Evaluation of graphic data corruptions impact on artificial intelligence applications

O Molokovich¹, A Morozov², N Yusupova³, K Janschek⁴

¹,³ Ufa State Aviation Technical University, Ufa, K. Marx str., 12, 450000, Russia
² University of Stuttgart, Stuttgart, Pfaffenwaldring str., 47, 70550, Germany
⁴ Technische Universität Dresden, Dresden, Nöthnitzer str., 43, 01187, Germany

E-mail o.molokovich@ya.ru, andrey.morozov@ias.uni-stuttgart.de, yussupova@ugatu.ac.ru, klaus.janschek@tu-dresden.de

Abstract. The article is devoted to the problem of the reliability of applications based on artificial intelligence. The authors made an attempt to evaluate the impact of the graphic data distortion at the input of a convolutional neural network on the result of image classification. The experiment is based on the fault injection method. A series of independent tests were carried out for such distortions as Gaussian noise, salt and pepper, speckle and Poisson noise, as well as median blur, motion blur, scene brightness changing, rotation, rain, and snow. The results showed that Gaussian noise was the least critical distortion; environmental conditions (rain, snow, brightness) and image rotation up to 20 degrees are less critical than focus losing and motion blur, while the most critical distortion is speckle noise. It was verified that preprocessing the input data of the neural network improves the accuracy of image recognition.

Keywords: Reliability; Cyber-physical system; Fault injection method; Neural network; Artificial Intelligence

1. Introduction

The Cyber-physical system (CPS) is an system consisting of interconnected intelligent and heterogeneous components that interact with each other. Various components of a CPS are prone to faults. A malfunction is an abnormal condition that can manifest itself as an error propagating through the CPS, and ultimately leading to a failure of the system as a whole. This paper is devoted to assessing the impact of CPS faults on its intelligent components using the example of graphic data distortions at the input of a convolutional neural network for image recognition as an example of an artificial intelligence (AI) application.

Many researchers are engaged in issue of AI applications robustness. For example, Qi et al. [1] drew attention to the problem of data quality in machine learning and data mining. Hynes et al. [2] developed a data preparation and transformation tool for machine learning. Huang et al. [3] discuss how an attacker modifying data can cause a machine learning model to behave incorrectly. Li et al. [4] created the CleanML benchmark to investigate the effect of data cleansing in machine learning. Several tools have also been created for introducing faults into images [5,6,7,8].
2. Cyber-physical systems reliability problems analysis
CPS mechanisms are controlled by computer algorithms and AI systems. AI applications are highly dependent on the input data quality. Measurement errors can be the cause of CPS dysfunction. There are CPSs that use a video camera to receive data from the outside world. The camera receives an image and transfers it to neural net, one of AI technologies. The network classifies the image and passes the result on to decide on the control action. An example of such a CPS is an autonomous driving system in which a camera is used to recognize traffic signs. Due to camera malfunctions or any network problems image distortions can occur. Then, the error extends to the driving scene recognition result, which may lead to erroneous vehicle behaviour. Image corruptions can be divided into two types:
1. Errors due to hardware malfunctions (Gaussian, shot, speckle, and Poisson noise, noise of salt and pepper, motion blur, defocusing).
2. Decreased quality due to environmental conditions (rain, snow, fog, brightness, image rotation).

3. Reliability evaluation method
In this paper, the error injection method is used to evaluate the reliability of AI applications. Faults are embedded in the data - images of the test dataset.

4. Classification problem interpreting
Video camera image recognition can be interpreted as one of the variants of the classification problem. The classical formulation of the pattern recognition problem is as follows [9]: $M$ is given set of objects that need to be classified. The set is represented by subsets $K_1,\ldots,K_l$, which are called classes: $M = K_1 \cup \ldots \cup K_l$. The partitioning $M$ is not fully specified, only some information $I_i$ about the classes $K_1,\ldots,K_l$ is given. Similarly, the valid object $S$, whose belonging to classes $K_1,\ldots,K_l$ is unknown, is determined by the values of some characteristics that make up $I(S)$. It is required according to information $I_i$ and description of object $I(S)$ to set the value of property $S \in K_j$, $j = 1,\ldots,l$ for each $j$.
In the last decade, solutions based on a neural network approach have been successfully solving the problem of pattern recognition. The current research also used a neural network for image recognition.

