# ROBY: a Tool for Robustness Analysis of Neural Network Classifiers

Paolo Arcaini
*National Institute of Informatics*
Tokyo, Japan
arcaini@nii.ac.jp

Andrea Bombarda
*University of Bergamo*
Bergamo, Italy
andrea.bombarda@unibg.it

Silvia Bonfanti
*University of Bergamo*
Bergamo, Italy
silvia.bonfanti@unibg.it

Angelo Gargantini
*University of Bergamo*
Bergamo, Italy
angelo.gargantini@unibg.it

***Abstract*—Classification using Artificial Neural Networks (ANNs) is widely applied in critical domains, such as autonomous driving and in the medical practice; therefore, their validation is extremely important. A common approach consists in assessing the network *robustness*, i.e., its ability to correctly classify input data that is particularly challenging for classification. We recently proposed a robustness definition that considers input data degraded by alterations that may occur in reality; the approach was originally devised for image classification in the medical domain. In this paper, we extend the definition of robustness to any type of input for which some alterations can be defined. Then, we present ROBY, a tool for ROBustness analYsis of ANNs. The tool accepts different types of data (images, sounds, text, etc.) stored either locally or on Google Drive. The user can use some alterations provided by the tool, or define their own. The robustness computation can be performed either locally or remotely on Google Colab. The tool has been experimented for robustness computation of image and sound classifiers, used in the medical and automotive domains.**

***Index Terms*—NN classifier, robustness, ML testing**

## I. Introduction

*Artificial Neural Networks* (ANNs) are increasingly used to perform different activities [4], among which classification is one of the most popular. ANN-based classification is used in critical domains [13] as, e.g., during autonomous driving [7], or in the medical practice [6]; therefore, their validation is of paramount importance. A desired property of an ANN to be tested, is its *robustness*, i.e., the ability of the network to correctly evaluate unknown (not seen during training) inputs. Typical approaches define the robustness using *adversarial examples*, i.e., inputs that are particular challenging for the network under test. However, it has been noted that, since adversarial examples are often created by exploiting the internal structure of the network [17], they may not reflect real inputs that could occur during the network usage [10], [11], [20]. Therefore, we have recently proposed to define the robustness by considering *real alterations* that may occur to input data. In [5], we defined the typical alterations (e.g., blur) that could occur during image acquisition in the medical practice of cancer detection using Convolutional Neural Networks (CNNs); moreover, we also provided a formal definition of robustness that assesses *to what extent* the network is robust against image alterations. In a similar way, Secci and Ceccarelli [15] defined alterations that could occur during the image acquisition of an RGB camera (e.g., condensation), and assessed the performance decrease of an autonomous driving agent that takes in input such altered images.

Given the widespread use of ANNs in critical contexts, there is an increasing need for ANN testing approaches that are actually implemented in usable tools, so that they can be adopted in the development practice. However, while several techniques have been constantly proposed for ANN testing [14], [21], their implementation is often not available: a recent survey on ML testing [14] found that 71% of the surveyed papers do not provide any artifact. Even when the implementation is available, the techniques are usually only applicable to the considered domain, and they require quite some effort to be adopted in other contexts.

Our definition of robustness [5] was originally proposed and implemented only for image classification, and it was experimented only in the medical domain. However, the definition is quite general, and it is applicable to the classification of different types of data (e.g., images, sounds, etc.), once the alterations typical of the domain (e.g., medical domain, autonomous driving, speech recognition for domotics, etc.) are defined. Therefore, in this paper, we generalize our approach, and we present ROBY, a tool for ROBustness analYsis. The tool has been engineered so that it can be used, with minimal effort, by different users in different domains for different types of data. A user must only specify:

- the location of their test data set;
- how to retrieve the correct classification of the test data;
- which alterations to apply to input data; the user can use standard ones provided by the tool, or specify their own;
- where to run the robustness computation (either locally or on Google Colab).

As a result, the tool computes the robustness values for the different alterations, and produces plots that visualize how the accuracy changes when alterations are applied and, in this way, visualizes the robustness.

The tool is available at:

https://github.com/fmselab/roby

and it can also be installed using the package manager pip:

P. Arcaini is supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST, and Engineerable AI Techniques for Practical Applications of High-Quality Machine Learning-based Systems Project (Grant Number JPMJMI18BB), JST-Mirai.

