# DEAN: Deep Ensemble Anomaly Detection

Simon Klüttermann
*Chair of Data Science and Data Engineering*
TU Dortmund University, Germany
simon.kluettermann@cs.tu-dortmund.de

Emmanuel Müller
*Chair of Data Science and Data Engineering*
TU Dortmund University, Germany
emmanuel.mueller@cs.tu-dortmund.de

***Abstract*—Ensemble methods are a key technology for many supervised machine learning tasks. However, there is little research that relates them to unsupervised and in particular unsupervised anomaly detection methods. An ensemble of anomaly detection might allow to combine the individual strengths of multiple models. This requires a comparable basis of outlier scores and profits from high variance between sub-models.**

**In this paper, we introduce Deep Ensemble Anomaly Detection (*DEAN*), which enables such a consistent scoring of anomalies, while also having very high intermodel variance.**

**We enhance weak but deep anomaly detectors into an ensemble that outperforms more complicated networks, while being much easier to optimize and allowing for explanations of its prediction.**

**In experiments, we show the quality, robustness, scalability and interpretability of our ensemble. We compare against shallow, deep, and ensemble competitors.**

***Index Terms*—Ensembles, Anomaly Detection, Unsupervised Deep Learning**

## I. Introduction

Anomaly detection is an essential sub-field of data analytics with a variety of applications in data cleaning and data pre-processing in general, but also in specific domains like financial fraud, customer risk, or health surveillance. In all of these domains, anomalies are rare, exceptional and interesting objects that are highly deviating from the residual (regular) data. Due to the variety of applications, there was always the need for a large variety of anomaly detection models [1]. Each of these proposed outlier models has the ability to detect specific types of anomalies, while it is an open research question how to combine multiple of these models into an outlier ensemble to improve their anomaly detection quality [41]. In particular, for creating powerful deep anomaly detection models [31] with complex parameter sets, ensemble methods have not yet been studied much.

We see the huge potential of ensemble methods that have been exploited for shallow anomaly detection [21], [24], [25] in various application domains. In contrast to this, recent deep anomaly detection methods [31] show advanced detection quality on image data, but limited abilities when combined into an ensemble of multiple models: We address the intersection of these two research fields (ensemble methods and deep anomaly detection). While deep methods allow for training feature spaces that can distinguish regular (normal) data from abnormal (anomaly) objects, this flexibility of arbitrary feature spaces lacks consistency in the resulting outlier score. For example, in Figure 1 we observe a toy example with normal (black dots) and abnormal (red cross) objects. The central

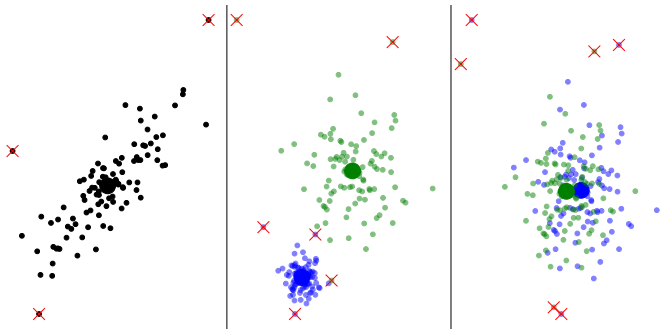

Fig. 1. Challenge of comparable scoring of outliers in deep anomaly detection models. Given the two-dimensional data on the left, a noncomparable model might transform it similarly to the middle distribution and use the difference to the big dots as an anomaly score. Since both models have different scales, the green distribution would be weighted higher when combined into an ensemble. A more consistent transformation is shown on the right, allowing for an unbiased combination of both models.

figure illustrates two spheres of trained one-class SVDD models [32] that vary in feature spaces but also in the reference center of each sphere. Hence, these models do not allow for the combination of scores due to a lack of consistency. The right figure illustrates two spheres with limited flexibility, as they share the same reference center, but who still allow for training arbitrary feature spaces.

Given comparable sub-models, we are much easier able to combine multiple models. Still, to improve the performance, scalability and robustness of the ensemble, we require the sub-models to learn different concepts: Even if we could effectively combine multiple deep anomaly detection methods, their ambition to create a singular good algorithm limits their variance and thus their performance. This variance could be achieved by combining different algorithms, but this again strongly limits their comparability [11].

With *DEAN*, we define a set of concise properties that allow for comparable scoring in deep anomaly detection, while achieving high intermodel variance. We show that existing deep methods such as reconstruction-based anomaly detection or deep-one-class methods do not fulfill these properties as they are too flexible in the allowed feature spaces but too restrictive in the training parameters learned. In contrast, we propose a more variable but consistent outlier score in a deep ensemble. This allows for improved anomaly detection due to the combination of scores as well as an interpretable expla-

nation of impact factors. We suggest a new loss function that solves these specific goals for outlier ensembles and propose its combination into a deep ensemble. Our contributions:

- definition of concise *DEAN* properties
- consistent high variance loss function for *DEAN*
- scalability optimization of *DEAN*
- interpretable description for anomaly ensembles

In experiments, we show the anomaly detection quality, parameter robustness, and runtime scalability of our novel *DEAN* model. We compare against shallow, deep, and ensemble competitors [10], [25], [32], [33].

## II. Related Work

Before introducing *DEAN*, we will briefly review related unsupervised anomaly detection methods and give an overview of open challenges in ensemble methods; in particular for deep anomaly detection. We summarize our contributions compared to related work in Table I

**Unsupervised outlier detection** refers to algorithms that search for abnormal, rare, or exceptional objects in a database of unlabeled objects [1]. Neither normal nor abnormal objects are known and labeled beforehand, which makes it a very challenging task to score anomalies directly or to train models representing normal/abnormal data characteristics. Many different applications exist, from approaches to anomaly detection on log data [40] to time series anomaly detection [3] and graph-based applications [19]. We will study here the most general application on tabular data, and leave extensions of our method for future work.

This task has been widely studied: Algorithms based on nearest neighbor methods [16], and One-Class Support Vector Machine (OC-SVM) [4] divide the feature space into normal and abnormal regions, while methods like LOF [8] exploit statistical properties to separate normal samples from outliers. As the complexity of this separation is limited, they cannot reach a high accuracy and tend to not work well with many features. This complexity problem is solved, by adding depth to the outlier detection algorithm. This allows methods like the *Autoencoder* based reconstruction introduced in [15], [33], [34], those based on training a generative adversarial network (*GAN*) [28], or those that extend a given shallow method [32] to drastically outperform shallow algorithms, especially for many features. However, none of the above methods natively includes ensemble methods, which, as we show later, limits their detection quality, robustness and often results in less interpretable models.

**Ensemble methods** [41] refer to meta-algorithms to combine multiple models into one that has a better performance than each constituent algorithm. In supervised machine learning, ensemble methods like bagging [7] and boosting [14] combine the output of multiple models to produce the desired output more accurately. This becomes more complicated for unsupervised training goals like outlier detection as there is usually no desired output available. One way to solve this is to combine the outlier scores of different methods. This works as different outlier detection methods are better at finding different kinds of anomalies [1] and results in an ensemble [2] being more general than each of their constituent models, theoretically having less bias and variance than the initial models.

However, as combining a set of anomaly detection methods requires a choice of this set of algorithms and thus introduces human bias, their trustworthiness is limited. Moreover, since every two models are not similar, combining the different outlier scores will become more complicated [11]. Also since the number of different algorithms is limited, the improvement is finite and methods like our interpretability in Section VI-E become impossible.

Because of this, ensembles of singular model types are usually preferable. To increase their variance, meta-algorithms like feature bagging [24] can be employed, making the outlier score more robust: The same model is trained multiple times considering only some random features in each version. Similar to this, subsampling considers parts of the training set in each model. These are useful ideas, but there is no deep method explicitly designed for ensembles, and thus their effectiveness is limited by the drawbacks of the method without them. Most notably, a model with good performance is often not the best choice to be included in an ensemble: The effectiveness of the ensemble is limited by their often high training time for each model and thus by the low maximum number of models included in the ensemble. Moreover, if every single model has a good performance, this generally means that its variance is reduced and thus also the robustness of the resulting ensemble.

**Ensembles for outlier detection** can be quite powerful, especially for shallow submodels. Methods like *Isolation Forest* [25], or subspace ensembles [21] are few but interesting approaches towards a combination of multiple outlier models.

In the ensemble of multiple isolation trees building an *Isolation Forest*, every single tree is a weak outlier detection method combined into a stronger one. We briefly explain this method in Section III, but since *Isolation Forests* are shallow, their performance is often limited compared to their deep alternatives.