5. Classification quality indicator
A natural assessment of the classification quality is the Mean Consequential Error:

$$MCE = \sum_{i=1}^{l} a(x_i) = y_i,$$

where $a(x_i)$ – predicted class, $y_i$ – original class, $l$ – number of objects in the sample.

The metric shows the percentage of objects for which the algorithm gave correct answers. When used for unbalanced classes, the accuracy is calculated as a weighted sum, the weights are equal to the weights of the classes.

6. Case study neural network
To study the consequences of input data distortions pretrained open-source convolutional neural network (NN) was used [9,10]. The class-unbalanced test dataset network classification accuracy is 96%. It is based on a network designed by Vamsi Ramakrishnan and has two parts: localization network with spatial transformer and convolutional neural network for classification. The network consists of 15 layers. First, the image is fed into the localization network, which determines the transformation applied to the image by the spatial transformer. It leads to greater invariance to displacement, scaling, rotation, and general deformation [12]. Then the picture is fed into the neural network, which consists of several stacked convolutional layers. This part of the network is based on the architecture described by K. Simonyan and A. Zisserman [13], with each stack having the same structure: multiple convolutions followed by a PReLU activation function, and one Max Pool layer. By combining the outputs of the last three Max Pool layers before entering fully connected layers, it is possible to obtain the semantics of several extracted features from different levels of abstraction, which improves object detection [14,15].
7. Training dataset
Ramakrishnan network was developed for recognition of the German Traffic Sign Recognition Benchmark (GTSRB) [11]. It consists of over 50,000 images from 43 unbalanced class frequencies and is divided into three subsets for training, validation, and testing.

8. Fault injection tools overview
AVFI is a tool for entering faults into autonomous vehicle control systems. It allows one to study how various sensor malfunctions affect performance [8]. DeepRoad is a deep neural network framework, which automatically synthesizes driving scenes without using image transformation rules. It can create scenes with different weather conditions [7]. DpEmu is a software environment for emulating data problems, it is possible to introduce errors like rotation, snow, rain, delay of sensor signals, and others into various types of data sets [5]. DeepMutation is a system close to DpEmu that generates artificial faults for data and model implementation. The capabilities of these two tools are similar [6]. OpenCV is an open source cross-platform library that allows to process images in various ways [16].

9. Computer experiment

9.1. Input data
The input image set consists of 2950 class balanced images from the GTSRB test dataset. The instances distribution by class is a set of pairs (name of the picture, class number).

9.2. Methodology
The experiment of the evaluation of images distortions impact on the recognition result consists of an independent series sequence. Series algorithm is shown in Figure 1. The main steps are:
1. Select the distortion type and determine the range its power values.
2. For each range value:
   A. Inject distortion into images.
   B. Classify
   C. Calculate image recognition accuracy.

![Figure 1. Series block diagram](image)

9.3. Software
The computer experiment was implemented using Python. DpEmu, and OpenCV was chosen as distortion injection tools. Both libraries are also written in Python. The environment was deployed using the Ubuntu-18.04 operating system.
9.4. Data corruption

Blurring is a filter that defines the output pixel value as the weighted sum of the output pixel values (formula 2):

\[ g(i, j) = \sum_{k,l} f(i+k, j+l) h(k, l) \]

where \( h(k, l) \) – filter coefficients, thus filter is a visualization of coefficients sliding across the image.

Gaussian Blur is a filter that uses a Gaussian function or normal distribution to compute transform applied to each pixel in an image (formula 3):

\[ G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

where \( x \) – distance from the origin in the horizontal axis, \( y \) – distance from the origin in the vertical axis, \( \sigma \) – the standard deviation of the normal distribution.

The formula result in two dimensions is concentric circles with a normal distribution from the center point. The values from this distribution are used to construct a convolution matrix that is applied to the original image. The new value of each pixel is set as a weighted average of the neighboring pixels. The original pixel takes on a heavyweight, the neighboring pixels get smaller.

Gaussian blur implementation using OpenCV is presented below:

\[
\text{output_image} = \text{cv2.GaussianBlur(input_image, (ksize.width, ksize.height), 0)},
\]

where output_image – distorted image, input_image - clear image, width and height of the sliding window are determined by formulas 4, 5:

\[
ksize.width = 2*n - 1, \quad ksize.height = 2*m - 1,
\]

where \( m, n \in \mathbb{N} \) – distortion parameters. Blurred image example is shown in Figure 2, Figure 3 shows the original image.

