Fall 2021

Twin Anomaly Detection System

Paaras Chand
San Jose State University

Twin Anomaly Detection System

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Paaras Chand

December 2021

The Designated Project Committee Approves the Project Titled

Twin Anomaly Detection System

by

Paaras Chand

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2021

Teng-Sheng Moh, Ph.D.	Department of Computer Science
Melody Moh, Ph.D.	Department of Computer Science
Nada Attar, Ph.D.	Department of Computer Science

ABSTRACT

TWIN ANOMALY DETECTION SYSTEM

by Paaras Chand

Anomaly detection performs well in situations where signature (and other rule-based) methods fail; there is no need to identify every threat as long as it is different from the norm. The tradeoff is that anomaly detection often results in a large number of false positives. While previous work has capitalized on the data imbalance problem to train models with only one set of data (one-class classification), few have utilized the limiting set for anything other than testing purposes. This paper seeks to utilize two anomaly detectors: one that is trained on the positive set and one that is trained on the negative set. By utilizing multiple detectors, we can encode more information about each class and ensure that a data point is not only different from one class, but also similar to the other. We present a new approach to anomaly detection and show its effectiveness at reducing false positives with limited effect on detection rates.

ACKNOWLEDGMENTS

I would like to thank my advisor, Teng Moh, for teaching me so much and encouraging me to tackle difficult problems. I look forward to our Wednesday research group meetings and am saddened that they may soon end.

I would also like to thank Melody Moh and Nada Attar for serving on my committee and providing feedback and continued support.

I have enjoyed my time at San Jose State and am thankful to have met such talented and caring people.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

An anomaly is defined as a point which deviates from the norm. While outliers may simply be noise or hold little value, an anomaly presents useful information. Anomaly detection is focused on identifying and alerting users of such events. It has been used in several applications including malware analysis, intrusion detection, and medical imaging.

The premise behind these applications generally involves establishing a notion of normal behavior by analyzing a single class of data points (the normal set). Anything that does not belong to this class is labeled as an anomaly. While intuitively simple, this type of model is well-suited for situations where the anomalous class is unknown or limited in size (class imbalance). Because the model is only trained on normal data, there is no need to find data points which define the other classes. As long as there is a distinction between normal and non-normal behavior, anomaly detection methods can perform well. In fact, they have achieved state of the art performance for several tasks [1].

The problem with this approach is that it tends to lead to a large number of false positives (normal points incorrectly labeled as anomalous). While detection rates (i.e. normal points labeled as normal; true positives) are generally high, several applications are much more sensitive to false positives. For example, malware detection applications are more likely to be disabled if they flag benign files as malware, even if they are able to detect all known malware. A system with a lower detection rate (i.e. a system which misses some forms of malware) may actually be preferred if the system can guarantee fewer false alarms (i.e. benign files labeled as malware).

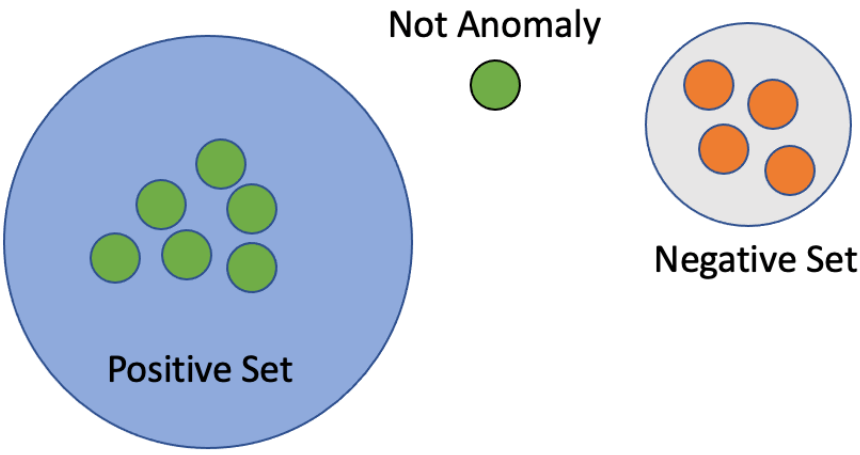

Fig. 1. Graphical representation of the multi-detector approach; two detectors can better identify true anomalies

2 LITERATURE REVIEW

Anomalies are generally rare occurrences. This makes it difficult to use supervised learning methods which require labeled instances of anomalies. Wang et al. have proposed a framework which utilizes oversampling (on the limited number of anomalies) and under sampling (on the normal data) [2]. The downside to such an approach is that it assumes that all anomalies can be represented by the samples in the dataset. Unfortunately, this is rarely the case. Anomalies have a broad range; they need only be different from the normal to qualify. Quantifying all possible outcomes is difficult and may lead to vulnerabilities in certain models. For example, an intrusion detection system trained on a few types of anomalous behavior may be susceptible to a zero-day attack.

One of the simplest solutions to establishing a sense of the normal is by observing the distribution of the data points. Coincidentally a Gaussian distribution is actually called a *normal* distribution; anything that falls outside of the bell curve is an anomaly. Several machine learning models assume that the data follows a Gaussian distribution, but this is not so true in real life. Dias et al. proposed utilizing normalizing flows to transform a simple probability distribution into a more complex one and computing an anomaly score based on the log-likelihood [3]. While useful and even computationally efficient, few anomaly detection systems have utilized normalizing flows to improve performance, preferring to focus on other parameters or data features instead.