There exist other ensembles which combine many versions of existing deep anomaly detection algorithms [10], [17], but the assumption that each submodel is a powerful deep learning model, generally limits the variance of sub-models and thus their performance.

We will compare here against *RandNet* [10], which combines many *Autoencoder* based models into an ensemble. To increase the variance of their sub-models, they randomly disable on average up to $\frac{1}{e} < 37\%$ of connections in each *Autoencoder*. Still, we will see that the variance this induces is limited, and thus their performance is not very good.

## III. Basic Notions and Open Challenges

We consider outlier definitions that train a model capturing the characteristics of normal data and apply the parameters extracted from this to score abnormal samples as highly deviating from the normal characteristics learned by the model.

| | depth | ensemble | consistent | variable | scalable | robustness | interpretable |
|---|---|---|---|---|---|---|---|
| shallow outlier models [8], [16], [20], [22], [35] | | | (✓) | | ✓ | | ✓ |
| shallow ensembles [21], [24], [25], [42] | | ✓ | | ✓ | (✓) | ✓ | (✓) |
| reconstruction-based [9], [34] | (✓) | | | | | (✓) | |
| deep outlier models [32], [36], [38] | ✓ | | | | (✓) | (✓) | |
| deep ensembles [10], [17] | ✓ | ✓ | (✓) | | | (✓) | |
| *DEAN* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

TABLE I
COMPARISON W.R.T. OUR KEY CONTRIBUTIONS WITH RELATED WORK

**Definition III.1** (Outlier Score)**.** For datapoints $x_i \in DB$ and some reference point $y_\theta \in DB$ representing the regular expectation of a model $f_\theta$ an outlier score is defined as

$$score(x_i) = |\,(f_\theta(x_i) - y_\theta)\,| \tag{1}$$

Different instantiations of this definition exist in the literature but are highly dependent on $f_\theta$ and especially on the loss function used to train $f_\theta$.

A **Reconstruction-based paradigm** simply assumes the original input object as reference point $y_\theta = x_i$ and measures the reconstruction error of an object $x_i$ before and $f(x_i)$ after the transformation. *Autoencoder* neural networks [34] are the most prominent example of the reconstruction-based anomaly detection paradigm. The model $f(x_i)$ is an encoder that compresses the feature space of the given training data into a lower-dimensional latent space representation, followed by a decoder that decompresses this representation again into the original space. Removal of redundancy by the latent space results in a lossy decompression, and hence, $|f(x_i) - x_i|$ can be used as a measure for anomaly detection (i.e., objects that do not compress well are considered anomalous).

However, *Autoencoder* do not profit much from an ensemble method. This is because their loss function

$$l_{AE} = \sum_{i=0}^{n} (f(x_i) - x_i)^2$$

is quite restrictive. Having each feature reconstructed well, limits the number of different parameter sets that achieve this. Furthermore, the hyperparameter of an *Autoencoder* are quite sensitive. Especially the latent space dimension is hard to optimize in general and heavily impacts the outlier detection quality. As we show in our experiments, having these restrictions for an ensemble results in relatively poor results.

**Autoencoder ensemble** One notable method of these autoencoder ensembles is *RandNet* [10], which combines the scores of many *Autoencoder* trained with different initialization into an ensemble score. To solve the limited variance, they randomly remove some connections (up to $1 - \frac{1}{e} < 37\%$) of their deep neural network. Afterwards they define their anomaly score as the median of their submodel scores $score(x) = median_i(score_i(x))$. We will later show that this approach might reduce the ensemble problems of the underlying *Autoencoder*, but does not solve them: Still, each submodel is quite similar, and thus the performance is limited. In comparison, we designed *DEAN* to address the submodel variance and outlier score comparability directly.

**Deep Anomaly Detection Algorithms** are an alternative to this reconstruction-based paradigm that adds a specialized training of $f_\theta$ optimized for anomaly detection. For example, by using a loss function inspired by one-class support vector machines [39] in the *DeepSVDD* method [32]: After training $f_\theta$ represents a data transformation in which the distance of each training sample $x$ to a reference point $\vec{c}$ is minimal. Outlier scores are computed by $score(x) = |f_\theta(\vec{x}) - \vec{c}|$, while the reference point $\vec{c}$ is here chosen as $c = mean(f_\theta^{untrained}(x))$, a function of the random network initialisation, and thus also almost randomly.

Unfortunately, because of this randomness of the reference point $\vec{c}$, the score lacks consistency, which, as shown in our experiments, hinders ensemble methods. Due to a heavy training objective [1]

$$l_{DeepSVDD} = \frac{1}{n} \sum_{i=1}^{n} \left\| f(\vec{x_i})) - \vec{c} \right\|^2 + \frac{\lambda}{2} \sum_{l=1}^{L} \left\| W^l \right\|_F^2$$

that does not allow for high model variances, the performance of our ensemble is again limited.

In contrast to such deep learning methods with heavy optimization schemes, **Isolation Forest** [25] use a simple binary tree classifier $\theta$ with random feature selection and random splits that isolate objects in each leaf node. It can be shown that deviating samples reach a leaf node earlier, and thus this depth can be used to differentiate between normal and abnormal samples: $score(x) = |avg_\theta(x) - max_{depth}|$

As *Isolation Forests* do not rely on a distance measure, they often outperform other shallow methods for high-dimensional data. Hence, we use them as a representative ensemble and shallow competitor in our experiments. However, we show that the anomaly detection quality is often still quite limited compared to our deep ensemble anomaly detector.

## IV. METHODOLOGY

Let us first define the general properties for a well-behaving deep ensemble anomaly detection and then discuss our concise loss function for our anomaly model and the details of our deep ensemble.

### A. Problem Definition

As ensemble methods increase the robustness of an outlier detection algorithm while reducing both the bias of the underlying models and decreasing the variance of the resulting

[1] $W^l$ represents the weights of layer l of the neural network, implementing a frobenius norm regularisation

model [1], they have the potential to outperform every other outlier detection algorithm. To reach an optimal ensemble, we suggest the following attributes.

**Scalability**: A perfect model to be combined into an ensemble would be generated quickly, making the combination of thousands of models possible and thus reducing the variance of the combination model further while making concepts like the interpretability described in Section VI-E possible. This also implies that using multiple algorithms or the same algorithm with different hyperparameters is less optimal, as this limits the maximum number of models in the ensemble while also introducing human bias in the selection: We prefer to use models that can be infinitely drawn from a distribution $f_j(x)$, for example, the initialization of a neural network.

**Depth**: For a good ensemble performance, we require the ability to describe complicated relations, and thus each model needs to be deep. This does not mean that the performance of each model needs to be high: To reach a good ensemble performance, the single model performance $f_i(x)$ is less important than its ability to profit from the ensemble.

**Variability**: It is impossible to learn something new, from the combination of multiple identical sub-models. So for the ensemble to reach a good performance, we require the variance between predictions $var(f_i(x))$ to be high. This is often mutually exclusive with the quality of every single model, as fewer parameter constellations are reaching a great performance than reaching a good one.

**Consistency**: Finally, we require an ensemble combination function that combines every single model into an ensemble and each model to work well with this function. In reference to Equation 1 this requires $score_i(x)$ to be similar for each model, which implies $y_\theta$ to be constant and the scale of $f_\theta(x)$ to be the same.

For example, a model is inconsistent if implicit unsubstantiated weights are applied to each model. *DeepSVDD* introduces these (inconsistent) weights as $\vec{c}$ which are drawn almost randomly, resulting in lower quality and robustness if combined into an ensemble. At the same time, the anomaly score of an *Autoencoder* does depend on the varying reference point $y_\theta = x_i$ of each sample, complicating the combination function and making the interpretability introduced in Section VI-E impossible.

In contrast to the existing anomaly detection methods, we aim to satisfy these four properties (scalability, depth, variability and consistency) for a novel deep ensemble method: *DEAN*. After describing how *DEAN* works in the following, we will show in experiments that *DEAN* satisfies all of these and results in the best quality, robustness, and runtime in our empirical results.

### B. The DEAN submodel

As we would like each model to have a high variance while being trained quickly, we choose a simple loss function: We train each model to output a constant value of 1. That means given a neural network $f(x)$, we minimize the loss given by Equation 2

$$l_{DEAN} = \left(f(x_{train}) - 1\right)^2 \tag{2}$$

Similar to *DeepSVDD* [32] we usually need to remove the learnable shift (also called bias) from the neural network $f(x)$ and relinquish partially constant activation functions to be able to gain non-trivial results. See Section IV-E for a more detailed study of this problem.

