On the Practical Generation of Counterfactual Examples*

D.E. Namiot, E.A. Ilyushin, I.V.Chizov

Lomonosov Moscow State University, Moscow, Russia

Abstract. One of the important elements in evaluating the stability of machine learning systems are the so-called adversarial examples. These are specially selected or artificially created input data for machine learning systems that interfere with their normal operation, are interpreted or processed incorrectly. Most often, such data are obtained through some formal modifications of the real source data. This article considers a different approach to creating such data, which takes into account the semantic significance (meaning) of the modified data - counterfactual examples. The purpose of the work is to present practical solutions for generating counterfeit examples. The consideration is based on the real use of counterfactual examples in assessing the robustness of machine learning systems.

Keywords: *machine learning, adversarial examples, counterfactual examples.*
DOI: 10.14357/20790279230109

Introduction

This article is a continuation of a series of publications devoted to robust machine learning models [1, 2]. It was prepared as part of the project of the Department of Information Security of the Faculty of Computer Science of Moscow State University named after M.V. Lomonosov on the creation and development of the master's program "Artificial Intelligence in Cybersecurity" [3].

The special impact on the elements of the pipeline of machine learning systems is called attacks on machine learning systems. Among such influences, the so-called adversarial attacks stand out. This is a special change in the input data, designed to change the operation of the machine learning system ("cheat" the system) or to achieve its desired behavior.

Adversarial attacks rely on the complexity of deep neural networks and their statistical nature to find ways to exploit them and change their behavior. There is no way to detect malicious activity with the classic tools used to protect software from cyber threats. Adversarial attacks manipulate the behavior of machine learning models. Most of the examples relate to working with images, but in reality, there are examples of attacks on text analysis systems, audio data classification (speech recognition), and time series analysis. In general, they can be considered as some universal risk for machine learning (deep learning) models [2]. There are various attempts to explain the nature of their existence. According to one hypothesis, adversarial attacks exist due to the non-linear nature of the systems, which leads to the existence of some data areas not covered by the generalization algorithm. According to others, it is, on the contrary, retraining of the system, when even small deviations from the training data set are processed incorrectly.

The term adversarial attack is often used loosely to refer to various types of malicious actions against machine learning models. And the term adversarial example describes the data used in an adversarial attack. The formal definition of a threat model for a classification problem (it is a typical use case for so-called critical applications) could be described via the following set of statements. Suppose we have an initial data set X and a finite set of class labels Y, it is necessary to find a mapping $f: X \rightarrow Y$. This mapping f is vulnerable to adversarial attacks when there is a mapping A such that for any $x \in X$ there exists $\tilde{x} = A(x)$ for which $f(\tilde{x}) \neq y$, given that $f(x) = y$.

The construction of adversarial examples, classically, is the search for minimal perturbations of the correct input data, which change the operation of the classifier. In this case, the search for such perturbations is performed purely formally (L-norms for images). In the article, we consider the issues of meaningful generation of test cases for machine learning systems in the form of so-called counterfactual examples. There

* This research has been supported by the Interdisciplinary Scientific and Educational School of Moscow University "Brain, Cognitive Systems, Artificial Intelligence".

are already a fairly large number of publications on this topic, but not all of the proposed approaches are really useful or even just applicable. The purpose of this work is to present exactly the approaches actually used in practice.

Common to all machine learning systems is, obviously, only the presence of a pipeline with standard stages: data selection (selection), model training, and testing (practical use) of the trained model. At each stage, developers have a sufficient selection of ready-made tools. On the one hand, this is undoubtedly a big plus. And technically, it's hard to imagine that there will be a single model, a single data cleansing tool, and so on. But, on the other hand, the practice of IT (and machine learning systems are, of course, IT systems) suggests that it is necessary to reuse models, architectures, etc. For economic success, IT projects cannot always be unique and must reuse some solutions (components of other solutions). For machine learning systems, an example of such reuse is AutoML solutions [36], where solutions are selected (fixed) for all individual elements of the machine learning pipeline. It is the requirements of reuse (in fact, these are economic requirements - the need for rapid implementation) that make it important to develop practical recommendations for reusable implementations of individual elements of the pipeline. And first of all, it is some standard (de facto standard) architectural solutions that are important. For many positions, we have many implementations that differ in quite specific aspects, and the question of choosing a specific implementation is not the most important one. Much more important is the overall solution architecture.

