Post-Robustifying Deep Anomaly Detection Ensembles by Model Selection

Benedikt Böing*
Department of Computer Science
TU Dortmund
Dortmund, Germany
benedikt.boeing@cs.tu-dortmund.de

Simon Klüttermann*
Department of Computer Science
TU Dortmund
Dortmund, Germany
simon.kluettermann@cs.tu-dortmund.de

Emmanuel Müller
Department of Computer Science
TU Dortmund
Dortmund, Germany
emmanuel.mueller@cs.tu-dortmund.de

***Abstract*—Anomaly detection has been a major research area in machine learning with deep ensemble models showing exceptional performance. However, formal verification of robustness for anomaly detection in general, and ensemble models in particular, has been mostly neglected. Moreover, given an already trained, non-robust model, there is no way to adapt it for robustness as a post-processing step as of yet.**
By harnessing properties of ensemble methods - in particular of the *DEAN* model - we are the first to post-robustify a model via submodel selection. Beyond this new capability, our method significantly increases verification scalability by employing the inherent properties of ensemble methods.
Our experiments show that the *DEAN* model is most suitable for our method: it proves to be the most robust from the start, allows for post-robustification and keeps a stable runtime across all datasets considered.

***Index Terms*—Robustness, formal verification, anomaly detection, ensemble methods**

I. Introduction

Anomaly Detection has been an actively researched machine learning problem for several decades. Its use cases range from fault detection in machines [3] to credit card fraud detection [24] as well as to safety-critical areas such as medical diagnosis [7] or infrastructure control [12]. More recently, outstanding performance was attained by anomaly detectors combining ensemble methods and deep learning.
While researchers made a lot of progress in designing algorithms for increased detection performance, so far formal verification of these deep anomaly detectors has been neglected. Yet, especially in safety-critical areas, it is of utmost importance to formally state and prove guarantees about a model's behavior. This need is exacerbated by the discovery of so-called adversarial samples: inputs designed to fool the model into a wrong prediction [20]. For the anomaly detection task, this corresponds to false positive samples indistinguishable from normal samples in the train dataset.

From these adversarial samples, a research branch defending against such attacks has emerged. Methods such as adversarial training [9], [13] make neural networks more robust against adversarial samples by adjusting their training. If however these methods yield a non-robust model, there is no method to fix it, i.e., robustify it as a post-processing step.

*equal contribution

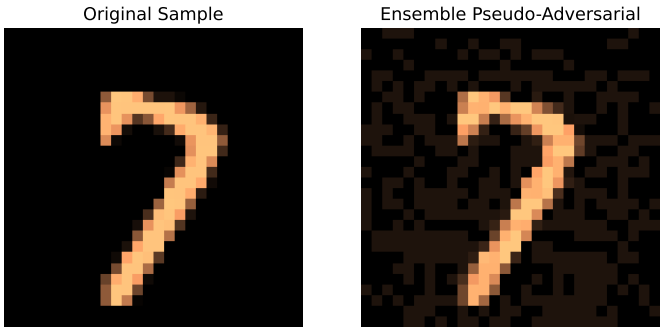

Fig. 1. Original MNIST sample (left) and its adversarial (right). Differences between the original and the adversarial have been enlarged to make them visible. The original sample is predicted as normal. The pseudo-adversarial *p-adv* on the other hand is predicted as anomalous even though it is nearly indistinguishable from the original.