*1) A single model as an anomaly detector:* The intuition why Equation 2 works is very similar to supervised regression. In theory, any regression task is also an anomaly detection task when we compare its model output to the truth we expect. This regression matches the desired value in most cases if the model is trained well. But if there is a notable difference between both, we can consider the sample anomalous.

In practice, this is a very restricted way of finding anomalies. It requires a manual choice of one feature as the label and is strongly biased to changes in this label. Minor differences in preprocessing can also easily lead to large changes in the model, which is why this method of anomaly detection is not used in practice.

With our loss function, we try to make this intuitive idea of anomaly detection viable.

1) First, we are searching for relations $g(x)$ that are approximately constant in our training set. $g(x) = k \cdot (1 \pm \delta)$. This is a much more general task compared to predicting features, especially as these relations are findable even if we change the preprocessing (for example, when we redefine our features). And since there are usually many approximately constant values constructible in each dataset, we have a high submodel variance.
2) Instead of finding a single equation to predict whether a sample is to be considered normal, we want to use many different relations in an ensemble. For this we rewrite the definitions of our relations so that all equal 1 on one side $f(x) = \frac{g(x)}{k} \approx 1$. This means that the scale of each model is now comparable.
3) Finally, we do not want to find any relations, but those relations that have the most meaning. We assume that these are the most constant relations, as a strongly variant relation also hides many anomalies in this randomness. Mathematically this means, we are searching for the lowest $\delta$.

This is achieved by optimizing Equation 2:

$$l_{DEAN} = \left(f(x_{train}) - 1\right)^2 = \left(\frac{k}{k} \cdot (1 + \delta) - 1\right)^2 = \delta^2 \tag{3}$$

Accordingly, we expect normal samples to produce $f(x) \approx 1$, while a high deviation implies anomalous samples.

*2) Outlier Scoring:* This allows us to score each sample by Equation 4.

$$Score = |\left(f(x_{test}) - q\right)| \tag{4}$$

where $q = \text{mean}\left(f(x_{train})\right)$. One might set $q = 1$, but deriving $q$ from the training set is a better choice, as the

loss without learnable shift prefers distributions around points lower than one: Consider a distribution of infinite samples with a density equally distributed between $1-\alpha$ and $1+\alpha$. After a quick calculation, one can find that $l_{DEAN} = \frac{2}{3} \cdot \alpha^3$. If you instead multiply each value with an optimizable variable $q$ you get a lower loss of $\frac{2 \cdot \alpha^3}{3+\alpha^2}$ at $q = \frac{3}{3+\alpha^2}$. As most relative uncertainties are $<< 3$, the difference to 1 is small, but setting this value right has a measurable effect. And as the value of $q$ depends on the width and shape of the distribution, it is hard to estimate, which is why we simply measure it from our training data. This level of optimization is only possible since our loss function is so simple, giving another reason why the combination of simple models can improve one complex model.

*3) Optimisations:* Because using some partially constant activation function (tanh, sigmoid) could trivially solve $l_{DEAN}$, we suggest a relu function as activation. Still, this can sometimes create problems with vanishing gradients depending on the initialization. We suggest switching to elu activations [12] if this appears and also not to activate the final layer.

As a preprocessing step, we normalize our data so that every training sample lies in the range $[0, 1]$.

For the application to the MNIST dataset, we also use an Autoencoder (This Autoencoder has five hidden layers with a latent dimension of 192), as its latent space concentrates useful features. Since this introduces a certain randomness into our ensemble, we combine the latent spaces of 5 Autoencoder. As the Autoencoder performs much worse on cifar10, we only use this optimization step on MNIST data, as we do not want to remove useful pieces of information.

### C. Ensembles

A single *DEAN* model works as an outlier detection method, but as a deliberately simple model, it is not very powerful. The benefit of this simplicity is its ability to combine into an ensemble of outlier detection methods easily. Therefore we train here multiple neural networks $\mathrm{f_i}(x)$ with the same loss function and model setup, but with different initializations.

We employ feature bagging so that each model only sees some ($bag$ many) of the features, as this is a native extension of our ensemble and results in almost constant runtime in the number of features, allowing us to ignore the higher number of features from using multiple Autoencoder latent spaces. It also increases submodel variance, while not affecting the comparability of our submodels.

Finally, we average the score of each model according to Equation 5

$$F(x) = \frac{1}{n} \sqrt{\sum_{i=0}^{n} \mathrm{f_i}^2(x)} \tag{5}$$

The simplicity of our model allows it to have a higher submodel variance and comparability, resulting in the quality we show in Section VI.

### D. Hyperparameters

In most of our analyses, we train 3000 models and combine them into one ensemble. As Figure 2 shows, this is unnecessary as convergence happens much earlier, but we like to show the best our method can achieve.

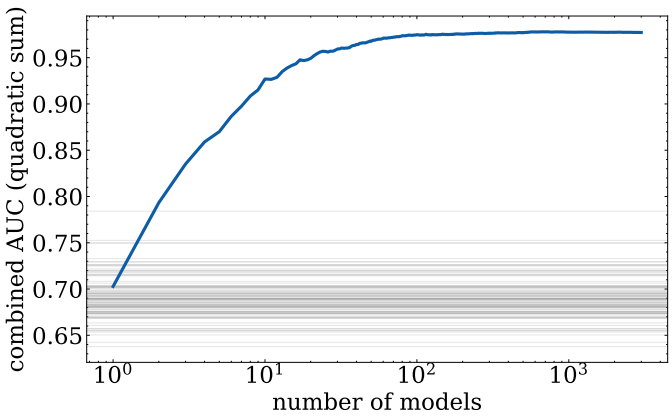


Fig. 2. Convergence behavior of our ensemble on the dataset "7" compared to the results of single models.

To employ feature bagging, we need to specify how many features each model can see (referred to as $bag$). We choose a value of 128 for MNIST data and a higher value of 384 for the higher-dimensional cifar10. Changing this hyperparameter, like changing any other, does not change the ensemble quality much (see Section VI-D). We use two hidden and one output layer with $bag$ neurons for our neural network in every layer. We activate them using the relu activation function and use a learning rate of 0.03 and a batch size of 100.

Each submodel trains for a maximum of 500 epochs but usually stops earlier, as we employ EarlyStopping [29] with a patience of 5 epochs. EarlyStopping normally decreases the runtime of a deep learning model, while increasing its variance and thus decreasing its average performance. But as submodel variance is a good thing for our ensemble, there is no reason not to use it here.

### E. Learnable shift

One drawback of a simple training objective is the possibility of trivial solutions. While for a complicated training objective like those of an *Autoencoder* it is almost impossible to create a model with low loss without understanding the training data, for our loss function of $\sum_i (f(x_i) - 1)^2$ the function $f(x) = 1$ is a perfect solution independently of the training data.

The same problem appears in *DeepSVDD* [32], where it is solved by removing the learnable shifts[2] employed in the setup of the neural networks. This is indeed also a solution for *DEAN* and, as we will see in Section VI, works quite well in practice.

[2]Learnable shifts are more commonly called biases, but we rename them here, not to be confused with the variance bias tradeoff

The drawback is that removing learnable shifts restricts the complexity of functions learnable by a neural network until they are no longer a universal function approximator: This phenomenon can be observed in Figure 3, showing the results of multiple complicated neural networks, tasked to learn part of a sinus curve: The learnable shifts are necessary to find this sinus curve. This is especially problematic, as this limits our submodel variance: When no submodel can learn a complicated relation, the resulting ensemble usually can not learn this relation either.

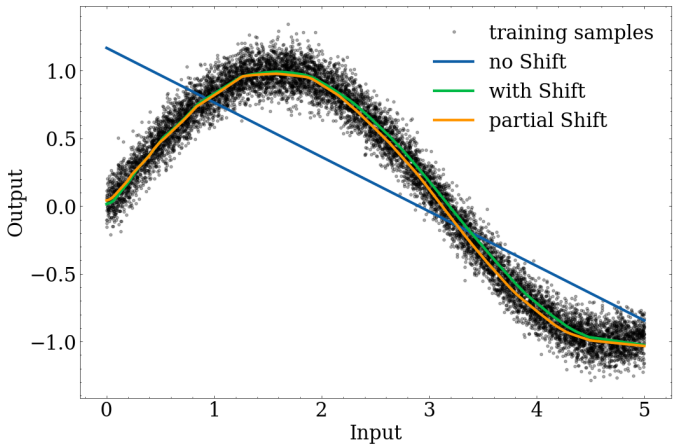


Fig. 3. Given complicated alinear data, the functions learned by three neural networks are shown. One network without learnable shifts cannot capture the structure of the underlying data, while both a network with learnable shifts in each layer and a network with learnable shifts in all layers except the last can describe the alinearity.

