How to build trust in AI systems with misclassification detectors and local misclassification explorations

Pål Vegard Bun Johnsen[1,*], Milan De Cauwer[1], Joel Bjervig[1] and Brian Elvesæter[1]

[1]SINTEF Digital, Forskningsveien 1, 0373 Oslo, Norway

Abstract

In the context of an AI system consisting of a machine learning model for classification, we present a framework denoted SafetyCage for systematically detecting and explaining misclassifications. We show how the framework can be used under deployment of the AI systems when true labels are unknown. Specifically, a misclassification detector measures the reliability in one particular model prediction and flags the prediction as either trustworthy or not. Unfortunately, most existing misclassification detectors are not easily interpretable for the purpose of finding the root cause of a misclassification. Hence, if the prediction is deemed untrustworthy, our approach provides additional so-called local misclassification explorations to further assess the trustworthiness of the prediction. The purpose of the framework is to be able to systematically explore the root cause of a particular misclassification, and hence incentivizing procedures to enhance the AI system even further. We showcase our framework with three ML models of different model architectures trained on images, tabular data and text respectively, and present three generic suggestions of local misclassification explorations, and how they can be adapted for each use case.

Keywords

AI systems, Trustworthy AI, Machine Learning

1. Introduction

The integration and use of AI systems has been influential in modern society. The widespread availability of so-called large language models (LLMs) is a good example of this. Another example is within the health sector where for instance radiologists can get assistance from an AI model to detect bone fractures based on X-ray images of patients [1]. As defined in the memorandum published by the Organisation for Economic Co-operation and Development (OECD) in 2024 [2]: "*An AI system is a machine-based system that, for explicit or implicit objectives, infers, from the input it receives, how to generate outputs such as predictions, content, recommendations, or decisions that can influence physical or virtual environments. Different AI systems vary in their levels of autonomy and adaptiveness after deployment*". The backbone of any AI system is the underlying AI model that provides the predictions. Most modern AI models are so-called machine learning (ML) models — a subset of AI models that are trained on historical data. This type of AI model has gained particular traction due to its superior performance within imaging, text and speech tasks. However, this often comes at the cost of reduced *explainability* - the degree to which one can understand the basis for a model's predictions.

Following the definition above, an AI system is not limited to providing predictions, but may also provide recommendations for decisions or even make decisions on behalf of a human. We emphasize the difference between the raw predictions, and the final recommendations and decisions that will in some way make use of the raw predictions in a decision-making process. Under the EU AI Act [3] which entered into force on 1 August 2024, high-risk AI systems are according to Article 15 required among other things to be resilient to errors that occur within the system. Moreover, under Article 14 natural persons must be able to effectively oversee the high-risk AI system during use. To ensure human oversight and control of the AI system, and to avoid negative consequences, it is important that the decision maker is efficiently able to assess the trustworthiness of a particular prediction. As

TRUST-AI: The European Workshop on Trustworthy AI. Organized as part of the European Conference of Artificial Intelligence - ECAI 2025. October 2025, Bologna, Italy.

*Corresponding author.

pal.johnsen@sintef.no (P. V. B. Johnsen)

described in [4], several aspects must be considered when assessing the trustworthiness of an AI system. In this work, we will focus on the aspects of accuracy and human oversight, and the extent to which we can assess and explore whether a particular prediction from the AI system is correct. We present the framework SafetyCage for detecting and exploring incorrect predictions using misclassification detectors. The advantage of having a misclassification detector together with the ML classification model is that the accuracy of *each* prediction can be assessed in contrast to an overall model-based accuracy assessment. In an AI system, this is a useful tool when human intervention is relevant or required, for instance for quality assurance, spot checks or when a particular prediction is suspicious. If both the ML model and the misclassification detector are sufficiently accurate, it can be relevant to use it actively such as by presenting the output from the misclassification detector together with the particular ML model prediction. In this way, the ideally small proportion of suspicious predictions flagged by the misclassification detector can be inspected in detail by a human. To build trust in the detection procedure, the framework includes analysis that help explore *why* a particular prediction is correct or not. We limit ourselves to classification models where the task is to predict whether an input sample (such as an X-ray image) belongs to a certain class (such as representing the presence of a bone fracture).

