Securing Deep Learning Models with Autoencoder based Anomaly Detection

Joana Kühne[1], Christian März[2], and Clemens Gühmann[3]

[1,2,3] *Technische Universität Berlin, 10587, Germany*
joana.kuehne@iav.de
christian01.maerz@iav.de
clemens.guehmann@tu-berlin.de

Abstract

With deep learning models on the rise in safety relevant applications, securing their predictions has become an important task. The most common approaches for this task include detecting anomalies in the data or predicting an additional safety score with the model itself or a separate supervisor model. In this study a joint model approach is presented, consisting of two neural network submodels: a predictive model and an Autoencoder. The data is first passed through the Autoencoder generating a reconstruction of the input. Then the reconstruction and the original data are both passed through the predictive model generating two outputs. A subsequent supervisor ensures the accuracy of the predictive model by comparing the two outputs. The developed supervisor works unsupervised, hence no anomalous data is required for training. In comparison to other Autoencoder based supervisors, there is no need to set a threshold on the Autoencoder's reconstruction loss. The approach was tested on image classification, multivariate time series classification and multivariate time series regression. On classification tasks with enough output classes, it was shown that it is superior to the traditional threshold based algorithms, ensuring high accuracy with a minimal drop of samples. The methodology was also adapted for regression problems, giving a good estimation of the prediction error for clean and noisy data, but it was not possible to ensure a satisfying accuracy on time delayed data.

1. Introduction

Deep learning models are on the rise in many scientific fields. Their ability to solve complex and nonlinear tasks has made them popular. However, in comparison to physical models, they struggle with extrapolation. Hence, it is important that the data in the production stage is similar to the data seen in training. Deviations to the training data often occur in real world applications due to sensor delays and drifts, aging of the system, and communication errors such as noise. Especially in safety relevant applications, securing those models against these influences, which can be seen as anomalies, is essential.

In the past, Autoencoders, especially Variational Autoencoders (VAEs), have been proven useful for anomaly detection. Many researches focus on improving the Autoencoder's separation ability for an optimal set anomaly threshold (Xu et al., 2018), (Lin et al., 2020). Choosing this threshold is not trivial but crucial for a good anomaly detection. Setting the threshold optimal becomes especially challenging if the anomaly is unknown in the training process, which is often the case in real world applications.

The proposed method combines a deep learning model with an Autoencoder. The input data is handed to the trained Autoencoder which reconstructs the input. If the data is similar to the training data, the Autoencoder should be able to reconstruct the input data accurately. Otherwise, an anomaly is suspected. The reconstruction and the original input data are both passed through the deep learning model, generating two predictions, which are then compared.

For applications with discrete result space the prediction of non-anomalous data should lead to the same class for both samples. For those tasks this allows us to sort out samples as anomalies for which the two results are not the same, hence the threshold becomes obsolete.

For applications with continuous output space the difference between the two predictions can be interpreted as a safety measure which gives an estimation of the expected error and has the same unit as the prediction. This is easier to grasp than the Autoencoder's reconstruction error, which has no unit and is typically used.

The advantage of this method is the distinction between samples that can or cannot be handled by the subsequent application model instead of just deciding if the input is anomalous.

This leads to a higher robustness of the joint model and a better usage of the prediction model's resources. Moreover, the Autoencoder and the prediction model are trained separately which makes the training a lot more stable than using coupled training methods. Furthermore, the supervisor can be applied to existing prediction models.

The proposed method is validated on both, classification and regression tasks. For the classification, the publicly available MNIST (LeCun & Cortes, 2010) and UEA multivariate time series classification (Bagnall et al., 2018) datasets are used. For regression, a dataset simulating a SCR catalyst as part of an automotive exhaust gas aftertreatment system is evaluated. For both tasks common anomalies are applied to the data.

2. Related Work

Autoencoders have been first introduced by Rumelhart et al. (1986) and were adapted to deep learning thirty years later by Hinton & Salakhutdinov (2006) for dimensionality reduction. The scope of Autoencoders is to produce a compressed representation of the input data. Autoencoders are built of an encoder and a decoder connected by a bottleneck layer which has a lower dimensionality. First the encoder compresses the input data into the size of the bottleneck, this representation of the data is called code, from which the decoder reconstructs the input. The loss is computed between the input and the reconstruction.

A special form of Autoencoders are Varational Autoencoders (VAE) (D. Kingma & Welling, 2014). The difference between a vanilla and a VAE architecture is that the code consists of two separate vectors, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, instead of one. The vectors $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are interpreted as the mean and standard deviation of a normal distribution from which a sample is drawn and then passed to the decoder. During training, in addition to the loss between input and reconstruction, the Kullback-Leibler divergence (Kullback & Leibler, 1951) between the normal distribution of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ and the standard normal distribution is computed. This regularization enforces the distribution to be close to the standard normal distribution, which is a desirable property, especially for data generation since new samples can be created easily by inputting values from the standard normal distribution into the decoder (Doersch, 2021).

Apart from dimensionality reduction and generative modeling, Autoencoders have been used for unsupervised anomaly detection. In the following, some examples of studies that use Autoencoders for anomaly detection for time series data are presented.

Xu et al. (2018) use VAE for anomaly detection for seasonal KPIs in web applications. To improve the performance, they sample L times from the latent space and then compute the reconstruction probability instead of the reconstruction error. As evaluation metric they use the area under curve (AUC) and the F1-score corresponding to the optimal set threshold.

Lin et al. (2020) propose VAE-LSTM for time-series anomaly detection. In addition to the standard VAE, their model consists of an Embedding and LSTM layers, both situated behind the encoder and a predicted embedding located in front of the decoder. They evaluate their model on five real world univariate datasets and compare the results to anomaly detection with VAE (An & Cho, 2015), LSTM-AD (Malhotra et al., 2015) and ARMA (Pincombea, 2007). As evaluation metrics they use precision, recall and F1-score. The value they use as threshold on the score function to determine an anomaly is set to give the best F1-score.

