Machine Learning in Applications
FP01 Project Report

Latino, Salvatore s314872@studenti.polito.it
Magliano, Paolo s314867@studenti.polito.it
Panuccio, Andrea s294603@studenti.polito.it
Spaccatrosi, Jacopo s285891@studenti.polito.it

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Machine Learning in Applications FP01 Project Report

***Abstract*—We compare four sota models for TSAD on the RoAD dataset, with a BNN and a K-NN models as baselines. Sota models are: LSTMVAE, Telemanom, TimeVQVAE and USAD. We consider different metrics and try to delve into the problem of defining a good one. Here the GitHub Repository:**

I. Introduction

The Time Series Anomaly Detection, often just called TSAD or TAD, is a well known problem in the statistical inference field. It tries to merge the two tasks of applying inference in a dynamic, potentially unpredictable setup, and the one of defining what onthologically defines an anomaly.
The Statistical Learning Theory highly relies on the informativeness and the iid assumptions to guarantee that a certain Hypothesis Class of functions will effectively infer unseen observations, once trained on a sample of the same population. Time Series per se break this assumption because every observation is highly depending in the preceding ones. The best way to account for this is to apply Windowing to the original population so that any new observation contains the information on the preceding ones. This procedure, despite useful to respect the informativeness assumption, makes the data size bigger and the sample size smaller. There is so a tradeoff between how much we assume our system is close to markovian and how much we search for complexity to increase the theorical performance.
But Windowing does not account for iid on its own. In order to get closer to this assumption, we should assume other two facts that usually are not: our training data should be independents and they should be identically distributed with respect to the test set ones. The training set is id but, after windowing, if the number of Time Series at disposal is low, observations will end up being highly dependents. Also, even if the training and test set will easily be independents, it's not so easy to grant for them being id. Given a certain system, either it's an open one, like for weather forcasting, or it's a closed one but the policy changes, applied to collect the training data, affected the system itself and were not maintained at test time.
This makes datasets with many short Time Series, modeling closed systems, with short time interactions, far easier to tackle, from both a theorical and practical standpoints.
The other problem that TSAD tries to face is defining what an anomaly is in the first place. At a first glance the problem seems as a binary classification task (i.e. Anomaly / Not Anomaly) but we can't really find regularities for them, since they are what differs from what we expected, i.e. what we define as normal. Also, for practical reasons, anomalies are rare, so data on them are too, and this leads to a strongly unbalanced dataset. Anomalies also occurs in multiple modalities, so the only way to represent them all would be to collect a huge amount of data, most of them being normal. So anomalies are unpredictable per se and that's why the most common approach is to model what's normal and define how likely is an observation to be drawn from that normal population. But this is nothing more than an Out of Distribution Detection (OOD Detection shortly) task, thus we have no other way than tradingoff our normal outliers in the filtering process. The only way to account for that is to better model the normal distribution.
In a mixup scenario like TSAD, a further problem emerges, as what we first defined as anomaly in the original population is shown to a statistical model in the windowed one, so an OOD detected by the model might be less effective and usually ad hoc estimators or heuristics are crafted to make them effective.

II. Materials and methods

A. *RoAD Dataset [1,]*

Robotic Arm Dataset, RoAD, is also named Kuka dataset, since it consists of observations collected from a Kuka arm robot. It consists of 6 Time Series, numbered from 0 to 5. 0, 2, 3 and 4 are considered normal while 1 and 5 are the collisions ones, containing anomalous intervals.
In order to create a fair setup for our models, without sacrificing a preprocessing ad hoc for a specific model, we split TS1 in two. The first 10% was used to create a new TS6, not present in the original RoAD dataset; we will use TS6 for calibration. The remaining 90% of TS1 was kept in TS1, so that a new shrunk TS1 was left; we will use TS1 and TS5 for testing. Since all of our models and baselines perform OOD from the normal class, normal TS will be used for training.

B. *Baselines*

Since defining a dataset complexity isn't a well defined concept, we decided to get two different baselines, of different complexity and that rely on different metrics.