The remainder of the paper is structured as follows. In Section 2 we include related work, and present challenges of using misclassification detectors in AI systems, and how we address them in this work. In Section 3, we present our SafetyCage framework for misclassification detection and exploration. Section 4 demonstrates the application of the framework in three different use cases involving ML models for images, tabular data and text. In Section 5 we discuss the results and limitations of the framework. Finally, Section 6 concludes the paper and presents directions for future work.

2. Related work - Misclassification detectors

Given a *pre-trained* (already trained) ML model. With an input sample $\mathbf{x}_i$, the ML model will output a prediction $\hat{y}_i$, $\hat{f}_\theta : \mathbf{x} \mapsto y$, satisfying $\hat{y}_i = \hat{f}_\theta(\mathbf{x}_i)$. The ML model is *trained* by fitting the parameters θ such as to mimic the true target function $f^* : \mathbf{x} \mapsto y$ satisfying $y_i = f^*(\mathbf{x}_i)$. This branch of machine learning is called *supervised learning* where the ML model is trained based on true labels $y_1, \ldots, y_N$ for every input $\mathbf{x}_1, \ldots, \mathbf{x}_{N_{ML}}$ with training size N_{ML}, denoted the *training data* $\mathcal{D}_{ML} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{ML}}$. In classification problems the labels are distinct and disjoint *categories* or *classes*. We assume only one class can be present for each input sample. In this study, we will deal with ML models where the output is a vector where each element, representing class j, is a value that predicts the probability that class j is present, and where the final prediction is equal to the class with the largest probability. We will further only explore ML models where the *softmax activation* function is applied to the output layer which ensures that all elements sum to one, following the law of total probability.

The goal of a *misclassification detector* (MD) is in general to detect whenever the ML model prediction is wrong for a given input $\mathbf{x}_i$ and prediction $\hat{y}_i$ for which $\hat{y}_i \neq y_i$. We define the MD to be a function $\mathcal{D}(\mathbf{x}_i, \hat{y}_i) \mapsto \mathbb{B}$ which given $\mathbf{x}_i$ and $\hat{y}_i$ outputs a boolean value where the output 1 means that the MD predicts the ML model prediction to be wrong, while the output 0 means the MD model prediction is correct. Ultimately, the MD is a binary classifier. When the output is 1 we say that the MD model *flags* the prediction as being wrong. Note that this problem is only relevant during so-called *deployment* of the ML model, which means whenever the ML model makes a prediction, but when we still don't know what the true label y_i is. The MD is itself a prediction model, which can flag wrongly, and hence we will throughout the paper refer to an *MD model*.

At the time of writing, there exist several misclassification detection models that have been published. What can be considered the baseline methods are the maximum softmax probability (MSP) method [5] and the DOCTOR method [6] respectively which can be applied to every ML model with a softmax output layer. For the MSP method, a prediction is considered incorrect whenever the maximum softmax value is less than a predefined threshold. For the DOCTOR method, a prediction is considered incorrect when the estimated odds of a misclassification is larger than a predefined threshold. Another MD

model called RED is described in [7] where the the uncertainty in the maximum softmax value is estimated using Gaussian processes. There also exist detectors that are made for particular model architectures, such as in [8] and [9] where the ML model is a feedforward neural network model. In these cases, the presence of an incorrect prediction is inferred by using the empirical distribution function of correctly and incorrectly predictions on historical data across different layers of the neural network, and a hypothesis test is then constructed to infer the likelihood that the prediction is wrong. All of the mentioned methods share the property that, for each input $\mathbf{x}_i$ and prediction $\hat{y}_i$, the raw output of any detector is a quantitative measure of the *reliability* or *trustworthiness* of the prediction. For instance, under the MSP method, the larger the maximum softmax value for a prediction, the more trustworthy the prediction is considered.

2.1. The challenge of using MD models in AI systems

This raw output from the misclassification detectors is in itself informative, and can ultimately be used to declare a prediction as trustworthy or not. In practice, this will require a *threshold* with respect to the quantitative reliability measure, which defines the border between correct and wrong predictions. However, in [5], the performance of the MSP method is evaluated using the AUROC and AUPR metrics. These metrics aggregate the model performance across several thresholds, and they are therefore threshold-independent performance metrics. While these are informative metrics to quantify general performance, they are not helpful for flagging a particular prediction as trustworthy or not. In [6], an MD model is evaluated by computing the proportion of correct classifications that are flagged by the MD model (Type I error) given that the proportion of all misclassifications that are flagged by the MD model (recall) is required to be greater than 95%. This requirement automatically determines the threshold. However, this evaluation metric does not account for other important performance metrics, such as the proportion of flags that are correct (precision), or the proportion of non-flagged samples that are correctly classified (negative predictive value). Importantly, for an MD model to be practical in an AI system, only a small proportion of flagged instances should be false alarms. Hence, the precision should be sufficiently high. For a threshold-based MD approach, this raises the question of how one should evaluate an MD model in a way that accounts for all aspects of MD model performance, and how one should estimate the threshold.