Vasheghani et al. proposed a system to identify anomalous patterns by clustering normal patterns, classifying new points with LDA, and using a Markov chain (probabilistic model) to determine if the sequence of patterns was anomalous [12]. The clustering is dependent upon k-means and the user knowing the number of patterns beforehand. It requires the data to be cyclic and may not work as effectively on complicated patterns.

Distance-based methods like k-nearest neighbor have also been used to identify anomalies. The problem is often in choosing the right value for k. Liu et al. proposed to use the natural neighborhood graph to eliminate the need to select the parameter [13]. Utilizing a system of weights and overlapping metrics, this solution allows for the selection of the "natural" neighbor. The idea is that normal data points will only slightly shift the center of the cluster, but anomalies will cause a significant change. The proposed solution relies on labeled data, but it is possible to craft a solution which does not. Choosing between the two depends on the available data.

Disentanglement learning attempts to "disentangle" the input into a select number of variables and encode them in a higher dimension. It is similar to high level vs low level of understanding. Higher level features involve general descriptions like man or woman; low level features describe details like hair color or height. Wang et al. built upon this concept by adding small changes to the low level of understanding and measuring the change at the high level [14]. The thinking was that changes in the low level would yield negligible changes for normal data points but high reconstruction error for anomalous points. This method utilizes variational autoencoders and was quite successful at identifying anomalies. The downsides are that the variations may lead to some false positives (but higher overall accuracy).

Most machine learning models need a lot of training data. One-shot learning attempts to learn from just a few data points. Instead of classifying different objects, this method measures the similarity between two objects. Ullah et al. utilized two CNNs (same weights, same parameters) and measured the similarity between the outputs [15]. This dual network system is known as a Siamese network and works well in situations where limited data is available but performs poorly in other situations. The biggest disadvantage is that the similarity score can have a large range, limiting the model's ability to correctly distinguish between anomalous and normal behavior.

Pang et al. developed a deviation network which utilizes a similar method to one-shot learning, except the network is only trained on anomalies and scored for "deviation" [16]. Normal data will tend to exhibit a similar pattern while anomalies will "deviate." The downside to this approach is that the network can only distinguish anomalies which are similar to the training set. It cannot identify new attacks and will fail when all points are anomalous because everything will deviate from the norm.

Isolation forests attempt to identify anomalies by exploiting the fact they are few and different. By creating a tree, the anomalies should be visible as the shortest paths, while normal values will have longer paths. This works a little bit differently from the other methods in that you don't need to explicitly define the normal; the anomalies will appear due to the tree structure.

Kumar et al. proposed a system to model unknown GDP data by removing anomalous data points from known countries [22]. The solution employed transfer learning to relate an unknown country with a similar one. The results were rather unusual in that the predicted value was almost identical to the others. This is likely due to the fact that the anomalies were removed prior to prediction. A better solution would be to remove the anomalies after prediction.

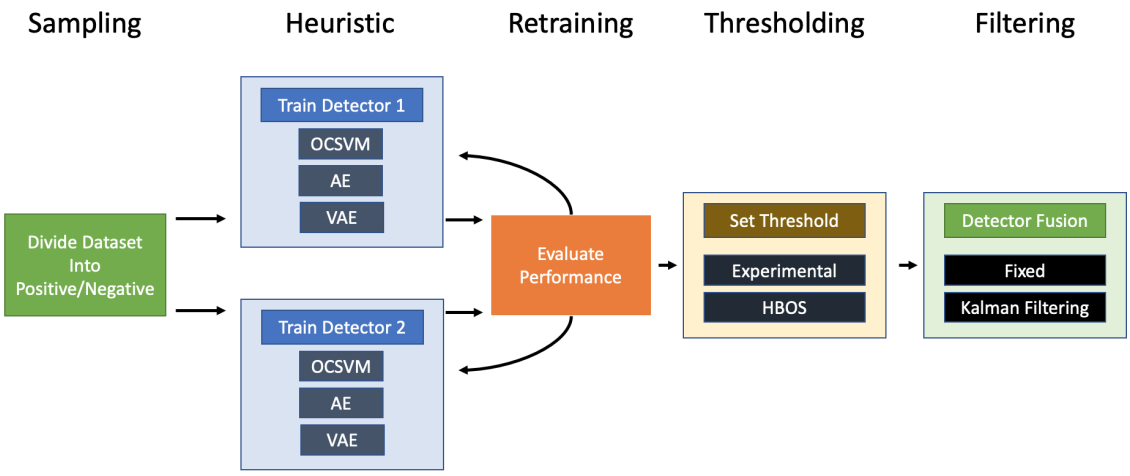

Fig. 2. Proposed Model Architecture

3 PROPOSED METHOD

In an attempt to reduce false positives, we propose a novel approach which relies on two anomaly detectors. The first detector is trained on only normal data (the positive set), similar to existing methods. The second detector is trained on known anomalous data points (the negative set). Before labeling a point as anomalous, both detectors must agree. For example, in the case of malware, detector one (D1) is trained on benign files and detector two (D2) is trained on malware files. In order for a file to be labeled malware it must be anomalous to the benign detector and normal to the malware detector. In the event that the detectors do not agree (e.g. D1 claims a file is malware while D2 claims it is benign), we generally err on the side of benign. This is done in an attempt to exploit the anomaly detector's high rate of false positives. Consequently, a detector's normal evaluation is generally more reliable than its non-normal rating. See table 1 for a complete listing of possible options.

