# Self-Attention as a Predictor of EEG Anomalies

Authors

Affiliations

**Abstract.** One of the main concerns when dealing with electroencephalographic signals (EEG) is assuring that clean data with a high signal-to-noise ratio is recorded. The relevant denoising methods tend to have a narrow scope of application as what is noise for one application might be useful signal for some other application and there no general-purpose approach (or even paradigm) that works best across domains and applications. Machine learning methods are often used for this task, by training Autoencoders and Transformers on reconstruction and prediction, and then assuming the reconstruction/prediction error as an indication of anomalies. These approaches only take into account the morphology of the stream, and are not aware of the different, often highly contextualized, aspects of artifacts.
In this article we explore the novel idea that we can create application-specific artifact detectors by training an attention-based deep neural network and then extracting from the attention layer information about what is ignored. This removes the most pressing challenge of artifact detection, namely that artifacts are vaguely defined and thus difficult to directly supervise, and allows application-specific artifact patterns to be extracted from non artifact-related supervision. We evaluated our method using electroencephalogram (EEG) signals on a sleep-stage labeling task. The performance of the proposed approach was compared against reconstruction/prediction error and against EEG-specific noise detection methods. The results indicate that the proposed method is a promising task-agnostic tool for anomaly detection in streaming data.

**Keywords:** Machine learning · Biomedical data · EEG · denoising

## 1 Introduction

One of the main concerns when dealing with electroencephalographic signals (EEG) is assuring that clean data with a high signal-to-noise ratio is recorded. The EEG signal amplitude is in the microvolts range, and it is easily contaminated with noise, known as artifacts, which need to be filtered from the neural processes to keep the valuable information needed for different applications.

In this domain, an artifact is denoted as any component of the EEG signal not directly produced by human brain activity, making the system register noise that contaminates the neural EEG data. The ability to recognize these artifacts is the first step in removing them. EEG artifacts can be classified depending on their origin, which can be physiological or external to the human body (technical/non-physiological).

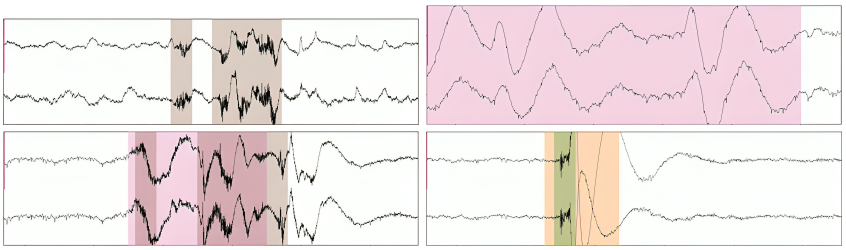

Fig. 1: Examples of the temporal visualization of EEG signals and coloring of noisy segments.

Fig. 1 shows some examples of EEG signals contaminated with noise. In this figure, pink-colored areas are typical of low-frequency noise (0.2-4Hz), which is mainly due to perspiration originating from small drops of sweat produced by the skin glands, which cause changes in the electrical baseline of the electrodes. Brown-colored and green-colored areas are typical of high-frequency noise (30-45Hz) that may originate from electrical activity produced by the muscles when they are contracted, like, for example, muscle tension in the jaw or forehead that can take place when clenching or frowning, respectively. Orange-colored areas are typical of high-amplitude noise that may be due to temporary failures in contact between the EEG sensor and the scalp produced by touching the sensor or by spontaneous changes in electrode-skin contact.

As understood from the above, there is a variety of artifacts, both technical and physiological, each with different characteristics. Furthermore, and in particular regarding the physiological artifacts, what is noise for one application might be useful signal for some other application. It is well-known that there is no general-purpose approach (or even paradigm) that works best across domains and applications and performance depends on the use case and the nature of the encountered anomalies [6].

