  $\mathcal{P}'^n$ 
10 for  $i = 1$  to  $n$  do
11    $\hat{F}(\mathcal{P}, \theta) \leftarrow \text{train}(\mathcal{P}_i, Y_i)$ 
12    $\hat{F}(\mathcal{P}, \theta) \leftarrow \text{train}(\mathcal{P}'_i, Y_i)$ 
12 return  $\hat{F}(\mathcal{P}, \theta)$ 

```

4 Results

Emperical results were obtained with a reference CMS monitoring data from the Secom dataset [15]. The Secom dataset was recorded from a semi-conductor manufacturing process and consists of process data with 590 attributes collected from sensor signals and variables during 1567 manufacturing cycles.

The reference CMS is implemented based on a DNN by using the python framework tensorflow [16]. The applied DNN architecture consists of 590 input neurons, 4 hidden layer with 590, 1180, 2360, 590 neurons each and one output neuron representing the conditions normal or anomaly. As activation function rectified linear unit (ReLU) is applied. Training the DNN was performed in a supervised manner for 1000 epochs using the Adam optimizer [17] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and a learning rate of 0.01.

Our CyberProtect implementation extends the reference CMS and is based on the Python library Cleverhans [18], an extension to the Tensorflow framework [16].

The reference CMS was extended with the CyberProtect algorithm to obtain results shown in Fig. 2.

The left column of Fig. 2 shows classification results from the CMS reference excluding CyberProtect. The mean classification rate is equal to 82% with a standard deviation of 7%, the best result is 95% and the worst result is 60%. The middle column shows classification results of the same CMS reference under AE attacks based on FGSM generation. The classification rate is reduced to a

Fig. 2. Classification rate of a common CMS without AE attacks (left), with AE attacks (middle) and a CMS using CyberProtect with AE attacks (right).

mean of 20% with a standard deviation of 7%, best result of 43% and worst result of 7%. In the right column, results are shown of the extended reference CMS utilizing the CyberProtect approach while being attacked with AEs. Here, CyberProtect significantly increases the classification rate 80% with a standard deviation of 9%, best result of 95% and worst result of 50%.

CyberProtect enables a DNN to nearly regain the classification rate despite AE attacks, as demonstrated in carefully designed experiments.

5 Conclusion

This paper presents the CyberProtect approach to prevent misclassification caused by adversarial example attacks on deep neural network based condition monitoring systems in the domain of cyber-physical production. Adversarial example attacks can result in a serious threat to production systems and employees, due to their ability to manipulation the monitored production process unrecognized.

This work formally defines generation of adversarial examples as a constrained search problem and uses adversarial examples to retrain a deep neural network. Empirical results prove that a deep neural network hardened by CyberProtect show a significant less misclassification rate despite being attacked.

In future work, prevention of misclassification caused by adversarial example attacks will be explored for discrete manufacturing, in which time-dependent machine learning approaches are utilized.

References

1. J. Otto, B. Vogel-Heuser, and O. Niggemann. Online parameter estimation for cyber-physical production systems based on mixed integer nonlinear programming, process mining and black-box optimization techniques. *at-Automatisierungstechnik*, 66(4):331–343, 2018.
2. G. Reinhart, S. Krug, S. Huttner, Z. Mari, F. Riedelbauch, and M. Schlogel. Automatic configuration (plug & produce) of industrial ethernet networks. In *Proc.*

- 9th IEEE/IAS International Conference on Industry Applications (INDUSCON)*, pages 1–6, Sao Paulo, Brazil, nov 2010.
3. J. Otto, B. Vogel-Heuser, and O. Niggemann. Automatic parameter estimation for reusable software components of modular and reconfigurable cyber-physical production systems in the domain of discrete manufacturing. *IEEE Transactions on Industrial Informatics*, 14(1):275–282, 2018.
 4. L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda. Cyber-physical systems in manufacturing. *International Academy for Production Engineering Annals*, 65(2):621–641, 2016.
 5. D. Hossain, G. Capi, M. Jindai, and S. Kaneko. Pick-place of dynamic objects by robot manipulator based on deep learning and easy user interface teaching systems. *Industrial Robot: the international journal of robotics research and application*, 44(1):11–20, 2017.
 6. S. Jeschke, C. Brecher, T. Meisen, D. Özdemir, and T. Eschert. *Industrial internet of things and cyber manufacturing systems*, pages 3–19. Springer International Publishing, 2017.
 7. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proc. of the 2nd International Conference on Learning Representations (ICLR)*, Banff, Canada, apr 2014.
 8. K. Stouffer, J. Falco, and K. Scarfone. Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16, 2011.
 9. I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proc. of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, USA, may 2015.
 10. N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *Proc. of the 38th IEEE Symposium on Security and Privacy (SP)*, pages 39–57, San Jose, USA, may 2017.
 11. Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.
 12. N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *Computing Research Repository (CoRR)*, abs/1605.07277, 2016.
 13. N. Papernot and P. McDaniel. On the effectiveness of defensive distillation. *Computing Research Repository (CoRR)*, abs/1607.05113, 2016.
 14. W. Xu, D. Evans, and Y. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *Computing Research Repository (CoRR)*, abs/1704.01155, 2017.
 15. M. McCann and A. Johnston. Uci ml repository secom dataset, 2008. [Online; accessed 2018-02-05].
 16. M. Abadi et al. Tensorflow: A system for large-scale machine learning. In *Proc. of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, volume 16, pages 265–283, Savannah, USA, nov 2016.
 17. DP. Kingma and J. Ba. Adam: A method for stochastic optimization. *Computing Research Repository (CoRR)*, abs/1412.6980, 2014.
 18. I. Goodfellow, N. Papernot, and P. McDaniel. cleverhans v2.0.0.: an adversarial machine learning library. *Computing Research Repository (CoRR)*, abs/1610.00768, 2016.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.