It is important to note that while the tie-breaking mechanism is straight-forward, the actual distinction between normal and not-normal states is dependent upon some

Table 1
Truth Table for Detector Outcomes

Detector 1 *Positive Set*	**Detector 2** *Negative Set*	**Output**
Normal - Benign	Not Normal - Benign	**Benign**
Not Normal - Malware	Normal - Malware	**Malware**
Normal - Benign	Normal - Malware	**Benign**
Not Normal - Malware	Not Normal - Benign	**Benign**

threshold. The output of the detector is generally some form of reconstruction error. Determining which level of error distinguishes two classes from one another is usually done experimentally (based on training data).

In this paper we compare the performance of three different detector architectures (One-Class SVM, Autoencoder, and Variational Autoencoder) on three different types of datasets (malware, network data, and images). We build upon our baseline results by providing heuristics for model selection and data types in section VI, explore different techniques to handle class imbalance in section VII, evaluate different thresholding options in section VIII, and provide a filtering technique to better handle output from multiple detectors in section IX.

3.1 One-Class SVM

Support Vector Machines (SVM) distinguish between different classes of data by finding the hyperplane which maximizes the margin between all classes. This method works by mapping the input from the feature space to a higher-dimensional space where it is easier to find a separating hyperplane. The tradeoff to moving to a higher-dimensional space is the increased computational and space complexity. Fortunately, the kernel trick allows us to work in the higher-dimensional space without paying for the higher mathematical cost. By selecting the appropriate kernel, we can model complex shape boundaries, all without actually computing the (costly) transformed coordinates.

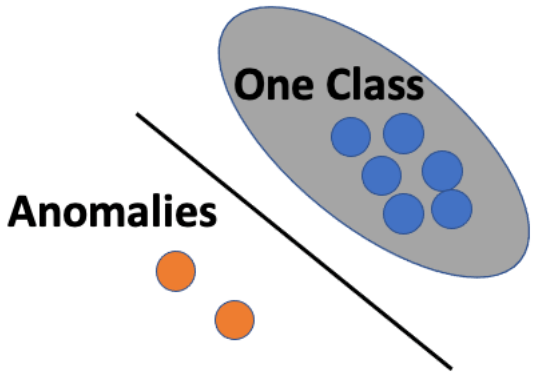

Fig. 3. Graphical representation of OCSVM. All normal (blue) data points are in one class, anything outside of boundary is an anomaly.

One-Class SVM (OCSVM) operates in much the same way as a regular SVM, except there is only one class; anything which doesn't belong to this class is labeled anomalous. The separating hyperplane attempts to maximize the margin between the origin and the data points. This approach was pioneered by Scholkopf and is mathematically described as [24]:

$$\begin{aligned} &\min_{\mathbf{w},\rho,\varepsilon} \tfrac{1}{2}\|\mathbf{w}\|^2 - \rho + \tfrac{1}{vn}\textstyle\sum_{i=1}^{n} \varepsilon_i \\ &\text{s.t. } < \mathbf{w}, \varphi(\mathbf{x_i}) > \geq \rho - \varepsilon_i, \varepsilon_i \geq 0, \forall i \end{aligned} \tag{1}$$

Another variation on one-class SVMs is known as Support Vector Data Description (SVDD). While OCSVM attempts to produce a hyperplane, SVDD attempts to produce a hypersphere. The goal is to find the smallest hypersphere which contains all of the normal data points; anything outside of the sphere is anomalous. The minimization function for this approach can be described as [25]:

$$\begin{aligned} &\min_{\mathbf{c},R,\varepsilon} R^2 + \frac{1}{vn}\sum_{i=1}^{n} \varepsilon_i \\ &s.t. \|\phi(\mathbf{x_i}) - \mathbf{c}\|^2 \leq R^2 + \varepsilon_i, \varepsilon_i \geq 0, \forall i \end{aligned} \tag{2}$$

In both OCSVM and SVDD, user parameters can relax the constraints of the hyperplane or hypersphere. For example, a smaller hypersphere can generally be used if the hypersphere need not contain all points (allowing for some false positives). A larger hypershphere would contain more points, some of which may be false negatives.

3.2 Autoencoders

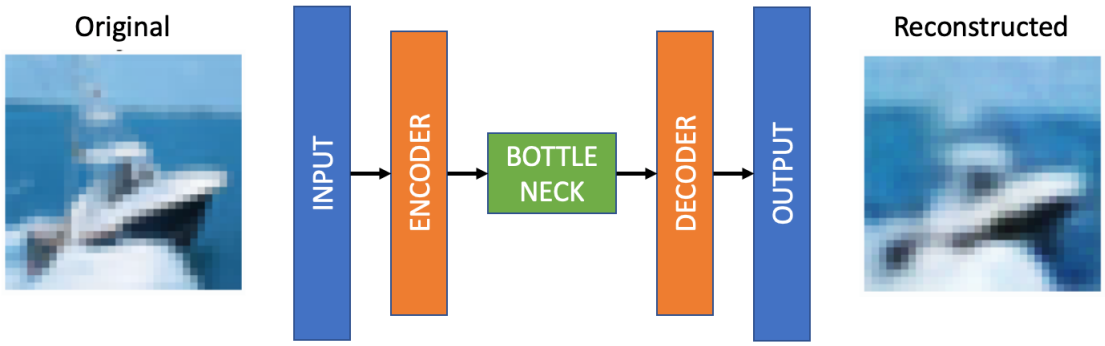

Fig. 4. Autoencoder architecture with example input and output

Autoencoders describe a neural network architecture where a bottleneck is introduced to force the network into compressing the input in the latent space. The network then attempts to reconstruct the input from the compressed representation. A well-trained network should be able to produce an output which is similar to the input. To achieve this goal, we calculate the reconstruction error as the absolute value of the difference between the output and the input. The training process attempts to reduce this reconstruction error.