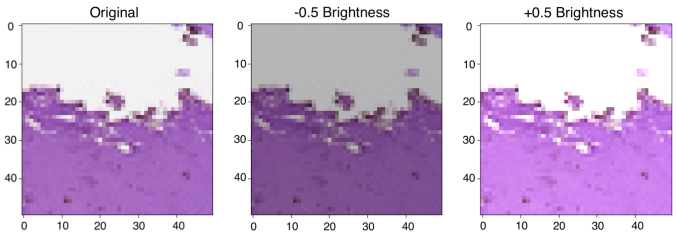


Fig. 1. Examples of brightness alteration

```
pip install roby
```

*Paper structure.* Sect. II introduces our definition of robustness. Sect. III explains the tool architecture, its requirements, and its usage. Then, Sect. IV shows its application to different domains, and Sect. V discussed some lessons we learned during the tool development. Finally, Sect. VI reviews some related work, and Sect. VII concludes the paper.

## II. Robustness definition

*Artificial Neural Networks* (ANNs) are used in many critical tasks, among which classification. For example, Convolutional Neural Networks can be used to analyze images.

In the following, we consider ANNs trained to be used as classifiers, i.e., to assign a label (taken from a set of possible *categories*) to an input (e.g., an image, a sound, a text, etc.).

**Definition 1** (Classifier)**.** A *classifier* $C$ for inputs $I$ can be seen as a function that assigns a label $l$ to an input $p \in I$, i.e., $C(p) = l$.

We measure the *quality* of an ANN classifier in terms of *accuracy*. However, other quality metrics could be used.

**Definition 2** (Accuracy)**.** The *accuracy* of a classifier $C$ w.r.t. a set of inputs $P \subseteq I$ is defined as the ratio of correctly evaluated inputs, i.e.,

$$acc(C, P) = \frac{|\{p \in P \mid C(p) = correctLabel(p)\}|}{|P|}$$

where $correctLabel$ gives the correct evaluation of an input $p$.

If we alter the input data in $P$, the accuracy of the classifier will likely decrease, and the decrement will be more evident for bigger alterations. Examples of images obtained with the brightness alteration are shown in Fig. 1. Fig. 2 shows how the accuracy of a CNN for image classification, changes by altering the brightness of the images in $P$.

Some alterations are more plausible than others, as they may occur in reality during any stage of the processing of the input data. Therefore, in [5] we proposed to compute the robustness of a network w.r.t. these alterations. We here recall our definitions of *alteration* and *robustness*.

**Definition 3** (Alteration)**.** An *alteration* of type $A$ of an input $t$ is a transformation of $t$ that mimics the possible effect on $t$ when a problem during its acquisition, or in its elaboration, occurs in reality. In the following, we identify with $[L_A, U_A]$ the range of plausible alterations of type $A$; moreover, we identify with $P^{A_i}$ the set of data obtained by altering all the input data in $P$ with an alteration of type $A$ of *level* $i \in [L_A, U_A]$. We require one element $u \in [L_A, U_A]$ to be the *unaltered value*, i.e., $P^{A_u} = P$.

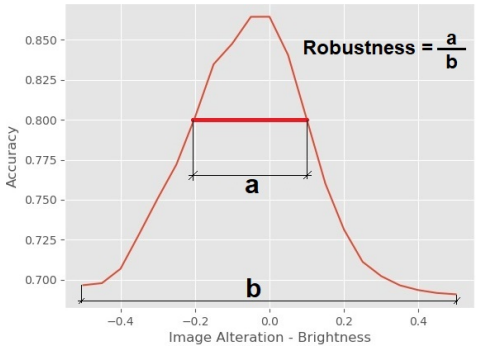


Fig. 2. Accuracy change when brightness is altered

Given a set of alterations, we define the robustness as follows.