1) K-NN: K Nearest Neighbours is a distance based discriminative model that considers as scores the distances of an observation with respect to the K nearest observations in the training set. The whole training set is so used at inference time and the number K is an hyperparameter to be defined. Different ways of weighting the K scores can be defined too, still K-NN assumes data are distributed with a globular shape; a usually unmet assumption in datasets having complex patterns.

2) Bayesian NN: Instead of directly passing windows to the model, a Feature Extraction step is performed on them, and statistics of the related window are passed to the model. The Fully Connected layers guarantee asymptotically a good enough approximation of the equivalent Gaussian Process.

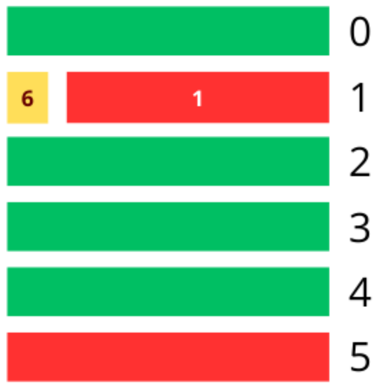

Fig. 1: This is how our RoAD splits look like: Green is for the training set, Yellow for the calibration set, Red for the test set.

Furthermore, adding Dropout layers has been proven to approximate a DeepGP [7,], thus also the learned latent variables are stochastic, and applying MC Dropout makes the model variational. The learned latent distributions are bimodal, with one of the modalieties having 0 mean, so more complex variants like VI on explicitly defined Multivariate Gaussian shaped latent variables will show better results. We kept this lightweight version since it had already been tested on the RoAD dataset. A Bayesian model is a generative one, so it learns the posterior distribution $P(C = normal|\ X = x)$. A threshold is defined, for the MC sample, to label as anomaly the too unlikely observations.

C. Models

1) LSTMVAE [2,]: LSTMVAE model implements a Variational Autoencoder, with LSTM layers. The Variational Autoencoder consists of sequential Encoder and Decoder that extract into latent space the useful information from input data and then reconstruct the input data from the latent space. The latent space represents the latent distribution of the input data. The long short-term memory-based variational autoencoder (LSTM-VAE) exploits the temporal dependency of time-series. Given a multimodal input xt at time t, the encoder approximates the posterior $p(z_t|x_t)$ by feeding an LSTM's output into two linear modules to estimate the mean μ_{z_t} and co-variance Σ_{z_t} of the latent variable. Then, the randomly sampled z from the posterior $p(z_t|x_t)$ feeds into the decoder's LSTM. The final outputs are the reconstruction mean μ_{x_t} and co-variance Σ_{x_t}. The paper uses the combination of KL divergence and reconstruction loss to train the network. The KL loss optimizes the progress-based prior $p(z_t)$. Unlike conventional static priors using a normal distribution $N(0, 1)$, the center of a normal distribution varies as $N(\mu_p, \Sigma_p)$, where μ_p and Σ_p are the center and co-variance of the underlying distribution of multimodal points of the time-series. The reconstruction loss is the negative log-likelihood of the input data The proposed implementations of LSTMVAE doesn't work for the KuKa dataset, because during the training loss diverges and the model doesn't learn anything. We tried to change the hyperparameters, but the model still doesn't work, so we propose little changes in the loss function that provides working model. The KL divergence is computed using normal distribution $N(0, 1)$, and the reconstruction loss is computed using the mean squared error between the input and output data. The paper version defines anomaly score as the negative log-likelihood of an observation with respect to the reconstructed distribution of the observation through an encoding-decoding model. The threshold is dynamically computed through a support vector regression that maps latent space to the anomaly threshold. We propose a different approach, we use the reconstruction loss as anomaly score. So for each time series we compute the mean squared error between the input and output data as multidimensional anomaly score. We use the anomaly score as input for the support vector classifier to classify the time series as normal or anomaly, previously calibrated on the calibration set.