Median blur assigns to each pixel the average value of neighboring pixels from some square neighborhood:

\[
\text{output_image} = \text{cv2.medianBlur(input_image, ksize)},
\]

where output_image – distorted image, input_image – clear image, window size is determined by the formula 6:

\[
ksize = 2*k - 1,
\]

where \( k \in \mathbb{N} \) – distortion parameters.

Motion blur is also implemented using OpenCV:

\[
\text{output_image} = \text{cv2.filter2D(input_image, (ksize.width, ksize.height), 0)},
\]

where output_image – distorted image, input_image - clear image, sliding window width and height are determined by formulas 7, 8:

\[
ksize.width = 2*n - 1, \quad ksize.height = 2*m - 1,
\]

where \( m, n \in \mathbb{N} \) – distortion parameters. Processed image example is shown in Figure 4.

Speckle noise is implemented as the sum of the pixel value and its product by the distortion parameter:

\[
\text{output_image} = \text{input_image} + \text{input_image} * n,
\]

where output_image – distorted image, input_image – clear image, \( n \in \mathbb{N} \) – distortion parameter. Image example in Figure 5.

Gaussian noise is implemented as the sum of the point value and the value of a random variable \( X \), obeying the normal distribution law (formula 9):

\[
X = f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.
\]
where \( \mu \) – expected value, \( \sigma \) – standard deviation. Image example in Figure 6.

Poisson noise is implemented as a change in the values of the components (0-255) of red, green and blue colours at \( H \) points of the picture. Random variable \( H \) obeys the Poisson distribution law (formula 10):

\[
H = \frac{\lambda^k}{k!} e^{-\lambda},
\]

where: \( k \) – events number, \( \lambda \) – observed interval.

Salt and pepper noise is implemented as a change of the red, green, blue components (0-255) at random picture points. The distortion parameter is distorted pixels’ percentage. Image example in Figure 7.

Image rotation is implemented using the DpEmu. Distortion parameter is rotation angle. Sample image in Figure 8.

Snowy and rainy scenes are also generated using dpEmu. In these cases, the parameter is the probability of a snowflake or raindrop in percent. Examples in figures 9 (for snow) and 10 (for rain).

Changing brightness is another distortion under consideration. Image example in the Figure 11.

10. Results

In total, 332 experiments were carried out to recognize a set of 2950 images. In each experiment, different distortions were applied to the pictures with different intensities. The graph of the recognition accuracy change depending on the filter and the strength of distortion is shown in Figure 12.

At the second iteration, the Gaussian blurred image recognition accuracy increases slightly. This can be explained by the realism of the GTSRB dataset. Blur reduces noise in the original images and contributes to better classification. However, the Median blur does not improve accuracy. Motion blur dramatically reduces the quality metric from 0.901 to 0.870. Then the accuracy almost returns to the original value and falls unevenly below. Slight picture blurring does not hinder recognition so much. But with an increase in the distortion power (by 5 iterations), the accuracy drops by almost 12 percent. The graph shows the influence of Gaussian noise on the metric depending on the mathematical expectation of its occurrence probability (\( \sigma = 0.3 \)). We can say that the distortion is not critical to the classification process. At the same time, Poisson noise almost uniformly decreases the accuracy over the entire range of variation in the percentage of noisy points. Although the parameter for salt and pepper noise is pixel value, its effect is similar to Poisson noise. Perhaps a similar effect arises from a similar
filter implementation. Speckle noise critically reduces the accuracy to 0.574% already at the first iteration, then the rate of decrease in accuracy slows down. The graph of classification accuracy changes of images with a small image rotation angle indicates the network resistance to image rotation. We can say that is due to the specifics of the training set (the meaning of some traffic signs does not change even if they are turned over) and the localization network. It determines which transformations need to be applied to the picture, then increases the invariance to turns. Excessive brightness reduces the efficiency of the network already at the first iteration. With an increase in the parameter value, the picture becomes more and more like a white canvas. Snowflakes create a "missing" area in the image, as a white spot appears instead of scene detail. Accuracy drops rapidly across the entire range. Jumps on the chart can be associated with the superposition of snowflakes on top of each other.

Rain affects accuracy in a similar way to snow. This is probably due to the rain implementation in dpEmu. The weather condition impact evaluation will be more objective if the distortion is implemented in a different way.