Interestingly, our ensemble structure gives us another solution to this problem: Since our loss function is quite simple, it is also easy to fulfill. And as neural network optimizers are not perfect, this means that they usually find a sub-optimal non-trivial solution, increasing the submodel variance and thus the ensemble performance.

So by using many neural networks, most still work with biases, and in the case they find trivial solutions, this is only a constant 0 contribution in our combination (Equation 5).

More problematic is that the combination of a trivial solution with a solution is also a solution: $f'(x) = \alpha + (1-\alpha) \cdot f(x)$.

This scales the anomaly scores and thus does not change the anomaly detection performance of a single model. Still, for an ensemble, this weights each model with a random weight, reducing the comparability of our ensemble models.

We try to combat this effect, by removing the learnable shifts only in the last layer of the neural network. These partial shift neural network are still a universal function approximator, as Figure 3 shows, but it is harder to create a trivial solution: Instead of $f(x) = x*0+1$ the function $f(x) = (0*x+1)*1$ has to be learned.

Still, both solutions have their merit. While partial shifts allow more complicated distributions to be learned, using no learnable shifts seems to be simpler to optimize and tends to converge a bit faster. We will compare both algorithms in the later section, showing that in practice, the difference seems to not matter too much, at least for the datasets we consider here.

## V. Understanding DEAN as a density estimator

A common way to understand anomaly detection is through the density of the underlying distributions [5], [8], [31]. This idea considers an anomaly detection algorithm to be good if its anomaly score can be understood as a measure of the underlying density of the training samples.

In practice, this is a very complicated goal, and often only approximately solvable: Describing an arbitrarily complicated probability density function requires an arbitrarily complicated function to describe it, with arbitrarily many parameters, while most methods generally have finitely many parameters.

Ensembles can be a solution to this, but for example, an *Isolation Forest* is not, as it is biased along its axes [18].

We think that the combined convergence of many models seen in *DEAN* might fit this expectation better.

Sadly it is hard to show if a model follows the density distribution mathematically for a complicated neural network, and especially for an ensemble of those.

Because of this, we will only try to motivate it experimentally. For this, we use the Run_or_walk dataset [27], as it is low dimensional, with two dominating features, and thus easy to visualize, while still having enough samples to measure densities.

As this is a binary classification dataset, we only consider the walking part for our anomaly detection. We could use the running part as anomalies, but as we only care if the densities of the normal sample are learned, we don't require it. We train a relatively small ensemble of 10 models without feature bagging as described above and show the anomaly scores on our test set, once for the ensemble and for the 10 models themself in Figure 4. As this dataset is six-dimensional, we use a RandomForest-Classifier on the whole dataset to find the most important features, which here are the acceleration in x and z directions and plot them (We still train on all six dimensions, but only visualize 2). We also show a histogram of the training data, to allow for density comparisons.

Notice how much the individual model varies, from considering both peaks to be normal (Model 3) to evaluate many of the samples of the less dense right peak as anomalous (Model 5). Still always, the region around the highest mode of the distribution is considered to be normal. Also consider the samples in the lower right corner. While each model considers them almost randomly different, the ensemble averages out these variations, providing a much more consistent anomaly score than each singular model.

## VI. Evaluation

In this section, we thoroughly evaluate our method[3] on outlier detection performance, robustness, and runtime. We compare our method to the algorithms explained in Section

[3]An implementation of *DEAN* is available from https://github.com/psorus/DEAN.

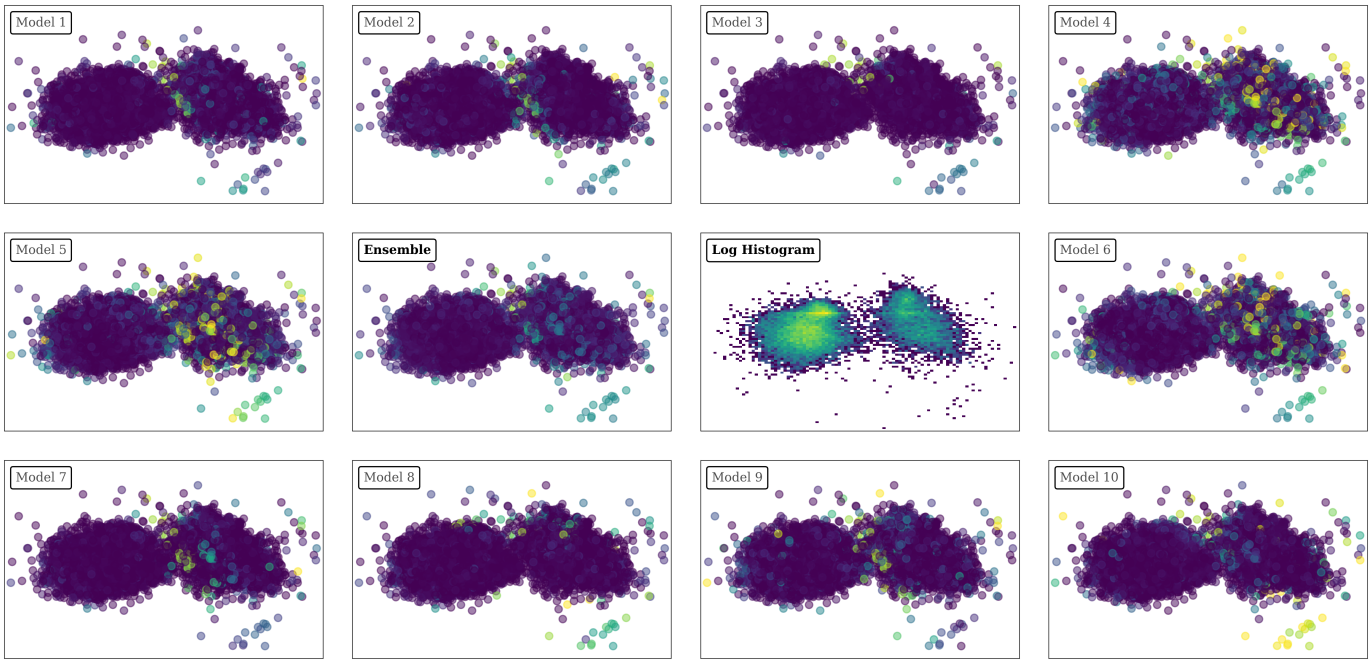


Fig. 4. An evaluation of a few *DEAN* models finding different normal samples of the same dataset more (green, yellow, blue) or less (magenta) anomalous. The ensemble (middle) averages out this randomness and follows the density distribution next to it.

III: *DeepSVDD*[4] [32], an *Autoencoder* [34], the *RandNet Autoencoder* ensemble [10] [5], and to *Isolation Forests* [6] [25]. For our version of *DEAN*, we show two versions: One using no learnable shifts and one using shifts as explained in Section IV-E.

To evaluate the outlier detection performance, we use the *ROC Area under Curve (AUC)* [6]. We generate 10 datasets from MNIST [13] images. Each dataset considers one digit normal, while each method tries to find every other digit as an outlier. Similarly, for our experiments on cifar10 [23], one group of images is considered normal, while the other groups are abnormal.

### A. Outlier Detection Performance

We show the ROC AUC score for all 10 MNIST Datasets and each model discussed before in Table II. Every experiment is repeated ten times, and the resulting AUC performance is averaged. Using this method, both our versions of *DEAN* outperform each comparison algorithm on every dataset considered. Seemingly complicated datasets like "2", "3" and "5" show some of the most significant performance improvements compared to the comparison algorithms. But also datasets with a simple anomaly detection task, like "1", "0" and "6" still show consistent improvement. This consistency of our method might be the most significant benefit of it: Each of the three alternative algorithms drops below 0.9 for multiple datasets while the dataset on which our ensemble performs worst ("8") still reaches a performance of $> 0.93$.

Additionally, our uncertainties are at least one order of magnitude smaller than each non-ensemble competitor and could likely be reduced further by increasing the number of models in our ensemble. We consider this helpful as outlier detection methods are usually trained without knowing all the outliers we would like them to find. This can make it complicated to differentiate between a well-trained model and a bad one; thus, a training procedure that consistently produces great models generally allows one to trust more in a given model.

These uncertainties are even drastically lower than those of the *Autoencoder* we use for preprocessing, showing that our ensemble averages out slight errors in our dataset.