In this paper, we consider one of the elements of the machine learning pipeline - testing machine learning systems. Such testing is obviously different from traditional software testing. First of all, because the nature of the results of a machine learning system is probabilistic in nature, the basis for the results (conclusions) often cannot be not only verified, but simply obtained. The contribution of this article is architectural solutions for the so-called adversarial testing, that is, the search for data examples that cause machine learning systems to work incorrectly. In addition to justifying the architecture itself, we provide examples of the use of specific software products. It should be noted that, as examples, they illustrate the proposed approach, but, of course, are not exclusive.

More generally, the task we are solving is to build a trusted environment for the development of artificial intelligence systems [37]. Such environments, in the end, are sets of products (software systems) that affect different aspects of the machine learning pipeline, designed to increase confidence in the results of the developed machine learning models. These products include, for example, tools for adversarial attacks [38], formal verification of machine learning models [1], data cleansing, and so on. Naturally, testing and adversarial testing are necessarily part of such an environment. To avoid being tied to specific vendors, and to enable the development of critical applications, we consider only systems with open source, which can be modified if necessary. The article discusses the choice of products for testing machine learning systems.

The remainder of the article is structured as follows. Section II describes the actual approach to using counterfactuals. Section III focuses on the use of counterfactuals in machine learning. Section IV compares counterfactual and adversarial examples. Section V contains direct guidance on practical application.

1. On counterfactual examples

The use of counterfactuals has become a hot and popular topic in the machine learning community for many reasons such as explainability, interpretability, checking algorithmic fairness, etc. The general idea is clear enough. If we have a known output of the model, that is, for a given input, we know the output (the result of work) of the model, then we will be interested in the change in this output (result) when the input changes. Under what input data (what changes in input data) will the output change?

So, there are several models (and definitions) for the counterfactuals. For example, in the classification task, an example counterfactual explanation provides the following information: "for an example that belongs to class A, what changes do we need to make to the input so that the output will be classified as B". As per another definition, if we consider A and B to represent events or facts and A precedes B in time in the statement "A and B is true", then the counterfactual statement is 'If A had not occurred, B would not have occurred'. So, a counterfactual analysis can help to find whether A is a cause of B (it is by supposing the non-occurrence of A and seeking for the effect of this assumption on B.

In NLP a counterfactual example is defined as synthetically generated text which is treated differently by a condition model. For example, given the text "This program is written in Python", the counterfactual text becomes "This program is written in Java". If we know how the original sentence was classified, then how the counterfactuals will be classified?

In fact, we are talking about the conclusion that a small change in the data changes the output (result) of the work. This obviously coincides with the definition

of the robustness of a machine learning model. Adversarial examples, in particular, are looking for precisely the minimum possible changes in the initial data that change the operation of the model. Accordingly, at least in theory, counterfactuals can be used to test the robustness of a machine learning model as well as for studying (checking) the fairness and transparency of algorithms. Counterfactual examples as tests should be more understandable (interpretable) as we tie them to some real conclusion. These examples will be created based on the interpretation (explanation) of the real conclusion (the result of the model).

In order to discuss counterfactuals, we have to turn to causality. The ability to understand causal relationships and to reason from them is one of the main human abilities [4]. Understanding physical causal relationships are fundamental to using any tool [5]. For example, people to people management (relations) is based on the understanding of psychological causal relationships. In many works, it is noted that it is convenient for human psychology to explain any conclusions by means of contrasting rather than direct explanations. We can explain, for example, a certain classification by giving reasons why only a certain class is chosen and why others are rejected. In other words, the explanation can be based on the choice and rejection of specific alternatives (results). This "discriminatory" explanation is counterfactual. For a machine learning system, counterfactual examples are input data that changes the result (classification, solution). The academic literature notes that this approach is more in line with emerging regulatory constraints, such as the General Data Protection Regulation (GDPR) [20]. The counterfactual approach helps to establish three important characteristics of the interpretability of models:

- determine how the interpretation of the model was made,
- provides opportunities for correcting unfavorable decisions
- provides hints for obtaining expected results in forecasting

As it is stated in [4], causal understanding thus maintains the kinds of cognition that have been proposed as part of the distinctively human cognitive toolbox. And in both the physical and psychological domains, causal knowledge is linked with sophisticated inferences about the counterfactual past. There are two distinctive features of causal knowledge, which are captured by causal models [4]:

1) causal knowledge supports a distinctive set of inferences involving interventions and counterfactuals. So, causal knowledge supports counterfactual claims;
2) causal knowledge involves not only specific relations between particular causes and effects but coherent networks of causal relations.

