IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 April 2024, accepted 15 June 2024, date of publication 25 June 2024, date of current version 23 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3418779

RESEARCH ARTICLE

Robust Representation Learning via Sparse Attention Mechanism for Similarity Models

ALINA ERMILOVA[1], NIKITA BARAMIIA[1], VALERII KORNILOV[1], SERGEY PETRAKOV[1], AND ALEXEY ZAYTSEV[1,2]

[1]Skolkovo Institute of Science and Technology, 121205 Moscow, Russia
[2]Risk Management, Sber, 121165 Moscow, Russia

Corresponding author: Alina Ermilova (A.Rogulina@skoltech.ru)

The work was supported by the Analytical center under the RF Government (subsidy agreement 000000D730321P5Q0002, Grant No. 70-2021-00145 02.11.2021).

ABSTRACT The attention-based models are widely used for time series data. However, due to the quadratic complexity of attention regarding input sequence length, the application of Transformers is limited by high resource demands. Moreover, their modifications for industrial time series need to be robust to missing or noisy values, which complicates the expansion of their application horizon. To cope with these issues, we introduce the class of efficient Transformers named **Regu**larized Trans**former**s (Reguformers). We implement the regularization technique inspired by the dropout ideas to improve robustness and reduce computational expenses without significantly modifying the pipeline. The focus in our experiments is on oil&gas data. For well-interval similarity task, our best Reguformer configuration reaches ROC AUC 0.97, which is comparable to Informer (0.978) and outperforms baselines: the previous LSTM model (0.934), the classical Transformer model (0.967), and three recent most promising modifications of the original Transformer, namely, Performer (0.949), LRformer (0.955), and DropDim (0.777). We also conduct the corresponding experiments on three additional datasets from different domains and obtain superior results. The increase in the quality of the best Reguformer relative to Transformer for different datasets varies from 3.7% to 9.6%, while the increase range relative to Informer is wider: from 1.7% to 18.4%.

INDEX TERMS Deep learning, efficient transformer, robust transformer, representation learning, similarity learning.

I. INTRODUCTION

Drilling wells is a time- and money-consuming process that plays an essential role in the exploration of a basin and its characteristic and prevention of drilling accidents [1]. Well and well-interval similarity can help make decisions about drilling reasonableness: it helps reconstruct the properties of an oil well by comparing it to other wells with known properties. Moreover, with a proper similarity learning approach, we can obtain low-dimensional representations of well-intervals that can help estimate their lithological and physical properties by having an appropriate well-interval representation of known wells and a new one.

The associate editor coordinating the review of this manuscript and approving it for publication was Brian Ng.

Many methods that estimate well and well-interval similarity exist as described in Section II, among which approaches dealing with sequential data structures seem the most promising. During drilling, engineers gradually collect data along the well in a sequential manner, producing a multivariate sequence of observations along the well's depth. The sequential nature of data encourages using an architecture that exploits its structure for maximum benefit. Recurrent Neural Networks (RNNs) have previously been used to obtain representations of well-intervals [2]. Despite the widespread use of RNNs, the attention mechanism has made a significant breakthrough in the field of natural language processing (NLP), leading to the creation of the Transformer architecture, which avoids the common problems of RNNs [3]. The most evident benefits are a better

long-term memory for our model and the equal treatment of whole sequences without focusing on the end of a sequence inherent to RNNs [4].

However, the processing of long sequences by Transformers requires significant computational time and memory resources and is prone to errors if the quality of input data is low [5], [6]. Moreover, the classical Transformers tend to be overconfident in their predictions, which leads to less robust models to noised or perturbed data. Recent approaches [7], such as Informer [8], Performer [9], DropDim [10], and LRformer [11] address these problems by proposing several improvements to the basic Transformer architecture, allowing to focus attention only on the relevant part of a sequence, more details are provided in Section III. Nevertheless, the robustness of all the existing time series models remains debatable. Some papers claim that Transformers are ineffective and of low quality, e.g., the authors of [12] show that simple linear-based models outperform Transformers in time series forecasting. Nonetheless, the authors of [13] propose Transformers' modification for time series data, which includes a patching strategy combined with a Transformer's backbone. This allows to outperform simple linear models proposed in [12].

So, these articles confirm that Transformers may perform poorly in some tasks, which also highlights the need for robust models. In addition, in our problem statement, expert labeling is not required as we can reformulate the task in terms of well-interval similarity instead of well similarity, which leads to the self-supervised nature of our method. This confirms the applicability of our approach across domains. Our experiments with data from three different areas support this statement.