In the work described here, we present a method for automatically adapting a deep-learned anomaly detector to different domains. In other words, instead of aiming at a general-purpose anomaly detector (which is unatainable for the reasons explained above), we aim at a general-purpose way to train a case-specific anomaly detector without direct supervision. The core of the idea is that anomaly detection is a pre-processing step for some sequence processing task. We operate under the assumption that there is supervision for this downstream task, although there is no supervision for what constitutes an anomaly. We then make the following research hypothesis: *Anomalies are the parts of the sequence that have the property that ignoring them gives superior performance despite the fact that decisions are made from fewer datapoints.*

One can easily see how this maps directly to specific instances of anomalies. For example, assuming a sleep-stage classification task on EEG data, electrical activity produced muscle contraction will have the same morphology regardless of the sleep stage and a successful classifier will learn to ignore it; But the same

patterns might be considered useful signal for a different application of EEG data.

Our contribution is the formulation of a methodology for leveraging the outputs of intermediate layers of deep neural networks in order to extract the level of significance the network places on the different parts of the sequence being learned. In the remainder of this article, we first provide the necessary background (Section 2) and then proceed to describe our methodology (Section 3), which we evaluate on an EEG dataset where we have supervision for both a downstream task (sleep stage prediction) *and* carefully curated anomaly annotations (Section 4). We then present and discuss the experimental results (Section 5) and conclude (Section 6).

## 2 Background

One of the main frameworks in sequence processing are *recurrent neural networks* where the sequence is presented to the network one token at a time and the network maintains a *hidden state* which distills the information needed from past tokens to provide a context for the processing of the current token. One of the most successful recurrent architectures is the *Long Short-Term Memory (LSTM)* where trainable *gates* control the flow of information to and from the hidden state [3]. This allows LSTM to capture long temporal dependencies in a low-dimensional state representation [4,2].

LSTMs are often combined with other techniques, such as convolution and encoder-decoder architectures. Encoder-decoder LSTMs, in particular, learn a compressed representation of the data. When training the decoder to reconstruct the original sequence back (which is known as *autoencoding*), the compressive encoding is trained to drop information that is not detrimental for the loss estimation. Anomalies (in the sense of patterns not encountered during training) can then be identified by higher reconstruction error [7,1]. Reconstruction error over autoencoding has been used extensively for anomaly detection in various architecture besides LSTM [9], and its limitations are well-understood: Autoencoders struggle with high-dimensional data, as it is up to the system designer to find the layer widths (effectively, the level of compression) that drop the correct amount of information. Further, Autoencoders are limited with the respect to the kind of information they drop: Since this decision is driven by a loss that compares the reconstructed sequences against the input sequence, a commonly occurring pattern will be retained even when it is an anomaly in a given context.

Besides recurrency, the other major approach to sequence-processing is the *Transformer* architecture. Transformers are Autoencoders based on the *self-attention* mechanism [8]. Unlike LSTMs and other recurrent architectures, Transformers receive the complete sequence as input and model relationships across the entire sequence. This allows gradients to flow directly across the entire sequence, rather than being propagated step by step, making it easier to discover long-distance dependencies. At the core of the Transformer architecture lies the self-attention mechanism, which enables the model to assign different levels of

importance to elements in the input sequence based on the value of *any other element in the sequence*. Specifically, Transformers learn three sets of weights which are applied to input of dimensionality $d$ to get the *query (Q)* vector, the *key (K)* vector, and the *Value (V)* vector. The attention mechanism computes the *similarity score* $QK^T$ which is a $d \times d$ matrix that determines the contribution of each element in the final representation of each other element. The similarity score is scaled and softmax'ed into a matrix of weights, which are applied to the value vector. This yields the representation $\text{softmax}\left(QK^T/\text{sqrt}(d)\right) \cdot V$ where each element contains information aggregated from the entire sequence, improving the model's ability to detect long-range dependencies.

## 3 Research Methodology

### 3.1 Research Hypothesis

As discussed in the previous section, reconstruction and prediction errors are widely used as key indicators for anomaly detection in time-series data due to their intuitive appeal and straightforward implementation. However, these approaches do not take into account differences in the nature of what is considered an anomaly for each use case.