What the MD models mentioned above also have in common is that while their mutual goal is to capture incorrect predictions, they do not provide any explanations as to *why* the prediction is incorrect. This lack of interpretability is problematic, and limits the trustworthiness of the MD model itself.

Our contribution in this work is the following:

1. We propose to use the Matthews correlation coefficient (MCC) as a suitable evaluation metric for threshold-based MD models
2. We propose a way to estimate the threshold which decides when to declare a prediction as trustworthy or not
3. We introduce the concept of *local misclassification explorations* with the aim of exploring predictions flagged by an MD model in order to monitor an AI system and to validate the flagging by the MD model

3. Methodological approach and framework

In this section our proposition points 1., 2. and 3. described in Section 2.1 will be described in detail in Section 3.1, 3.2 and 3.3-3.4 respectively.

3.1. Evaluation of misclassification detectors

As noted earlier, misclassification detectors (MDs) can themselves produce inaccurate outputs, so their trustworthiness must also be quantified. The underlying ML model in an operative AI system must

be sufficiently accurate. Therefore, it is essential that the MD correctly identifies the small portion of incorrect predictions, while minimizing false alarms on correct ones. This imbalance between correct and incorrect predictions must be accounted for when evaluating MD performance. Moreover, the MD model performance should be evaluated based on all aspects covered by the performance metrics recall, precision, specificity (1-Type I error) and negative predictive value. We therefore recommend the use of the Matthews Correlation Coefficient (MCC), which is specifically designed to provide a reliable performance measure for binary classifiers under class imbalance [10], and which is also constructed such that a large MCC value is only possible if also the recall, precision, specificity (1-Type I error) and negative predictive value are large. The MCC values range from -1 to 1 where -1 means the classifier is perfectly wrong in all cases, 0 means the classifier performs as well as a coin tossing classifier, while 1 means the classifier predicts correctly in all cases. Another favourable feature is that $\text{MCC} = 0$ for classifiers that predict only one class in every case, whereas the classification accuracy for instance would be equal to 0.99 in the setup where the majority class covers 99 % of all samples. It is also therefore we follow the same procedure as presented by [9] where the optimal threshold during training on an MD model is estimated based on data such as to maximize the Matthews correlation coefficient (MCC).

3.2. Misclassification detectors and how to construct them

To avoid optimism bias due to potential model overfitting, the threshold should be estimated on data independent of the data used to train the ML model [11]. For methods such as in [8] and [9], parameter fitting is required as well, and so this should also be fitted under data independent of the training data. We denote this as $\mathcal{D}_{MD} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{MD}}$ where $\mathcal{D}_{ML} \cap \mathcal{D}_{MD} = \varnothing$. When $\mathcal{D}_{MD}$ is applied on the ML model, we can aqcuire the training data for the MD model which we denote $\mathcal{D}^*_{MD} = \{(\mathbf{x}_i, z_i)\}_{i=1}^{N_{MD}}$ with $z_i = \mathbb{1}(\hat{y}_i = y_i)$. Summarized, the construction of a misclassification detector can be visualized in Figure 1.

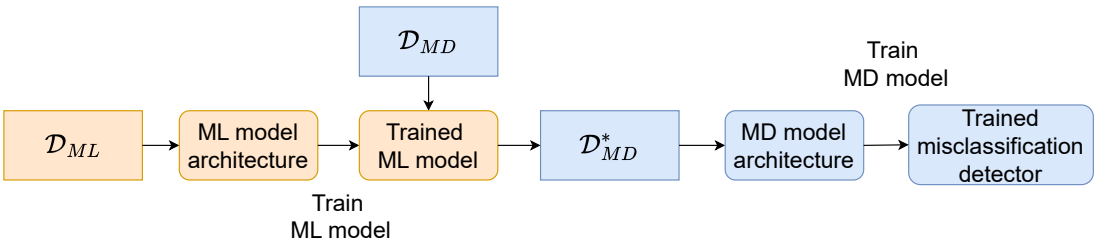


Figure 1: Methodological overview of the construction of a misclassification detector given a pre-trained ML model and disjoint datasets $\mathcal{D}_{ML}$ and $\mathcal{D}_{MD}$.