And one of the main statements from [4]: "counterfactual and intervention reasoning, and Bayesian learning all involve the same cognitive machinery: the ability to consider events that have not occurred". This is, in fact, a direct reference to the basic idea of machine learning. We train the network on a training dataset with the idea that a generalization will be built that will work correctly on the rest of the data, which, generally speaking, are unknown to us. We want to generalize the learning outcomes for the entire general population. That is, train the network to process data (events) that do not yet exist.

2. The usage of counterfactuals in machine learning

Counterfactual explanations are gaining attention as a way to explain the decisions of a machine learning model. There are several technical ways to generate and evaluate counterfactuals, such as feature-based explanations, prototype explanations, example-based explanations, or causal explanations [6].

We define a feature-highlighting explanation as an explanation that points to specific features in the model that matter to the individual decision. Of course, each type of feature-highlighting explanation may define this "matter" differently. There are two types of feature-highlighting explanations: counterfactual explanations and principal reason explanations. Principal reason explanation is defined in [8] as the reasons defined by law. Or more broadly, we could describe them as reasons based on some predefined set (some vocabulary).

The goal of counterfactual explanations is to explain how things could have been different, as well as provide a set of features changes for reaching a different output of the model in the future. Counterfactual explanations are generated by identifying such features that, if minimally changed, would alter the output of the model. For example, counterfactual explanations are trying to find the "nearest" hypothetical point that is classified differently from the point currently in question [7].

In other words, identifying the set of features results in the desired prediction while remaining at a minimum distance from the original set of features describing the individual [7]. It is illustrated in Fig. 1.

Suppose we are going to present counterfactual explanations for classification models, which are functions mapping input feature vectors $x \in X$ into label $c \in \{C_1, C_2, \ldots C_n\}$. Actually, the most of research

papers in this area have applied counterfactual explanations to classification tasks. Given a classification model [7]

$$f : X \rightarrow \{ C_1, C_2, \ldots C_n \},$$

we can define the set of counterfactual explanations for a (factual) input $\hat{x} \in X$ as $CF_f(\hat{x}) = \{x \in X \mid f(x) \neq f(\hat{x})\}$. In other words, $CF_f(\hat{x})$ contains all the inputs x for which the model f returns a classification different from $f(\hat{x})$.

For prediction models, this is defined similarly, only the mapping will be carried out into the set {0, 1}

Based on the above-defined counterfactual space $CFf(\hat{x})$, we would like to produce counterfactual explanations for the output of a model f on a given input by trying to find a nearest counterfactual, which is defined as: $\hat{x}^* \in argmin\ d(x, \hat{x})\ for\ x \in CF_f(\hat{x})$

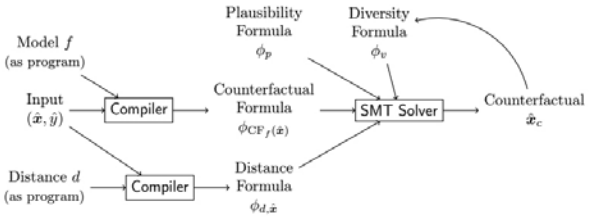

Fig. 1. Architecture Overview for Model-Agnostic Counterfactual Explanations (MACE) [7]

Prototype-based counterfactual explanations are discussed in the paper [9]. Since we cannot interpret the black box, our first task is to create an interpreted view for it. This is exactly what prototyping-based methods do. The work of the black box is being prototyped. As a first step, we need to find a representative dataset. The maximum mean discrepancy (a distance on the space of probability measures) is used to calculate the representativeness. After that prototype-based explanations provide the nearest prototype as explanations for a given test instance [10]. In some papers, a similar approach is called an example-based explanation: example-based approaches seek to find data points in the vicinity of the explainee data point. They either offer explanations in the form of data points that have the same prediction as to the explainee data point or the data points whose prediction is different from the explainee datapoint [11]. In other words, example-based approaches are another kind of explainability technique used to explain a particular outcome.

In general, the explainability problem for machine learning systems can be presented as model explanation or outcome explanation problems. As per definition, a model explanation is about an interpretable and transparent explanation of the original model.