Chen et al. (2019) built a joint model consisting of a VAE and a LSTM model to address anomaly detection and trend prediction together. The univariate data is first passed through the VAE and an anomaly label is given according to the reconstruction error and a threshold $k\sigma_r$ where k is a fixed value determined by the validation dataset and σ_r is the standard deviation of the absolute reconstruction error. The reconstruction is cleaner than the original data in the sense that the VAE reduces the noise. The cleaned reconstruction is thus used as input for the LSTM for trend prediction. In comparison to this study, prediction performance and anomaly detection are evaluated separately while the present approach uses the anomaly detection for confidence estimation of the prediction model. Additionally, in the model built by Chen et al. (2019) the models have to be trained together and a threshold needs to be set for anomaly detection which is not needed in this study's approach.

Furthermore, Zhang et al. (2018) developed a Multi-Scale Convolutional Recurrent Encoder-Decoder for anomaly detection in multivariate timeseries. They compare their model to eight different baseline methods including a one-class SVM model, Deep Autoencoding Gaussian Mixture model (Zong et al., 2018), History Average (HA), Auto-Regression Moving Average (ARMA) (Hamilton, 1994) and LSTM encoder-decoder (LSTM-ED) (Cho et al., 2014) on a synthetical and a power plant dataset. Their model is able to outperform all baseline models with regard to precision, recall and F1 score. They set the anomaly detection threshold according to

$$\tau = \beta \cdot s(t)_{valid} \tag{1}$$

where $s(t)_{valid}$ are the anomaly scores over the validation data and $\beta[0, 1]$ was set to maximize the validation F1-score. For all methods using the F1-score a validation dataset with anomalies must be available.

Apart from the sole task of anomaly detection, Autoencoders can be used as network supervisors. Network supervisors detect inputs that are dissimilar to the training set. Using Autoencoders as network supervisors is very similar to anomaly detection since the inputs that are dissimilar to the training data can be seen as anomalies. Henriksson et al. (2019) provide a framework for comparing the performance of network supervisors evaluated with seven different metrics including area under receiver operating characteristic (AUROC), area under precision recall curve (AUPRC), the true positive rate at 5% false positive rate, the precision at 95% recall and others. Their study showed that the performance of a network supervisor strongly depends on the metric used for evaluation.

Determining the anomaly threshold is a commonly known problem where no ideal solution exists. This paper introduces a new approach to ensure a model's accuracy without the need to set a threshold.

For the purpose of anomaly detection, besides Autoencoders, Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have been proven useful (Schlegl et al., 2017). GANs consist of a generator and a discriminator network. The latter learns to distinguish between generated and real data. GAN based methods rely on concurrent training of generator and discriminator, producing and detecting adversarial examples during training. During inference, the discriminator is able to distinguish between adversarial or anomalous and real samples. The present approach focuses on supervising a prediction regarding typical anomalies occurring in production stage, like noise, instead of adversarial examples, therefore GANs are not further considered in this study.

Besides autoencoder based methods, there exist other network supervision approaches. Hendrycks & Gimpel (2016) monitor the activation of a classifier's softmax layer. They propose that correctly classified samples have greater maximum softmax probabilities than falsely classified and out-of-distribution samples. DeVries & Taylor (2018) introduce a classifier network that learns to estimate its own confidence. They add a second branch, consisting of one or more fully connected layers, in parallel to the softmax prediction. That confidence branch produces a scalar output between 0 and 1, representing the network's confidence. During training, they modify the network's predictions according to the confidence of the network such that they are closer to the target probability distribution.

The most similar study was conducted by Geifman & El-Yaniv (2019) who introduced SelectiveNet which is a network for end-to-end learning of selective classification. The selective model is defined through a prediction function f and a binary selection function g. For a sample x, if $g(x)$ is 1, the selective model does a prediction $f(x)$, otherwise no prediction is conducted. SelectiveNet is trained with the selection prediction objective as an additional term of the loss function optimizing the model to a target coverage. They compare their model to Softmax Response and Monte Carlo Dropout on several image classification and on one regression dataset.

3. METHOD

The traditional process of supervising a deep learning network with an Autoencoder is schematically displayed in figure 1. Here the network that is to be supervised is called prediction model. The prediction model and the Autoencoder use the same input $\boldsymbol{X}$. While the Autoencoder reconstructs the input resulting in $\boldsymbol{X_R}$, the prediction model estimates the output Y_P. If the Autoencoder's reconstruction loss is rated non-anomalous by the supervisor, the prediction is expected to be secure otherwise the prediction is likely to be inaccurate (Kühne et al., 2020).

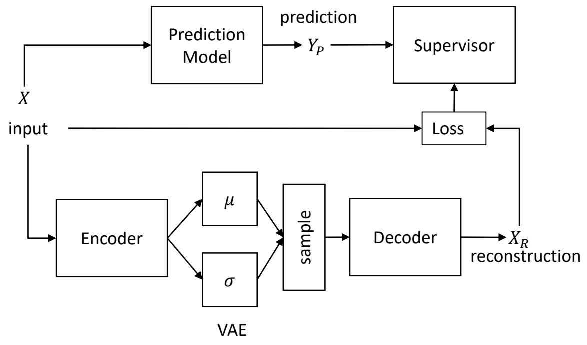

Figure 1. Schematic overview of a classical VAE supervisor.

In the present paper a new approach for supervision of machine learning models is provided, which is shown in figure 2. The two submodels (Autoencoder and prediction model) are the same as in the classical approach. The difference is that the Autoencoder's reconstruction $\boldsymbol{X_R}$ is used as input for the prediction model instead of computing of the reconstruction loss. The prediction model thus makes two separate predictions: one from the original input $\boldsymbol{X}$ and one from the reconstructed input $\boldsymbol{X_R}$ leading to Y_P and Y_R, respectively. The supervisor compares the two predictions and depending on the prediction model's task, a statement about the risk of the prediction can be made.