Once an MD model is trained we can use it during deployment of the ML model to flag potential misclassifications, see Figure 2 where we see the MD model takes as input the input sample, $\mathbf{x}_i$, the ML model prediction, $\hat{y}_i$ as well as information about the ML model architecture if needed.

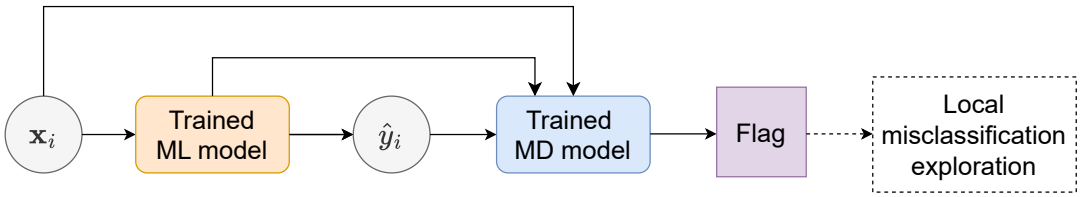


Figure 2: Deployment phase of ML model when MD model can be utilized to flag potential misclassifications. Essential is the ability to further explore the input sample when the MD model flags the predictions as wrong.

3.3. The SafetyCage framework for monitoring AI systems during deployment

The performance of an ML model can be evaluated based on *historical data* where we know the incidents where the predictions were correct or wrong. This can be achieved by computing accuracy metrics such as *classification accuracy* or the *F1 score*, and the performance can be visualized for instance with a *confusion matrix*. These metrics can also be calculated across different subsets of the data that share similar characteristics, helping to identify specific conditions under which the model performs better or worse. These results can give an overall impression of when the ML model fails, but to a limited degree the reason why.

Recent work has emphasized the value of creating *local explanations* — that is, explanations for individual predictions. This includes identifying which input variables the ML model found most important for a given prediction [12]. Such explanations are valuable because they incentivize questions such as "Should this variable in the input sample be this important?" or "Is it reasonable that such a small change in the input sample flips the prediction?". If the answer to any of these questions is "no", it automatically lowers the confidence in the prediction. However, less effort has been put to locally explore misclassifications. In this particular case, the inspection may lead to hypotheses as to why something went wrong in particular cases, and altogether this insight may lead to concrete actions for enhancing the ML model when retrained. During *deployment* of the AI system, hence under decision-making, we do not have true labels and the decision-maker must predict the best action to take under available information. In this process, we can only use the single ML model prediction during the decision-making process. To combat the limitation of the framework above, we present a framework for using a misclassification detector during the deployment of an AI system. As the misclassification detector typically does not reveal why a prediction is flagged as being wrong, we additionally provide *local misclassification explorations*. Summarized, the SafetyCage framework is given in Algorithm 1.

Input: Input sample $\mathbf{x}_i$, trained ML model f, misclassification detector $\mathcal{D}$
Output: Trust decision on prediction $\hat{y}_i$

$\hat{y}_i \leftarrow f(\mathbf{x}_i)$; `// Obtain ML model prediction`
flag $\leftarrow \mathcal{D}(\mathbf{x}_i, \hat{y}_i)$; `// Detect if prediction may be incorrect`
if *flag = 1* **then**
  Generate local misclassification explorations;
end
Make trust decision based on flag and local misclassification explorations;

Algorithm 1: Framework for use of misclassification detector together with local misclassification explorations during deployment of an AI system.

Note that the term SafetyCage has been previously used as a particular type of misclassification detection model first introduced in [8] and [9], authored among others by the main author of this workshop paper. The interpretation of SafetyCage is from hereon extended to mean the framework in which a misclassification detector acts within an AI system.

3.4. Generating local misclassification explorations

As the previous section suggests, we are interested in generating local misclassification explorations for a sample $\mathbf{x}_i$ whenever $\mathcal{D}(\mathbf{x}_i, \hat{y}_i) = 1$.

In this section, we present three generic open-ended misclassification exploration methods. We approach this as a hypothesis-generating study rather than a hypothesis-testing one.