As the developed techniques for neural networks explanations, we could mention decision trees [12, 13] and rule sets [14, 15]. As per software tools, there are some model-agnostic packages. For example - Partition Aware Local Model (PALM) [16]. PALM allows you to study the structure of model conclusions using its approximation by surrogate models. This is to some extent a general approach, also called partial modeling in the literature. In such a model, we are trying to approximate the general black box with "understandable" models. This, in particular, should help in debugging models. PALM approximates a neural network using a two-part surrogate model, which includes a meta-model that partitions the training data, and a set of sub-models that approximate the patterns (solutions) within each partition. The paper [17] describes an approximation algorithm, GoldenEye, to select sets of attributes that influence the work of the classifier. In fact, this is combinatorics, when the possible combinations are sorted out.

Outcome explanation needs to provide an explanation for just a specific prediction from the model. This type of explanation does not affect the internal logic of the models, but only deals with inferences. There are model-specific approaches like Grad-CAM [18] and model agnostic approaches like LIME [19] have been proposed. All of them are provided either feature attribution or model simplification methods.

3. Counterfactual and adversarial examples

In general, the performance (the predictive of classification performance) for any machine learning model is based on the assumption of a statistical similarity of the distributions of training and production (testing) data.

Note that in the general case, the general set of data is unknown to us. Accordingly, it is unknown not only how the training and test data correlate with each other, but also how the test and training data separately correlate with the general population.

In the classic example [22], we have (an actually unknown) some sine curve, the test and training data for which just happened to be on different crests (Fig. 2). Both the training and test data are perfectly (very accurately) approximated by some straight line, but these are completely different straight lines (different angles of inclination).

And in the general case, we cannot assume similar distributions. The only way to somehow guarantee this, obviously, involves exhaustive knowledge of the population. In some tasks, this is really possible, but this is usually the point that is not discussed in the works on machine learning, although, of course, it deserves separate consideration.

Since, in fact, when creating a model, we only know the training data and, accordingly, their statistical characteristics, studying the operation of the model on data whose distribution differs from the distribution of the training data looks like a natural process. As noted in [23], evaluation of out-of-distribution data is a common practice in NLP and image proceedings.

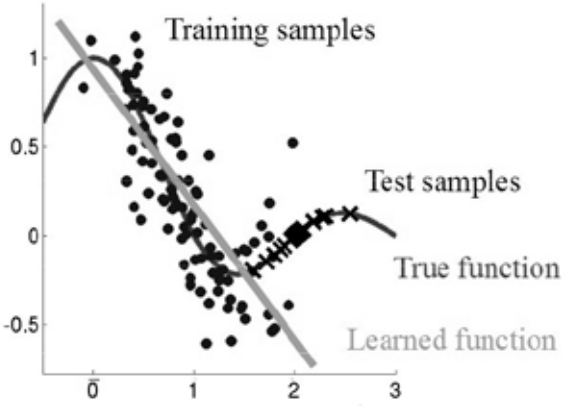

Fig. 2. Training and test samples [22]

And the reason for this is the different distributions of test and training data (Fig.3).

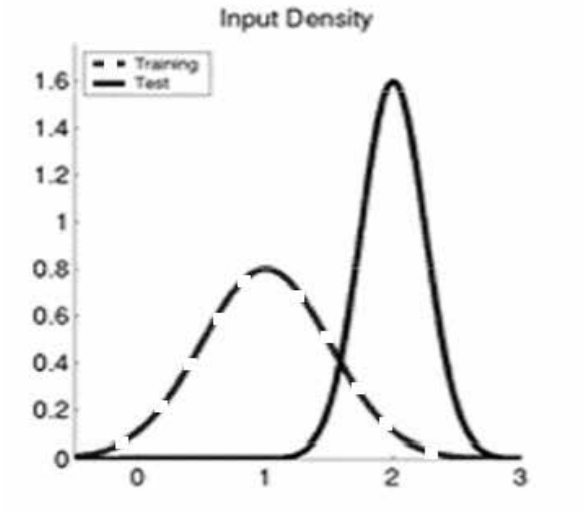

Fig. 3. Distribution shift [22]

One fact may be noted here. Model errors (poor generalization) are often associated precisely with what is called network overtraining [24]. This happens when the models rely on some training dataset-specific biases and artifacts rather than intrinsic properties of the data. When these biases do not exist in the production data, the performance of the models can drop dramatically. The keywords here are "intrinsic properties". And the way to identify them is just counterfactual examples. We change the data, the solution of the system is reversed, which allows us to assume that the changed (deleted) data are the main characteristics based on which the machine learning model makes a decision.