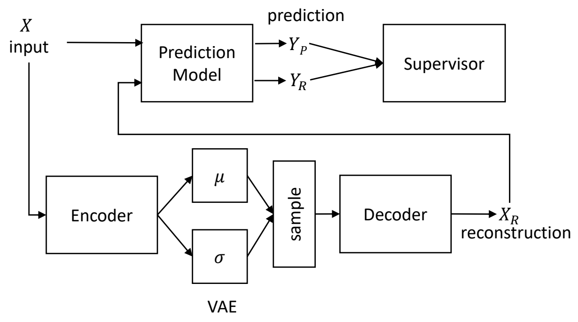

Figure 2. Schematic overview of the proposed method.

If the prediction model is a **classifier** with distinct classes, the supervisor compares Y_P and Y_R. If the result is not identical, the prediction is labeled insecure, otherwise it is labeled safe.

The goal of this method is to sort out samples that are dissimilar to the training data, retaining a high prediction accuracy for the remaining samples. The performance of the joint model can thus be measured by two parameters risk and coverage defined in (3) and (4) where N_{total} is the total number of test samples, $N_{classified}$ is the number of samples that are found to be non-anomalous by the supervisor and $N_{correct}$ is the number of non-anomalous samples that are classified correctly. The definitions of risk and coverage for selective classification were first introduced by El-Yaniv & Wiener (2010).

$$Accuracy = \frac{N_{correct}}{N_{classified}} \tag{2}$$

$$Risk = 100 \cdot (1 - Accuracy) \tag{3}$$

$$Coverage = 100 \cdot \frac{N_{classified}}{N_{total}} \tag{4}$$

In some safety relevant tasks it is better to have no prediction at all, than to get an inaccurate result. The advantage of this method is that the decision whether a sample is valid or not does not only depend on the Autoencoder and its strictness but also on the prediction model's robustness. The classification cases that can occur using the joint model are displayed in table 1. There exist five possible combinations of correctly or falsely classifying the original sample Y_P and the reconstructed sample Y_R. The chance of obtaining true negatives, which lead to a drop in accuracy and therefore increase in risk, decreases if several output classes are present.

Table 1. Possible outcomes of the joint model used for classification tasks.

Y_P	Y_R	$Y_P = Y_R$	**classification case**
correct	correct	true	true positive
correct	false	false	false negative
false	correct	false	true negative
false	false	true	false positive
false	false	false	true negative

If the prediction model is performing a **regression** task, the application is more complex since it is not possible to just compare the predicted classes to one another. Instead, a safety measure is defined according to (7). This safety measure estimates the deviation ΔY_T of the prediction model's output Y_P from the ground truth data Y_T at any given point, using the deviation between two predictions ΔY_R. This is useful in a production stage application where the ground truth values are not available.

$$\Delta Y_T = |Y_P - Y_T| \tag{5}$$

$$\Delta Y_R = |Y_P - Y_R| \tag{6}$$

$$Safety = \sigma_R \Delta Y_R \tag{7}$$

In equation (7) σ_R is a calculated constant, which can be interpreted as a safety factor and differs for any given feature. The optimal value for σ_R can be calculated with (8), which can be interpreted as the minimal distance between ΔY_T and ΔY_R, where n is the number of samples. The same distance can later in the test cases be used as a Key Performance Indicator (KPI) for the supervisor. If the KPI is zero, ΔY_T and ΔY_R are identical for every time step which would be an ideal error estimation. On the other hand, a large KPI indicates an insufficient error estimation.

$$\sigma_R = \underset{\sigma_R}{\text{argmin}} \underbrace{\left(\frac{1}{n} \sum_{i=1}^{n} |\sigma_R \Delta Y_{Ri} - \Delta Y_{Ti}| \right)}_{=KPI} \tag{8}$$

In contrast to optimizing a threshold, optimizing σ_R does not require anomalous data.

4. Evaluation

4.1. Image Classification

To prove the proposed concept a study was conducted on the MNIST (LeCun & Cortes, 2010) dataset. This dataset consists of images of handwritten digits and is mostly used for classification tasks. It was chosen for the concept validation since it is publicly available and widely used as benchmark dataset. The dataset is divided into training and test set, with 60000 and 10000 samples, respectively.

Table 5 shows the architectures chosen for the VAE and the Classifier. ReLU was used as activation function and Adam (D. P. Kingma & Ba, 2015) as optimizer. Training was conducted separately for the two models with the unmodified training dataset. To test the supervisor property of the joint model normal distributed noise was added to the test data with a standard deviation of 0.3. Figure 3 shows the joint model's risk and the corresponding coverage as defined in equations (3) and (4). In addition to the joint model, the data was evaluated with a threshold based approach with the same models for varying thresholds. The upper boundary of the threshold range is set to result in a $Coverage$ of 100%. Two prominent thresholds are marked differently: the average training loss and the threshold for which 90 percent of the train samples are labeled non-anomalous. The lower the threshold is set, the more samples are labeled anomalous resulting in a small coverage. The objective is to have a high coverage while not losing accuracy in comparison to the unmodified data.
For the non-anomalous case, displayed in figure 3a, the risk does not vary a lot with an increasing threshold. The train threshold and the 90 percentile threshold are both valid choices resulting in a low risk with a coverage of 50% and 85%, respectively. The joint model's risk and coverage are close to the 90 percentile threshold result. However, when introducing noise to the test data (figure 3b), the set thresholds become unsuitable. The risk for the same threshold is lower than for

the non-anomalous case and the reconstruction errors of all samples are higher than many of the set thresholds, including the train threshold and the 90 percentile threshold. The joint model is able to retain the low classification risk even for the high noise magnitudes and still achieve a high coverage. Table 4 depicts the differentiated classification outcomes of the joint model as defined in table 1 for noises with standard deviations of 0.1, 0.2 and 0.3. The number of false positives, where the first prediction Y_P and the second prediction Y_R are both false but by chance the same, is low (0.68-1.12%). This confirms the proposed hypothesis that the joint model works well for classification tasks with several output classes.