We address the aforementioned challenges by using the inherent properties of ensemble methods. Given an already trained ensemble model, we first **assess its robustness by a divide-and-conquer approach**. By splitting it up into its submodels, solving a verification problem for each submodel and merging the intermediate results, we obtain both upper and lower bounds for the largest anomaly score in a predefined region.
Moreover, the intermediate results allow us to distinguish between submodels that either do or do not harm the ensemble's robustness. Thereafter we can create a new, robust ensemble model by using only the non-problematic submodels. Thus we obtain a **post-processing method that** - within certain bounds - **robustifies any such ensemble** to the desired degree.
Beyond post-robustification, our method is the **first to produce** a so-called **pseudo-adversarial** for an ensemble method as shown in Figure 1. These are inputs to the ensemble that are most likely being predicted as anomalous by the ensemble model and narrow down the approximation gap between the upper and lower bound of the anomaly score.
Using ensemble methods **alleviates another major issue of neural network's formal verification: scalability**. As shown by [25], the runtime of a verification method increases exponentially with the complexity of neural networks. However, the use of ensemble methods allows to elegantly circumvent

this problem as it is much faster to verify many small neural networks compared to one large neural network.
We highlight the use of our robustification method on the *DEAN* ensemble method [11]. This model combines several advantageous properties: since it employs feature bagging with a large set of simple submodels, we can scale up the verification to datasets of any dimension. Moreover, due to its simple architecture, any particular submodel can be verified fast and with almost no approximation loss by an SMT solver. With our experiments we show that we can successfully post-robustify a given *DEAN* ensemble without impairing its predictive performance. Moreover, we compare the robustness of the *DEAN* model to other well-known deep anomaly detectors. As it yields the best performance both in terms of scalability and robustness we deem it most suitable for verifiable anomaly detection.

II. Related Work

A. Deep Anomaly Detection Methods

Neural networks have shown exceptional performance in the task of anomaly detection. They assign an anomaly score to each input and compare it to a threshold to determine anomalousness. Due to their implicit feature learning, they are particularly well-suited for complicated distributions on high dimensional datasets [8]. For the purpose of post-robustification, we must combine this property with formal verification. However, formal verification proves to be a notoriously difficult task for large-scale neural networks. Therefore we need models that are complicated enough to model complex distributions and simple enough to be verified.
In this work, we primarily employ the *DEAN* model [11] because it is an ensemble of many simple submodels. Thereby each particular submodel can be verified while the entire ensemble retains its predictive capability.
Moreover we compare *DEAN* to two representative alternatives: an autoencoder-based ensemble called *RandNet* [4] and *DeepSVDD* [18]. Both models show state-of-the-art performance in anomaly detection, yet as we will see, they are not competitive in terms of both robustness and scalability.

B. Neural Network Verification

Formal verification of neural networks can be categorized into exact approaches such as SMT solvers [6], [10] and approximation approaches such as abstract-interpretation [19], [23], [27]. For our purposes, we will make use of SMT solvers as this allows us to obtain two types of results for a predefined region: an upper bound on the anomaly score as well as a lower bound derived from a pseudo-adversarial. Thus we can estimate the approximation gap to the true largest anomaly score in that region. Abstract-interpretation based approaches on the other hand, could only provide us with an upper bound and do not yield a pseudo-adversarial for ensembles.
However, exact solvers are notorious for being much slower than approximation approaches. We counteract this deficiency by feature bagging and limiting the complexity of each submodel. Overall this leads to an acceptable runtime while producing well-approximated robustness results.

C. Robustifying against Adversarials

Most existing methods, including [5], [9], [13], [26], try to incorporate robustness into their training methods. However, if this training does not yield a robust model, one can only retrain it without the guarantee of obtaining a robust model thereafter. Instead, we aim to adjust an already trained network to become provably robust.

III. Recall: *DEAN* Model

This section gives the necessary background on the *DEAN* model [11] we use to highlight the use of post-robustification. Beyond being on par with other state-of-the-art deep anomaly detectors, *DEAN* has some very favorable properties for robustness verification. The *DEAN* model is a particular type of deep anomaly detector given by an ensemble $D = (f_1, \ldots, f_m)$ of relatively simple neural networks. Each submodel f_i consists of a fully-connected neural network with ReLU activations in each hidden layer without constant bias. The input dimension b of f_i is set as a hyperparameter because *DEAN* employs feature bagging on the input: for an input $x \in \mathbb{R}^N$ to D and submodel f_i, the input $\tilde{x}_i \in \mathbb{R}^b$ denotes the projection of x onto the features for f_i. These networks are trained to transfer normal points close to a constant $q_i \in \mathbb{R}$ using the following equation:

$$L_{f_i}(\tilde{x}_i) = (f_i(\tilde{x}_i) - q_i)^2$$

As [11] suggest, we set this constant $q_i = 1$ for training and use $q_i = mean(f_i(x_{train}))$ for evaluation.
Since the submodels do not have constant bias terms, they cannot simply learn the constant function $f_i(\cdot) = q_i$. Instead the network needs to learn parameters in such a way that normal inputs result in a low deviation from q_i while all other inputs exhibit large deviations. Therefore this loss can be used to measure anomalousness for a given input similar to [18].
Finally we define the anomaly score obtained by *DEAN* given by

$$Anom(x) = \sqrt{\frac{1}{m}\sum_{i=1}^{m} L_{f_i}(\tilde{x}_i)}$$

combining all the outputs and thereby incorporating all input features of x. Moreover it will average out too large deviations occurring in e.g. just one submodel, resulting in a statistically robust model.

IV. Problem Setting

There are several notions of robustness both for deep anomaly detectors [15], [29] and for neural networks in general [28]. While a lot of emphasis has been put on training models to be robust, once a model exists it can only be shown or measured whether it is robust. However, given a non-robust model, it would be useful to just slightly adapt it in order to make it robust, instead of retraining a new model from scratch.

Therefore this section poses the challenge to post-process a given model such that it becomes robust. To this end, we will formally introduce the post-robustification problem and provide the necessary definitions.

A. Post-Robustification

Inspired by the robustness against adversarial samples in the realm of supervised learning, we want to locally post-robustify models for anomaly detection around a given input x^*. More precisely, this paper addresses the following problem:

Problem 1 (Post-Robustification)**.** *Given an evaluation metric m and a non-robust model-input pair (D, x^*), create a new model D^* such that (D^*, x^*) is robust and $m(D^*) - m(D)$ is maximized.*

This problem definition reflects that we do not only want to robustify our model on x^*, but that we also do not want to trade off too severely with respect to a given evaluation metric. Otherwise post-robustification might result in a degenerate model that maps all inputs to the same output: a model that is very robust but not at all useful.
Note that, assuming a monotone m, we do not want to minimize $|m(D^*) - m(D)|$: if D^* performs even better according to the evaluation metric, we do not consider this a problem.
As this problem definition is very general, in order to work with it we need to make it more concrete by giving the model, the evaluation metric and the notion of robustness: in this paper, we choose the previously defined *DEAN* model evaluated by the standard AUC score reflecting the predictive capability in the task of anomaly detection. The precise definition of robustness will be given in the next section.

B. Adversarial Robustness

Our robustness definition is the direct adaptation of adversarial robustness from supervised learning to anomaly detection. Essentially, we consider the anomaly detector as a binary classifier and apply the definition of [13] to it.

Definition 1 (ε-adv-rob)**.** *Let D be an anomaly detector, $dist$ a distance function and $x^* \in \mathbb{R}^N$ such that $Anom(x^*) < \tau$. We say that D is ε-adversarial-robust at x^* if and only if for all inputs in $x^* \pm \varepsilon := \{x \in \mathbb{R}^N : dist(x, x^*) \leq \varepsilon\}$ the **L**argest **A**nomaly **S**core is less than τ:*

$$LAS(D, x^* \pm \varepsilon) := \max_{x \in x^* \pm \varepsilon} Anom(x) < \tau$$

According to this definition, a normal input x^* is robust if and only if all surrounding inputs are normal as well. Thus if we can prove ε-adversarial-robustness, we know that there cannot be an adversarial sample in $x^* \pm \varepsilon$.
Typically $dist$ will be given by an L_p distance such as L_1 or L_∞ and for the remainder of this paper we will work with the L_∞ distance.
Please note that this is a much stronger notion of robustness than testing against a finite set of adversarial samples. In contrast, we aim to verify the model against infinitely many points defined by the ε environment of x^*. It corresponds to the same notion of robustness against adversarial attacks in supervised learning [13].