In ideology, this is similar to competitive examples, but their search is carried out not through a sequential selection of modifications that change the solution, but by one-time changes that change the "meaning" of the image. The emphasis is on pairs: the image and its counterfactual example(s).

In this connection, we could cite works that discussed generalization from a causal perspective [25, 26]. To provide generalization, the model must reflect the real causal mechanisms behind the data.

One practical example that can be mentioned (and actually used) in this situation.

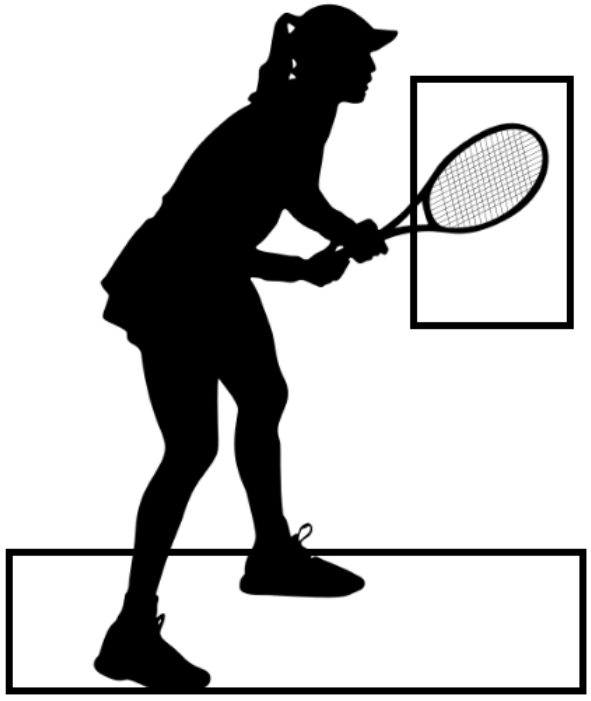

Fig.4. Attention areas [27]

Using the well-known approach for evaluating areas of the image that attract attention [27], one can try to build counterfactual examples by removing just these areas. In this work, this was illustrated by a tennis player, where attention was drawn to the racket and the surface (Fig. 4). Strictly speaking, it was on these parts of the image that a person evaluated the image. In our case, this was used on a traffic sign recognition system (Fig. 5), and images (Fig. 6)

The counterfactual examples constructed in this way were used for adversarial training and increasing robustness.

Fig.5. Road sign: original (left) and counterfactual (right)

Fig. 6. Images: original (left) and counterfactual (right)

Summarizing these results, we can say that counterfactual examples are adversarial examples built with image semantics in mind. If we talk about the classical form of constructing adversarial examples, then we operate with pixels in the framework of L-norms, and not with image fragments. Of course, we should be talking about data in general here, but in practice, we are talking about images (as in the vast majority of works related to sustainable machine learning) and, as discussed in the next section, texts.

Note that the analysis of the content of images corresponds to the ideas outlined in the pioneering work [28], where the authors propose a new model for deep learning based precisely on enlarged fragments. The current state of research suggests that, within the framework of per-pixel processing, the robustness of machine learning systems cannot be ensured.

4. On practical examples

Which in the end can be used to prepare counterfactual examples in practical applications?

A. Counterfactual examples for texts

Counterfactual examples for text represent the simplest and most clearly interpretable model. Several approaches to generating such examples are described in the literature, in a simple form they can be presented:

If we have a sentence in the “Who did what” format, then building a counterfactual example is, in fact, a well-known exercise from foreign language textbooks - “make a new sentence that denies the original statement (“No one did it” format)

For example, such negation formats are given in [29] for the source text “I am very disappointed with the service”:

There is practically no difference between the approaches, since technically they do pretty much the same thing. As an example of software (a toolkit that can be used in your own projects), you can cite, for example, Checklist [30].

It is a system built on the basis of templates. Here is a typical example (source data https://github.com/marcotcr/checklist)

```
import checklist
from checklist.editor import Editor
import numpy as np
editor = Editor()
ret = editor.template('{first_name} is {a:profession} from {country}.',
profession=['lawyer', 'doctor', 'accountant'])
np.random.choice(ret.data, 3)
```

and here is the result:

```
['Mary is a doctor from Afghanistan.',
'Jordan is an accountant from Indonesia.',
'Kayla is a lawyer from Sierra Leone.']
```

To generate the Checklist uses the set of predefined templates, lexicons, generic perturbations, and context-sensitive sentences. The main limitation of these pattern-based or rule-based approaches is that they cannot generate meaningful diversity [29].