**Definition 4** (Robustness)**.** Let $\Theta$ be a threshold representing the minimum accepted accuracy. The *robustness* of a classifier $C$ w.r.t. alteration of type $A$ in the range $[L_A, U_A]$ (using a set of inputs $P$) is defined as the percentage of alteration values for which the accuracy is above $\Theta$. Formally:

$$rob_A(C, P) = \frac{\int_{L_A}^{U_A} H(acc(C, P^{A_i}) - \Theta) di}{U_A - L_A} \quad \text{where} \quad H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Fig. 2 shows the robustness representation with $\Theta = 80\%$. Directly computing robustness is not feasible, since it requires to use an infinite number of alterations. Therefore, we approximate it as follows.

**Definition 5** (Approximate robustness)**.** Given $n$ equi-distributed points $SP = \{i_1, \ldots, i_n\}$ sampled in the interval $[L_A, U_A]$ of all the possible alterations of type $A$, the *approximate robustness* is defined as:

$$rob_A(C, P) = \frac{|\{i \in SP \mid acc(C, P^{A_i}) \geq \Theta\}|}{|SP|}$$

## III. ROBY tool

ROBY (ROBustness analYzer) is a Python tool to perform robustness analysis of neural network classifiers, as defined in Sect. II. Given a trained model, an already labeled data set, and the list of classification classes, ROBY applies the selected alterations and computes the robustness based on the defined accuracy threshold.

Regarding the alterations, users can apply predefined alterations or define their own. Currently, ROBY provides, as predefined alterations, those usually applied to images: vertical translation, horizontal translation, compression, Gaussian noise, blur, brightness, and zoom. Once an alteration has been applied, the tool computes the robustness, displays the robustness result, and plots the accuracy over different levels of alteration. In case the data set can be subject simultaneously

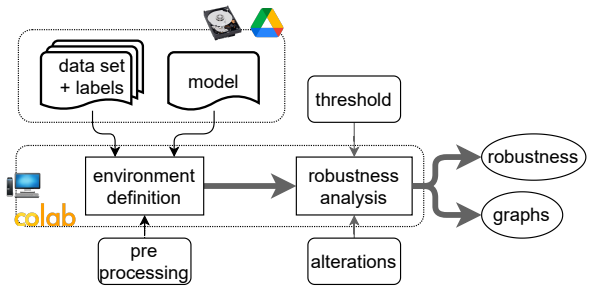


Fig. 3. ROBY workflow

to more than one alteration (e.g., an image could be translated and blurred), the user can apply a sequence of alterations to test the robustness of the network and simulate the effect of a composed alteration.

Since the data set could require pre-processing activities (e.g., removing white space from an image) before the input is given to the network for classification, the tool allows the user to define functions for the pre-processing tasks.

The tool has been designed with the following main characteristics set as goals. (i) *Usability*: for instance, we provide some examples and a wiki; moreover, the code is documented and uses typing annotations to guide the user in the use of methods. (ii) *Portability*: the tool can be used locally on any computer with Python or on Google Colab. (iii) *Extensibility*: the user can extend classes and redefine methods (see Sect. III-C) based on the specific case study.

### A. Requirements

Before using ROBY tool, the user must make sure that their system satisfies the following requirements:

- The model represents a classifier and is written in a format supported by Keras, i.e., `HDF5` or `SavedModel`.
- All the inputs in the test data set must be previously labeled and must be expressible in the `np.ndarray` format (note that it is a rather general format in which images, audio, text, and video can be represented).
- It is possible to represent each alteration as an input modification between a minimum and a maximum threshold.

### B. ROBY workflow

Fig. 3 shows all the activities required by ROBY that the user has to carry on in order to perform the robustness analysis of a given ANN. In particular, the user has to:

1) Supply a data set, together with the labels (possibly given as a function, `labeler`, applicable to each input data and giving as output the correct label) and a model. Data set and model can be stored either in the local hard drive or on Google Drive.
2) Define the environment that optionally includes, besides the input data together with their labels and the model, also the pre-processing function.
3) Select suitable *alterations* among those provided by ROBY or define new ones in accordance with the domain.
4) Specify the desired *threshold* for computing the robustness.
5) Run ROBY which computes the robustness measure, and produces plots showing how the accuracy of the model changes w.r.t. different levels of alterations. The robustness analyzer can run either on the local PC or on Google Colab[1].

### C. Extension points

ROBY has been designed to be extensible as much as possible. The user can define a method to load the data set, a method to adapt to different data formats, and another method to assign labels to input data. Moreover, ROBY allows the user to define custom alterations and pre-processing functions. These extension points are better explained in the following.