As we framed our work in a context where there is no anomaly supervision, it follows that it is also not possible to select training data that is not contaminated with anomalies. This means that the model may inadvertently learn to reconstruct anomalies. Additionally, reconstruction-based methods often struggle to detect contextual anomalies, where an observation may be anomalous only in specific temporal or multivariate contexts. For example, a high-temperature reading might be expected in the summer but anomalous in the winter, and reconstruction models may overlook such contextual nuances.

Similarly, prediction error, which measures deviations between predicted and actual values, can be susceptible to noise and non-stationarity in time-series data. In highly dynamic systems, normal variations may result in significant prediction errors, leading to false positives. Moreover, models relying on prediction error often assume that future patterns can be reliably forecasted based on past observations, an assumption that may not hold in volatile or chaotic systems.

On the other hand, the attention mechanism offers an alternative, currently unexplored, way to extract indications about what parts of the sequence are anomalies. The hypothesis is that the part of the sequence that receives the least attention while performing a task relevant to the use case, then is this part is an anomaly. The intuition is that neural networks will happily overfit the data when they are given enough parameters to do so. Such a network might not be good to actually perform the task, but is good at recognizing two ways in which a sub-sequence is anomalous: (a) it does not follow any pattern that was boosted (gradient-wise) by the training; (b) it follows a pattern that is inconsistent, it is sometimes associated with one class on the task and sometimes with an another, so it alternates between being boosted and penalized by the loss function, again resulting in low attention.

Table 1: Methods under comparison.

| *Acronym* | *Architecture* | *Detection* |
|---|---|---|
| *LSTM* | LSTM Autoencoder | Reconstruction error |
| *C-LSTM* | Convolutional LSTM Autoencoder | |
| *AE_err* | Attention-based Autoencoder | |
| *TP_err* | Transformer Predictor | Prediction error |
| *AE_att* | Attention-based Autoencoder | Attention |
| *TP_att* | Transformer Predictor | |
| *MNE* | IIR filter | |

To investigate this hypothesis, we experimented with several methods for anomaly and noise detection ranging from attention-based methods to state-of-the-art machine learning approaches, as well as conventional anomaly detection methods. These are listed in Table 1 and described in more detail in the remainder of this section.

### 3.2 Reconstruction and Prediction Error

The *LSTM Autoencoder (LSTM)* uses a sequence-to-sequence architecture with LSTM layers for both encoding and decoding. The encoder compresses the input into a fixed-size latent representation by processing the input sequence and retaining the final hidden state of the LSTM. This is achieved using an LSTM layer, followed by a dropout layer for regularization, and a fully-connected layer for dimensionality reduction. The decoder then processes this compressed representation using the reverse architecture (FC, dropout, and LSTM) to reconstruct the original data.

In this architecture, fully connected (FC) layers process the final hidden state of the LSTM (the last timestep in the sequence) as a single, comprehensive representation of the entire input sequence. This output is then projected into a lower-dimensional latent space through a linear transformation. While this approach is computationally efficient, it assumes that the LSTM's final hidden state sufficiently captures all relevant temporal dependencies. As a result, it can struggle to retain fine-grained temporal details, especially for data like EEG signals, where localized patterns are crucial.

The *Convolutional LSTM Autoencoder (C-LSTM)* enhances feature extraction by combining LSTM layers with convolutional layers. The key difference between the LSTM and C-LSTM architectures lies in how dimensionality reduction is achieved in the encoder and decoder: the first uses fully connected (FC) layers, while the latter uses Conv1D layers. These operate directly on the