There does not seem to be an important difference between *DEAN* and *DEAN incl Shift*, implying that both our solutions to trivial solutions, as discussed in Section IV-E, work. Still, for almost all datasets, the inclusion of shifts seems to help the performance a bit, and the average uncertainty is also even a bit lower. Because of this, we only mark the best results in our tables .

For cifar10 (Table III), we were not able to calculate performances with *RandNet*, as this does not scale well, requiring around 26 days of computation time per ensemble on our hardware.

Compared to the other algorithms, *DEAN* still performs best, but now there are three datasets on which it is outperformed. On the dataset "Ship", *DeepSVDD* performs better, while on the datasets "Bird" and "Deer", a simple *Isolation*

[4]For our experiments, we use the code and hyperparameters suggested in https://github.com/lukasruff/Deep-SVDD-PyTorch.

[5]We implement a version of *RandNet* here https://github.com/psorus/RandNet/

[6]We use an *Isolation Forest* given by sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

*Forest* seems to perform best.

Still, the errors of our algorithm are even lower, which we attribute to us no longer using preprocessing. Correspondingly, the difference between *DEAN* and *DEAN incl Shift* is now also completely neglectable (The biggest difference between both, is an AUC difference of $< 0.0003$ for the "Horse" dataset, which less than one uncertainty)

For the rest of this paper, we will show most other properties on MNIST data, as the generally higher performance gives them more practical usage.

### B. Runtime

The computational effort of our ensemble is a linear function of the number of models used and thus almost entirely arbitrary. Figure 2 shows that adding more models statistically increases the outlier detection performance. We can not be sure that this continues for arbitrarily many models, but we have looked at up to $50000$ models (MNIST), and there does not seem to be a fluctuation on the level of accuracy that we could measure (As the size of the MNIST test set implies differences $<< 0.0001$ are meaningless). This allows us to reduce the variance of our model by an arbitrary amount, but it also makes defining the runtime of our ensemble not trivial. We could show the runtime of a single model (This time would sometimes even outperform an *Isolation Forest*), but this does not represent the optimizations we suggest. Alternatively, we could show the runtime of an ensemble with a fixed number of models (For 500 models, the runtime would be 25% lower than those of *DeepSVDD*), but this scales with an arbitrary variable and is thus not an appropriate way of comparing times.

We choose to show the training time needed until our ensemble outperforms every alternative. This represents a good optimization without introducing any arbitrary parameter. For most datasets, this means considering $< 30$ models, but for datasets "8" and "9", the similar performance of *DeepSVDD* makes us require $60$ models to outperform it. These outliers would distort the average time, which is why we show in Figure 5 the median wallclock time needed for the training process instead. Because this calculation is impossible for most cifar10 datasets, we only give the runtime on MNIST data here. Similarly, it is impossible for the *RandNet* ensemble as their performance is too weak. We will choose here the lowest amount of models recommended by their original paper (20), while we will use the highest amount of models they recommend (300) for the quality comparison before. We hope this gives a lower bound for the runtime and an upper one for the quality for us to compare to. The bars in Figure 5 are sorted by outlier detection performance. Generally, we would expect a simpler algorithm to be faster and have a weaker performance: The more complicated and powerful a model is, the higher its computational effort is. The deep ensembles are the outlier in this comparison, we require less computational effort than *DeepSVDD* to reach the best performance overall, while *RandNet* requires drastically more time than we would have expected based on their performance.

The model runtime could still be optimized. We include preprocessing time for *DeepSVDD* and *DEAN*. In both algorithms, this time takes up about $\frac{2}{3}$ and $\frac{5}{6}$ of the time shown respectively, but in *DEAN*, this time is dominated by us suggesting the combination of 5 different autoencoder to reduce their variance. So by using fewer autoencoder, we could drastically reduce the time at the cost of some accuracy.

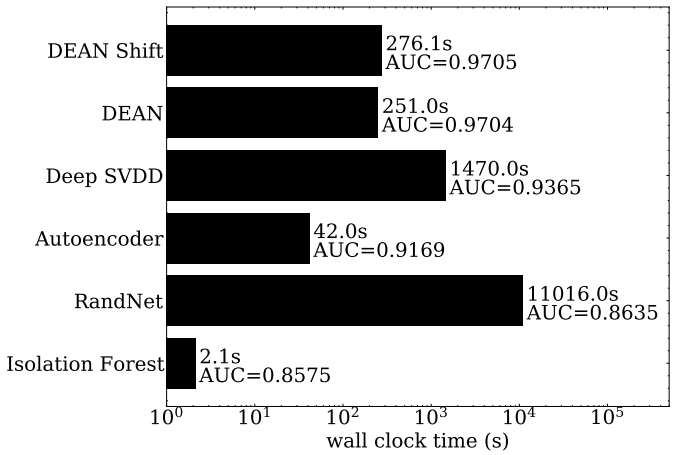


Fig. 5. The median wallclock training time of the models we used, sorted by outlier detection performance. Since this is hard to define for our ensemble, we use the minimum number of models needed to outperform each competitor.

We also give the median runtime per submodel in Table IV. Notice that both *Dean* and *Dean incl shift* outperform their competitors quite quickly, both only requiring about $21$ models. Still, the training cost for *Dean incl shift* is more than $50\%$ higher, thanks to the additional complexity we allow the models to learn.

Consider also, how the time to train each model is lower on the higher dimensional cifar10 data. This is quite counterintuitive, as the higher bagging size generates about nine times bigger neural networks. We believe this to be a product of the EarlyStopping we employ, and the lack of clear concepts present in cifar10 data.

### C. Ensemble Structure

Sections VI-A and VI-B showed why ensembles are an excellent choice for anomaly detection methods, and here we show why a simple loss function motivated in Section IV-A and especially our loss function introduced in Section IV-B produces anomaly detection models which are good choice to be combined into an ensemble. To do this, we show in Figure 6 the performance of using the same combination function (Equation 5) to combine up to 100 *DeepSVDD* models or *Autoencoder* into one ensemble and compare it to our proposed ensemble and *RandNet*. We do not show an ensemble of *Isolation Forest*s, as these are already based on ensembles.

The most significant benefit of the ensemble for the *Autoencoder* is the consistency, as their improvement flats out nearly immediately ($< 10$ models). It does this at a model performance not comparable to the other algorithms, while also requiring $20$ times more computation power than *DEAN*

TABLE II
OUTLIER DETECTION PERFORMANCE (AUC) FOR DIFFERENT METHODS ON MNIST DATA

| Dataset | DEAN incl Shift | DEAN | DeepSVDD | Autoencoder | RandNet | Isolation Forest |
|---|---|---|---|---|---|---|
| 0 | **0.9917** $\pm$ 0.0005 | 0.9900 $\pm$ 0.0007 | 0.9756 $\pm$ 0.0087 | 0.9669 $\pm$ 0.0169 | 0.9677 $\pm$ 0.0016 | 0.9656 $\pm$ 0.0071 |
| 1 | **0.9988** $\pm$ 0.0001 | 0.9985 $\pm$ 0.0007 | 0.9946 $\pm$ 0.0015 | 0.9963 $\pm$ 0.0012 | 0.9868 $\pm$ 0.0002 | 0.9938 $\pm$ 0.0009 |
| 2 | **0.9527** $\pm$ 0.0023 | 0.9524 $\pm$ 0.0020 | 0.8887 $\pm$ 0.0219 | 0.8689 $\pm$ 0.0121 | 0.7788 $\pm$ 0.0033 | 0.7518 $\pm$ 0.015 |
| 3 | **0.9625** $\pm$ 0.0007 | 0.9600 $\pm$ 0.0020 | 0.897 $\pm$ 0.0156 | 0.9030 $\pm$ 0.0081 | 0.8563 $\pm$ 0.0026 | 0.8351 $\pm$ 0.0092 |
| 4 | **0.9732** $\pm$ 0.0016 | 0.9704 $\pm$ 0.0014 | 0.937 $\pm$ 0.0127 | 0.8944 $\pm$ 0.0090 | 0.8784 $\pm$ 0.0016 | 0.8764 $\pm$ 0.0074 |
| 5 | **0.9660** $\pm$ 0.0037 | 0.9632 $\pm$ 0.0051 | 0.8667 $\pm$ 0.0216 | 0.9245 $\pm$ 0.0076 | 0.7189 $\pm$ 0.0029 | 0.7441 $\pm$ 0.0159 |
| 6 | **0.9867** $\pm$ 0.0012 | 0.9851 $\pm$ 0.0009 | 0.9783 $\pm$ 0.0068 | 0.9662 $\pm$ 0.0053 | 0.8694 $\pm$ 0.0025 | 0.8726 $\pm$ 0.0161 |
| 7 | **0.9743** $\pm$ 0.0012 | 0.9755 $\pm$ 0.0020 | 0.9437 $\pm$ 0.0095 | 0.9539 $\pm$ 0.0036 | 0.9111 $\pm$ 0.0005 | 0.9131 $\pm$ 0.0054 |
| 8 | 0.9314 $\pm$ 0.0030 | **0.9360** $\pm$ 0.0005 | 0.9223 $\pm$ 0.0137 | 0.7741 $\pm$ 0.0106 | 0.7892 $\pm$ 0.0032 | 0.7418 $\pm$ 0.0159 |
| 9 | **0.9680** $\pm$ 0.0022 | 0.9657 $\pm$ 0.0017 | 0.9602 $\pm$ 0.0068 | 0.9479 $\pm$ 0.0041 | 0.8788 $\pm$ 0.0008 | 0.8809 $\pm$ 0.0058 |
| Average | **0.9705** | 0.9704 | 0.9365 | 0.9169 | 0.8635 | 0.8575 |