We can also use the reconstruction error to differentiate between normal and anomalous points. This can be done by setting a threshold during training. For example, if most normal training data points have a reconstruction error of 0.4, any points with a reconstruction error over 0.4 would be labeled as anomalous.

The physical architecture of an autoencoder generally follows the encoder-bottleneck-decoder format, but there is no limiting description for the encoder or

decoder blocks. They can be composed of regular multi-layer perceptrons (MLP), convolutional neural networks (CNN), recurrent neural networks (RNN/LSTM/GRU), or any other type of architecture. The best choice will depend upon the data and application.

3.3 Variational Autoencoders

Variational autoencoders (VAE) are generative models which apply many of the same underlying principles as traditional autoencoders (AE). A limitation of AEs is that the latent space is deterministic. VAEs map to a latent space which is continuous. This means that instead of simply mapping to a value, the latent space maps to a distribution.

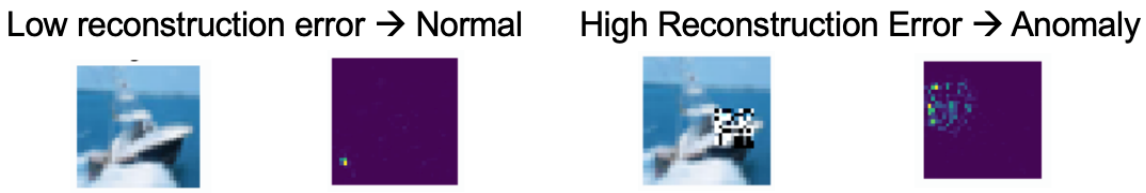

Fig. 5. Reconstruction error in images

While AEs attempt to reduce only reconstruction error, VAEs attempt to reduce both reconstruction error and latent loss. This stochastic representation allows us to utilize not only a fixed value, but also a probability for scoring the output. Instead of simply using a fixed threshold, we could utilize statistical values like the p-value to make the final determination. In most cases, VAEs tend to perform better than AEs [6]. The loss function can be described as:

$$\begin{aligned} \log p(x) &\geq \log p(x) - \mathrm{KL}\left(q_\phi(z \mid x) \| p(z \mid x)\right) \\ &= \mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x \mid z) - \mathrm{KL}\left(q_\phi(z \mid x) \| p(z)\right) \end{aligned} \tag{3}$$

3.4 Parameters

For OCSVM, we tested 4 kernels (linear, poly, rbf, sigmoid) and achieved the best results with the rbf kernel. We left the degree and gamma parameters to their defaults value (3 and ‘scale’ respectively).

For our AE model, we created encoder and decoders with a CNN architecture. The encoder network had three convolutional layers of decreasing size (512, 128, 64). The decoder network followed the same structure in reverse (64, 128, 512). We used RELU activation functions and mean squared error loss. The model was trained over 50 epochs with the Adam optimizer.

For the VAE model, we utilized virtually the same architecture and parameters as the AE model, but used the evidence lower bound (ELBO) loss.

4 DATASETS

4.1 Malware Dataset

Our malware dataset consists of 5500 samples of malware from 5 different families (Cycbot, Rbot, Vobfus, Zbot, and Zeroaccess). The samples are a subset of the Malicia project and are made up of executable files [26]. The benign set is made up of 5500 files from the Windows system folder (C:Windows) which includes a range of executable and non-executable files. The dataset was split into 80% train (4,400 malware, 4,400 benign) and 20% test (1,100 malware, 1,100 benign).

For preprocessing, we calculated the Shannon entropy of the raw byte values. We used a sliding 256-byte window with a slide length of 128 bytes to compute an entropy feature vector. For each 256-byte block B we computed the entropy as:

$$H(B_i) = -\sum_{j=0}^{255} p_j \log_2(p_j) \tag{4}$$

The probability p was calculated by simply dividing the frequency of the bit in the block by the total size (256). This results in feature vectors of different sizes (depending on the size of the file in question). We calculated the average feature vector size and truncated or padded each of the vectors until they were the same size. Padding was done by appending the average entropy value of the existing vector.

4.2 Intrusion Detection Dataset

The KDD 99 dataset is an intrusion detection dataset made to mimic US Air force LAN traffic. The dataset consists of the raw TCP dump during both normal and attack scenarios. There are a total of 41 features and five labels (4 attack and 1 normal). We only used a subset of these features (all 18 continuous values). We trained the model with 742,224 samples of normal traffic and 742,225 samples of attack data. The test set

consisted of 230,557 samples of normal and 230,557 samples of attack data. We truncated out dataset in order to have an equal number of normal and anomalous samples.

4.3 Image Dataset

The CIFAR10 dataset includes 60,000 small images of size 32x32. We divided our dataset into 50,000 train samples and 10,000 test samples. To define our anomalous set, we added perturbations (random noise) to a duplicate copy of the normal set. The perturbations were an equal contribution of *nxn* squares (where *n* is a number between 1 and 20) that were randomly positioned on the image.