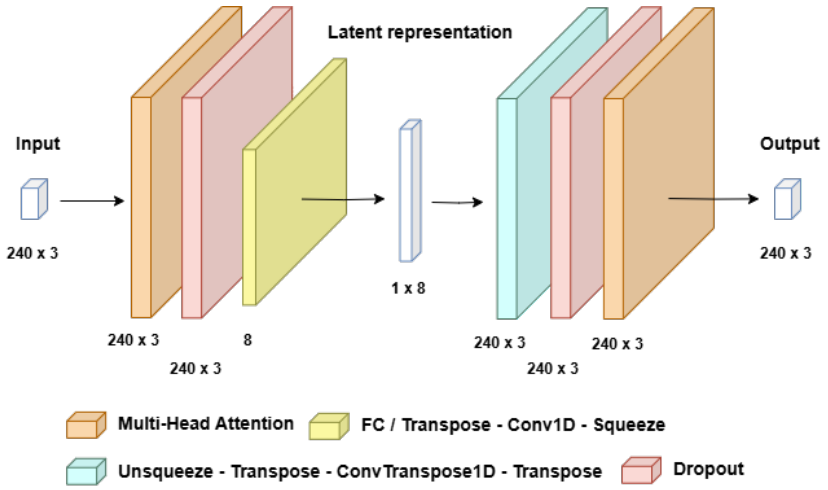


Fig. 2: AE_err/AE_att architecture.

sequence of hidden states produced by the LSTM. By applying a kernel across the temporal dimension, they extract localized patterns and dependencies within the sequence. This convolutional operation integrates information from multiple timesteps, creating a more nuanced and structured representation. After convolution, the output is reduced to a lower-dimensional latent space, where temporal features are preserved and compactly encoded. This method emphasizes localized temporal dynamics while reducing dimensionality.

The *Attention-based Autoencoder (AE_err)* combines convolutional layers with an attention mechanism to reconstruct the input data, again using the convolutional layers to capture local features while replacing LSTM with attention to capture long-term patterns. Fig. 2 illustrates this architecture. The input to the encoder consists of a sequence of 240 measurements from two simultaneous channels, along with a same-size sequence of time representation. The input data passes through the encoder and is compressed into a latent representation of size $1 \times 8$, where the first dimension represents a single compressed time step, and the second dimension corresponds to eight learned features. This is achieved by applying multi-head attention to the input data, followed by a dropout layer for regularization, and a convolutional layer for dimensionality reduction. The decoder then processes this compressed representation using the reverse architecture (transposed convolution, dropout, and attention) to reconstruct the original data, capturing both trends and amplitudes.

In all three autoencoding systems, anomaly detection is based on the assumption that the compressed latent representation preserves only recurring, periodic fluctuations and trends. Atypical spikes, often caused by noise, do not cause enough loss to be worth the space to represent them (in terms of nodes in the latent representation). Therefore, the level of anomaly is estimated as the

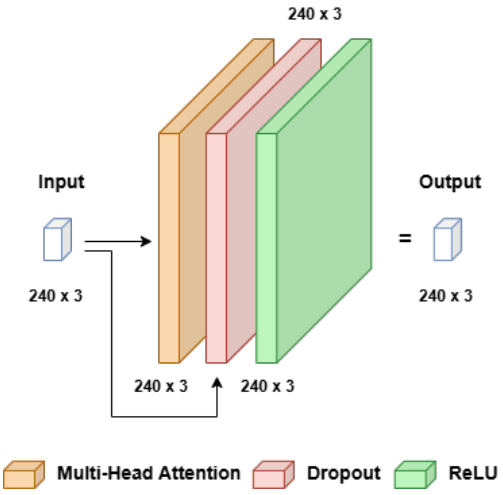


Fig. 3: TP_err/TP_att architecture.

reconstruction error, the difference between the original input sequence and the sequence decoded from the latent representation.

The *Transformer Predictor (TP_err)* method utilizes a Transformer that uses multi-head attention for time-series forecasting. It consists of a sequence-to-sequence architecture, with multi-head attention layers in both the encoder and decoder components. The encoder captures temporal dependencies in the input sequence, while the decoder predicts the next sequence based on these encoded features. Fig. 3 illustrates this architecture. The input consists of a sequence of 240 measurements from two simultaneous channels, along with a same-size sequence of time representation. This data passes through the embedding layer, which applies multi-head attention to capture key temporal patterns. The output is then processed with dropout regularization and passed through a LeakyReLU activation function to introduce non-linearity.