In general, we can say that:

- Gaussian noise turned out to be the least critical distortion;
- environmental conditions (rain, snow, brightness) and image rotation up to 20 degrees turned out to be more critical;
- focus losing of and motion blur most strongly affect the accuracy;
- the most critical distortion is speckle noise.

The considered network turned out to be quite resistant to basic distortions applied with little force. In some cases, the small number of data errors in some cases slightly increased the classification accuracy. However, we shouldn't make any mistake that errors have a beneficial effect on network results. Due to the limitations of the hardware used, the experiment was conducted on a quarter of the GTSRB test data.

The results of the experiment highlighted the importance of pre-processing AI applications input data. Due to the peculiarities of the considered neural network architecture, namely the localization network in the first part, it turned out to be resistant to small rotations of the picture.

11. Conclusion
The article attempts to evaluate the impact of image distortions on artificial intelligence applications used in security-critical CPS.

The results of the experiment allowed us to conclude that:
1. The training dataset plays an important role, especially if it contains corrupted data.
2. Network resilience is highly architecture dependent.
3. Similar methods of injecting different distortions can lead to similar classification results.
4. It is necessary to carry out a similar experiment for different neural networks. This will allow identifying architectural features that contribute to improving network resilience.
5. Fault injection tools should be expanded. It will reduce the dependence of the results on the fault injection implementation and will increase research objectivity.
6. A lot of experiments carried out, perhaps, will allow deriving some recommendations for creating a fault resilient neural network.

12. Acknowledgment
The work is supported by the DAAD «Ostpartnerschaften» funds.

References
[1] Z. Qi, H. Wang, J. Li, and H. Gao 2018 Impacts of Dirty Data: and Experimental Evaluation
[2] N. Hynes, D. Sculley, G. Brain, M. T. Google Brain, and M. Terry 2017 The Data Linter: Lightweight, Automated Sanity Checking for ML Data Sets NIPS MLSys Workshop
[3] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar 2011 Adversarial machine learning Proc. of the 4th ACM workshop on Security and artificial intelligence - AISec ’11 (New York, USA: ACM Press)
[4] P. Li, X. Rao, J. Blase, Y. Zhang, X. Chu, and C. Zhang 2019 CleanML: A Benchmark for Joint Data Cleaning and Machine Learning [Experiments and Analysis] 4
[5] K. Jukka, Nurminen, T. Halvari, J. Harviainen, J. Myllari, A. Roysko, J. Silvennoinen and Tommi Mikkonen 2011 Software Framework for Data Fault Injection to Test Machine Learning Systems” Proc. of 30th Annual IEEE International Symposium on Software Reliability Engineering (ISSRE 2019); 4th IEEE International Workshop on Reliability and Security Data Analysis
[6] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao and Y. Wang. 2018 DeepMutation: Mutation Testing of Deep Learning Systems” Proc. International Symposium on Software Reliability Engineering, ISSRE, vol. 2018-October IEEE Computer Society, 11 2018
[7] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid 2018 DeepRoad:GAN-based metamorphic testing and input validation framework for au-tonomous driving systems” Proc. 33rd ACM/IEEEInt. Conf. AutomatedSoftw. Eng.
[8] S. Jha, S. S. Banerjee, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer 2018 AVFI: Fault Injection for Autonomous Vehicles Proc. 48th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks Workshops
[9] V. Ramakrishnan 2017 Traffic Sign Recognition
[10] M. Beyer 2019 Evaluation of the Effects of Common Hardware Faults on the Accuracy of Safety-Critical AI Components
[11] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel 2012 Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition Neural Networks (2012) issn: 0893-6080
[12] M. Jaderberg, K. Simonyan, A. Zisserman and Koray Kavukcuoglu 2015 Spatial Transformer Networks CoRR abs/1506.02025 (2015). arXiv: 1506 . 02025
[13] K. Simonyan and A. Zisserman 2014 Very Deep Convolutional Networks for Large-Scale Image Recognition CoRR abs/1506.02025 (2015). arXiv: 1506 . 02025
[14] S. Bell, C Lawrence Zitnick, K. Bala and R. Girshick 2016 Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks Proc. of the IEEE conference on computer vision and pattern recognition (USA, Las Vegas)
[15] B. Hariharan, P. Arbelaez, R. Girshick and J. Malik 2015 Hypercolumns for object segmentation and Fine-grained localization Proc. of the IEEE conference on computer vision and pattern recognition (USA, Boston)
[16] OpenCV official documentation