TABLE III
OUTLIER DETECTION PERFORMANCE (AUC) FOR DIFFERENT METHODS ON CIFAR10 DATA

| Dataset | DEAN incl Shift | DEAN | DeepSVDD | Autoencoder | Isolation Forest |
|---|---|---|---|---|---|
| Airplane | 0.6863 $\pm$ 0.0003 | **0.6867** $\pm$ 0.0003 | 0.6061 $\pm$ 0.0138 | 0.5373 $\pm$ 0.0958 | 0.6609 $\pm$ 0.0130 |
| Automobile | **0.6267** $\pm$ 0.0001 | **0.6267** $\pm$ 0.0001 | 0.6145 $\pm$ 0.0073 | 0.4397 $\pm$ 0.0679 | 0.4371 $\pm$ 0.0083 |
| Bird | 0.5798 $\pm$ 0.0002 | 0.5798 $\pm$ 0.0002 | 0.491 $\pm$ 0.0047 | 0.6101 $\pm$ 0.0552 | **0.6446** $\pm$ 0.0051 |
| Cat | **0.6357** $\pm$ 0.0002 | 0.6356 $\pm$ 0.0003 | 0.5783 $\pm$ 0.0063 | 0.4876 $\pm$ 0.0110 | 0.5057 $\pm$ 0.0069 |
| Deer | 0.6876 $\pm$ 0.0002 | **0.6879** $\pm$ 0.0002 | 0.5687 $\pm$ 0.012 | 0.6291 $\pm$ 0.0334 | **0.7462** $\pm$ 0.0029 |
| Dog | **0.6299** $\pm$ 0.0004 | 0.6298 $\pm$ 0.0004 | 0.623 $\pm$ 0.012 | 0.4950 $\pm$ 0.0445 | 0.5172 $\pm$ 0.0076 |
| Frog | **0.7292** $\pm$ 0.0002 | **0.7292** $\pm$ 0.001 | 0.5956 $\pm$ 0.0179 | 0.6004 $\pm$ 0.0311 | 0.7114 $\pm$ 0.0062 |
| Horse | **0.6291** $\pm$ 0.0003 | 0.6288 $\pm$ 0.0003 | 0.6144 $\pm$ 0.0183 | 0.4814 $\pm$ 0.0158 | 0.5355 $\pm$ 0.0060 |
| Ship | 0.7305 $\pm$ 0.0002 | 0.7303 $\pm$ 0.0002 | **0.7726** $\pm$ 0.0108 | 0.5295 $\pm$ 0.0965 | 0.6995 $\pm$ 0.0108 |
| Truck | 0.6989 $\pm$ 0.0003 | **0.6993** $\pm$ 0.0004 | 0.6854 $\pm$ 0.0173 | 0.4322 $\pm$ 0.0475 | 0.5396 $\pm$ 0.0124 |
| Average | **0.6634** | **0.6634** | 0.6150 | 0.5242 | 0.5998 |

TABLE IV
RUNTIME COMPARISON OF SINGLE *DEAN* SUBMODELS

| | Dean | Dean incl shift | Randnet |
|---|---|---|---|
| Time/Model | 1.84$s$ | 2.92$s$ | 550.82$s$ |
| Models needed | 21 | 21 | |
| time cost | 41.26$s$ | 66.29$s$ | |
| Time/Model cifar | 1.58$s$ | 1.88$s$ | |

per submodel. This low improvement of the combination of multiple models is a clear sign of lacking submodel variance.

This effect is less strong for *DeepSVDD*, and so it improves through the ensemble structure a bit more, but also flats out at a low number of models ( 30). Between 50 and 70 models, there is even a region where adding more models decreases the ensemble quality. We think this is because each model is weighted arbitrarily by the random choice of $\vec{c}$. This implies that adding more models will not improve the quality even further and means that the variance of these models falls slower as a function of the number of models. Also, the final quality is worse than those from *DEAN*, while this ensemble requires about 800 times more computational effort than ours.

*RandNet* seems to behave quite similarly to the direct Autoencoder ensemble, also flatting out quickly. So it seems their approach to increasing submodel variance is ineffective. And since the overall quality of *RandNet* also seems to be worse, we can not recommend it for our applications.

Our model performs best on the shown dataset while requiring the lowest time. Because of different axis limits, you see more variation here than in Figure 2, highlighting that our model flats out a lot less than the other curves and showing that our approaches to high submodel variance worked.

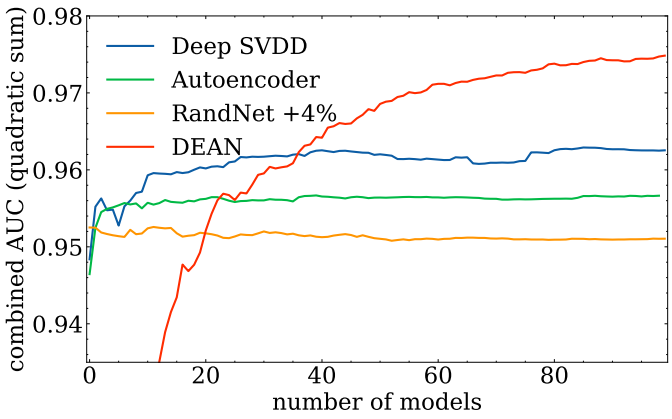


Fig. 6. Quality of ensembles of other models in dataset "7" compared to *DEAN*. To compare the quality of *RandNet* we had to add 4% AUC to it.

### D. Hyperparameter invariance

In Section VI-A we have already shown that our ensemble is much more consistent and reliable than alternative models. This is helpful as there is often no way to evaluate the strength of an outlier detection algorithm. We would like to have outlier detection algorithms with fewer hyperparameters for the same reason. Since our method is based on deep learning, we have some hyperparameters that define our neural network: depth, width, initialization, activation, batch size, and learning rate.

Also, we have some variables that change the behavior of our ensemble, namely the number of models used and the number of features used in each model. Figure 2 shows that the number of models used is easily optimized even in an unsupervised manner: More models generally only increase the quality. For the rest of the hyperparameters, we find that a variation of them does not change the quality much in reasonable parameter regions[7]. For example, in Figure 7, we show the ensemble outlier detection quality for 7 different submodel depths, while only showing the expected performance drop from using shallow networks with less than 2 hidden layers: Choosing a different depth for our networks does not change the performance much.

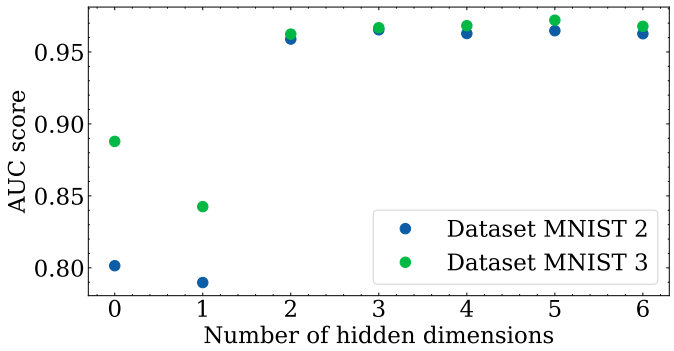


Fig. 7. Ensemble outlier detection quality on two MNIST datasets "2" and "3" as a function of the number of hidden dimensions of each model used. At least two hidden dimensions are needed, after which the quality stays almost constant.

### E. Interpretability

While earlier sections show the highly competitive anomaly detection performance of *DEAN*, the performance comes at the price of millions of parameters describing the ensemble, resulting in an apparently very un-interpretable method. In this section, we want to show that this is not the case since we can use the ensemble structure itself to create a novel kind of interpretability.