5 RESULTS

Our results indicated that the VAE model performed best for nearly all tests. Autoencoder detectors outperformed OCSVM in most cases but with a lower margin than the VAE models. This makes sense, VAE models have achieved state-of-the-art performance and are better adept at encoding complex data features. The multi-detector approach is also reliant upon different detectors learning different representations. If OCSVM and AE models learn similar representations then the benefit of multiple detectors may be reduced. While OCSVM had the lowest performance, it was the fastest to train. Training time may be more important in certain applications.

Table 2
Malware Detection (Malicia)

	OCSVM		**AE**		**VAE**	
	True Malware	*True Benign*	*True Malware*	*True Benign*	*True Malware*	*True Benign*
Predicted Malware	67%	29.8%	68.3%	27.8%	79.8%	19.8%
Predicted Benign	33%	70.2%	31.7%	72.2%	20.2%	80.2%

Table 3
Image Detection (CIFAR10)

	OCSVM		**AE**		**VAE**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	68.3%	30.4%	65.7%	35.2%	82.0%	18.9%
Predicted Normal	31.7%	69.6%	34.3%	64.8%	18%	81.1%

We also compared the performance of a single detector (using the highest performing model) with a two detector configuration. We noticed a decrease in false positives for all three datasets. In this situation, the reduction in false positives came at the expense of true negatives (anomalous points passing along as normal).

Table 4
Network Intrusion (KDD99)

	OCSVM		**AE**		**VAE**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	73.07%	28.24%	72.57%	23.86%	77.4%	20.00%
Predicted Normal	26.93%	71.76%	27.43%	76.14%	22.6%	80.00%

Table 5
Malware - Single vs Multiple Detectors

	Single		**Multiple**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	82.9%	**27.1%**	79.8%	**19.8%**
Predicted Normal	17.1%	72.8%	20.2%	80.2%

Table 6
CIFAR10 - Single vs Multiple Detectors

	Single		**Multiple**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	82.8%	**28.3%**	82.0%	**18.9%**
Predicted Normal	17.2%	71.7%	18.0%	81.1%

Table 7
KDD99 - Single vs Multiple Detectors

	Single		**Multiple**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	78.16%	**25.39%**	77.49%	**20.00%**
Predicted Normal	21.84%	74.61%	22.51%	80.00%

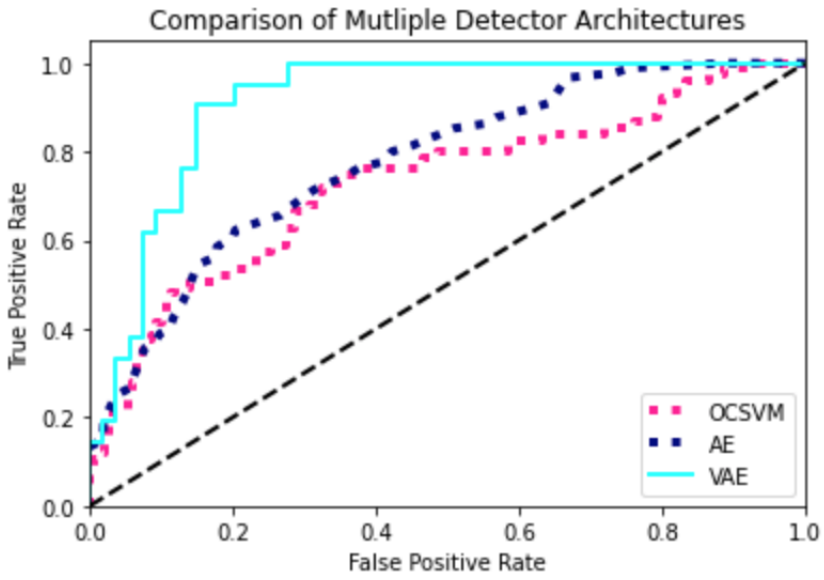

Fig. 6. ROC curves comparing different detector architecture performance

6 HEURISTIC

6.1 Mix-and-Match Models

Training multiple detectors can be an expensive and time-consuming process. While the VAE architecture tends to have the best performance, it may not be practical for the multi-detector approach. Additional detectors may increase costs significantly while adding only slight improvements. We can exploit this fact by using a combination of different model architectures. Ideally, the multi-detector approach should be able to augment the output of both a strong and weak model to produce results which are better than a strong model alone. We conducted tests combining the weakest model (OCSVM) with two stronger models (AE and VAE). This also allows us to provide a heuristic to determine if the multi-detector approach is worthwhile. If there is no performance improvement using a fast OCSVM pair then it is likely impractical to spend time and resources on more complex architectures for limited gains.

Table 8
Mixed Model Performance (KDD99)

	2 OCSVM		**AE + OCSVM**		**VAE + OCSVM**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	73.07%	28.24%	73.58%	23.86%	72.33%	24.33%
Predicted Normal	26.93%	71.76%	26.42%	76.14%	27.67%	75.67%

6.2 Univariate vs Multivariate Data

Univariate data involves data which corresponds to one variable while multivariate data corresponds to multiple variables. Most deep learning models tend to perform best in multivariate situations, while simpler techniques tend to do well with univariate data. We tested our model against the Melbourne Housing Prices dataset to see how additional variables impacted performance.