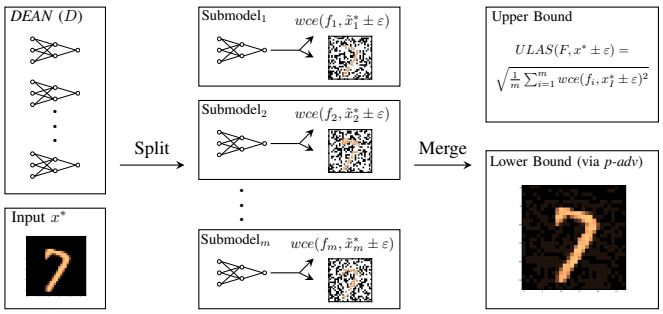

Fig. 2. Verification Process of the DEAN model. We first split DEAN and the input (left) up into the submodels and their features. Thereafter on each submodel both wce and *u-adv* are calculated (middle) followed by merging the results into the upper bound and the pseudo-adversarial *p-adv* (right). Note that due to feature bagging each submodel's *u-adv* does not cover all input dimensions.

V. Solution Framework

This section details how to check a given *DEAN* model for robustness and how to post-robustify it if necessary.

A. Robustness Verification

The general procedure of our verification framework is given by splitting up the ensemble model, calculating the so-called worst-case-error (wce) and *u-adv* [2] on each submodel and merging the results. In the following we provide details on these steps.

1) Splitting: We split the *DEAN* model into each of the submodels it consists of. Thus, if $D = (f_1, ..., f_m)$ is the *DEAN* model, we consider each f_i separately in the next step. Note also that for the next step we need to respect the feature bagging. If we want to check robustness for a given input x^* each submodel f_i is verified on $\tilde{x}_i^* \pm \varepsilon$.

2) Submodel Verification: For each submodel f_i we solve an adapted version of the worst-case-error problem posed by [2]. To be precise, for each model f_i we will approximately calculate

$$wce(f_i, \tilde{x}_i^* \pm \varepsilon) := \sup_{x \in \tilde{x}_i^* \pm \varepsilon} \|q_i - f_i(x)\|_\infty.$$

by employing SMT solvers. These solvers can check whether $\|q_i - f_i(x)\|_\infty$ is greater than a given δ and thus allow us to narrow down wce with a binary search over δ. Moreover, if the SMT solver returns *True* for a given $\hat{\delta}$, it will additionally return a sample $\hat{x}$ such that $\|q_i - f_i(\hat{x})\|_\infty > \hat{\delta}$. By keeping track of the largest realized error we can thus obtain an unsupervised adversarial *u-adv* which is an input that realizes wce up to a predefined accuracy[1].

[1]More details can be found at www.github.com/KDD-OpenSource/Robustify

3) Merging Outputs: From the two outputs obtained for each submodel, we will extract an upper and a lower bound for $LAS(D, x^* \pm \varepsilon)$. Recall that if the upper bound is lower than the anomaly threshold τ we prove local robustness. The lower bound on the other hand is used to estimate the approximation gap to $LAS(D, x^* \pm \varepsilon)$.

We **U**pper bound the **L**argest **A**nomaly **S**core by:

$$ULAS(D, x^* \pm \varepsilon) = \sqrt{\frac{1}{m}\sum_{i=1}^{m} wce(f_i, \tilde{x}_i^* \pm \varepsilon)^2}$$

We replace the error of each model with the largest error that can possibly manifest for each submodel. This is an overapproximation of $LAS(D, x^* \pm \varepsilon)$ because different submodels might realize their wce on different inputs. However the *DEAN* model must of course be evaluated on a single input only. If the upper bound is low enough, it serves as proof that the ensemble model is robust.

