Airbus Anomaly Detection Challenge

Archit Jain [1] **Aleksei Tcysin** [1] **Orhan Solak** [1]

Abstract

In Airbus, the researchers are interested in automatic methods of detecting anomalies in sensor data. This would provide useful tools for system health monitoring and preventive maintenance, yet manually labeling the data places enormous burden on engineers. We test various techniques and discuss their effectiveness in context of IA 2020 Airbus Challenge.

1. Introduction

The task of Airbus Challenge is to detect anomalies from sensor data for the purposes of system health management and predictive maintenance of helicopters. Normal data is available in abundance - most of the time systems stay in good condition throughout the test flight. Anomalies are rare events, and collection or generation of anomalous data is impossible due to extreme costs and complexity of the system. This setting corresponds to a semi-supervised anomaly detection, where we have access to normal data but have no idea about possible abnormalities.

Semi-supervised anomaly (novelty) detection is a well-researched topic in literature with a lot of application domains including medicine, cyber-security, image processing and many more. Subject matter is so vast several surveys and reviews have been published over the course of past years. (Chandola et al., 2009a) conduct general survey of anomaly detection techniques; (Pimentel et al., 2014) provides exhaustive review of novelty detection techniques, while recent survey of (Chalapathy & Chawla, 2019) focuses on Deep Learning techniques for anomaly detection.

(Chandola et al., 2009b) and (Cheboli, 2010) explore novelty detection techniques in the context of time-series data, which is directly applicable to our problem setting. Unfortunately, dealing with time series complicates the original problem of semi-supervised anomaly detection. (Cheboli, 2010) discusses the challenges of time series novelty detection, some of which are presented below.

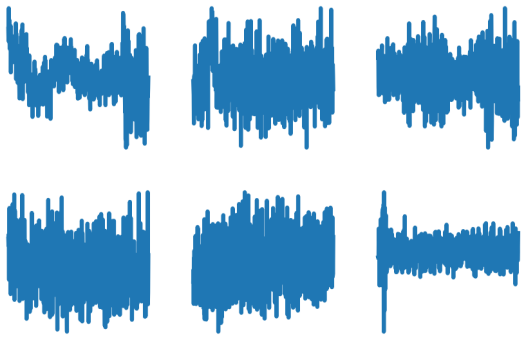

Figure 1. Examples of normal sensor data.

1. There are many ways in which anomaly can be defined within times series. Otherwise normal event within a specific context may be anomalous; a subsequence within a time series may be anomalous; or an entire time series may be anomalous with respect to a database of normal examples.
2. Best distance/similarity measure highly depends on a problem at hand.
3. Some algorithm perform poorly in the presence of noise, since differentiating anomalies from noise may be quite difficult.
4. Long sequences increase computational complexity and introduce curse of dimensionality.

Good system should be able to catch all three types of anomalous behavior, since we do not know the nature of possible anomalies. Further, it should be able to deal with noise and high dimensionality.

In Section 2 we briefly describe methodologies we employed in our experiments. In Section 3 we present experiments themselves and preliminary results on validation set. In Section 4 we present and discuss final results on a test set, as well as draw a conclusion to the project.

[*]Equal contribution [1]Department of Machine Learning and Data Mining, University of Jean Monnet, Saint-Etienne, France. Correspondence to: Archit Jain <archit.jain@etu.univ-st-etienne.fr>, Aleksei Tcysin <aleksei.tcysin@etu.univ-st-etienne.fr>, Orhan Solak <orhan.solak@etu.univ-st-etienne.fr>.

2. Methodology

2.1. Piecewise Aggregate Approximation

In order to reduce the dimensionality of time series and deal with possible noise, we employ Piecewise Aggregate Approximation (PAA) technique described in (Lin et al., 2003).

To reduce the time series from d dimensions to k dimensions, the data is divided into k equal sized frames. The *mean* value of the data falling within a frame is calculated and a vector of these values produces PAA representation, as shown in Figure 2. Instead of specifying new dimensionality one may provide frame size w.

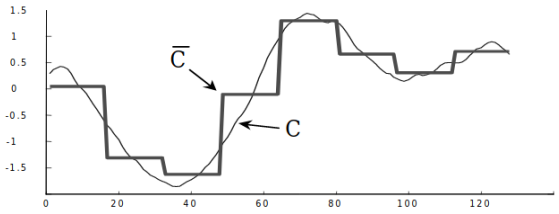

Figure 2. PAA representation can be visualized as an attempt to model a time series with a linear combination of box basis functions. In this case, a sequence of length 128 is reduced to 8 dimensions.