- **Exploration 1**: Historically explore *similar* samples with same misclassified prediction
- **Exploration 2**: Evaluate whether the sample is an *outlier*, and in what way

- **Exploration 3**: Perturb the input sample, without changing the true label, and explore the effect it has on the corresponding prediction

Exploration 1. requires the definition of what are similar samples. This will depend from use case to use case. In Section 4.1 we will show a particular example of this. The purpose of **Exploration 2.** is to explore whether the sample deviates significantly from other previous observations in the training dataset, hence being an outlier. This is based on the fact that ML models typically perform poorly in these circumstances. The presence of an outlier may be due to measurement errors in the sample, but it might as well also be due to the limited representativeness of the training data. In Section 4.2 we show a practial example. **Exploration 3.** explores how the ML model behaves if we modify the original input sample without loosing its essence. If the corresponding prediction changes, we automatically loose trust in the original prediction. The challenge is to know how we should perturb the input sample without changing its true label. In Section 4.3 we show one example for how to deal with this.

4. Use cases and experiments

4.1. Image classification with the MNIST database

We demonstrate the approach of generating **Exploration 1.** of a flagged sample to the MNIST dataset [13]. The dataset consists of 70 000 grayscaled images (28*28 pixels) representing digits from 0-9. Given an unseen image of a digit, the classification task is to predict which class that image belongs to.

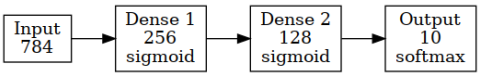
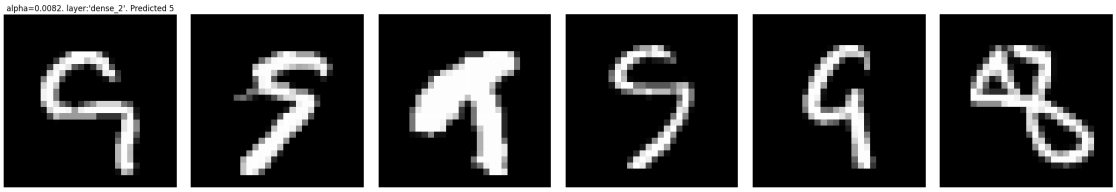


Figure 3: A simple fully connect neural network solving MNIST

We trained a simple feed-forward neural network designed to solve the MNIST classification problem. As illustrated by Figure 3, the model consists of an input layer, two hidden layers with sigmoid activation functions, and an output layer of size 10 with a softmax activation function. This model achieves an overall accuracy of 97% on unseen test data of size 1000.

We trained a SPARDACUS detector on the $\mathcal{D}_{MD}$ data using the Dense 2 layer as a source of information. The training procedure lead to an associated $\mathrm{MCC} = 0.48$ on testing data. Specifically, the trained SafetyCage flagged 2.61% of the 1000 test samples as not trustworthy.



(a) (b) (c) (d) (e) (f)

Figure 4: (a) A sample (correctly) flagged by SPARDACUS with true label 9 and predicted label 5. (b-f) The nearest neighbors to (a) in latent space *dense 2* from the set of training samples that were misclassified as label 5.

Figure 4a shows a sample, wrongly predicted to be the digit 5, from the testing set correctly flagged by the SPARDACUS detector. We show an example of applying **Exploration 1**, namely by looking at previous samples with same prediction, however misclassified, and similar to the input sample (4a). The notion of similarity must be explicitly defined, which is not straightforward to do between raw images. However, with a neural network, one approach is to instead base the similarity between samples on the corresponding representations (activation values) at a particular layer. Specifically, given a new input sample $\mathbf{x}_i$ flagged by SafetyCage , we can search for previous samples, having the same class

prediction and known to be misclassified, that are closest to $\mathbf{x}_i$ with respect to a distance metric in a particular layer. Here, we use the penultimate layer of the neural network model (Dense 2 layer), and deploy the KNN distance metric introduced by [14]. Figures 4b-f, depict the 5 nearest neighbors of the flagged sample under scrutiny. These samples are suggesting to the end-user in what way digit 9 can be misclassified to digit 5.