Given all this, we present our Transformer-based approach to similarity learning for time series data. The main contributions of this work are as follows:

1) We propose a new regularized variant of the Transformer architecture, *Reguformer*, working in the self-supervised regime. This model adopts the dropout ideas to create more accurate, efficient, and robust models for time series data processing.
2) The basic idea of these models, attention dropout, makes the models both more efficient and more robust, which leads to the increase of the Reguformer's performance on missing or noisy data.
3) One of the requirements for our model is working in the self-supervised regime, so expert labels are not needed during our model's training phase. However, they can be applied to fine-tuning. This architecture improves existing results for time series similarity estimation and representation learning tasks.
4) We adapt all modifications of the Reguformer for the oil&gas data and evaluate them on the problem of similarity estimation between well-intervals. Our models significantly improve existing results, proving the usefulness of Reguformers for processing logging data for oil&gas wells. This follows from a better similarity estimation and overall better representations.
5) To confirm the advantages of our Reguformers, we test them on the three additional datasets consisting of multivariate time series from incredibly different areas: finance, weather, and crimes. The results of time series similarity estimation and Reguformers' robustness prove their great utility.

II. RELATED WORK

To compare wells, we consider well logging information. It is represented as sequential data of petrophysical numerical properties recorded by lowering a variety of sensors into an oil well [14].

The simplest way to estimate the correlation between wells is a rule-based approach. The authors of [15] rely on prior expert knowledge to identify logical rules for the well-to-well correlation. In [16] and [17], the use of geometric distances and Jaccard and Overlap similarities are proposed for wells' similarity calculation. The authors of [18] adopt synchronization likelihood and visibility graph similarity for well similarity assessment. However, most of these methods consider oversimplified physical well properties and are unproductive in terms of obtaining well or well-interval representations. Moreover, they work with only two features, which may not be enough to get a complete picture of the wells' lithological and physical properties. It is reasonable to assume that using a larger number of features will reveal more complex data dependencies, as well logs consist of many characteristics.

The experiments presented in [14] show better performance of classical machine learning models than rule-based approaches. The model based on gradient boosting shows a sufficiently high quality with PR AUC 0.943 and ROC AUC 0.861. Although the predictions can be used for distance calculation and clustering, representations reflecting the geographical and physical characteristics of the wells are impossible to obtain in this case due to the model's architecture.

Deep neural networks are useful for generating well-interval representations (embeddings). The authors of [2] obtain well-interval representations via an LSTM. In [19], the authors implemented a transformer-based approach to well diagnosis. In [20], autoencoder-based approaches are considered for generating data representations applicable to different oil&gas field problems. The authors of [21] adapt non-contrastive approaches like BYOL and Barlow Twins for logging data representations. In [22], the authors train transformers to predict the wells' productivity. This approach is claimed to allow the model to be transferred to another well, increasing the accuracy of the bottomhole pressure or flow rate evolution. There are also studies devoted to the adoption of CNN [23] and autoencoders [24] for oil flow rate prediction.

As mentioned above, well logs can be considered sequential data, specifically, time series. The classical approaches

to estimating the similarity without additional information about separate time series come from unsupervised learning. Among all the models in this area, autoencoders [25] seem the most promising in terms of learning data representations. RNNs such as GRU [26] and LSTM [27] are also a popular group of methods. However, all unsupervised learning methods suffer from inaccurate results because of the absence of labels.

The interpretability of models in the industry is as crucial as the quality and the usage to downstream tasks. Frequently used RNNs such as LSTM have a bottleneck related to the concentration of all information in a single vector that follows the encoder in sequence to sequence models [4]. However, we can overcome the difficulties associated with this problem by changing RNNs to the Transformer model as the attention matrix provides deep insight into the object and its internal relations [28]. Moreover, attention exposes the contribution of each input to each specific output of the model. In particular, scores of attention to input areas or intermediate objects are interpreted as a measure of the contribution of the visited element to the output of the model [29].

Self-supervised models perform better than unsupervised ones, producing a more aligned outcome with the current task. The two main ideas of self-supervised learning are the following:

1) Generative. The most prominent representative is autoregression [30], which predicts the future using representations. Autoencoders considered above can also be used in the context of self-supervised learning.
2) Contrastive. The key idea of these approaches is that similar objects should have similar representations, and dissimilar objects should have distant representations. The most common loss functions in this class are Siamese and Triplet.