We construct the lower bound by combining the adversarials *u-adv* of each submodel f_i into an input point for *DEAN*. To this end we leverage a property of the adversarials *u-adv* obtained by our subroutine: usually they are at a corner of the input space $\tilde{x}_i^* \pm \varepsilon$. Thus for $(\tilde{x}_i^*)_k$ being the k'th dimension of $\tilde{x}_i^*$ they are given by $y_k \in \{(\tilde{x}_i^*)_k + \varepsilon, (\tilde{x}_i^*)_k - \varepsilon\}$.

Taking the perspective of a particular dimension k, there are several submodels that have this dimension as input due to feature bagging. To combine the adversarials we simply employ a majority vote among these submodels to determine which side of the corner we choose.

Thus let J be the index set of submodels using feature k as input and $\{y_k^j : j \in J\}$ be the corner points obtained for dimension k by their respective unsupervised adversarials. Then we construct a pseudo-adversarial for the ensemble as

$$\textit{p-adv}(x^* \pm \varepsilon)_k := \text{mode}\{y_k^j : j \in J\}.$$

From this pseudo-adversarial we extract a lower bound for $LAS(D, x^* \pm \varepsilon)$ by simply calculating $Anom(\textit{p-adv}(x^* \pm \varepsilon))$. Essentially, we try to combine the adversarials of each submodel in such a way, that many of the submodel's errors become large thereby tailoring a pseudo-adversarial for the ensemble model.

Even though this results in a lower bound for the *DEAN* model, we will experimentally show that it is close to the upper bound. Thus we achieve a small approximation gap.

B. Robustify

Endowed with the capability to determine robustness, we will now present a simple procedure with which we can post-process a non-robust model input pair based on each submodel's wce such that it becomes robust: we sort all submodels by their wce and remove them one by one starting with the largest wce until $ULAS(\hat{D}, x^* \pm \varepsilon)$ of the remaining models $\hat{D}$ is below the anomaly threshold τ. This way we can guarantee that in $x^* \pm \varepsilon$ the resulting ensemble will never predict *anomalous* ensuring robustness.

This seemingly simple procedure has to be executed with care given the following caveats: first, by reducing the number of models we might impair the predictive capability of the ensemble. We will show experimentally that this trade-off is not severe, but of course this depends on the level of robustness one wants to achieve. Secondly, there is a limit of robustness that we cannot overcome simply given by the smallest worst-case error of any submodel. Thus, if ε is too large (if we want too much robustness) post-robustification by removing submodels becomes impossible. Note that this algorithm can be applied to any anomaly detection ensemble and is not limited to *DEAN*.

VI. Experiments

This section highlights the use of post-robustification for a given *DEAN* model. We will start by looking into a single ensemble model trained on MNIST, showing what useful results can be obtained with our method. Thereafter we will compare *DEAN* with a) an autoencoder ensemble (*RandNet*) [4] and b) the *DeepSVDD* [18] model on 8 other real-world datasets highlighting that *DEAN* models are more robust from the beginning, allow for post-robustification on each of these datasets, and - in contrast to RandNet and DeepSVDD - keep a constant runtime across all datasets.

The code and more details on our experiments can be found at www.github.com/KDD-OpenSource/Robustify.

A. Deep Dive MNIST

Our first experiments are conducted on the MNIST dataset. Here we train an ensemble of 1000 *DEAN* submodels with feature bagging of size 32 to highlight properties of our *Robustify* method on a single ensemble model.

1) Remaining Predictive Capability: Assuming that we started with a powerful predictor, our first experiment addresses whether by robustifying the model loses its predictive capability. As shown in Figure 3, after post-robustification for one point we can keep 856 of the original models and sacrifice almost no AUC. Indeed we could have deleted more than 50% of the submodels before witnessing a severe drop in AUC score.