4.2. Heart disease prediction based on clinical variables

We now discuss another possible approach to explaining misclassified samples. Consider a tabular dataset of 918 rows linking 11 patient features (age, sex, resting blood pressure, ...) to a 0/1 variable stating whether the patient had a cardio-vascular incident. An efficient method to model the binary classification problem $\hat{y}_i = f_\theta(X_i)$ with $\hat{y}_i \in \{0, 1\}$ is to use a decision tree based algorithm such as light gradient boosting machines (LGBM). We relied on the open source implementation of LGBM: LightGBM [15]. Binary and categorical features from the dataset were encoded using an *Ordinal Encoding* strategy. Some model hyperparameters were set explicit values; The maximum depth of trees (*max depth*) was set to 5, the bagging fraction to 0.5 and the number of training rounds to 100. Those hyperparameter that were not set explicitly took their default values. With this setting the ML model achieved an overall accuracy of 0.85 on test data with precision and recall equal to 0.84 and 0.90 respectively.

In this use case, the DOCTOR missclassification detector is used to flag test set samples. The method flagged 30 samples (13%) as potentially wrongly classified, achieving an overall accuracy of 87% at correctly flagging samples and $\mathrm{MCC} = 0.45$ on the testing set.

Figure 5 (Left) shows test set samples as flagged by the detector and ECOD [16] (Empirical Cumulative Outlier Detection), a simple, computationally efficient and explainable outlier detection method. The figure suggests that sample 22 is detected as an outlier in the testing dataset while also having been successfully flagged as wrongly predicted by SafetyCage . The ECOD outlier detection algorithm allows us to measure the feature's individual contribution to the sample's overall outlier score as seen on Figure 5 (Middle). As can be seen on the figure, the outlier detection method suggests that the *age* feature is contributing significantly to the overall score and is also above the 99% cutoff band making the sample very likely to be an outlier.

To conclude **Exploration 2**, we can present the age distribution (Figure 5 (Right)) to the ML model end-user and propose that the *extreme* value of the age can explain why the ML model misclassified the sample.

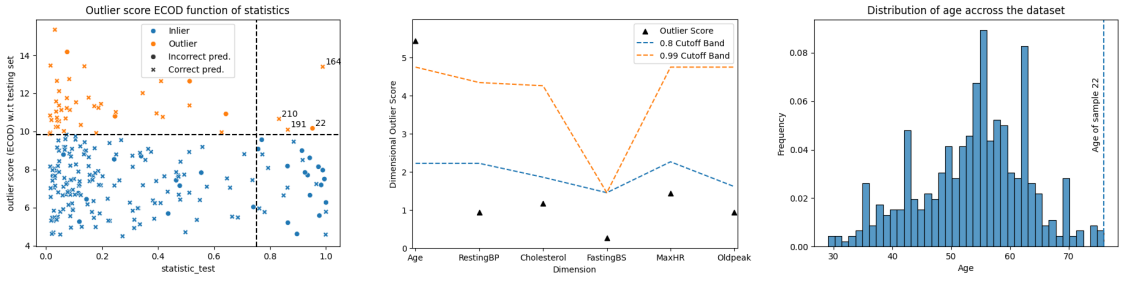


Figure 5: Using an outlier detector method to explain the misclassification of a flagged sample (#22). The label on the x-axis of the leftmost figure, *statistic_test*, is the reliability quantity measured by the MD (DOCTOR). The horizontal and vertical line represent the thresholds for declaring a sample as an outlier and the prediction to be wrong respectively. The upper right window represents samples both declared as an outlier, and where the corresponding prediction is flagged by the MD.

4.3. Review sentiment prediction based on the IMDb Dataset

We apply a pre-trained Roberta transformer model [17] to predict sentiment (either negative or positive) of movie reviews from the IMDb dataset [18]. We collect 1000 samples, and train an MSP misclassification detector on half of it, leaving the remaining 500 as test data. The MSP detector achieves an $\mathrm{MCC} = 0.38$ on the test data. See left column of Table 1 which shows a movie review which was wrongly classified to have a positive sentiment by the Roberta transformer model.

Original Review (shortened)	**Paraphrased Review**
This looks like one of these Australian movies done by "talented" students and funded by the government. It is chock full of smart shots of colors and shapes and verbal excursions into Freudian psychology to be appreciated by art students and teachers alike, but in general it is perceived a stupid mockery of good cinema, good storytelling and generally good taste...	This movie is thought to be a stupid mockery of good cinema because it is full of smart shots of colors and shapes and verbal excursions into Freudian psychology to be appreciated by art students and teachers alike.

Table 1
Side-by-side comparison of an original and paraphrased movie review, using the PEGASUS transformer model, which flips the prediction from the Roberta transformer model from having a positive review to a negative review. The prediction of the original review was wrong, and the MSP misclassification detector flagged it as likely being wrong.