- **Data loading:** ROBY works for ANNs used as classifiers receiving `np.ndarray` input data. Users can create a testing environment `EnvironmentRTest` by giving either the paths for all the input data or a list of data already in the array format. In the former case (i.e., paths are given), the user shall specify the way to be used to convert the file data into the `np.ndarray` format by declaring a `reader(file_name)` function receiving the path as input and giving the array representation as output.
- **Data labeling:** correct labels for input data can be given either with a list of all the labels or by using a `labeler` function. In the former case, the list must be of the same size as the data set, while in the latter case the user must define a function receiving a data (in either `np.npdarray` format or in string representing the file path) and returning a string representing the real label. Typical `labeler` functions will use the image name or folder name to extract the correct classification.
- **Custom alterations:** ROBY introduces the abstract class `Alteration` that can be extended to create custom alterations, expressible as input modifications within a minimum and a maximum threshold. When extending the abstract class, the user must implement the functions `name()`, to return the name of the alteration, and `apply_alteration(data, alteration_level)`, which, given the data in a `np.ndarray` object, returns the data (still in `np.ndarray`) with the alteration applied with a desired alteration level. Moreover, ROBY provides a class `AlterationSequence`, that can be used to represent an alteration caused by the composition of more alterations.
- **Pre-processing:** ANNs could have been trained with data of different shapes, sizes, or formats than the ones used for testing. For these reasons, during the declaration of the testing environment `EnvironmentRTest`, users can specify a *pre-processing* function. This function is applied to each input data, possibly after an alteration is applied, before its recognition by the ANN. Typical `pre-processing` functions remove white border from images, resize or extract relevant part from the input, or clip the volume of an audio track.

[1]https://colab.research.google.com

TABLE I
SUMMARY OF THE CHARACTERISTICS OF THE EXAMPLE CASE STUDIES

| Case Study | Model format | Labeling method | Pre-processing function | Local | Cloud | Alterations | Input data |
|---|---|---|---|---|---|---|---|
| Breast cancer | .model | labeler function | X | X | Google Colab + Drive | standard | histological exam images [8] |
| MNIST | .h5 | labels list | X | X | | standard | MNIST data set [3] |
| GTSRB | .h5 | labels list | X | X | | camera failures [15] | GTSRB data set [16] |
| MNIST Audio | .h5 | labels list | X | X | | audio Gaussian noise | MNIST audio data set [1] |

## IV. DEMONSTRATION

To further evaluate the efficacy and to demonstrate the functionalities of ROBY, we have tested it on several case studies, which differ in terms of model format, the way in which the user gives the correct labels, input type and size, and type of alterations. The summary of these case studies is reported in Table I. For all the case studies, we have tested the models using a limited number of input data in the test set, since we are only interested in showing the functionalities of the tool —and this can be done with a limited number of input data too— and not in computing the precise robustness of the models.

### A. Breast cancer diagnosis

Breast cancer diagnoses, in particular for Invasive Ductal Carcinoma (IDC), are based on the analysis of images of histological features of tissue or cells removed with surgery or biopsy. These images are captured by a microscope and examined by pathologists to make a decision about the benignity or the malignity of the suspected cancer. The images we used for robustness analysis are those from a publicly available data set curated by [8], while the model is the same we presented in [5]. A particular aspect of the data set we used is that the real label of the images is contained in the name of the files. Thus, we have defined a `labeler` function, extracting the class (0-negative or 1-positive) from the file names.

For this case study[2], we have implemented two solutions:

- A local analysis (`breast_cancer_local.py`) in which the model and the data set are available locally, and the robustness evaluation is run on the user's PC.
- A cloud solution (`breast_cancer_colab.ipynb`), in which the data set, the model, and categories definition are loaded from Google Drive, and the robustness analysis is performed on Google Colab. To make this possible, we have exploited the functionalities offered by the `CloudTools` module included in ROBY.

Using ROBY, we have obtained the results in Table II, when threshold $\Theta = 0.8$ is used. The last row in Table II shows the robustness of the model when a sequence of alterations is used. In this case, for each image, a defined level of zoom and brightness alterations is applied. This can be useful to represent alterations made up of the composition of other sub-alterations.