The TP_err method estimates noise through prediction error, i.e. the difference between the model's predicted output and observed values. When the model predicts the next time step in the sequence, a significant difference between the predicted and actual values suggests that the input data may be noisy.

### 3.3 Attention-Based Detection

The exact same architectures as in AE_err and TP_err above are also used for the respective attention-based detection methods *Attention-based Autoencoder (AbAE_att)* and *Transformer Predictor (TP_att)*, except that now anomaly is estimated by the attention weights.

The key idea is that when a part of the sequence receives a low attention weight, it suggests that the information at that moment is likely noisy and, therefore, unimportant for the task.

### 3.4 Conventional Noise Detection

Alongside machine learning methods, conventional techniques provide a reliable alternative for detecting noise in time-series data and are commonly utilized. *Multinomial Noise Exponential filtering (MNE)* is a method used to remove unwanted frequencies from time-series data, typically EEG signals. It applies a bandpass filter to the data, allowing signals within a specific frequency range to pass through while reducing frequencies outside this range.

## 4 Experimental Setup

This study employs a case analysis to validate the efficacy of the proposed noise estimation methods. The focus is detecting noise within electroencephalographic (EEG) signals, recorded via headbands utilized during slumber to monitor various stages of the user's sleep cycle.

In this analysis, EEG signals collected from a headband with two EEG channels synchronized with medical-grade EEG devices are considered. The recorded signal frequency for each channel of the headband is 128Hz. These signals are grouped into consecutive 30-second segments, and each segment is further annotated by three experts, into one of five sleep stages: Wake, N1, N2, N3, and REM. These stages correspond to specific brain activity patterns, such as slow eye movements, sleep spindles, and other characteristic waveforms. The goal of the current analysis is to detect the presence of noise, if any, at any of these segments. In total, 56 recordings were examined. Each recording refers to the EEG signals acquired by the headband of a user during a night-long sleep. These signals are grouped into 30-second consecutive segments.

As a ground-truth method to evaluate the results of the proposed noise estimation methods, the original estimation method used by the headband manufacturer is considered. This method consists of task-specific algorithms to automatically estimate noise and evaluate the overall quality of EEG signals, identifying the artifacts in recordings made using the wearable textile headband.[1]

### 4.1 Data Pre-processing

Data normalization is required for the machine learning methods to ensure robust model training and reliable outcomes. It is performed using statistics derived from the entire training dataset instead of relying on per-batch calculations. The median and *interquartile range (IQR)* are used instead of the mean and standard deviation, respectively, to improve robustness against outliers. The median measures central tendency and is resistant to extreme values, and IQR is a measure of statistical dispersion that represents the spread of the middle

[1] The device manufacturers and data providers are co-authors, so further details are withheld to preserve anonymity. More details about the device and the data (including data access) will be made available in the camera-ready.

Table 2: Absolute number and percentage of segments with artifacts in each channel and recording session.

| RecID | A | B | C | D |
|---|---|---|---|---|
| HB1 | 0 | 57 (5%) | 27 (3%) | 9 (1%) |
| HB2 | 151 (14%) | 56 (5%) | 26 (3%) | 9 (1%) |

50% of a dataset and reduces the influence of outliers by focusing on the range within which the central portion of the data lies.

PCA and MNE require raw data that preserves the characteristics of the original signal, so no normalization is applied for these methods.

To train the machine learning models, the available recordings were randomly split into training (43 recordings), validation (3 recordings), and testing subsets (10 recordings). Among the testing recordings, six had zero or one noisy segments are not reported here as it makes little sense to compare methods on having identified a single datapoint. For the record, only PCA identified the noisy segment in only one of the segments.

Table 2 gives the absolute and relative noise density for the remaining four recordings, arranged in descending order. Specifically, the table gives the count of 30-second segments within which at least one artifact is annotated as noise in the ground-truth labeling. Note that recordings are *not* of equal length.

What is noteworthy is that Recording A has considerably different noise rates between the two channels. Since the signals from the two channels are heavily correlated, this presents an opportunity for the machine learning methods to demonstrate learning a comparative model that exploits the fact that the noisy parts of HB2 are not correlated to their corresponding parts in HB1. However this is a difficult theory to construct, since it requires 'discovering' correlation first.