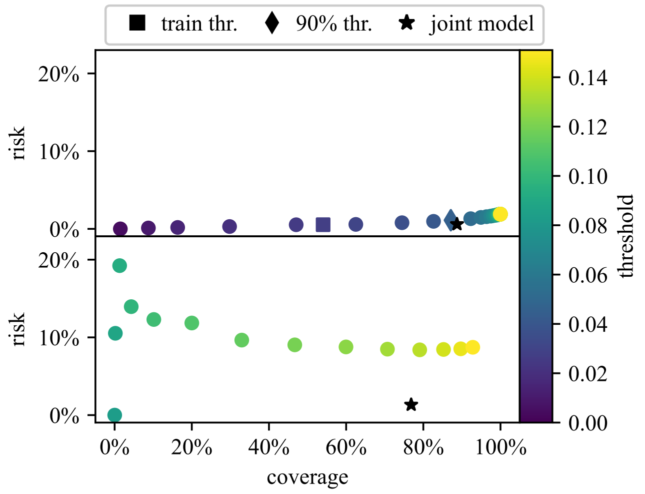

Figure 3. Risk over coverage for the MNIST (LeCun & Cortes, 2010) dataset. The upper plot (a) is generated with non-anomalous data, the lower plot (b) with normal distributed noise with a standard deviation of 0.3. The round markers display the results for arbitrary chosen thresholds for a traditional VAE supervisor model. The upper boundary of the threshold range is set to let all non-anomalous test samples pass through the supervisor resulting in a coverage of 100%. The square and diamond marker indicate the set threshold on the train threshold and 90% coverage, respectively. The star marker indicates the joint model's performance.

4.2. Time Series Classification

Analogous to the evaluation on image data, the joint model is evaluated for classification of time series data. The evaluated time series originate from the UEA multivariate dataset archive (Bagnall et al., 2018). For the proposed approach a certain number of output classes is required, otherwise the number of samples Y_P and Y_R that are the same by chance is too high. As minimum number of classes six was found to be suitable. With that restriction six datasets are chosen for evaluation. The specifics of the datasets are summarized in table 2. For each dataset a VAE and a classifier are trained separately. The architecture specifics are shown in table 5 and are the same for all the time series classification datasets except for the bottleneck dimension.

Table 2. Timeseries classification dataset specifics.

Dataset	train	test	dim.	seq len	classes
Articulary Word Recognition	275	300	9	144	25
Cricket	108	72	6	1197	12
ERing	30	270	4	65	6
Libras	180	180	2	45	15
PenDigits	7494	3498	2	8	10
UWave Gesture	120	320	3	315	8

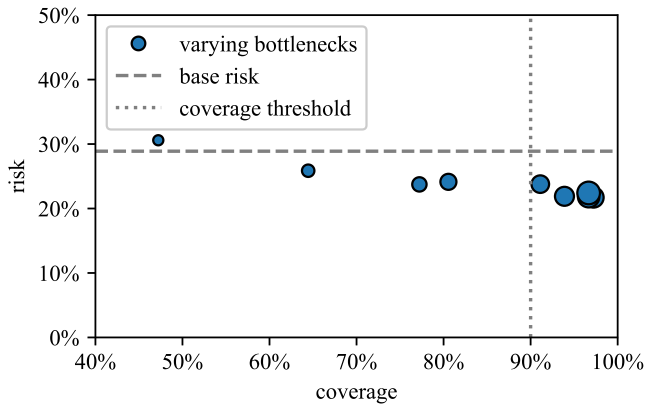

Figure 4. Bottleneck search on the Libras dataset. The grey lines represent the two objectives for risk and coverage while the size of the markers represents the bottleneck size.

The bottleneck dimension is an important hyperparameter since it determines the strictness of the Autoencoder. If the bottleneck dimension is too large, samples that are not similar to the the training data can be reconstructed well, if it is too small even the training data cannot be reconstructed sufficiently. To determine a suitable bottleneck size two objectives were defined for the performance of the joint model on the training data:

1. the joint model's risk (according to equation 3) should be at least as low as the classifier risk when used stand-alone
2. the coverage (according to equation 4) should be at least 90%.

Starting with a size of two, the bottleneck size was gradually increased until both objectives were satisfied. This procedure is visualized in figure 4 for the Libras dataset. The resulting bottleneck dimensions for all time series classification datasets are summarized in table 6. Choosing the bottleneck dimension and the other architecture hyperparameters influences the supervisor performance of the VAE. The advantage of the presented joint model is the ambiguity of a reconstruction loss threshold, which is an additional hyperparamter of