TABLE II
ROBUSTNESS RESULTS FOR THE BREAST CANCER DIAGNOSIS CASE STUDY

| Alteration | # intervals | $[U_A, L_A]$ | Robustness [%] |
|---|---|---|---|
| Gaussian noise | 20 | $[0, 1]$ | 100.00 |
| Compression | 20 | $[0, 1]$ | 4.76 |
| Vertical translation | 20 | $[-1, 1]$ | 100.00 |
| Horizontal translation | 20 | $[-1, 1]$ | 100.00 |
| Blur | 20 | $[0, 1]$ | 38.10 |
| Brightness | 20 | $[-0.5, 0.5]$ | 19.05 |
| Zoom | 20 | $[0, 1]$ | 71.43 |
| Seq (Zoom, Brightness) | 20 | $[0, 1], [-0.5, 0.5]$ | 19.05 |

TABLE III
ROBUSTNESS RESULTS FOR THE HAND-WRITTEN DIGITS CLASSIFICATION CASE STUDY

| Alteration | # intervals | $[U_A, L_A]$ | Robustness [%] |
|---|---|---|---|
| Gaussian noise | 20 | $[0, 1]$ | 100.00 |
| Compression | 20 | $[0, 1]$ | 4.76 |
| Vertical translation | 20 | $[-1, 1]$ | 100.00 |
| Horizontal translation | 20 | $[-1, 1]$ | 100.00 |
| Blur | 20 | $[0, 1]$ | 4.76 |
| Brightness | 20 | $[0, 1]$ | 4.76 |
| Zoom | 20 | $[0, 1]$ | 76.19 |

### B. Hand-written digits classification

MNIST (Modified National Institute of Standards and Technology database) is a well-known data set containing a lot of images of hand-written number digits [3]. It is also shipped with Keras, in which the images are already in the required `np.ndarray` format and are accompanied by a list containing each image label.

To test ROBY on this case study, we have analyzed the same standard alterations used for the breast cancer recognition model, by testing a publicly available model [2] only on the first 200 images of the data set[3] and we have obtained the results in Table III using $\Theta = 0.8$.

### C. Traffic signs recognition

GTSRB (German Traffic Sign Recognition Benchmark) is the data set coming from a multi-class, single-image classification challenge [16] including more than $50,000$ images, classified in more than $40$ classes, of German traffic signs. The recognition of traffic signs is challenging, since it is performed by cameras under different environmental conditions, and the robustness of systems recognizing them must be proved, since

[2]https://github.com/fmselab/roby/tree/main/code/roby_Use/breast-cancer

[3]https://github.com/fmselab/roby/tree/main/code/roby_Use/mnist

TABLE IV
ROBUSTNESS RESULTS FOR THE TRAFFIC SIGNS RECOGNITION CASE STUDY

| **Alteration** | **# intervals** | $[U_A, L_A]$ | **Robustness [%]** |
|---|---|---|---|
| Ice | 20 | $[0, 1]$ | 4.76 |
| Condensation | 20 | $[0, 1]$ | 4.76 |

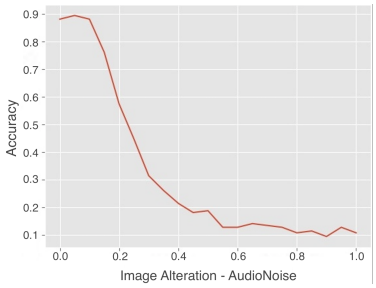


Fig. 4. Digit recognition from sound case study – Accuracy over Gaussian audio noise alteration

they are usually performed by self-driving cars or, in general, by safety-critical systems.

To simulate the effects of weather on the recognition of the traffic signs, we have defined custom alterations exploiting the ones proposed by [15], and we have assessed the robustness of the model w.r.t. condensation and rain[4], obtaining the results in Table IV, when $\Theta = 0.8$ is used. As we can see from these results, ROBY allows us to prove that, even though the model we have used was very accurate ($acc = 97.5\%$), a small alteration degrades significantly the accuracy of the classification.