2.2. KNN-based Outlier Detection

(Ramaswamy et al., 2000) and (Angiulli & Pizzuti, 2002) present simple method for outlier detection, which can be easily adopted for the problem at hand.

The idea is to assign a *rank* to each point based on the distance to its k^{th} nearest neighbors. After ranking is done, we declare top n farthest points to be outliers. This implicitly selects a distance *threshold*, which can later be used to classify new examples.

2.3. One-class SVM

The idea of One-class SVM approach proposed in (Schölkopf et al., 2000; Campbell & Bennett, 2001) is to define a novel boundary in the feature space corresponding to a kernel, by separating the transformed training data from origin in the feature space, with maximum margin. See Figure 3 for intuition. This approach requires fixing *a priori* the percentage of positive data allowed to fall outside the description of the *normal* class.

2.4. Isolation Forest

(Liu et al., 2012) propose a different type of model-based method that explicitly isolates anomalies rather than profiles

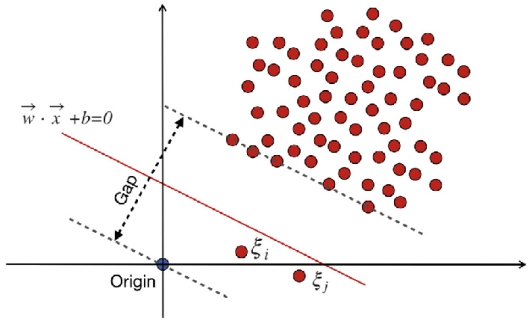

Figure 3. Diagram of One-class SVM. Normal data is mapped to a new space F such that maximum separation from origin is achieved.

normal instances. The method assumes anomalies are in minority relative to normal examples and have attribute-values that are very different from those of normal instances.

Isolation Forest *isolates* observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node.

This path length, averaged over a forest of such random trees, is a measure of normality and our decision function. Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produce shorter path lengths for particular samples, they are highly likely to be anomalies.

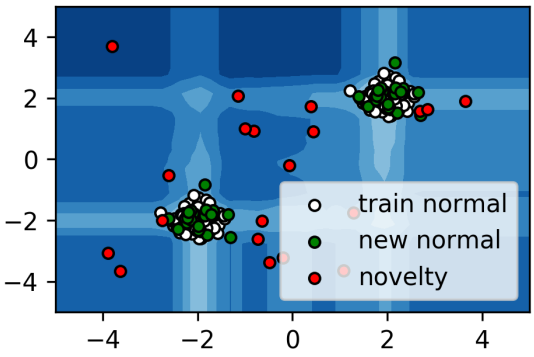

Figure 4. Isolation Forest applied on 2-dimensional data.

2.5. LSTM Autoencoder

LSTM. (Hochreiter & Schmidhuber, 1997) introduced LSTM with the objective of modeling long-term dependencies and determining the optimal time lag for time series problems.

The main part of LSTM is a memory block, depicted in Figure 5, containing memory cells with self-connections memorizing the temporal state, and a pair of adaptive, multiplicative gating units to control information flow in the block. It also has input gate and output gate controlling the input and output activations. Generally, LSTM can learn a long-term dependencies and forget what's not redundant and learn what seems like a trend.

Autoencoder. (Bengio, 2007) proposed Autoencoder (AE) as a dimensionality reduction model. It is a neural network that learns to copy its input to its output. Formally, it takes an input vector x and transforms it into a latent representation z, as presented in Figure 6. The resulting latent representation z is then mapped back into the original feature space y. The network is trained by minimizing the reconstruction error $(x - y)^2$.

LSTM is used as an intermediate layer in Autoencoder, which should aid in preserving temporal properties of the data.

2.6. Variational Autoencoder

Variational Autoencoder (VAE) (Kingma & Welling, 2013; Sölch et al., 2016; Rezende et al., 2014) is a more expressive version of standard AE as it replaces latent representation z with stochastic variables. For each observation, latent feature vector is replaced by a *distribution*, which allows for richer representation.