### 4.2 Training Process and Hyper-parameters

The implementations of the methods in Table 1 that we used in our experiments are publicly available. Our implementation of the machine learning methods have been reposited in Zenodo.[2] MNE is used as implemented and configured in the 'MNE Tools' Python package for exploring, visualizing, and analyzing human neurophysiological data.[3] MNE is a bandpass filter with pre-configured cut-offs specifically targeting EEG data.

[2] To preserve anonymity the Zenodo record is not published but left as a draft; reviewers can access it at https://zenodo.org/records/14842198 (must use this link to include the access token)

[3] Available from `https://mne.tools` and Pypi.

To process the data with the *AE* and *TP* methods, we split each 30-second segment of EEG data (3840 samples per channel) into 16 smaller chunks of 240 samples each. This point is moot for *LSTM*, *C-LSTM*, and *MNE*.

The Autoencoder methods are trained on sequence reconstruction. The Predictor methods are trained on predicting sleep stage. To balance the autoencoders' focus on both the amplitude and trends of the time-series data, we define a custom loss function, called *BlendedLoss*. This function combines the median and mean of the powered absolute differences between the predicted values ($\hat{x}$) and the target values ($x$):

$$\text{Loss} = (1 - b) \cdot \text{median}(|\hat{x} - x|^p) + b \cdot \text{mean}(|\hat{x} - x|^p)$$

where $p$ is the power parameter that controls the sensitivity of the loss to differences and $b$ the *blend factor* that controls the trade-off between learning the overall trend (median error) and closely following local patterns (mean error).

Revisiting our research hypothesis, we expect methods trained on predicting sleep stage to outperform methods trained on autoencoding since they have access to task-specific labels. More generally our methods are ordered as follows in terms of task-specific knowledge they have access to:

1. *MNE* is specifically, expertly designed to detect noise in EEG signals and has proven to be very effective on this task.
2. *AE_ att* and *TP_ att* implement our hypothesis that the attention mechanism can exploit supervision unrelated to anomaly detection to automatically extract task-specific knowledge of anomalies.
3. *LSTM*, *C-LSTM*, *AE_ err* and *TP_ err* are general-purpose autoencoders with minimal access to task-specific knowledge in the form of a task-specific loss function.

This ordering reflects our prior expectation regarding their relative performance. *None of the above (neither machine learning nor manual calibration) has ever had any access to noise annotations in the training data, but only to sleep-stage annotations.*

For the training configuration, we set the batch size to 512 and trained the models for a maximum of 1000 epochs with patience of 30 epochs, meaning that if the validation loss does not improve for 30 consecutive epochs, training will stop. We set the learning rate to 1e-4 and use the Adam optimizer to adjust the model weights. Additionally, we use *ReduceLROnPlateau* scheduling to reduce the learning rate if the validation loss plateaus.

## 5 Results and Discussion

We prepared three validation/testing setups. In the first setup validation and testing scores are calculated on the original 30-sec segments where each segments is annotated as 'noise' if it includes at least one sample marked as 'noise' in the ground-truth annotation. In the 5min and 10min-window setups segments

evaluation was performed on 5min and 10min windows. If a method reports noise on any part of the window and the ground truth also annotates as noisy any part of the window, the window counts as a true positive. Obviously, these are easier tasks than the original 30-sec segmentation on two grounds: (a) positives get less sparse so task becomes less of a needle-in-a-haystack problem, and (b) learners have access to a longer context for the same number of input token, which know to be an important factor in sequence processing [5].

Table 3 presents F-score, Precision, and Recall metrics for the evaluated methods on all three evaluation setups. Since all methods give a numerical estimation and not a binary decision, a threshold was established. The threshold was calculated on the training data, separately for each method, as the threshold that makes 1% of the training data come out as noise. This is the only piece of prior domain knowledge shared by all learners. F-score is calculated for $\beta = 2$, in order place more weight on recall than precision. This is due to the fact that the dataset is very unbalanced and noise instances are rare, even in the 10min-window setup. F-score is adapted accordingly to reflect that recall is harder to achieve than precision.