This interpretability method uses the variance of the prediction of each model and correlates it to the features used for the same model. Given the anomaly score each model would predict on its own $p_i(x),\ i < N$ and the set of features considered by the same model $F_i$ we think of a feature $f$ as important if using it changes the prediction of the ensemble.

$$\frac{1}{N}\cdot\sum_i^N p_i(x) \neq I^f \equiv \frac{1}{|S^f|}\cdot\sum_i S_i^f(x)\ with\ S^f = \{p_i,\ f \in F_i\} \tag{6}$$

To generate a meaningful score $I^f$ the size $|S^f|$ needs to be much larger than the bagging size (Equation 7), as else the randomness of the feature bagging selection dominates the score: Imagine a single feature is quite important. It will be combined into models with $bag-1$ other features in each model considering it, who all get the same high contribution to $I^f$. If every feature is now only chosen $\sim 1$ time, $bag-1$ random features also have high importance.

$$|S^f| \gg |F_i| = bag \tag{7}$$

As $\sum_f |S^f| = N\cdot bag$, this condition is independent of $bag$: We require more models than features.

$$N \gg number_{features} \tag{8}$$

Still experimentally it seems like a lower $bag$ contributes to clearer images, while a minimum value of $bag$ is required to capture complicated relations ($bag = 1$ would ignore any correlations between features).

This interpretability score $I^f$ is theoretically applicable to each anomaly detection ensemble with feature bagging, but the high number of models required and their comparability make it especially suitable to *DEAN*.

We test this here on two image-based datasets, as image-based data allows for easy visualization and thus evaluation of our method.

### F. MNIST

First, we look at the $I^f$ for an MNIST-trained ensemble of 50000 models. We choose here the consider the digit 7 to be normal, and thus we want to see deviations and similarities to the expectation of a 7 highlighted. We also choose a bagging size $bag = 32 < 128$ lower than in earlier experiments, as this seems to be enough to generate good explanations, while seemingly generating nicer images. Since we require many models, we also choose not to include shifts to save computation effort.

We show 5 different MNIST images together with their explanations in Figure 8. Their explanations $I^f$ can be clustered into three groups:

1) Dark pixels (low $I^f$) have on average a lower anomaly score and can thus be considered expected. In our examples here these seem to be the pixels that our input image shares with a similar 7 (see the left-most or the middle image). Please notice how these similar 7 are not the same in each explanation, but depend on the input (see the left two).
2) Bright pixels (high $I^f$) have on average a higher anomaly score, and thus we can consider them the reason these images (except for the right-most) are flagged as anomalies. In these examples they seem to be pixels we should change to generate a 7: Considering for example the left-most image, this 2 becomes a 7 when we would remove the horizontal line at the bottom. And if we look at the second-most left image, this 4 becomes a 7 if we remove the horizontal line in the middle, and add it again at the top.
3) Still most pixels have an almost random behavior. This is because there is no contribution from these pixels, and so the average anomaly score is only generated by the features it is paired with. As this noise shrinks with the

[7]A model depth below 3, an activation like sigmoid or an unreasonable feature bagging size like 1 all decrease the model performance for apparent reasons.

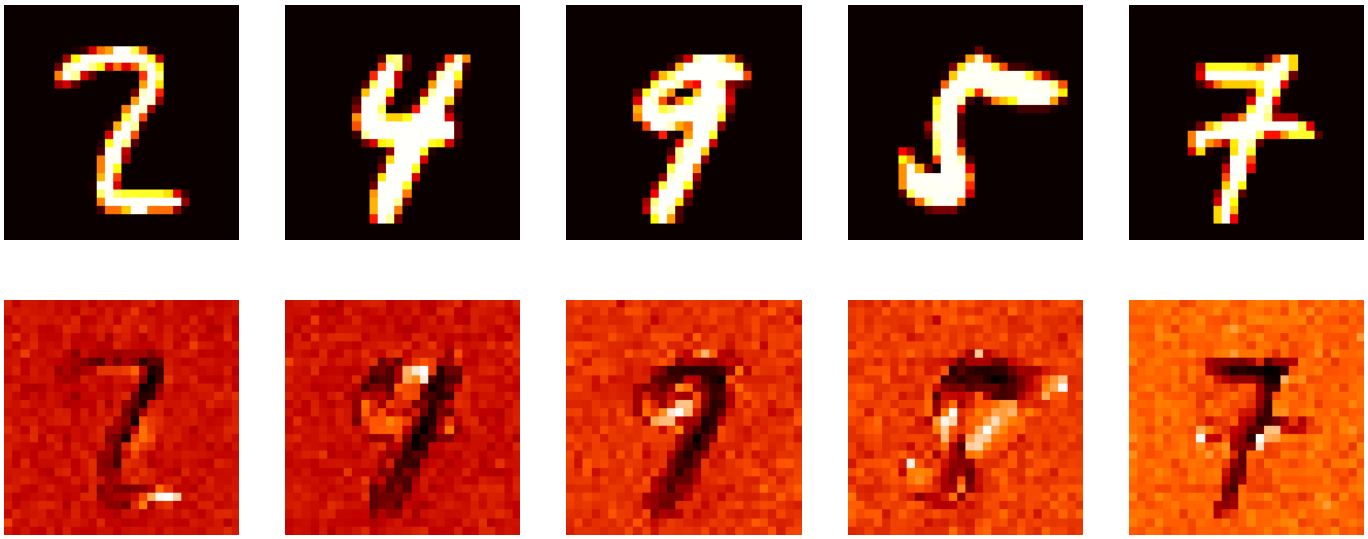

Fig. 8. 5 MNIST test Images on the top and the bottom showing the non "7" reason (the expectation) for each. 1: A "2" is similar to a 7 with another line at the bottom. This line is found as abnormal while the rest is considered similar to a "7". 2: Also, missing Values can be an anomaly. 3: Most of the image is similar to a 7, just one region is not expected. 4: Even significant differences to the expected value can be well explained. 5: This sample is considered normal. Still, the ensemble does not expect the second horizontal line. The background color is slightly different for each sample, as the color scheme is dominated by the high values of $I^f$ of the sample.

number of models we train, we use the appearance of structures as a good indicator for when the number of models is high enough. One also sees some structures when training only $O(1000)$ models, but the randomness of the surrounding pixels makes it harder to interpret.

Notice that these explanations don't only match our understanding of what should be considered anomalous, but tend to generally find less likely situations. The right-most image shows how a 7 with two horizontal lines can be considered more normal, by removing the middle horizontal line. To show that we did not only cherry-pick the nicest images, but you can also find explanations like this for the entire MNIST test set here[8].

### G. CelebA

While our explanations for MNIST are quite easy to interpret, this is also a fairly simple example, since every pixel is (almost) binary, most pixels are not used and the images are quite small. For this reason, we want to try to use our method to explain a more complicated dataset. We choose here a subset of the celebA dataset [26].

The main problem with bigger image-based datasets is that there are many different concepts to learn in a complicated dataset. And when we expect a model to only find a certain concept of what an anomaly is, this is only one of the tasks that the model has to understand during training to understand this dataset. And since these concepts can be quite abstract, to the point where they are also fulfilled on the anomalous samples the anomaly detection can become confused and thus the performance drops. We see this happen in Chapter VI-A: Even though our performance on cifar10 is competitive, a ROC-AUC score of $0.7$ is still not very good. Here we choose celebA not only because of the high number of features and samples but also because every image is taken in a very similar position (limiting the number of abstract hidden concepts) and the attributes allow us to restrict it further.

We choose here to consider images with the attributes "female", "attractive", "not bald" and "young" as normal, while inverting these filters for our anomalies. We choose these features to maximize the number of training images ($68531$), while also increasing the similarity between images[9]. To simplify the evaluation, we convert our images to grey scale, but we keep the number of pixels intact ($218 \times 178 = 38804$ pixels) to not change the complexity of our explanation task.

Since equation 8 suggests that many features require more models to generate images of similar quality, we choose here to train and combine the output of $500000$ models. Since it also requires a high training time ($O(10)s$ per model) we are not able to optimize $bag$ anymore, but we assume that the more complicated structure warrants using a higher bagging size of $bag = 128$.

We show the results of our explanations in Figure 9. Notice first that we see the same three groups of pixels and can thus interpret them easily. Also, notice how there seems to be a reconstruction of (at least some) facial features in each explanation.

We will try to interpret the features we see, but as we generate many explanations (one for each pixel), this can become a lot more subjective than our interpretation of MNIST samples.