Well, in Variational Inference technique, we aim to learn the latent features by approximating to the unknown posterior

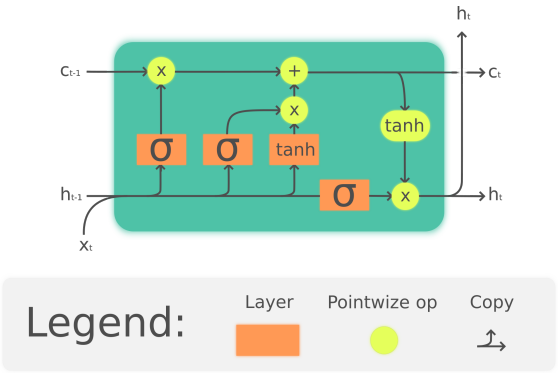

Figure 5. LSTM memory block.

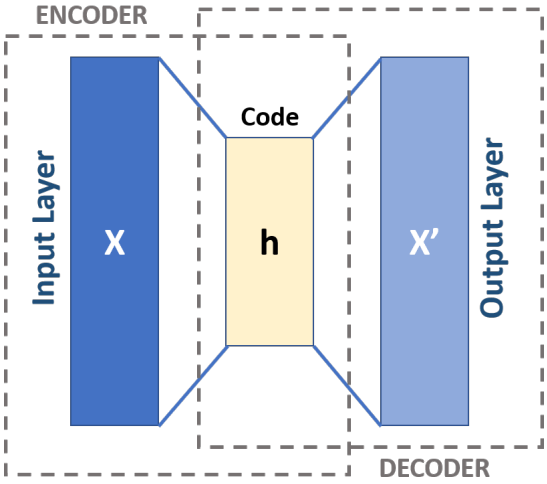

Figure 6. Schematic picture of Autoencoder.

$p(z|x)$ using a decoding distribution $q_\phi(z|x)$, and the encoding distribution $p_\Theta(x|z)$ implements a simple graphical model. Decoder and encoder are parametrized by ϕ and Θ, respectively, e.g., neural networks. This allows to overcome intractable posterior distributions by learning.

First, an encoder network turns the input samples x into two parameters in a latent space, which we will note z_μ and z_σ. Then, we randomly sample similar points z from the latent normal distribution that is assumed to generate the data, via $z = z_\mu + e^{(z_\sigma)} \times \epsilon$, where ϵ is a random normal tensor. Finally, a decoder network maps these latent space points back to the original input data.

The parameters of the model are trained via two loss functions: a reconstruction loss forcing the decoded samples to match the initial inputs (just like in our previous autoencoders), and the KL divergence between the learned latent distribution and the prior distribution, acting as a regularization term. You could actually get rid of this latter term entirely, although it does help in learning well-formed latent spaces and reducing overfitting to the training data.

3. Experiments on validation set

We fine-tune hyperparameters of the algorithms and show the best results on validation set in Table 1.

For all experiments we pre-process each time series in the database using Piecewise Aggregate Approximation with window size $w = 100$, thus reducing number of time steps from 61440 to 615. Each processed time series is treated as a 615-dimensional point. This allows us adapt standard anomaly detection techniques for the problem.

Table 1. Evaluation results for various algorithms on validation set.

ALGORITHM	F1-SCORE	SENSITIVITY
ONE-CLASS SVM	**0.93**	**0.87**
KNN-BASED	0.89	0.79
ISOLATION FOREST	0.83	0.73
LSTM-AE	0.88	0.86
VAE	0.92	0.86

We fit One-class SVM with rbf kernel, *gamma* set to $scale$ and *nu* 0.01.

For Isolation Forest we use default parameters from Scikit-learn.

KNN-based outlier detection algorithm is fitted with percentage of the outlier data set to 0.001, k-nearest neighbors set to 11 and *euclidean* distance measure. To assign a rank we use *largest* distance to k-th nearest neighbors.

For LSTM-Autoencoder, we use a couple of LSTM layers to encode the input layer, then use a RepeatVector layer to distribute the encoded vector across the decoder, which is then decoded in form of reconstructed input data (layer). We use Adam optimiser and MSE loss. Network is trained for 35 epochs using batch size 64.

For VAE, we first encode our input data using Dense layer into two vectors z_μ and z_σ, then we sample a point as $z_\mu + e^{0.5 z_\sigma} \times \epsilon$ to serve as input into Dense decoding layer. The loss function is defined as $R + KL$, where R is reconstruction loss, and KL is Kullback-Leibler divergence between the true and approximate posterior. Model is trained for 10 epochs using RMSprop optimizer and batches of size 128.