The first observation is that by comparing the three sub-tables we can immediately see that the three windows behave as expected, with results improving with longer windows. By comparing horizontally from right to left we also see that results improve as noise gets denser.[4] We cannot speculate on whether the top-to-bottom improvement is due to the longer context or due to denser positives. But either way, we have clear indications about how task difficulty scales across the table.

What is directly relevant for our hypothesis are the comparisons between (a) attention-based anomaly detection (AE_att and TP_att) against the error-based anomaly detection from the same models (AE_err and TP_err), and (b) models trained on the sleep stage task (TP) against autoencoded models (AE). TP_att generally outperforms all four TP/AE_err/att combinations, although autoencoding proved better on HB1-C (primarily LSTM and secondarily AE_att) with the gap becoming more pronounced on the denser setups. Although the conditions present in HB1-C are worth investigating, the results show that the ideas presented here are validated and promising.

What is also noteworthy is that TP_att outperforms MNE on HB2-A and HB2-B. Regarding HB2-A it is worth investigating whether the transformer has 'discovered' the lack of correlation between HB1-A and HB2-A on noisy segments. Regarding HB2-B, a possible explanation is that TP_att is more robust to noise density considerably above the hard-wired 1%: it suffers the unavoidable precision loss just as all methods, but it gets a comparatively higher F-score through perfect recall.

[4] Note that HB1-A is out-of-order in this respect, as it has no noise.

Table 3: F2-score ($\beta = 2$) on HB1 (left) and HB2 (right) for different segment lengths. When precision and recall not within three percentile points of the F2 score, they are also given in parenthesis. Empty cells where no segment was marked as noise.

(a) Evaluation on 30sec windows.

| RecID | A | B | C | D | A | B | C | D |
|---|---|---|---|---|---|---|---|---|
| LSTM | | **2%** | | | **2%**(6%,1%) | **4%**(7%,4%) | **4%**(3%,4%) | |
| C-LSTM | | | **4%**(11%,4%) | | | | | |
| AE_att | | **4%** | | | **1%**(20%,1%) | | | |
| TP_att | | **2%** | | | **15%**(13%,15%) | **10%**(5%,14%) | **4%**(2%,8%) | **3%**(1%,11%) |
| MNE | | | | | **2%**(40%,1%) | | | |

(b) Evaluation on 5min windows.

| RecID | A | B | C | D | A | B | C | D |
|---|---|---|---|---|---|---|---|---|
| LSTM | **13%**(6%,18%) | | | | **19%**(12%,22%) | **22%**(8%,38%) | **11%**(3%,35%) | |
| C-LSTM | | | **21%**(10%,30%) | | **1%**(2%,1%) | | | **4%**(1%,11%) |
| AE_err | **16%**(27%,14%) | | | | **4%**(10%,3%) | | | |
| AE_att | **9%**(3%,18%) | **10%**(3%,22%) | | | **7%**(18%,6%) | | | |
| TP_att | **14%**(6%,19%) | | | | **42%**(14%,84%) | **26%**(7%,100%) | **16%**(4%,100%) | **5%**(1%,100%) |
| MNE | | | | **96%**(82%,100%) | **15%**(38%,13%) | | | **14%**(100%,11%) |

(c) Evaluation on 10min windows.

| RecID | A | B | C | D | A | B | C | D |
|---|---|---|---|---|---|---|---|---|
| LSTM | **19%**(7%,37%) | **5%**(2%,11%) | | | **28%**(12%,42%) | **25%**(7%,64%) | **9%**(2%,42%) | |
| C-LSTM | | | **34%**(11%,67%) | | **1%**(1%,1%) | | | **21%**(5%,100%) |
| AE_err | **31%**(30%,32%) | | | | **6%**(8%,5%) | | | |
| AE_att | **12%**(3%,35%) | **26%**(7%,89%) | | | **9%**(14%,9%) | | | |
| TP_att | **21%**(7%,39%) | **2%** | | | **41%**(13%,90%) | **22%**(5%,100%) | **14%**(3%,100%) | **5%**(1%,100%) |
| MNE | | | | **96%**(82%,100%) | **25%**(35%,23%) | | | **96%**(82%,100%) |