### D. Digit recognition from sound

The previous examples are all based on image classifiers. However, ROBY supports all kinds of classifiers, for which input data can be given in `np.ndarray` format. To demonstrate the applicability of our tool to other input data types, we have tested a model for digit recognition from sound: different people pronounce digits and the model aims to recognize them. It is publicly available in [1], along with the data set that contains the real label of each sound in the name of the file.

For this case study, we have defined a customized alteration, namely a *Gaussian noise* for audio signals, and we have analyzed the robustness of the ANN, obtaining the result in Fig. 4, corresponding to a robustness $rob = 14.28\%$ when a threshold $\Theta = 0.8$ is chosen.[5]

## V. LESSON LEARNED

Using ROBY, we have been able to compute the robustness and identify the weaknesses of the analyzed models. This is a very important feature, since looking only at the accuracy of the model can be not enough for assessing the quality of a given ANN. For instance, the model presented in Sect. IV-C has a really high accuracy, but a very low robustness.

ROBY has been designed in an incremental way. At the beginning, we tried to implement a tool executing the same operations we have presented in [5], so that it was applicable to image classifiers. During the development, we have realized that our definition of *robustness* was general enough to extend the tool to all the domains in which a classifier is used, independently on the type of input data. This has required the introduction of abstract classes and methods, to allow users to define their own alterations, based on the input data type. Having chosen Python as programming language, we noticed that it was hard to check the correctness of programs that use or extend ROBY, since static code analyzer (e.g., myPy or Pylint) could not recognize type errors. Thus, we have decided to include type annotations to aid the final user in choosing the right variable types.

Moreover, it is widely known that machine learning applications are highly demanding in terms of time and computational resources, since they include a lot of data to be processed and perform complex operations. For this reason, during the development process of ROBY, we have tried to re-engineer the code in order to improve the performance. To deal with the time-consumption of machine learning applications, we have included in ROBY modules to allow a cloud execution on Google Colab. At the beginning we thought that Google Colab would have been faster than a standard desktop. At the end of the day, we have discovered that the free version of Google Colab has more or less the same performances than a regular desktop. Nevertheless, the advantage of using the cloud is that a user has immediately a working environment that can be scaled up easily.

We have decided to implement ROBY as a library and not as an interface (either GUI or CLI), since a rigid interface would have limited the extensibility which ROBY has been designed for. Thus, we assume that the final user has a good expertise in developing machine learning applications, or can modify the provided examples to adapt ROBY for their own application. However, in the future we may add an interface in order to make ROBY usable to non-experienced users.

## VI. RELATED WORK

Testing machine learning models is, recently, a hot topic which has been defined over several properties, e.g., correctness, robustness, and fairness [21]. However, it is difficult to test machine learning applications using software testing techniques originally designed for code [14]. Robustness, intended as the ability of the model to give the correct output even when a small perturbation of the input occurs, is often evaluated in machine learning testing. To address the complexity of testing, several solutions have been proposed, but many of them are domain-specific and cannot be generalized to all kinds of neural networks or input types. An example is the one presented in [15], where the authors test the behavior of a model in autonomous driving applications when RGB camera failures occur. Testing models used in autonomous driving vehicles is

[4]https://github.com/fmselab/roby/tree/main/code/roby_Use/trafficsigns-imagefailures

[5]https://github.com/fmselab/roby/tree/main/code/roby_Use/speech_recognition

also performed by DeepTest [18], a systematic testing tool for automatically detecting erroneous behaviors of DNN-driven vehicles that can potentially lead to fatal crashes, and by DeepRoad [22], a framework to perform metamorphic testing and input validation. Some solutions, such as PRODeep [9], aim at evaluating the robustness of a model using several approaches, such as constraint-based, abstraction-based, and optimization-based robustness checking algorithms. However, PRODeep is a standalone software and not a library, and, therefore, it cannot be extended with customized alterations or integrated in other software to perform combined analysis. Other solutions are focused on robustness w.r.t. adversarial examples. For example, [12] proposes a Python library supporting developers and researchers in defending Machine Learning models against adversarial threats. Another example of tool focusing on adversarial robustness is EvalDNN [19] which aims to evaluate deep learning systems in general, supporting, besides adversarial robustness, also neuron coverage and top-k accuracy. This is a different concept of robustness, compared to the one we propose with ROBY, which is more general and focused on plausible alterations. However, ROBY can be used to evaluate also adversarial robustness. In ROBY, the user could define a customized alteration which generates adversarial examples in a defined range, and test the robustness of the model w.r.t. them.