We showcase how **Exploration 3** can be utilized in this case for the movie review in Table 1. We employ the PEGASUS transformer model fine-tuned for *paraphrasing* [19, 20], which rephrases a given input while preserving its original meaning. We generate ten paraphrases of the original review, and compute the sentiment prediction of the Roberta transformer model. See right column of Table 1 to see one such paraphrase generated. We recompute the sentiment predictions by the Roberta model for each paraphrase. It turns out that the prediction flips from positive to negative sentiment for seven out of ten paraphrases. Provided that the paraphrases are of particular quality, this shows the lack of robustness for the sentiment prediction model for this particular input sample, and hence reduces the trustworthiness of the prediction.

5. Discussion

We have presented a system, denoted SafetyCage , for detecting and exploring misclassifications in an AI system for the purpose of general human oversight and finding the root cause of a misclassification. We showed three generic examples. Note the important distinction between exploration and explanation. A next step would be to construct hypotheses of root causes, and further to evaluate them. With this in mind, we regard the quality of a local misclassification exploration as the degree to which one can generate formal hypothesis testing to infer the root cause. Misclassifications where the root cause have been found is valuable information for the future use of an AI system. How to actually materialize this to further improve the AI system is yet another natural step.

The SafetyCage framework may be a compliance enabler for high-risk AI systems with respect to the EU AI Act, for instance with respect to Article 14 and 15 which among other things require that high-risk AI systems are resilient to errors, and that natural persons can efficiently oversee the system during use. The quantification of the prediction uncertainty and the flagging procedure can make the AI system more resilient to the prediction errors, and the local misclassification explorations can contribute to human oversight. As SafetyCage assesses the likelihood of an ML model to produce a prediction error, the framework can be integrated into a risk management system. This is especially relevant given that Article 9 of the EU AI Act requires a risk management system for high-risk AI systems. A potential integration of SafetyCage in a high-risk AI system also creates new obligations regarding the technical documentation (Article 11). This should for instance include detailed information about the flagging procedure of SafetyCage , and its performance quantified with metrics (such as the MCC).

The examples provided in Section 4 serve as illustrative cases intended to showcase the practical potential of the framework, and should not be interpreted as the default behaviour of flagged samples by a misclassification detector. The successfulness of the framework highly depends on the performance of the ML model, the MD model as well as the local explorations. The purpose of this work is to introduce the framework, and how to couple different components together in a bigger picture.

From the use cases in Section 4, we see the MCC on the test data ranges from 0.38 to 0.48 across different MD models. While this means the MD models are informative, there is still room for improvement by catching more misclassifications and reducing false flags, which effectively would increase the MCC towards 1.

A premise for the motivation of local misclassification explorations is the limited interpretability of existing misclassification detectors. One alternative way forward is to construct interpretable misclassification detectors where it is also possible to see the reason why there is a misclassification such as by using linear models or decision trees. However, increased interpretability often comes at the cost of reduced accuracy of the misclassification detector.

6. Conclusion and future work

In this work, we proposed the framework SafetyCage for how to actively use misclassification detectors in AI systems. SafetyCage acts as a guardrail mechanism during decision-making, aiming to reduce the risk of poor decisions due to incorrect predictions from AI classification models. We recommend the use of local misclassification explorations, which can help uncover the underlying causes of erroneous predictions. These insights can support the development of hypothesis-driven tests and guide AI model retraining or modification to improve the overall quality and reliability of the decision-making.

We have identified several directions for future research and development to enhance the SafetyCage framework:

- **Improving the misclassification detectors:** Even though the MDs covered in this work are informative, there is still room for improvement. How to construct MD models that can achieve higher MCC values will be important future research.
- **Improving local explorations:** Enhancing the quality and interpretability of local explorations is a key challenge. In particular, identifying root causes of misclassifications and briding the gap between exploratory insights and actionable explanations is essential.
- **Interpretable misclassification detectors:** By using an interpretable model architecture, one can not only predict the presence of a misclassification, but also be able to interpret the reason why.

Acknowledgments

This work has received funding from the THEMIS 5.0 project under the European Union's Horizon Europe research and innovation programme (grant agreement No 101121042). THEMIS 5.0 aims to develop an AI-driven, human-centered trustworthiness optimization ecosystem that enables users to assess, influence, and enhance the fairness, transparency, and accountability of AI systems.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT for grammar and spelling check, paraphrasing and rewording. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] L. Pettersen-Johansen, A. Faxvaag, Bruk av kunstig intelligens i bildediagnostikk - Hvordan påvirker en kunstig intelligens (KI) applikasjon for tolkning av beinbrudd, arbeidsflyt og oppgaveløsning for helsepersonell i klinisk praksis?, Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2024. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3190984?show=full.