2) Approximation of the Largest Anomaly Score: As we cannot directly calculate LAS, we must approximate it. Recall that we obtained an upper bound on the LAS by aggregating over each submodel's wce and a lower bound by combining each submodel's *u-adv* into a pseudo-adversarial for the *DEAN* model. We can use both bounds to test how accurate the approximation of LAS for the *DEAN* ensemble is. Since we have no theoretical guarantee on the size of the approximation gap, we empirically evaluate it for a given *DEAN* model on 20 subsamples. Surprisingly, the relative error between the approximation gap and the actual LAS is always less than 2% showing that our approximation scheme is very precise.

3) Local vs. Global Robustness: Finally, we investigate the global effects of local robustification.

Figure 4 shows that we significantly decrease $ULAS$ for other normal samples by local robustification. It appears that the models we delete by robustifying one sample also cause

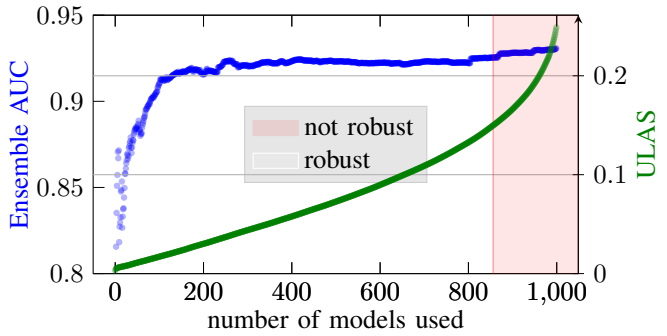

Fig. 3. ROC-AUC Score and $ULAS$ of the *DEAN* Ensemble as a function of the number of models used to generate it. Here we use only the submodels with the lowest error and consider an ensemble with $ULAS \leq 0.15$ to be robust as defined by τ. Calculating these individual errors does not require labels, so our process is still unsupervised.

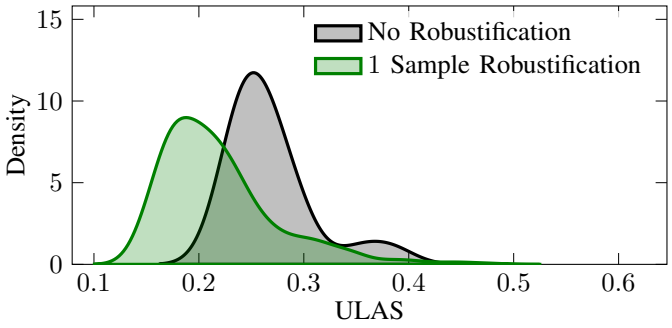

Fig. 4. Gaussian KDE plot of $ULAS$ of 20 samples before (black) and after (green) application of Robustify. After robustifying on a particular sample we calculated $ULAS$ on different samples to obtain the green density.

high wces on other samples. Therefore our method increases robustness not only locally but also globally.

B. Comparison to RandNet and DeepSVDD

This section compares *DEAN* to two representative, alternative models: *RandNet* [4] and *DeepSVDD* [18]. While *RandNet* consists of an ensemble of autoencoders, thereby being directly comparable to *DEAN*, *DeepSVDD* consists of a single, large neural network. Thus even though we cannot directly apply our post-processing method to *DeepSVDD*, we highlight how *DEAN* outperforms it with respect to runtime and robustness.

1) Datasets: We choose eight different datasets with varying number of features. These are chosen from [17] and [21], such that each algorithm achieves a similar AUC score as shown in Table I.

2) Ratio Comparison: As the anomaly scores of different models can have different scales, equation 1 defines the so-called Change Ratio (CR) to make the results on different models comparable. It indicates by what factor $ULAS$ needs to be reduced to obtain a robust model:

$$CR(D, x^*, \varepsilon) := \frac{ULAS(D, x^* \pm \varepsilon)}{\tau(D)} \quad (1)$$

Dataset	Features	RandNet	DEAN	DeepSVDD
pageblocks	10	0.9231	0.9577	0.8748
segment	18	0.9982	0.9998	0.9981
steelplates [16]	27	0.7521	0.7329	0.718
wbc	30	0.941	0.9751	0.9336
satellite	36	0.8321	0.8196	0.8233
qsarbiodeg [14]	39	0.8734	0.7425	0.821
gasdrift [22]	128	0.9791	0.9319	0.9562
har [1]	561	0.9786	0.9529	0.9371
Average		0.9097	0.8890	0.8828