[8]https://tu-dortmund.sciebo.de/s/nLKnv7jKQ7VOuIX

[9]The license of celebA prohibits us from sharing our data selection, but you can easily reproduce it. Sadly it also does not allow us to share all our generated explanations.

Fig. 9. Image and Explanation for 5 celebA images. The three leftmost images are considered normal, while the two images on the right represent anomalies.

Since we constructed our task in the hope that the network would find seemingly feminine features, we search for examples in these images. One might be the earring in the second image from the left.

Still, our network is conditioned to find statistically rare features, and as these are complicated images, these don't have to be feminine. Notice for example the glasses in the last image from the left: As most images don't contain glasses, these are considered strange and thus highlighted. Another example might be the strong wrinkle in the second to last image from the left, becoming the most unexpected feature of that image. This can also contradict the feminine features we expect: While bangs are more commonly a feminine feature ($20\%$ of women, but only $8\%$ of the men in celebA have bangs), they are still rare and thus considered anomalous in the second image.

Finally, there seems to be a bias towards bright hair colors. While the blond hair in the first image from the left is considered normal, the darker hair color in the second image is generally anomalous. And also in the third image, the parts of the hair seemingly bright are considered normal, while the rest is anomalous. This bias is problematic as it limits our anomaly detection performance by being a concept orthogonal to our desired definition of anomaly. But even more critically it also seems to extend to skin color (see the second image from the left). Importantly this is less an effect of our anomaly detection method, but an effect of the unbalanced distribution of skin colors of the celebA dataset. And using our interpretability we are able to detect such learned concepts to hopefully correct them.

Overall we think that even in very high dimensional data, these are quite intuitive reasons our method can extract, even when they don't necessarily fulfill the anomaly detection task. As our method also produced valuable insights on normal images, we think that it can contribute to solving the task of anomaly detection for very complicated datasets, by offering insights into the concepts learned. This could allow data augmentation to control them, and thus for anomaly detection to better follow the desired behavior.

## VII. Conclusion

In this work, we defined the benefits of using a simple anomaly detection model in an ensemble. We also suggested a new anomaly detection method that satisfies our conditions and works well in an ensemble. We have shown its effectiveness and efficiency on finding anomalies in the MNIST and cifar10 datasets, especially its robustness and simplicity. We have also used the way we define our ensembles to introduce a new method for explaining the reason for a given anomaly, resulting in arbitrarily clear high-level reasons being generated.

In future work, we believe that more advanced ideas from the ensemble community can be used for deep learning and, in particular, for unsupervised anomaly detection. We think that methods exploiting the ensemble structure, like the continuously retraining of single trees used to combat feature drift [37], could also be extended to *DEAN*. Similarly, also other ensemble methods like pruning [30] could extend our method. Also, more advanced combination functions and query strategies could improve the efficiency of our interpretability method.

## REFERENCES


[1] Charu C. Aggarwal. *Outlier Analysis*. Springer, 2013.

[2] Charu C. Aggarwal. Outlier ensembles: Position paper. *SIGKDD Explor. Newsl.*, 14(2):49–58, apr 2013.

[3] Paul Boniol, Michele Linardi, Federico Roncallo, and Themis Palpanas. Automated anomaly detection in large sequences. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1834–1837, 2020.

[4] Abdenour Bounsiar and Michael G. Madden. One-class support vector machines revisited. In *2014 International Conference on Information Science Applications (ICISA)*, pages 1–4, 2014.

[5] Jiayu Zhou Boyang Liu, Pang-Ning Tan. Unsupervised anomaly detection by robust density estimation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36, 2022.

[6] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[7] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[8] Markus Breunig, Hans-Peter Kriegel, Raymond Ng, and Joerg Sander. Lof: Identifying density-based local outliers. *ACM Sigmod Record*, 29:93–104, 06 2000.

[9] Christian Callegari, Loris Gazzarrini, S. Giordano, Michele Pagano, and Teresa Pepe. A novel pca-based network anomaly detection. In *IEEE International Conference on Communications*, pages 1 – 5, 07 2011.

[10] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga. Outlier detection with autoencoder ensembles. In *SIAM International Conference on Data Mining*, pages 90–98, 06 2017.

[11] Ziheng Chen and Hongshik Ahn. Item response theory based ensemble in machine learning. In *International Journal of Automation and Computing*, 2019.

[12] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations*, 2016.

[13] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[14] Yoav Freund and Robert E. Schapire. A short introduction to boosting. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406. Morgan Kaufmann, 1999.

[15] Adam Goodge, Bryan Hooi, See Kiong Ng, and Wee Siong Ng. Ares: Locally adaptive reconstruction-based anomaly scoring. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2022.

[16] Xiaoyi Gu, Leman Akoglu, and Alessandro Rinaldo. Statistical analysis of nearest neighbor methods for anomaly detection. In *Conference on Neural Information Processing Systems*, 2019.

[17] Xu Han, Xiaohui Chen, and Li-Ping Liu. Gan ensemble for anomaly detection. In *AAAI Conference on Artificial Intelligence*, 2020.

[18] Sahand Hariri, Matias Carrasco Kind, and Robert J. Brunner. Extended isolation forest. *CoRR*, abs/1811.02141, 2018.

[19] Renjun Hu, Charu C. Aggarwal, Shuai Ma, and Jinpeng Huai. An embedding approach to anomaly detection. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 385–396, 2016.

[20] Tomoharu Iwata and Makoto Yamada. Multi-view anomaly detection via robust probabilistic latent variable models. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[21] Fabian Keller, Emmanuel Muller, and Klemens Bohm. Hics: High contrast subspaces for density-based outlier ranking. *2012 IEEE 28th International Conference on Data Engineering*, pages 1037–1048, 2012.

[22] Edwin Knorr, Raymond Ng, and Vladimir Tucakov. Distance-based outliers: Algorithms and applications. *The VLDB Journal*, 8:237–253, 02 2000.

[23] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[24] Aleksandar Lazarevic and Vipin Kumar. Feature bagging for outlier detection. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 21, pages 157–166, 01 2005.

[25] Fei Tony Liu, Kai Ting, and Zhi-Hua Zhou. Isolation forest. In *International Conference on Data Mining*, pages 413 – 422, 01 2009.

[26] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[27] Viktor Malyi. Run or walk dataset, Jul 2017.

[28] Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. A survey on gans for anomaly detection, 2021.

[29] Lutz Prechelt. Early stopping — but when? In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade: Second Edition*, pages 53–67, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[30] Shebuti Rayana and Leman Akoglu. Less is more: Building selective anomaly ensembles. *ACM Trans. Knowl. Discov. Data*, 10(4), may 2016.

[31] Lukas Ruff, Jacob R. Kauffmann, Robert A. Vandermeulen, Gregoire Montavon, Wojciech Samek, Marius Kloft, Thomas G. Dietterich, and Klaus-Robert Muller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, May 2021.

[32] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402. PMLR, 10–15 Jul 2018.

[33] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In David E. Rumelhart and James L. Mcclelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[34] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, page 4–11, New York, NY, USA, 2014. Association for Computing Machinery.

[35] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. Discovery-driven exploration of olap data cubes. In *Proceedings of the 6th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '98, page 168–182, Berlin, Heidelberg, 1998. Springer-Verlag.

[36] Hanyu Song, Peizhao Li, and Hongfu Liu. Deep clustering based fair outlier detection. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1481–1489. ACM, 2021.

[37] Swee Chuan Tan, Kai Ming Ting, and Fei Tony Liu. Fast anomaly detection for streaming data. In *International Joint Conference on Artificial Intelligence*, 2011.

[38] Ta-Wei Tang, Wei-Han Kuo, Jauh-Hsiang Lan, Chien-Fang Ding, Hakiem Hsu, and Hong-Tsu Young. Anomaly detection neural network with dual auto-encoders gan and its industrial inspection applications. *Sensors*, 20(12), 2020.

[39] David M.J. Tax and Robert P.W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, Jan 2004.

[40] Arthur Vervaet. Monilog: An automated log-based anomaly detection system for cloud computing infrastructures. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2739–2743, 2021.

[41] Arthur Zimek, Ricardo J.G.B. Campello, and Jörg Sander. Ensembles for unsupervised outlier detection: Challenges and research questions a position paper. *SIGKDD Explor. Newsl.*, 15(1):11–22, mar 2014.

[42] Arthur Zimek, Matthew Gaudet, Ricardo J.G.B. Campello, and Jörg Sander. Subsampling for efficient and effective unsupervised outlier detection ensembles. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, page 428–436, New York, NY, USA, 2013. Association for Computing Machinery.