[2] Explanatory memorandum on the updated OECD definition of an AI system, Technical Report, Organisation for Economic Co-Operation and Development (OECD), 2024. doi:10.1787/623da898-en.

[3] Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act), 2024. URL: https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng.

[4] N. Díaz-Rodríguez, J. Del Ser, M. Coeckelbergh, M. López de Prado, E. Herrera-Viedma, F. Herrera, Connecting the dots in trustworthy Artificial Intelligence: From AI principles, ethics, and key requirements to responsible AI systems and regulation, Information Fusion 99 (2023) 101896. doi:10.1016/j.inffus.2023.101896.

[5] D. Hendrycks, K. Gimpel, A baseline for detecting misclassified and out-of-distribution examples in neural networks, 2018. URL: https://arxiv.org/abs/1610.02136. arXiv:1610.02136.

[6] F. Granese, M. Romanelli, D. Gorla, C. Palamidessi, P. Piantanida, Doctor: A simple method for detecting misclassification errors, 2021. URL: https://arxiv.org/abs/2106.02395. arXiv:2106.02395.

[7] X. Qiu, R. Miikkulainen, Detecting Misclassification Errors in Neural Networks with a Gaussian Process Model, 2022. URL: http://arxiv.org/abs/2010.02065. doi:10.48550/arXiv.2010.02065.

[8] P. V. Johnsen, F. Remonato, Safetycage: A misclassification detector for feed-forward neural networks, in: T. Lutchyn, A. Ramírez Rivera, B. Ricaud (Eds.), Proceedings of the 5th Northern Lights Deep Learning Conference (NLDL), volume 233 of *Proceedings of Machine Learning Research*, PMLR, 2024, pp. 113–119. URL: https://proceedings.mlr.press/v233/johnsen24a.html.

[9] P. V. Johnsen, F. Remonato, S. Benedict, A. Ndur-Osei, SPARDACUS SafetyCage: A new misclassification detector, in: Proceedings of the 6th Northern Lights Deep Learning Conference (NLDL), PMLR, 2025, pp. 133–140. URL: https://proceedings.mlr.press/v265/johnsen25a.html.

[10] D. Chicco, G. Jurman, The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation, BMC Genomics 21 (2020) 6. doi:10.1186/s12864-019-6413-7.

[11] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning: data mining, inference and prediction, 2 ed., Springer, 2009.

[12] M. T. Ribeiro, S. Singh, C. Guestrin, "why should i trust you?": Explaining the predictions of any classifier, 2016. URL: https://arxiv.org/abs/1602.04938. arXiv:1602.04938.

[13] L. Deng, The mnist database of handwritten digit images for machine learning research, IEEE Signal Processing Magazine 29 (2012) 141–142.

[14] Y. Sun, Y. Ming, X. Zhu, Y. Li, Out-of-Distribution Detection with Deep Nearest Neighbors, 2022. doi:10.48550/arXiv.2204.06507.

[15] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, Advances in neural information processing systems 30 (2017) 3146–3154.

[16] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, G. H. Chen, Ecod: Unsupervised outlier detection using empirical cumulative distribution functions, IEEE Transactions on Knowledge and Data Engineering 35 (2023) 12181–12193. URL: http://dx.doi.org/10.1109/TKDE.2022.3159580. doi:10.1109/tkde.2022.3159580.

[17] Aychang, aychang/roberta-base-imdb, https://huggingface.co/aychang/roberta-base-imdb, 2020. Fine-tuned RoBERTa model on the IMDb sentiment classification task.

[18] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, C. Potts, Learning word vectors for sentiment analysis, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Portland, Oregon, USA, 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[19] J. Zhang, Y. Zhao, M. Saleh, P. J. Liu, Pegasus: Pre-training with extracted gap-sentences for abstractive summarization, 2019. arXiv:1912.08777.

[20] Tuner007, tuner007/pegasus_paraphrase, https://huggingface.co/tuner007/pegasus_paraphrase, 2021. Fine-tuned PEGASUS model for paraphrasing, hosted on Hugging Face.