Token-based Substitution in Table 1 uses either single word replacements or some templates to generate multiple test cases. Adversarial examples, as usual, do not evaluate the text at all (do not appreciate the meaning).

Other approaches are used, for example, to generate text GPT-2, BERT, or bag-of-words models [31]. In general, we can characterize this direction as quite developed from a practical point of view, with ready-to-use tools.

B. Counterfactual examples for images

Originally, Search for EviDence Counterfactual (SEDC) is the model-agnostic search algorithm

Table 1

TEXT COUNTEFACTUALS

Input Sentence Token-based Substitution	Adversarial attack	Controlled Counterfactual Generation
I am very pleased with the service. I am very happy with the service.	I am very impressed with the service. I am very witty with the service.	I am very happy with this service. I am very pleased with the service.

(SEDC) to find counterfactual explanations for document classifications. According to this algorithm, the explanation can be considered as an irreducible set of characteristics (for example, for text documents – words), which, in their absence, would change the classification of the document.

The modification presented in [32] as a set of characteristics uses segments into which the original image is divided. As noted above, for images, again, not pixel processing is used, but manipulations with significant elements of the image. Accordingly, the explanation for an image is an irreducible number of segments, the removal of which will change the classification of the image.

As per [32], consider an image I assigned to class c by a classifier C the objective is to find a counterfactual explanation E as an irreducible set of segments that leads to another classification after removal. Formally:

$E \subseteq I$ (segments in image)

$C(I \setminus E) \neq c$ (class change)

$\forall\ E' \subset E : C(I \setminus E') =$ c (irreducible)

The counterfactual explanation, in this case, is the classification of the image that is obtained after removing the segments.

The original work thus defined the "minimal" image, which was still classified as an "airplane" (fuselage without wings). Approaches similar to this have been illustrated above.

In [33], with the telling title "Explanations based on the missing", this is described as pertinent positive (PP) and pertinent negative (PN). The PP is a factor that is minimally required for the justification of the final decision and the PN is a factor whose absence is minimally required for justifying the decision.

The basic implementation of SEDC is represented by the resource [34]. In the basic case, the image is segmented, and then the segments are removed one by one until the classification of the remaining "image" changes. In a modified version of SEDC-T [32], segments are also removed one at a time, until the classification reaches the specified value. In other words, SEDC-T gives a more detailed explanation of why the image is not predicted as the correct class (removing which segments leads to a given misclassification).

As for the actual segmentation of images, many approaches can be used here, in addition to the attention map presented above. In fact, you can use a simple grid to divide the image into segments or use more complex approaches that are widely presented in open implementations [35]. The obvious advantages of semantic segmentation are the possible explainability of the results and the ability to use the results for physical attacks (for example, to hide part of the image with a patch). We also note that semantic image segmentation is also present in popular packages for machine learning, such as Keras and Tensorflow [39].

C. Counterfactual examples for sounds

Counterfactual examples for sound classification present a more exotic challenge. In practice, we can only give an example from [21]. Here, the validity of automatic speech recognition (ASR) models was studied. The same text was recorded in different voices (for different ethnic groups, different sexes, and ages). At the same time, the work of the recognition system should not be disturbed. The paper practically shows that the widely used automatic speech recognition systems are unfair, since some groups of users had a higher error rate than others. One way to define fairness in ASR is to require that changing any person's demographic group (for example, changing their gender, age, education, or race) does not change the probability distribution between the possible speech-to-text transformations. In the counterfactual justice paradigm, all variables that do not depend on group membership (for example, the text read by the speaker) remain unchanged, while variables that depend on group membership (for example, the speaker's voice) change counterfactually. Therefore, one can attempt to achieve a fair ASR performance by teaching the ASR to minimize the change in the probabilities of recognition outcomes despite the counterfactual change in human demographics.

Conclusion

In this article, we focused on generating adversarial tests for machine learning systems. As we noted in previous works, testing machine learning systems is robustness testing.

Traditional methods, considered as an optimization problem of finding the smallest modifications that change the results of the classification, give, in the end, very limited results in terms of increasing stability. And, most importantly, the proposed modifications are completely artificial by their nature, in no way connected with possible attacks. In this regard, in this paper, we justified the use of counterfactual examples for generating tests, since they are related to the semantical analysis of data.