2) Telemanom [3,]: Telemanom is an LSTM (Long Short-Term Memory) model used for detecting anomalous time series. An LSTM model was chosen because it is particularly effective in capturing long-term dependencies in sequential data, thanks to its internal structure. This allows the network to ”forget” irrelevant information and retain relevant data, which is crucial for learning complex time series and accurately predicting future events. The model operates in an unsupervised manner. During the training phase, it attempts to learn the expected normal behavior from the data, using several previous telemetry data points to predict the current element. Once it has learned the normal data behavior, during the inference phase, it assesses how much the behavior of data with anomalies deviates from the expected behavior to predict anomalies. The anomaly prediction process begins by generating an estimated forecast of the telemetry value for each time interval, using historical and contextual data. This forecast is the result of the LSTM's sequential learning, which considers long-term correlations among the data. Once the predicted value ŷ(t) is obtained, the model calculates the prediction error by measuring the discrepancy between the predicted value and the actual value y(t). The error, expressed as the absolute value of the difference, immediately reflects any significant deviation from the expected behavior, which could indicate a potential anomaly. To address data volatility and reduce sensitivity to isolated peaks, which are common in complex time series, the error is smoothed using an exponentially weighted mitigate the effects of peaks in the data that should not categorically be associated with anomalous data behavior, maintaining a more consistent and stable view of error trends over time. Subsequently, the model establishes a dynamically calculated anomaly threshold based on the mean and standard deviation

of the smoothed errors.
This threshold helps to differentiate between normal fluctuations and statistically significant variations, thus identifying anomalies. Errors that exceed this threshold are classified as anomalies, suggesting that the system's behavior has significantly deviated from the predicted model. Finally, each detected anomaly is evaluated with a score to determine its severity. This is done by analyzing how much the smoothed errors exceed the defined threshold, thus quantifying the magnitude of the anomaly relative to the normal variability of the data. This analysis provides not only confirmation of the anomaly but also a measure of its criticality.
Several structural modifications have been applied to the original Telemanom model code. A calibration of the model weights before the training process for each channel was introduced. Additionally, PyTorch libraries have been used instead of the original TensorFlow ones. The original Telemanom model, although multivariate, predicted anomalies for individual channels without integrating predictions across various channels. Therefore, additional logic has been introduced to sum the various predictions, dynamically calculating an anomaly score threshold to prune anomalies below this threshold. Furthermore, logic for calculating metrics and for graphically displaying the predictions of individual channels and the final prediction has been added. It is important to note that the model does not divide the dataset into temporal sections but makes a point prediction for the predictions.

3) TimeVQVAE [4,]: TimeVQVAE is a Vector Quantized Variational Autoencoder, tailored for Multivariate Time Series. It receives as input a window tensor BatchxChannelsxLength (BxCxL) and applies the STFT to pass from time domain to time-frequency domain, so a 4 rank tensor BxCxHxW is spread into Low and High frequencies.
Two different AutoEncoders, deterministics per se, are used to learn meaningful discrete embeddings, known as codewords.
A second stage of training is then used to replace the Encoders with GITMask, a Bidirectional Transformer, to learn the distribution in the latent space [8,].
Despite the TimeVQVAE team itself proposed an Anomaly Detection application [9,], during our work the related code was still under review, so we applied an original customization. Since the model was pretty big, we entirely dropped the second stage, only leveraging the AutoEncoders as deterministic. In order to actually infer the last timestamp we also applied a mask to the last slice, the Lth one.
We then put a SVC over it, passing as input 4 reconstruction losses.

4) USAD [5,]: USAD (UnSupervised Anomaly Detection) model is an unsupervised artificial neural network based on an autoencoder architecture but inspired by GANs, managing to avoid the instability issues that commonly affect them. It combines the advantages of autoencoders and adversarial training while compensating for the limitations of each technique. During the development of USAD, the training time and energy consumption of the model were considered as performance criteria, resulting in an architecture that allows fast and energy-efficient training. The model is formulated as an AE architecture within a two-phase adversarial training framework. It is composed of two autoencoders, AE1 and AE2, sharing the same encoder network. Training follows two phases. First, the two AEs are trained to learn to reconstruct the normal input windows W. Secondly, the two AEs are trained in an adversarial way, where AE1 seeks to fool AE2, and AE2 aims to learn whether the data is real (coming directly from W) or reconstructed (coming from AE1). After the original input is reconstructed by AE1, it is fed into AE2, which then attempts to reconstruct it again. The training objectives are structured where the first term takes into account the capabilities of both AE1 and AE2 to reconstruct the original input, while the second term considers AE1's ability to minimize the difference between W and the output of AE2, and AE2's goal to maximize this difference. At inference time, the anomaly score for each window is defined as:

$$f(\hat{W}) = \alpha||\hat{W} - AE_1(\hat{W})||_2 + \beta||\hat{W} - AE_2(AE_1(\hat{W}))||_2$$