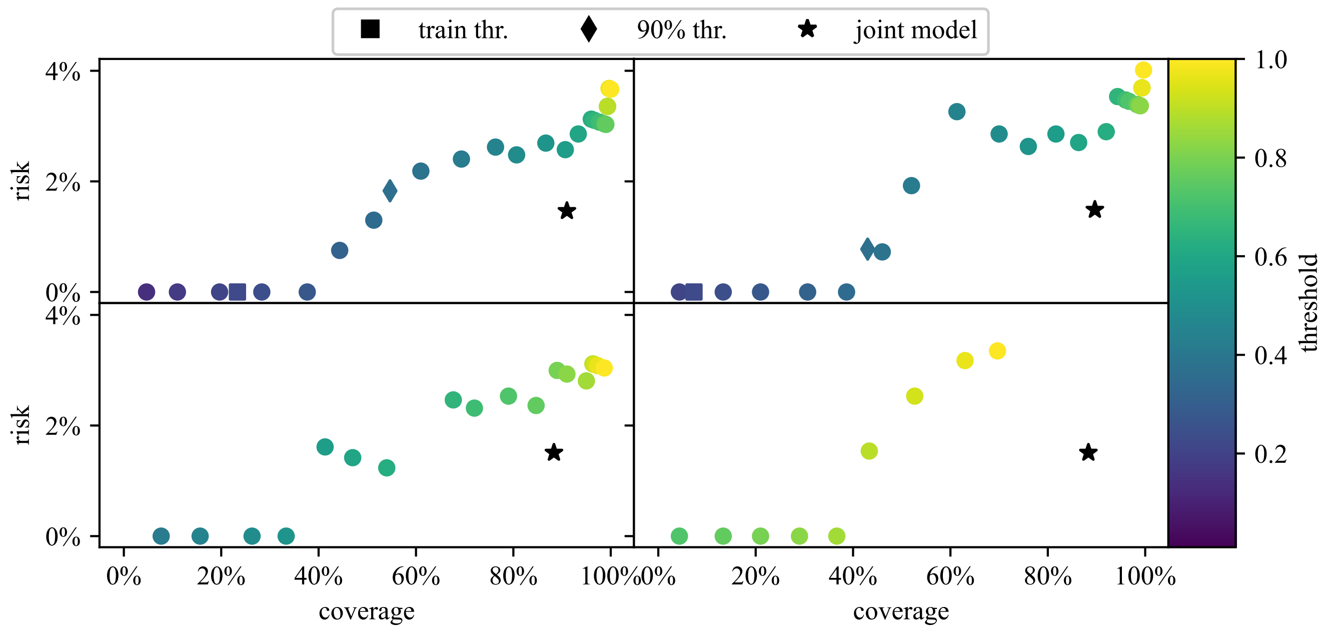

Figure 5. Risk over coverage for the Articulary Word Recognition dataset. The upper left plot (a) is generated with non-anomalous data, the upper right plot (b) with noise with a standard deviation of 0.25, the lower left (c) with 0.5 and the lower right (d) with 0.75.

the classical approach shown in figure 1. For the comparison of the two methods the exact same autoencoder is used.

The joint models were then tested on the original test dataset and four modifications: three noisy datasets, where noise is applied to the test dataset with standard deviations of 0.25, 0.5 and 0.75, and a dataset with a mirror anomaly. For the latter, dimension zero was mirrored for each time series, so the last time step is the first and so on. This procedure creates a contextual anomaly (Chandola et al., 2009), meaning that the values are feasible for the channel but not in the context of the other channels. Figure 5 displays the joint model results for the normal and noisy cases exemplary for the Articulary Word Recognition dataset. The results are similar to the MNIST results (figure 3). For the non-anomalous case the train threshold and 90 percentile threshold both lead to acceptable results (figure 5a). With increasing noise, however, all samples are above the train and 90 percentile threshold so no samples are labeled secure. In contrast, the joint model has still a coverage of 70% with the highest noise applied while keeping the same risk as for the normal data (figure 5d).

Figure 6 displays the results of the mirror anomaly. The joint model again excels since the coverage is with 70% significantly higher than for the 90 percentile threshold (15%) while resulting in the same risk. Moreover, for all the anomalies (figures 5 and 6), the joint model marker is above the varying threshold markers, meaning even with an optimal set threshold, the combination of coverage and risk cannot be achieved with the threshold based method.

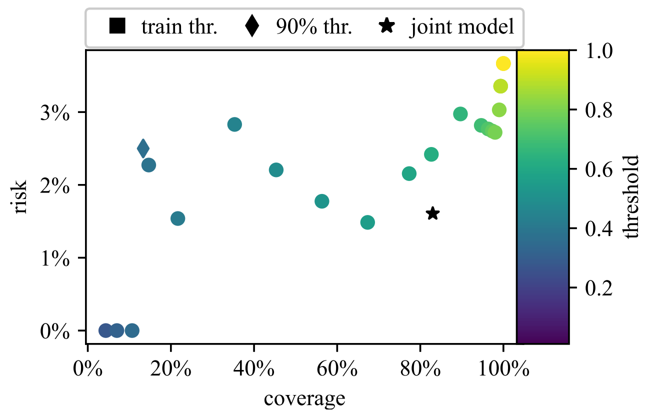

Figure 6. Risk over coverage for the mirror anomaly applied on the Articulary Word Recognition dataset.

Table 7 provides an overview of the results for all examined time series classification datasets. Comparing the joint model's risks for the different anomalies, it is noticeable that remaining the risk steady works better on datasets with a high number of classes, e.g. Articulary Word Recognition and Cricket, as it was anticipated beforehand. The accuracy for the mirror anomaly is significantly lower than for the non-anomalous case for the Pen Digits, ERing and UWave Gesture and Libras datasets. All those datasets have four or less dimensions (compare table 2), therefore mirroring one dimension is has a great impact and cannot be dealt with well by the joint model.

4.3. Time Series Regression

For the time series regression problem the multivariate dataset from März et al. (2020) is used. It is a result of a 1D physical simulation of an SCRF catalyst. For the present application the regression of the wall temperature signal is evaluated. The feature space has seven distinct features and the last two time steps are taken into account.

To find a well fitting prediction model, an architecture search was conducted. All models are trained with ReLU activation functions, using an Adam optimizer. The chosen architecture can be found in the appendix in table 5. As supervisor network an Autoencoder instead of a VAE is used since it shows a better performance for the task at hand.

Next up, the optimization problem (8) is solved for the network, finding the parameter σ_R. The behavior of sigma in comparison to the KPI is displayed in figure 7, leading to $\sigma_R = 1.16$.

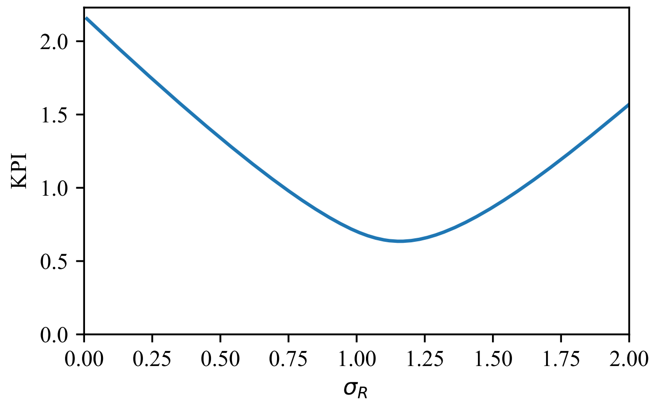

Figure 7. KPI over σ_R for the regression model. According to equation (8), σ_R is chosen where the KPI is minimal, resulting in $\sigma_R = 1.16$ at $KPI = 0.63$.