The purpose of this paper was to present practical reusable solutions for generating counterfactual examples for various types of input data. The result of our research, based on the practical use of various products, is the presentation of a pipeline for constructing counterfactual examples in image recognition and text classification problems.

Creating counterfactual examples for text classification is currently a purely technical task. The

question is only in choosing the most convenient software implementations. The algorithms are quite transparent and can be built into your own applications. We propose to use template-based systems, like the above-mentioned Checklist.

In terms of building counterfactual examples for images, the best choice, in our opinion, is the semantic segmentation of images. We propose to use the open source implementation of SEDC-T. Alternative methods to some extent reproduce approaches to constructing adversarial examples and are based on a formal assessment of the change in the quality of the system when modifying images.

Counterfactual examples for sound classification (important, for example, for biometric identification systems) are the least developed area. To date, we cannot offer practical solutions in this direction. One reason for this is the nature of existing classification systems, which rely on various artificially created characteristics. For example, wavelet transforms, etc. With their use, the reverse transition to modifications of the original sound characteristics becomes unclear.

We are grateful to the staff of the Department of Information Security of the Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University for valuable discussions of this work.

References

1. *Namiot, Dmitry, Eugene Ilyushin, and Ivan Chizhov*. "On a formal verification of machine learning systems." International Journal of Open Information Technologies 10.5 (2022): 30-34.
2. *Li, Huayu, and Dmitry Namiot.* "A Survey of Adversarial Attacks and Defenses for image data on Deep Learning." International Journal of Open Information Technologies 10.5 (2022): 9-16.
3. Artificial Intelligence in Cybersecurity. http://master.cmc.msu.ru/?q=ru/node/3496 (in Russian) Retrieved: May, 2022
4. *Buchsbaum, Daphna, et al.* "The power of possibility: Causal learning, counterfactual reasoning, and pretend play." Philosophical Transactions of the Royal Society B: Biological Sciences 367.1599 (2012): 2202-2212.
5. *Sterelny, Kim.* "Language, gesture, skill: the co-evolutionary foundations of language." Philosophical Transactions of the Royal Society B: Biological Sciences 367.1599 (2012): 2141-2151.
6. *Kasirzadeh, Atoosa and Andrew Smart.* "The use and misuse of counterfactuals in ethical machine learning." Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. 2021.
7. *Amir-Hossein Karimi, Gilles Barthe, Borja Belle, and Isabel Valera.* 2019. Model-Agnostic Counterfactual Explanations for Consequential Decisions. arXiv preprint arXiv:1905.11190 (2019)
8. *Barocas, Solon, Andrew D. Selbst, and Manish Raghavan.* "The hidden assumptions behind counterfactual explanations and principal reasons." Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. 2020.
9. *Duong, Tri Dung, Qian Li, and Guandong Xu.* "Prototype-based Counterfactual Explanation for Causal Classification." arXiv preprint arXiv:2105.00703 (2021).
10. *Yadav, Chhavi, and Kamalika Chaudhuri.* "Behavior of k-NN as an Instance-Based Explanation Method." arXiv preprint arXiv:2109.06999 (2021).
11. *Verma, Sahil, John Dickerson, and Keegan Hines.* "Counterfactual explanations for machine learning: A review." arXiv preprint arXiv:2010.10596 (2020).
12. *Thiagarajan, Jayaraman J., et al.* "Treeview: Peeking into deep neural networks via feature-space partitioning." arXiv preprint arXiv:1611.07429 (2016).
13. *Boz, Olcay.* "Extracting decision trees from trained neural networks." Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. 2002.
14. *Santos, Raul T., Júlio C. Nievola, and Alex A. Freitas*. "Extracting comprehensible rules from neural networks via genetic algorithms." 2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks. Proceedings of the First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (Cat. No. 00. IEEE, 2000.
15. *Andrews, Robert, Joachim Diederich, and Alan B. Tickle.* "Survey and critique of techniques for extracting rules from trained artificial neural networks." Knowledge-based systems 8.6 (1995): 373-389.
16. *Krishnan, Sanjay, and Eugene Wu.* "Palm: Machine learning explanations for iterative debugging." Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics. 2017.
17. *Henelius, Andreas, et al.* "A peek into the black box: exploring classifiers by randomization." Data mining and knowledge discovery 5 (2014): 1503-1529.