TABLE I
ROC-AUC SCORES ON THE 8 DATASETS USED IN THIS PAPER

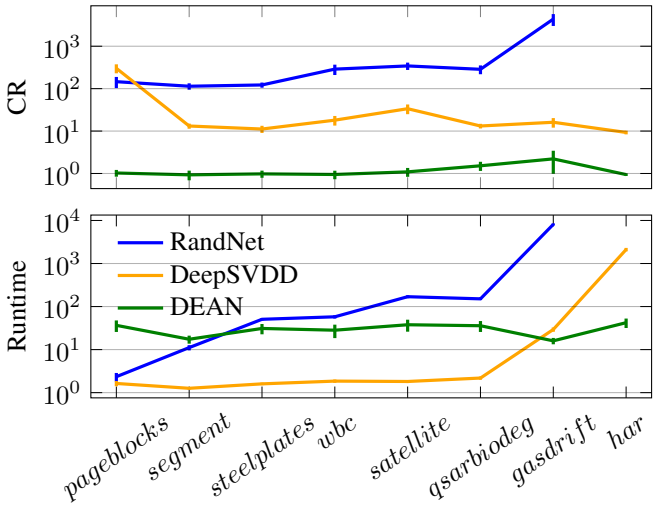

Fig. 5. Change Ratio and runtime on eight datasets averaged over 10 runs and 20 samples each. The computational effort for *RandNet* and *DeepSVDD* increases with the number of features (note that the datasets are sorted by their number of features). For the highest dimensional dataset "har" with 561 features, it was impossible to verify the autoencoder due to timeout. Notice that in contrast to the other algorithms, DEAN has constant runtime and the lowest Change Ratio.

The upper part of Figure 5 shows that for every dataset, *DEAN* is already the most robust, often having a $CR \leq 1$. While *DeepSVDD* is already less robust, *RandNet* has change ratios more than 100 times higher than those of *DEAN*. At least partly this is due to a larger approximation error of $ULAS$ as the different algorithms use different norms (L_2 or L_∞) for their anomaly scores. Therefore the simple architectures of each of *DEAN*'s submodels favor a precise calculation of $ULAS$.

The bottom part of Figure 5 shows the runtime of the verification algorithm. As the x-axis is sorted by the number of features in each dataset, we show the drastic increase in required verification time both for *RandNet* and, to a lesser extent, also for *DeepSVDD*. In contrast, the required verification time for the *DEAN* model stays constant over all datasets as feature bagging allows keeping the number of ReLU nodes in each submodel the same.

Even though *DEAN* seems to have a larger runtime on lower-dimensional datasets, note that - unlike *DeepSVDD* - its

verification process can be parallelized along the submodels. Moreover it depends linearly on the number of submodels verified. Therefore the eventual runtime of *DEAN* verification can be controlled both with the number of submodels to be verified and with the number of CPU cores available.

VII. Conclusion

In this paper, we are the first to study the formal verification of ensembles for anomaly detection. We show that especially the *DEAN* model seems suitable for this task, as it outperforms common alternatives in terms of robustness. Also, using feature bagging, *DEAN* achieves a verification time independent of the number of features of the dataset. To our knowledge, this is the only algorithm that achieves sub-exponential verification time.

We also introduce *Robustify*, a method allowing to further improve the robustness of the ensemble. We show using *DEAN* that this method allows to drastically increase the local robustness of the model while maintaining the same anomaly detection capability.

However, to achieve global robustness, we require a more powerful algorithm. This could be a method similar to boosting, scaling the impact of less robust submodels differently and thus reaching global safety against adversarial attacks. We leave these ideas for future work.