Table 3. KPI for Regression task.

Figure	Anomaly Type	Anomaly Value	KPI
8a	-	-	0.63
8b	noise	5°C	0.60
8c	noise	10°C	0.67
8d	delay	2s	1.26

In figure 8a, the two deviations ΔY_T and $\sigma_R \Delta Y_R$ are shown for non-anomalous data. The ground truth values and the prediction of the model can be found in the appendix in figure 9. In the real life scenario ΔY_T is an unknown property, which is estimated by ΔY_R. Hence, in the optimal case both curves are nearly identical. This cannot be achieved completely but the general trend and scale are matching. Especially at the beginning, where the difference between input and wall temperature is high, and thus the prediction of the output temperature is difficult, a good approximation of the error is given by the joint model. As with the classification data, in the next phase different errors are applied to the original data. Figure 8b, c and d display ΔY_T and ΔY_R for different errors. The error is applied starting at 500s. In the plots 8b and c, noise errors of 5°C and 10°C are applied to the input temperature, respectively. In both cases the joint model is able to estimate the accuracy of the prediction model. In plot 8d a time delay error of 2s on the input temperature is applied. The resulting deviation is not very large and the joint model is not able to predict the deviation in a significant way.

For further evaluation, the KPI for the different tests are evaluated and summarized in table 3. For a perfect error estimation the resulting KPI would be zero. It becomes clearly visible that the joint model is able to estimate the error on noisy data as good as on non-anomalous data, but its estimation of the error on time delayed data is not reliable enough.

5. Conclusion

The present study shows that the proposed joint model approach is superior to the traditional threshold based supervisor models for image and time-series classification. It works especially well if the number of distinct output classes is high enough. The results on the regression problem indicate that the model is a good estimator for the accuracy of the model, if the data is clean or noisy. The denoising property of Autoencoders is a well known feature and can here be used to great effect. However, on time delayed data the performance is not consistent. This can be further researched and maybe fixed with a recursive prediction model. Another interesting study would be to adapt the joint model approach on reinforcement tasks, where the supervisor can activate a fall back strategy for unsafe predictions. The proposed joint model could be, for example, very useful in the medical field in the application of tumor classification, where it is more important to ensure that predictions are accurate rather than to predict every sample.

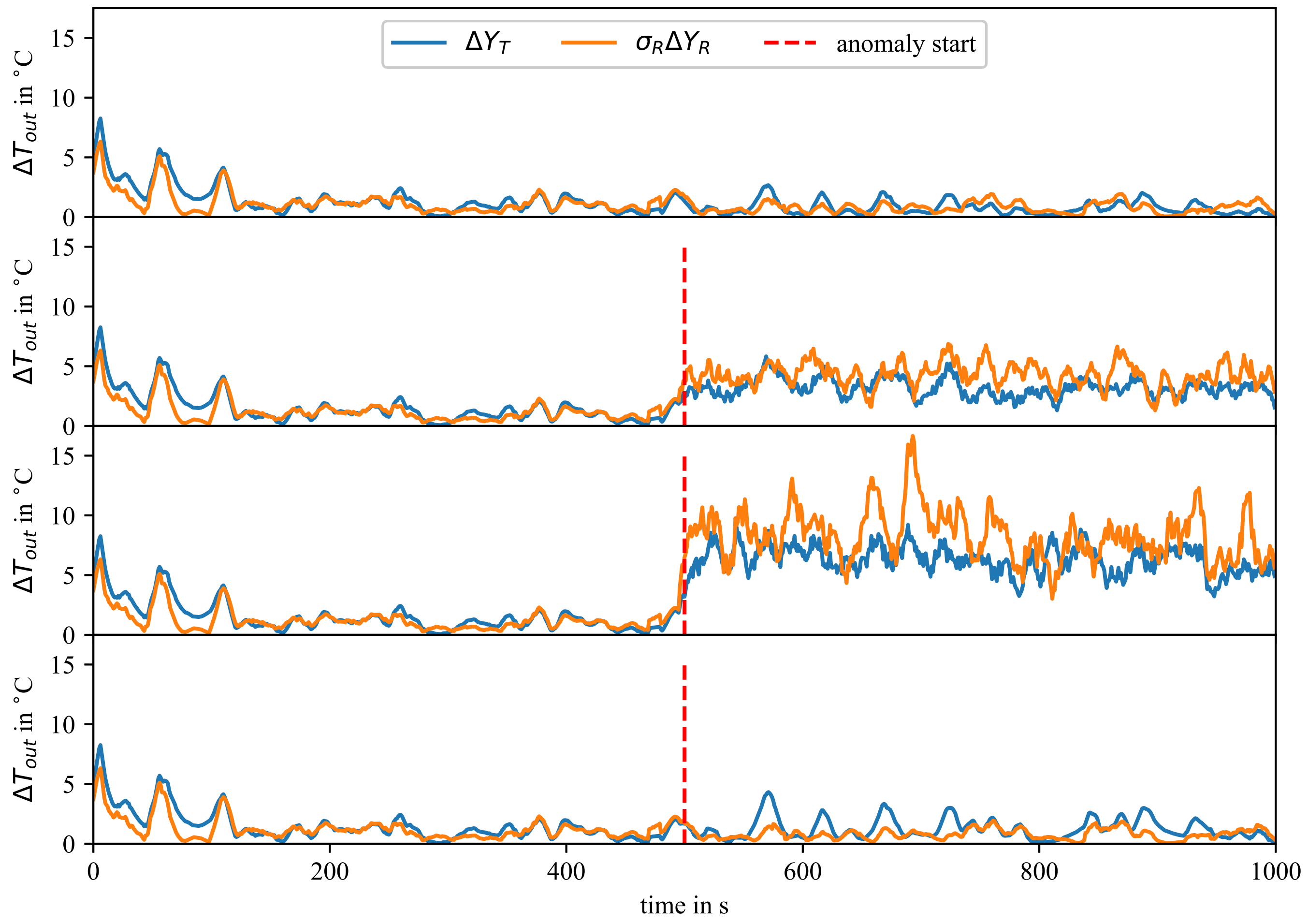

Figure 8. Deviation and safety for different anomalies applied on SCR dataset. The upper plot (a) is without anomaly, the next (b) and (c) with noise of $5°C$ and $10°C$ and the lower plot (d) with a time delay of 2s.