, where $\alpha + \beta = 1$, and they are used to parameterize the trade-off between false positives and true positives. If α is greater than β, the number of true positives is reduced along with the number of false positives, leading to a low detection sensitivity scenario. Conversely, if α is less than β, the model's sensitivity increases, raising the number of true positives at the cost of also increasing the number of false positives.

D. Metrics

1) F1: F score with $\beta = 1$, corresponds to the harmonic mean of Precision and Recall. In formula:

$$F1 = \frac{2\,Precision \cdot Recall}{Precision + Recall}$$

2) F1-PA%K [6,]: In TSAD simply applying F1 score means defining the task as to predict the anomalous labels, for all of the windows in the test dataset. Since systems are able to recover even if they receive a skewed signal of anomaly, a more used metric is F1-PA, that defines TSAD as the task to label as anomalous a segment, as far as at least one of the intervals is anomalous. The result is that, while F1 underestimates models'performance, F1-PA overestimates it. F1-PA%K applies F1-PA but considers the segment anomalous only if a K percentage is anomalous, and does this iteratively from K close to 0 to K equal 100. The score is the area under the so made curve in the F1-PA%K vs K plot. It is also shown that the metric point evaluation is equivalent to F1-PA @ K close to 0 and to F1 @ K equal 100, as expected.
Tested against uninformed baselines, this metric has shown a more robust and plausible behaviour than the related two.

3) AUROC: The Area Under the Receiver Operating Characteristic, computes the False Positive Rate(FPR) and the True Positive Rate(TPR) for a moving threshold. The score is the area under the so made curve in the TPR vs FPR plot. Despite its simplicity to compare different models, it shows limitations in case of class imbalance. On the other side it weights all the errors the same.

4) AUPRC: The Area Under the Precision-Recall Curve, computes Precision and Recall for a moving threshold. The score is the area under the so made curve in the Precision vs Recall plot. It is usually preferred to AUROC in case of unbalanced datasets but it has been shown to be biased towards errors since it weights high scoring errors the most [10,], so subpopulations that score the higher are the ones that lower AUPRC the most.

III. Results and discussion

A. For every model we have:

- sampling rate 0.1s
- windowing with window_size = 20
- feature selection that filters all of the features related to temperature and keep the remaining 55 ones

B. K-NN:

- K is set to 1

C. BNN:

- Since it assigns a prediction to the whole input window, we padded the same prediction for all of the timestamps, i.e. window_size times

D. USAD and LSTMVAE:

- Use a common dataset class (every other model has its own implementation)

The results are shown below:

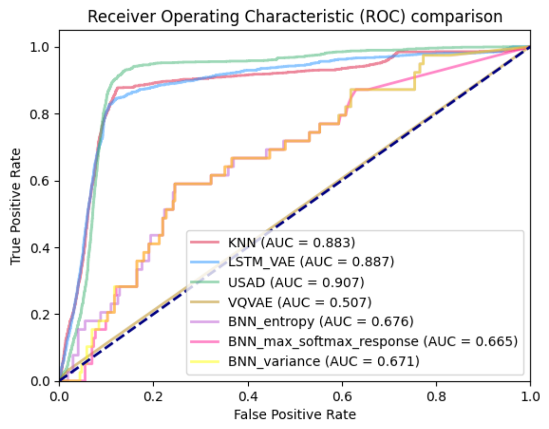

Fig. 2: AUROC

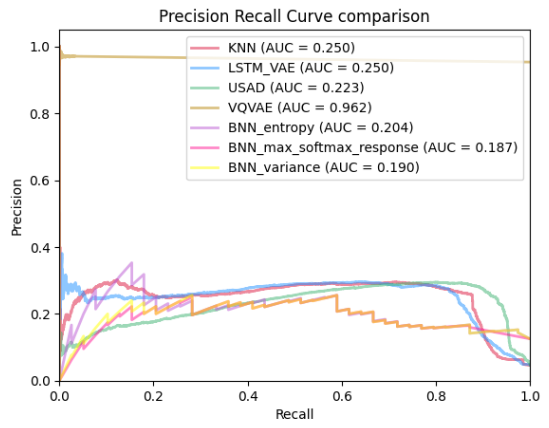

Fig. 3: AUPRC

Model	F1	AUROC	AUPRC
K-NN	0.419	0.845	0.254
BNN Entropy*	0.344	0.676	0.204
BNN Variance*	0.336	0.671	0.190
BNN Softmax*	0.304	0.665	0.187
LSTMVAE	0.375	0.774	0.226
Telemanom	0.429	0.932	0.187
TimeVQVAE	0.976	0.507	0.962
USAD	0.430	0.907	0.223

TABLE I: The scores of our models and baselines. * For the BNN model the F1 score has been computed on a threshold t=0.9, corresponding to the empirical max.

As we can see, regarding the F1 score, all the models (except for TimeVQVAE, which outperforms all the others with 0.976) range between 0.3 and 0.4, with 1-NN proving to be a strong baseline by achieving an F1 score of 0.419. Regarding the AUROC achieved by the various models, we can observe three trends: VQVAE stands at 0.507, BNNs achieve about 0.67, with no significant differences between Entropy, Variance, and Softmax, and all the other models exceed 0.75. The three trends reduce to two when looking at the AUPRC, where all models settle around 0.2, with the sole exception of TimeVQVAE, which achieves an incredible 0.976. With only four time series, our data are quite correlated indeed, so models try to learn those patterns and generalize poorly. TimeVQVAE clearly overfits, due to its too high amount of parameters.

IV. Conclusions and future works

We have shown that for problems when the dataset is unbalanced and far from iid, simpler and distance based models perform better and learning from data is an ill posed problem, highly relying on inductive bias to reach good performance. In cases like this, it worths to try a wide variety of different models so Hypothesis classes, to find the most suited one. Further studies can be conducted with unsupervised alternatives, like MERLIN++.

References

[1] A. Mascolini, S. Gaiardelli, F. Ponzio, N. Dall'Ora, E. Macii, S. Vinco, S. Di Cataldo, and F. Fummi, "Robotic arm dataset (road): A dataset to support the design and test of machine learning-driven anomaly detection in a production line," in *IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society*, 2023, pp. 1–7.

[2] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," 2017. [Online]. Available: https://arxiv.org/abs/1711.00614

[3] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18. ACM, Jul. 2018. [Online]. Available: http://dx.doi.org/10.1145/3219819.3219845

[4] D. Lee, S. Malacarne, and E. Aune, "Vector quantized time series generation with a bidirectional prior model," 2023. [Online]. Available: https://arxiv.org/abs/2303.04743

[5] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 3395–3404. [Online]. Available: https://doi.org/10.1145/3394486.3403392

[6] S. Kim, K. Choi, H.-S. Choi, B. Lee, and S. Yoon, "Towards a rigorous evaluation of time-series anomaly detection," 2022. [Online]. Available: https://arxiv.org/abs/2109.05257

[7] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” 2016. [Online]. Available: https://arxiv.org/abs/1506.02142

[8] D. Lee, E. Aune, and S. Malacarne, “Masked generative modeling with enhanced sampling scheme,” 2023. [Online]. Available: https://arxiv.org/abs/2309.07945

[9] D. Lee, S. Malacarne, and E. Aune, “Explainable time series anomaly detection using masked latent generative modeling,” *Pattern Recognition*, vol. 156, p. 110826, Dec. 2024. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2024.110826

[10] M. B. A. McDermott, L. H. Hansen, H. Zhang, G. Angelotti, and J. Gallifant, “A closer look at auroc and auprc under class imbalance,” 2024. [Online]. Available: https://arxiv.org/abs/2401.06091