18. *Selvaraju, Ramprasaath R., et al.* "Grad-cam: Visual explanations from deep networks via gradient-based localization." Proceedings of the IEEE international conference on computer vision. 2017.
19. *Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin.* "Model-agnostic interpretability of machine learning." arXiv preprint arXiv:1606.05386 (2016).
20. *Gohel, Prashant, Priyanka Singh, and Manoranjan Mohanty.* "Explainable AI: current status and future directions." arXiv preprint arXiv:2107.07045 (2021).
21. *Sari, Leda, Mark Hasegawa-Johnson, and Chang D. Yoo.* "Counterfactually Fair Automatic Speech Recognition." IEEE/ACM Transactions on Audio, Speech, and Language Processing (2021).
22. Francisco Herrera Dataset Shift in Classification: Approaches and Problems http://iwann.ugr.es/2011/pdf/InvitedTalk-FHerrera-IWANN11.pdf Retrieved: Sep, 2021
23. *Teney, Damien, Ehsan Abbasnedjad, and Anton van den Hengel.* "Learning what makes a difference from counterfactual examples and gradient supervision." Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16. Springer International Publishing, 2020.
24. *Roelofs, Rebecca, et al.* "A meta-analysis of overfitting in machine learning." Proceedings of the 33rd International Conference on Neural Information Processing Systems. 2019.
25. *Heinze-Deml, Christina, and Nicolai Meinshausen.* "Conditional variance penalties and domain shift robustness." arXiv preprint arXiv:1710.11469 (2017).
26. *Meinshausen, Nicolai.* "Causality from a distributional robustness point of view." 2018 IEEE Data Science Workshop (DSW). IEEE, 2018.
27. *Das, Abhishek, et al.* "Human attention in visual question answering: Do humans and deep networks look at the same regions?." Computer Vision and Image Understanding 163 (2017): 90-100.
28. *Bengio, Yoshua, Yann Lecun, and Geoffrey Hinton.* "Deep learning for AI." Communications of the ACM 64.7 (2021): 58-65.
29. *Madaan, Nishtha, et al.* "Generate your counterfactuals: Towards controlled counterfactual generation for text." arXiv preprint arXiv:2012.04698 (2020).
30. *Ribeiro, M.T., Wu, T., Guestrin, C. and Singh, S.* 2020. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. arXiv preprint arXiv:2005.04118 .
31. *Dathathri, Sumanth, et al.* "Plug and play language models: A simple approach to controlled text generation." arXiv preprint arXiv:1912.02164 (2019).
32. *Vermeire, Tom, and David Martens.* "Explainable image classification with evidence counterfactual." arXiv preprint arXiv:2004.07511 (2020).
33. *Dhurandhar, Amit, et al.* "Explanations based on the missing: Towards contrastive explanations with pertinent negatives." arXiv preprint arXiv:1802.07623 (2018).
34. SEDC implementation https://github.com/yramon/edc Retrieved: May, 2022
35. *Van der Walt, Stefan, et al.* "scikit-image: image processing in Python." PeerJ 2 (2014): e453.
36. *He, Xin, Kaiyong Zhao, and Xiaowen Chu.* "AutoML: A survey of the state-of-the-art." Knowledge-Based Systems 212 (2021): 106622.
37. *Namiot, Dmitry, Eugene Ilyushin, and Oleg Pilipenko.* "On Trusted AI Platforms." International Journal of Open Information Technologies 10.7 (2022): 119-127. (in Russian)
38. *Ilyushin, Eugene, Dmitry Namiot, and Ivan Chizhov.* "Attacks on machine learning systems-common problems and methods." International Journal of Open Information Technologies 10.3 (2022): 17-22. (in Russian)
39. *Dadhich, Abhinav*. Practical Computer Vision: Extract Insightful Information from Images Using TensorFlow, Keras, and OpenCV. Packt Publishing Ltd, 2018.

D.E. Namiot. Dr. of Sci., Lomonosov Moscow State University, MSU, Faculty of Computational Mathematics and Cybernetics, Russia, 119991, Moscow, GSP-1, 1-52, Leninskiye Gory, dnamiot@gmail.com (correspondent author)

E.A. Ilyushin. MSU, Faculty of Computational Mathematics and Cybernetics, Russia, 119991, Moscow, GSP-1, 1-52, Leninskiye Gory, john.ilyushin@gmail.com

I.V. Chizov. PhD, docent, Lomonosov Moscow State University; Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, MSU, Faculty of Computational Mathematics and Cybernetics, Russia, 119991, Moscow, GSP-1, 1-52, Leninskiye Gory, ichizhov@cs.msu.ru