Table 9
Univariate Performance

	Single		**Multiple**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	95.5%	**7%**	90.5%	**11%**
Predicted Normal	4.5%	93%	9.5%	89%

Table 10
Multivariate Performance

	Single		**Multiple**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	96%	**6.5%**	99%	**1.25%**
Predicted Normal	4%	93.5%	1%	98.75%

Our multi-detector configuration performs comparably to single detector systems in univariate settings but outperforms single detector systems in multivariate situations. In the univariate case, an anomalous value only needs to differ over one measurement and it is easy to establish a distribution (normal) and distinguish one class from another. In multivariate data, a sample may have anomalous values for one variable, but normal values for another. It is hard to establish a single distribution which can account for all variables. The multi-detector approach allows for the model to encompass a more complex distribution because each detector is tasked with encoding a different class of data (positive set vs negative set). AE and VAE models are built from the ground-up to handle dimensionality reduction and are good candidates for multivariate settings. OCSVM is also strong in this area but can also perform well in univariate settings.

6.3 Data Types

We conducted tests to evaluate the multi-detector approach over three different data types: categorical, continuous (numerical), and images.

The multi-detector approach tends to perform best over continuous data. This makes sense as it is easier for multiple detectors to encode complex data representations which can be used effectively when the outputs are combined. Categorical data tends to have a lot of overlap between detectors (i.e. both detectors learn very similar representations) so there is less to be gained, causing performance to be similar to single detector solutions. Images contain lots of data points which opens the door to using dimensionality reduction and complex filtering techniques but can be difficult for detectors to pinpoint and divide effectively. Images may be a good candidate for using more than two detectors since there is such a large space for encoding different data features.

Table 11
Data Type Performance

	Categorical		**Continuous**		**Image**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	85.8%	15.1%	92.4%	5.4%	87.8%	10.1%
Predicted Normal	14.2%	84.9%	7.6%	94.6%	12.2%	89.1%

7 CLASS IMBALANCE

Class imbalance lies at the heart of anomaly detection. One of the greatest features of one-class classification is that you only need training data for one class (the larger one). Our proposed model is akin to using two one-class classifiers. In this scenario, we need data samples from both classes and class imbalance becomes an issue. Generally, class imbalance will create one very strong classifier and one (comparably) weaker one. We tested our model over two different scenarios (excess normal data vs excess anomalous data) using three different levels of excess (1X, 2X, 3X). We used different sampling techniques to pad or truncate our dataset.

7.1 Random Undersampling

Table 12
Class Imbalance - Excess Normal

	1X		**2X**		**3X**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	73.9%	30.4%	71.8%	27.2%	72.2%	26.8%
Predicted Normal	26.1%	69.6%	28.2%	72.8%	27.8%	73.2%

Table 13
Class Imbalance - Excess Anomalous

	1X		**2X**		**3X**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	73.9%	30.4%	73.6%	29.8%	72.2%	28.3%
Predicted Normal	26.1%	69.6%	26.4%	70.2%	27.3%	71.7%

Random undersampling is the most basic method to data truncation. It simply removes data points randomly until the desired number is reached. When using random

undersampling the results were mixed. In some cases, balanced data did yield better performance but in other situations it made almost no difference. This is likely because of the data itself. If most data points are densely populated, then there will likely not be much difference due to under sampling. In cases where there is low density and a large cluster area, under sampling may cut important edge points.

7.2 Other Sampling Techniques

While random undersampling is effective at creating balanced data sets, it does not take advantage of any of the inherent features of the data (e.g. distribution, variance, etc). We compared the performance of 5 techniques: random oversampling, SMOTE, random undersampling, Tomek Links, and balanced bagging classifier. We found that most techniques resulted in comparable results to random undersampling but the ensemble-based bagging classifier produced the best results.

Table 14
Random Undersampling vs Balanced Bagging

	Undersampling		**Balanced Bagging**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	73.9%	30.4%	74.6%	24.6%
Predicted Normal	26.1%	69.6%	25.6%	75.4%

8 THRESHOLDING

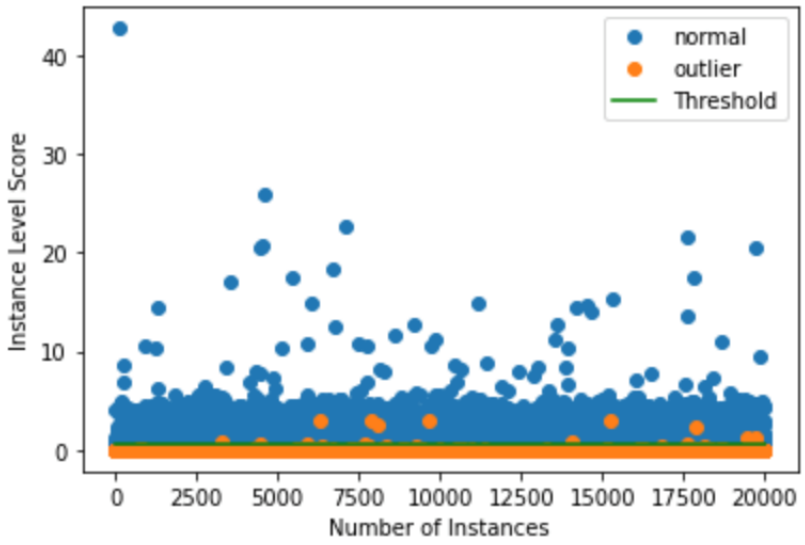

Fig. 7. Experimental threshold selection

Threshold selection is often done experimentally. Be varying the threshold on the training data, we can select a threshold which results in the highest evaluation metric. This is a good choice when the training data is well-representative of the real use case. Unfortunately most anomaly detection scenarios involve dynamic environments with skilled adversarial attackers. To help address this situation and automate threshold selection, we propose utilizing a histogram-based outlier score (HBOS).