## 6 Conclusions and Future Work

This paper investigates the extraction of anomaly indicators from *intermediate* layers of a deep neural network trained on a sequence processing task. By removing the need to train on a sequence reconstruction/prediction task, we are free to train on whatever task-specific supervision is available. This alleviates the most pressing challenge of anomaly detection, namely that anomalies are vaguely defined and thus difficult to directly supervise, while simultaneously allowing task-specific knowledge to be extracted from whatever other (non anomaly-related) supervision might be available.

Our ideas are positively validated on EEG signals sourced from wearable sleep-monitoring devices. This dataset has the great advantage of having both task supervision (sleep stage detection) and noise supervision, both expertly annotated by the device manufacturer. Our current findings already indicate that the attention layer of a Transformer trained on the sleep stage task exhibits promising performance when used as an anomaly detector.

Subsequent steps will pursue two directions: exploring the current experimental setup at more depth and expanding the breadth of the results with diverse datasets and tasks. Regarding the first direction, we mostly plan to investigate how the components of the attention layer (Q, K, and V) interact in order to understand whether a subset of these components is more tightly fitted on recognizing patterns that should be ignored. Another investigation would be to understand if the performance on HB2-A really indicates that the Transformer has 'discovered' the lack of correlation between HB1-A and HB2-A on noisy segments or is incidental. Despite the inherent difficulty in trying to understand the inner workings of neural networks even for single-layer attention, as the the field of *explainable machine learning* evolves the required tools are starting to materialize. Regarding the second direction, the most immediate goal is to jointly train on multiple tasks, which we expect to help the network distinguish between noisy segments and segments that happen to be irrelevant for a given task.

## References

1. Basora, L., Olive, X., Dubot, T.: Recent advances in anomaly detection methods applied to aviation. Aerospace **6**(11) (2019). https://doi.org/10.3390/aerospace6110117
2. Duja, K.U., Khan, I.A., Alsuhaibani, M.: Video surveillance anomaly detection: A review on deep learning benchmarks. IEEE Access **12** (2024). https://doi.org/10.1109/ACCESS.2024.3491868
3. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8) (1997). https://doi.org/10.1162/neco.1997.9.8.1735
4. Hojjati, H., Ho, T.K.K., Armanfard, N.: Self-supervised anomaly detection in computer vision and beyond: A survey and outlook. Neural Networks **172** (2024). https://doi.org/10.1016/j.neunet.2024.106106
5. Lee, M.C., Lin, J.C., Gran, E.G.: How far should we look back to achieve effective real-time time-series anomaly detection? In: Advanced Information Networking and Applications. Springer (2021). https://doi.org/10.1007/978-3-030-75100-5_13

6. Mejri, N., Lopez-Fuentes, L., Roy, K., Chernakov, P., Ghorbel, E., Aouada, D.: Unsupervised anomaly detection in time-series: An extensive evaluation and analysis of state-of-the-art methods. Expert Systems with Applications **256** (2024). https://doi.org/10.1016/j.eswa.2024.124922
7. Sgueglia, A., Di Sorbo, A., Visaggio, C.A., Canfora, G.: A systematic literature review of iot time series anomaly detection solutions. Future Generation Computer Systems **134**, 170–186 (2022). https://doi.org/10.1016/j.future.2022.04.005
8. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems 30 (2017), http://papers.nips.cc/paper/7181-attention-is-all-you-need
9. Zamanzadeh Darban, Z., Webb, G.I., Pan, S., Aggarwal, C., Salehi, M.: Deep learning for time series anomaly detection: A survey. ACM Comput. Surv. **57**(1) (Oct 2024). https://doi.org/10.1145/3691338