Nomenclature

$N_{classified}$	number of samples that are labeled non-anomalous by the supervisor
$N_{correct}$	number of correctly classified samples
N_{total}	number of test samples
$\boldsymbol{\mu}$	mean
$\boldsymbol{\sigma}$	standard deviation
σ_R	safety factor
$\boldsymbol{X}$	input data
$\boldsymbol{X_R}$	reconstructed input
Y_P	predicted output from original input
Y_R	predicted output from reconstructed input
Y_T	ground truth output

References

An, J., & Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability..

Bagnall, A. J., Dau, H. A., Lines, J., Flynn, M., Large, J., Bostrom, A., … Keogh, E. J. (2018). The UEA multivariate time series classification archive, 2018. *CoRR, abs/1811.00075*. Retrieved from `http://arxiv.org/abs/1811.00075`

Chandola, V., Banerjee, A., & Kumar, V. (2009, 07). Anomaly detection: A survey. *ACM Comput. Surv., 41*. doi: 10.1145/1541880.1541882

Chen, R., Shi, G., Zhao, W., & Liang, C. (2019). Sequential VAE-LSTM for anomaly detection on time series. *CoRR, abs/1910.03818*. Retrieved from `http://arxiv.org/abs/1910.03818`

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, *abs/1406.1078*. Retrieved from http://arxiv.org/abs/1406.1078

DeVries, T., & Taylor, G. W. (2018). *Learning confidence for out-of-distribution detection in neural networks.*

Doersch, C. (2021). *Tutorial on variational autoencoders.*

El-Yaniv, R., & Wiener, Y. (2010, 05). On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, *11*, 1605-1641.

Geifman, Y., & El-Yaniv, R. (2019, 09–15 Jun). SelectiveNet: A deep neural network with an integrated reject option. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (Vol. 97, pp. 2151–2159). PMLR. Retrieved from http://proceedings.mlr.press/v97/geifman19a.html

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). *Generative adversarial networks.*

Hamilton, J. D. (1994). *Time series analysis*. Princeton university press.

Hendrycks, D., & Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *CoRR*, *abs/1610.02136*. Retrieved from http://arxiv.org/abs/1610.02136

Henriksson, J., Berger, C., Borg, M., Tornberg, L., Englund, C., Sathyamoorthy, S. R., & Ursing, S. (2019). Towards structured evaluation of deep neural network supervisors. *CoRR*, *abs/1903.01263*. Retrieved from http://arxiv.org/abs/1903.01263

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. Retrieved from https://science.sciencemag.org/content/313/5786/504 doi: 10.1126/science.1127647

Kingma, D., & Welling, M. (2014, 12). Auto-encoding variational bayes..

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings.* Retrieved from http://arxiv.org/abs/1412.6980

Kullback, S., & Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79 – 86. Retrieved from https://doi.org/10.1214/aoms/1177729694 doi: 10.1214/aoms/1177729694

Kühne, J., März, C., Werfel, J., Gelbert, G., Behrendt, F., & Guehmann, C. (2020, 09). Hybrid modeling of a catalyst with autoencoder based selection strategy. In *Sae technical paper.* SAE International. Retrieved from https://doi.org/10.4271/2020-01-2178 doi: 10.4271/2020-01-2178

LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. Retrieved 2016-01-14 14:24:11, from http://yann.lecun.com/exdb/mnist/

Lin, S., Clark, R., Birke, R., Schönborn, S., Trigoni, N., & Roberts, S. (2020). Anomaly detection for time series using vae-lstm hybrid model. In *Icassp 2020 - 2020 ieee international conference on acoustics, speech and signal processing (icassp)* (p. 4322-4326). doi: 10.1109/ICASSP40776.2020.9053558

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015, 04). Long short term memory networks for anomaly detection in time series..

März, C., Werfel, J., Kühne, J., & Scholz, R. (2020, 01). Approaches for a new generation of fast-computing catalyst models: How artificial intelligence and machine learning could revolutionize catalyst simulations. *Emission Control Science and Technology*, *6*. doi: 10.1007/s40825-019-00153-y

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pincombea, B. (2007). Anomaly detection in time series of graphs using arma processes..

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1: Foundations* (p. 318–362). Cambridge, MA, USA: MIT Press.

Schlegl, T., Seeböck, P., Waldstein, S., Schmidt-Erfurth, U., & Langs, G. (2017, 03). Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In (p. 146-157). doi: 10.1007/978-3-319-59050-9_12

Xu, H., Feng, Y., Chen, J., Wang, Z., Qiao, H., Chen, W., … et al. (2018). Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18.* Retrieved from http://dx.doi.org/10.1145/3178876.3185996 doi: 10.1145/3178876.3185996

Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W., … Chawla, N. V. (2018). A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. *CoRR, abs/1811.08055*. Retrieved from `http://arxiv.org/abs/1811.08055`

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., & Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *6th international conference on learning representations, ICLR 2018, vancouver, bc, canada, april 30 - may 3, 2018, conference track proceedings.* OpenReview.net. Retrieved from `https://openreview.net/forum?id=BJJLHbb0-`