## VII. Future Work and Conclusion

In this paper, we have presented ROBY, a tool for robustness analysis of neural network classifiers. In our preliminary experiments, we have tested ROBY on several case studies, demonstrating that it is applicable to different input data formats, data locations, alterations, and in different execution environments (either locally or on the cloud). However, the tool can be further extended. As future work, we plan to enhance the functionalities of ROBY by adding functions to generate augmented data sets starting from the user-defined alterations. Indeed, as we have already demonstrated in [5], using such augmented data during the training phase can significantly increase the robustness (and often the accuracy) of a neural network. Furthermore, we are extending ROBY to analyze the robustness of other neural networks in addition to those used for classification, for example for data prediction, and with other models beside the Keras format. In terms of performance, we are investigating techniques suitable for reducing the required time, such as prioritization or approximation methods.

## Acknowledgment

We would like to thank Danilo Bertocchi for the preliminary work done for his master thesis.

## References

[1] Digit Recognition from Sound. https://github.com/adhishthite/sound-mnist.

[2] Handwritten digit recognition with MNIST and Keras. https://github.com/Curt-Park/handwritten_digit_recognition.

[3] The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[4] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[5] P. Arcaini, A. Bombarda, S. Bonfanti, and A. Gargantini. Dealing with robustness of convolutional neural networks for image classification. In *2020 IEEE International Conference On Artificial Intelligence Testing (AITest)*, pages 7–14, 2020.

[6] Q. Fu, F. Yang, J. Zhao, X. Yang, T. Xiang, G. Huai, J. Zhang, L. Wei, S. Deng, and H. Yang. Bioinformatical identification of key pathways and genes in human hepatocellular carcinoma after CSN5 depletion. *Cellular Signalling*, 49:79–86, sep 2018.

[7] H. Fujiyoshi, T. Hirakawa, and T. Yamashita. Deep learning-based image recognition for autonomous driving. *IATSS Research*, 43(4):244–252, dec 2019.

[8] A. Janowczyk and A. Madabhushi. Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases. *Journal of Pathology Informatics*, 7(1):29, July 2016.

[9] R. Li, J. Li, C.-C. Huang, P. Yang, X. Huang, L. Zhang, B. Xue, and H. Hermanns. PRODeep: A platform for robustness verification of deep neural networks. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, pages 1630–1634, New York, NY, USA, 2020. Association for Computing Machinery.

[10] R. Mangal, A. V. Nori, and A. Orso. Robustness of neural networks: A probabilistic and practical approach. In *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results*, ICSE-NIER '19, pages 93–96, Piscataway, NJ, USA, 2019. IEEE Press.

[11] D. Marijan, A. Gotlieb, and M. K. Ahuja. Challenges of testing machine learning based systems. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, pages 101–102, April 2019.

[12] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards. Adversarial robustness toolbox. *CoRR*, 1807.01069, 2018.

[13] W. Rawat and Z. Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29(9):2352–2449, sep 2017.

[14] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella. Testing machine learning based systems: a systematic mapping. *Empir. Softw. Eng.*, 25(6):5193–5254, 2020.

[15] F. Secci and A. Ceccarelli. On failures of RGB cameras and their effects in autonomous driving applications. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 13–24, 2020.

[16] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, aug 2012.

[17] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings*, 2014.

[18] Y. Tian, K. Pei, S. Jana, and B. Ray. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, ICSE '18, pages 303–314, New York, NY, USA, May 2018. ACM.

[19] Y. Tian, Z. Zeng, M. Wen, Y. Liu, T. yang Kuo, and S.-C. Cheung. EvalDNN: A toolbox for evaluating deep neural network models. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*. ACM, jun 2020.

[20] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See. DeepHunter: A coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2019, pages 146–157, New York, NY, USA, 2019. ACM.

[21] J. M. Zhang, M. Harman, L. Ma, and Y. Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, pages 1–1, 2020.

[22] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid. DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, pages 132–142, New York, NY, USA, 2018. Association for Computing Machinery.