HBOS is a statistical method which uses histograms to calculate an outlier score. It can handle both univariate and multivariate data by generating a histogram for each feature. The main appeal to HBOS is that it has linear time complexity; it is fast even when dealing with large datasets. HBOS is useful at finding global outliers but struggles with detecting local outliers. It is particularly useful when dealing with datasets where the distribution is unknown.

The multi-detector approach utilizes HBOS to quickly determine which points are global outliers. It then sets the threshold to the minimum level which encompasses all

global outliers. In practice, this performs better than selecting the experimental threshold. A more general threshold ensures that the system is robust against temporary environment changes and ensures equal balance of detectors (i.e. relying on one detector over another).

In our experimentation, we found OCSVM-based detectors to benefit most from the use of HBOS thresholding. OCSVM coupled with HBOS allow for increased efficiency which still maintaining high performance.

Table 15
Experimental vs HBOS Thresholding (OCSVM)

	Experimental		**HBOS**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	73.1%	**28.2%**	73.8%	**24.6%**
Predicted Normal	26.9%	71.8%	16.2%	75.4%

9 FILTERING

Multiple detectors have many advantages over relying on a single detector alone. They add a layer of redundancy which ensures reliable output even when one detector fails. Unfortunately, combining the output of multiple detectors into one final output is difficult. One approach is to use a fixed weighting (as shown previously). This is feasible for applications where detector performance is relatively stable but loses out on improvements from a more dynamic solution. A detector may be more reliable in some instances but less reliable in others. This problem is similar to the one encountered with sensor fusion in control theory.

9.1 Kalman Filtering

Kalman filtering is one solution used in sensor fusion and can be applied here. Kalman filtering estimates the joint probability distribution for points in a time series. It involves a two step process where the first step is used to predict the next state by estimating current values along with uncertainties. The second step uses a weighted average to correct the gain, state estimate, and covariance.

$$Prediction$$

$$x' = Fx + u$$

$$P' = FPF^T + Q$$

$$Correction$$

$$K = P'H^T S^- 1$$

$$x = x' + Ky$$

$$P = (I - KH)P'$$

We compared Kalman filtering against the fixed approach and found Kalman filtering improved performance for detectors with lower accuracy. However, detectors with high accuracy had little impact or, in some cases, worse performance. Kalman filtering works best when the errors have a normal distribution. This applied to most of our test cases. However, detectors with higher accuracy had fewer instances where there was disagreement between detectors (e.g. D1 flags a sample as malware but D2 does not). In these cases, Kalman filtering will not be useful for determining the correct output. This opens the door to using Kalman filtering to improve performance for the worst case scenario (multiple detectors with mediocre detection).

Table 16
Kalman Filtering vs Fixed Weighting (OCSVM)

	Fixed		**Kalman**	
	True Anomaly	*True Normal*	*True Anomaly*	*True Normal*
Predicted Anomaly	73.1%	**28.2%**	76.9%	**22.9%**
Predicted Normal	26.9%	71.8%	23.1%	77.1%

10 DISCUSSION

Anomaly detection systems tend to perform best in situations where anomalous behavior is hard to define. The multi-detector approach utilizes known examples of anomalous behavior in an en effort to augment traditional one-class classification, not to create a multi-class classifier. Two models trained on different data are able to model different distributions. When used together, they can identify features beyond even sophisticated single detector systems.

Class imbalance plays a significant role in any anomaly detection problem, but is especially important for our approach. Models trained on imbalanced data will result in over-reliance on one detector over another. Such a situation is undesirable not only for performance, but also defeats the purpose of using multiple detectors. In this situation a single detector would deliver faster performance at only a slight degradation in detection (based on the level of reliance). The best case scenario is a situation where the multi-detector approach can combine multiple mediocre detectors and produce results which outperform a strong single detector.

The multi-detector approach may also perform poorly in situations where detectors have a high level of accuracy. In order to improve performance, the filtering method would need to choose the correct detector more often than the incorrect detector. This can be difficult in practice and actually lead to a degradation in performance (compared to the single detector scenario). Such a situation would likely be a poor fit for the multi-detector approach even with an efficient filtering method due to the high cost of training for a low increase in performance.

11 CONCLUSION

Our proposed approach is able to reduce false positives when compared with a traditional single detector solution. The reduction in false positives often comes at the cost of true positive detection. The loss in detection rate is small when compared to the accompanying reduction of false positives. We believe that the tradeoff is worth the cost for applications where false positive rates are the top priority. Training and using multiple detectors adds overhead and complexity but can be reduced when using our proposed heuristic and filtering methods. Multiple detectors work best when each detector encodes a different representation of the data. In models where single detector accuracy is high, multiple detectors may result in a loss in performance. In situations where single detector performance is low, a combination of even low performance detectors may outperform a strong single detector (in line with other ensemble methods).

12 FUTURE WORK

Our experimentation only used two detectors. This method could be improved by using more than two detectors. While performance will likely improve, the additional overhead may not be worthwhile. Future work could explore the impact on performance with each additional detector and develop a heuristic to determine the optimal number of detectors.