Acknowledgements. This work was supported by the Research Center Trustworthy Data Science and Security, an institution of the University Alliance Ruhr.

The authors are thankful for the computing time provided on the Linux HPC cluster at Technical University Dortmund, helped by the Large-Scale Equipment Initiative by the German Research Foundation as project 271512359.

References

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. Training computationally efficient smartphone–based human activity recognition models. *Artificial Neural Networks and Machine Learning – ICANN*, 2013.

[2] Benedikt Böing, Rajarshi Roy, Emmanuel Müller, and Daniel Neider. Quality guarantees for autoencoders via unsupervised adversarial attacks. In *Machine Learning and Knowledge Discovery in Databases - European Conference - ECML PKDD*, 2020.

[3] Lucas Costa Brito, Gian Antonio Susto, Jorge Nei Brito, and Marcus Antonio Viana Duarte. Fault detection of bearing: An unsupervised machine learning approach exploiting feature extraction and dimensionality reduction. *Informatics*, 2021.

[4] Jinghui Chen, Saket Sathe, Charu C. Aggarwal, and Deepak S. Turaga. Outlier detection with autoencoder ensembles. In *International Conference on Data Mining - ICDM*, 2017.

[5] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of relu networks via maximization of linear regions. In *International Conference on Artificial Intelligence and Statistics - AISTATS*, 2019.

[6] Rüdiger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *Automated Technology for Verification and Analysis - ATVA*, 2017.

[7] Tharindu Fernando, Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Deep learning for medical anomaly detection - A survey. *ACM Computing Surveys*, 2022.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[9] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations - ICLR*, 2015.

[10] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification - CAV*, 2017.

[11] Simon Klüttermann and Emmanuel Müller. Dean: Deep ensemble anomaly detection. https://github.com/psorus/DEAN.

[12] Van Khoa Le. *Detection of atypical events for security in critical infrastructure*. Theses, 2018.

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations - ICLR*, 2018.

[14] Kamel Mansouri, Tine Ringsted, Davide Ballabio, Roberto Todeschini, and Viviana Consonni. Quantitative structure–activity relationship models for ready biodegradability of chemicals. *Journal of Chemical Information and Modeling*, 2013.

[15] Naji Najari, Samuel Berlemont, Grégoire Lefebvre, Stefan Duffner, and Christophe Garcia. Radon: Robust autoencoder for unsupervised anomaly detection. In *International Conference on Security of Information and Networks - SIN*, 2021.

[16] Semeion Research Center of Sciences of Communication. Steel plates faults data set.

[17] Shebuti Rayana. Odds library, 2016.

[18] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International Conference on Machine Learning - ICML*, 2018.

[19] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems - NeurIPS*, 2018.

[20] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations - ICLR*, 2014.

[21] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *SIGKDD Explorations*, 2013.

[22] Alexander Vergara, Shankar Vembu, Tuba Ayhan, Margaret A. Ryan, Margie L. Homer, and Ramón Huerta. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 2012.

[23] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane S. Boning, and Inderjit S. Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning - ICML*, 2018.

[24] Tung-Yu Wu and Youting Wang. Locally interpretable one-class anomaly detection for credit card fraud detection. In *International Conference on Technologies and Applications of Artificial Intelligence - TAAI,*, 2021.

[25] Kai Yuanqing Xiao, Vincent Tjeng, Nur Muhammad (Mahi) Shafiullah, and Aleksander Madry. Training for faster adversarial robustness verification via inducing relu stability. In *International Conference on Learning Representations - ICLR*, 2019.

[26] Huan Zhang, Minhao Cheng, and Cho-Jui Hsieh. Enhancing certifiable robustness via a deep model ensemble. *CoRR*, abs/1910.14655, 2019.

[27] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems - NeurIPS*, 2018.

[28] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *IEEE Conference on Computer Vision and Pattern Recognition - CVPR*, 2016.

[29] Chong Zhou and Randy C. Paffenroth. Anomaly detection with robust deep autoencoders. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2017.