APPENDIX

Table 4. Differentiated results for MNIST (LeCun & Cortes, 2010) classification in percent. [1] Y_P false and Y_R correct [2] Y_P and Y_R false

	noise 0.1	noise 0.2	noise 0.3
true positive rate	87.76	84.83	76.14
false negative rate	10.09	11.61	14.84
true negative rate [1]	0.88	1.57	5.51
false positive rate	0.68	0.83	1.12
true negative rate [2]	0.59	1.16	2.39

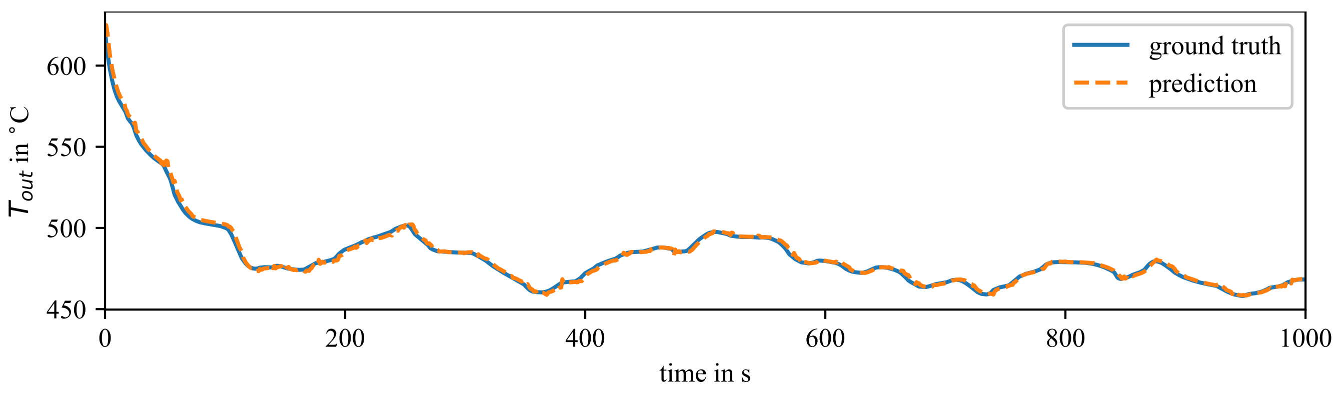

Figure 9. Real and predicted output temperature of the SCR catalyst without any anomalies.

Table 5. Architectures of the Autoencoder and prediction model for the different applications. PyTorch (Paszke et al., 2019) is used for implementation and the hyperparameters for the different layer types (Linear, Convolutional, Pooling, Dropout and Softmax) not displayed in the table are the default values. All models are trained with Adam (D. P. Kingma & Ba, 2015) optimizer and have ReLU as activation function.

dataset	**model**		**layers**	**hidden dim**	**kernel size**
MNIST	**VAE**	Encoder	Linear	512	-
			Linear	256	-
			Linear	4	-
		Decoder	Linear	256	-
			Linear	512	-
			Linear	784	-
	Classifier		Conv.	16	5
			Pool.	-	-
			Conv.	22	3
			Pool.	-	-
			Conv.	22	3
			Pool.	-	-
			Linear	50	-
			Dropout	-	-
			Linear	10	-
			Softmax	-	-
UEA datasets	**VAE**	Encoder	3 Conv.	20	1
			Linear	bottleneck dim	-
		Decoder	Linear	flat dim	-
			3 Tr. Conv.	20	1
	Classifier		2 Conv	64	1
			Dropout	-	-
			Conv.	classes	1
			Linear	1	-
			Softmax	-	-
SCR	**AE**	Encoder	Linear.	12	-
			Linear	10	-
		Decoder	Linear	12	-
			Linear	14	-
	Regression		Linear	64	-
			Linear	64	-
			Linear	1	-

Table 6. Bottleneck dimensions of the VAEs used for the time series classification datasets.

Dataset	**bottleneck dimension**
Articulary Word Recognition	9
Cricket	12
ERing	3
Libras	7
PenDigits	4
UWave Gesture	5

Table 7. Risk and coverage for Classification Tasks for non-anomalous data and for four different anomalies in percent.

Dataset	**evaluation point**	**non-anomal.**		**noise 0.25**		**noise 0.5**		**noise 0.75**		**mirror**	
		risk	cov.	risk	cov.	risk	cov.	risk	cov.	risk	cov.
Articulary Word Recognition	train	0.00	23.33	0.00	5.67	-	0	-	0	-	0
	90-percentile	1.90	55.33	0.84	41.67	-	0	-	0	2.56	13.00
	joint	1.47	91.00	1.09	91.33	1.12	89.67	1.49	89.33	1.62	82.33
Cricket	train	0	20.83	-	0	-	0	-	0	-	0
	90-percentile	0	36.11	0	31.94	-	0	-	0	-	0
	joint	1.82	76.39	1.85	75.00	1.92	72.22	2.27	61.11	2.44	56.94
ERing	train	0	13.70	0	8.89	-	0	-	0	-	0
	90-percentile	4.76	23.33	0	16.67	-	0	-	0	-	0
	joint	6.90	75.19	9.52	77.78	6.50	74.07	8.81	71.48	53.03	48.89
Libras	train	33.96	29.44	-	0	-	0	-	0	56.00	13.89
	90-percentile	38.05	62.78	41.94	17.22	-	0	-	0	63.33	33.33
	joint	41.89	82.22	44.37	83.89	48.78	68.33	52.53	55.00	68.46	72.22
Pen Digits	train	15.64	60.15	23.86	8.75	-	0	-	0	33.33	2.66
	90-percentile	19.32	85.96	23.85	47.34	42.03	1.97	-	0	60.01	15.09
	joint	16.74	90.02	19.40	80.16	28.08	60.26	41.97	44.68	72.64	59.78
UWave Gesture	train	5.26	11.56	0	3.44	-	0	-	0	-	0
	90-percentile	19.55	41.56	12.50	30.00	-	0	-	0	32.00	15.62
	joint	14.10	70.94	14.53	73.12	12.82	73.12	12.73	68.75	53.88	52.81