A large part of this approach is reliant upon handling multiple detectors. Our experiments have compared fixed detector selection vs Kalman filtering. Other data fusion methods may perform better or be more applicable to different use cases. A dynamic weighting algorithm could greatly improve performance and also dictate the way future detectors should be trained.

The multi-detector approach works best for applications where anomalous samples are plentiful and false positives are common. One such application is medical imaging. Single detector solutions tend to perform poorly because they only flag items which are different from the norm. Additional detectors would ensure that the samples are not only different from the norm but also similar to dangerous cases, adding distinction between anomalous benign and anomalous malignant. Medical imaging is also well-suited for using synthetic data. Extensions of the multi-detector approach could utilize synthetic data to create large balanced datasets and compare their performance against non-augmented training sets.

Literature Cited

[1] P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining (2nd Edition)*. Pearson, 2nd ed., 2018.

[2] J. Wang, R. Martins de Moraes, and A. Bari, "A predictive analytics framework to anomaly detection," in *2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService)*, pp. 104–108, Aug 2020.

[3] M. L. D. Dias, C. L. C. Mattos, T. L. C. da Silva, J. A. F. de Macêdo, and W. C. P. Silva, "Anomaly detection in trajectory data with normalizing flows," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2020.

[4] O. Ibidunmoye, T. Metsch, and E. Elmroth, "Real-time detection of performance anomalies for cloud services," in *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, pp. 1–2, June 2016.

[5] Y. Zhou and J. Li, "Research of network traffic anomaly detection model based on multilevel autoregression," in *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, pp. 380–384, Oct 2019.

[6] Y. Choi, H. Lim, H. Choi, and I.-J. Kim, "Gan-based anomaly detection and localization of multivariate time series data for power plant," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 71–74, Feb 2020.

[7] Z. Wu, W. Zhu, J. Chanussot, Y. Xu, and S. Osher, "Hyperspectral anomaly detection via global and local joint modeling of background," *IEEE Transactions on Signal Processing*, vol. 67, pp. 3858–3869, July 2019.

[8] P. Adey, O. Hamilton, M. Bordewich, and T. Breckon, "Region based anomaly detection with real-time training and analysis," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 495–499, Dec 2019.

[9] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. Van Den Hengel, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1705–1714, Oct 2019.

[10] Z. Chkirbene, A. Erbad, R. Hamila, A. Gouissem, A. Mohamed, and M. Hamdi, "Machine learning based cloud computing anomalies detection," *IEEE Network*, vol. 34, pp. 178–183, November 2020.

[11] J. Zhao, Y. Li, H. He, and F. Deng, "One-step predictive encoder - gaussian segment model for time series anomaly detection," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, July 2020.

[12] I. Vasheghani Farahani, A. Chien, R. E. King, M. G. Kay, and B. Klenz, "Time series anomaly detection from a markov chain perspective," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1000–1007, Dec 2019.

[13] R. Liu and Q. Zhu, "A network anomaly detection algorithm based on natural neighborhood graph," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, July 2018.

[14] S. Wang, T. Chen, S. Chen, S. Nepal, C. Rudolph, and M. Grobler, "Oiad: One-for-all image anomaly detection with disentanglement learning," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2020.

[15] A. Ullah, K. Muhammad, K. Haydarov, I. U. Haq, M. Lee, and S. W. Baik, "One-shot learning for surveillance anomaly recognition using siamese 3d cnn," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2020.

[16] G. Pang, C. Shen, and A. van den Hengel, "Deep anomaly detection with deviation networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '19, (New York, NY, USA), p. 353–362, Association for Computing Machinery, 2019.

[17] K. Wawryn and P. Widuliński, "Detection of anomalies in compiled computer program files inspired by immune mechanisms using a template method," *Journal of Computer Virology and Hacking Techniques*, vol. 17, p. 47–59, Mar 2021.

[18] A. H. Hamamoto, L. F. Carvalho, L. D. H. Sampaio, T. Abrão, and M. L. Proença, "Network anomaly detection system using genetic algorithm and fuzzy logic," *Expert Systems with Applications*, vol. 92, pp. 390–402, 2018.

[19] G. Kandilogiannakis and P. Mastorocostas, "Refuzztid: A recurrent neurofuzzy model for anomaly detection in time series," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, July 2020.

[20] P. Boniol and T. Palpanas, "Series2graph: Graph-based subsequence anomaly detection for time series," *Proc. VLDB Endow.*, vol. 13, p. 1821–1834, July 2020.

[21] N. An, A. Duff, G. Naik, M. Faloutsos, S. Weber, and S. Mancoridis, "Behavioral anomaly detection of malware on home routers," in *2017 12th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 47–54, Oct 2017.

[22] S. Kumar, A. K. Shukla, and P. K. Muhuri, "Isolation forest based multi-source unsupervised transfer learning for missing gdp prediction," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2020.

[23] S. Maya, K. Ueno, and T. Nishikawa, "dlstm: a new approach for anomaly detection using deep learning with delayed prediction," *International Journal of Data Science and Analytics*, vol. 8, p. 137–164, Sep 2019.

[24] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, (Cambridge, MA, USA), p. 582–588, MIT Press, 1999.

[25] D. M. Tax and R. P. Duin, "Support vector data description," *Machine Learning*, vol. 54, p. 45–66, Jan 2004.

[26] A. Nappa, M. Z. Rafique, and J. Caballero, "The malicia dataset: identification and analysis of drive-by download operations," *International Journal of Information Security*, vol. 14, p. 15–33, Feb 2015.