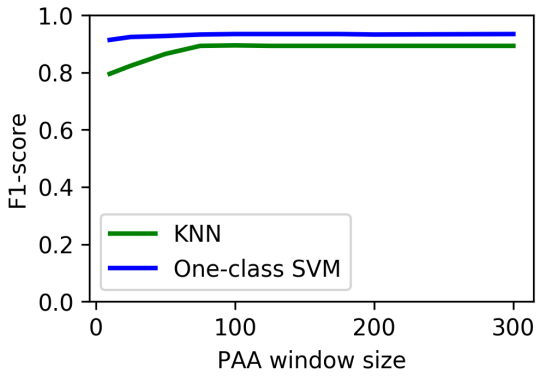

Figure 7. Performance of classifiers with respect to varying frame size w of PAA.

As Tables 1 show, all approaches perform reasonably well, with Once-class SVM and VAE having an edge over the rest. Detailed evaluation indicates that One-class SVM is able to detect all anomalies marked by other methods and 3 more on top, making it superior on validation set. We are not sure why this is the case.

We also examine the performance of Once-class SVM and KNN-based detector with respect to frame size for PAA. To our surprise, the performance does not change significantly, as can be seen form Figure 7.

4. Final results and discussion

Due to technical difficulties and being late we uploaded only one submission based on One-class SVM. We fit One-class SVM on the data consisting of the whole training set and normal examples from validation set.

The resulting scores are presented in Table 2.

Table 2. Results on final test set.

METRIC	SCORE
F1-SCORE	0.34
PRECISION	0.62
RECALL	0.23

In hindsight, we would've liked to try more methods and their ensembles. For example, applying domain adaptation or metric learning techniques.

There are a lot of interesting suggestions found in research literature; however, often they are not available out-of-the-box and need to be re-implemented, which makes it hard to quickly try new things. Some methods require deep technical knowledge.

Getting to know the data in more detail would be beneficial. For example, we would've liked to check how close are misclassified examples to separating hyperplane and see if there is anything we can do to represent the data better.

5. Citations and References

References

Angiulli, F. and Pizzuti, C. Fast outlier detection in high dimensional spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 15–27. Springer, 2002.

Bengio, Y. Learning deep architectures for ai (technical report 1312). *Université de Montréal, dept. IRO*, 2007.

Campbell, C. and Bennett, K. P. A linear programming approach to novelty detection. In *Advances in neural information processing systems*, pp. 395–401, 2001.

Chalapathy, R. and Chawla, S. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3): 1–58, 2009a.

Chandola, V., Cheboli, D., and Kumar, V. Detecting anomalies in a time series database. *UMN TR09-004*, 2009b.

Cheboli, D. Anomaly detection of time series. Master's thesis, Facility Of The Graduate School, The University Of Minnesota., 2010.

Chollet, F. et al. Keras. https://keras.io, 2015.

Faouzi, J. pyts: a Python package for time series transformation and classification, May 2018. URL https://doi.org/10.5281/zenodo.1244152.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Lin, J., Keogh, E., Lonardi, S., and Chiu, B. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pp. 2–11, 2003.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):1–39, 2012.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pimentel, M. A., Clifton, D. A., Clifton, L., and Tarassenko, L. A review of novelty detection. *Signal Processing*, 99: 215–249, 2014.

Ramaswamy, S., Rastogi, R., and Shim, K. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 427–438, 2000.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Rossum, G. Python reference manual. 1995.

Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., and Platt, J. C. Support vector method for novelty detection. In *Advances in neural information processing systems*, pp. 582–588, 2000.

Sölch, M., Bayer, J., Ludersdorfer, M., and van der Smagt, P. Variational inference for on-line anomaly detection in high-dimensional time series. *arXiv preprint arXiv:1602.07109*, 2016.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E. W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and Contributors, S. . . SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv e-prints*, art. arXiv:1907.10121, Jul 2019.

Zhao, Y., Nasrullah, Z., and Li, Z. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019. URL http://jmlr.org/papers/v20/19-011.html.

6. Software

In our research we used the following software:

- *Python* programming language (Rossum, 1995).
- *NumPy, pandas* packages from SciPy ecosystem for data extraction and manipulation (Virtanen et al., 2019).
- *Scikit-learn* package for building and evaluating models (Pedregosa et al., 2011).
- *pyts* package for time series processing (Faouzi, 2018).

- *pyod* package for KNN-based outlier detection (Zhao et al., 2019).
- *Keras* package for LSTM-AE and VAE methods (Chollet et al., 2015).