In this paper, we focus on contrastive learning approaches and address the following problems of the classical Transformer model: 1) the high computational cost of the attention matrix; and 2) the poor performance of the classical Transformer on middle- and short-sized sequences [31].

The main approaches to reduce attention's computational expenses include attention matrix sparsification, for example, Informer [8] and BigBird [32], attention matrix low-rank representation as presented in Performer [9] and Linformer [33], memory downsampling as in Longformer [34], bucket processing application, for instance, Reformer [35]. More details are available in the recent review [7], and some performance comparisons can be found in [36].

The papers [10] and [11] address the problem of the Transformer's robustness and overconfidence and solve them via the adaptation of the dropout technique and approximation of the attention formula in the way it satisfies the bi-Lipschitz condition.

However, the abovementioned solutions have not been applied yet to similarity and representation learning tasks.

In this paper, we propose the Regularized Transformer (Reguformer) class, adopt models from this class for solving the well-interval similarity task, and compare different regularization techniques with the baselines, namely, the classical Transformer and its recent efficient most promising modifications Informer, Performer, LRformer, and DropDim. Moreover, some papers address other problems of the vanilla Transformer, such as overfitting and overconfidence. We also compare Reguformers with the representatives of this class of models, DropDim [10] and LRformer [11].

Despite the existence of the papers [31], [37] creating the models capturing both long- and short-term patterns in sequences, they lack the adoption of the classical Transformer architecture for working with short- and middle-sized sequences. We expect to fill this gap.

Upon thorough analysis, we see that these efficient Transformers have not been developed for and applied to time series similarity, especially well-intervals similarity, problems due to challenges related to the length of time series intervals and the low amount of data used. We expect to fill this gap by proposing an efficient and interpretable attention-based model trained in a self-supervised manner.

III. METHODS

A. DATA OVERVIEW

1) MAIN DATASET: WELL LOGS

As the main dataset, we consider an open-access dataset provided by the New Zealand Petroleum & Minerals Online Exploration Database [38] and the Petlab [39]. It consists of wells from the Taranaki basin, New Zealand. We use the data from the largest formation, Urenui, containing the information about 28 wells.

In our work, we consider similarity not between whole wells but between well intervals of length 100 which corresponds to 33 ft. According to the experts [2], such selection allows appropriate aggregation of the local properties of rock.

In all our experiments, we use the following four features: porosity inferred from density log (DRHO), density log (DENS), gamma-ray (GR), and sonic log (DTC). These features are also approved by the experts [2].

2) ADDITIONAL DATASETS

To prove the performance quality and robustness of our Reguformers, we use three more datasets from the different domains:

- Crimes.[1] This dataset [40], [41], [42], [43] contains the records of crimes in 12 Boston's districts. In our experiments, we consider four features: codes of crimes (OFFENSE_CODE), shooting indicator (SHOOTING), latitude (Lat), and longitude (Long) of incidents. We formulate the similarity task as determining whether two crime sequences belong to the same district (DISTRICT feature in the dataset) or not.

[1] https://www.kaggle.com/datasets/AnalyzeBoston/crimes-in-boston

- Weather.[2] We consider the climate data [44], [45], [46] from 242 weather stations. We train our models to identify whether the intervals of two weather station logs belong to the same station (STATION feature) using the following features: MXWDSP, WDSP, TEMP, STP, SLP, PRCP, DEWP, LATITUDE, LONGITUDE, ELEVATION.
- Stocks.[3] Stock market data [47], [48] consists of stock prices of 33 companies. We solve the problem of determining whether two intervals of stock data belong to the same company (Name column) using five features: prices at the beginning (Open) and at the end (Close) of each day, the minimum (Low) and the maximum (High) prices during each day, and the number of stocks sold (Volume).

We set the sequence length to 100 for all datasets during the interval generation procedure. The detailed description of data preprocessing is provided in Appendix A.

B. METHODOLOGY

1) SIMILARITY PROBLEM STATEMENT

In this Section, we consider in detail the oil wells similarity problem statement. For the other datasets, we use a similar procedure, the main difference being the number of features.

We look at intervals of wells. Each interval consists of four features measured at 100 consecutive points (depth-wise) during drilling. We aim to provide a model that can report a similarity between two intervals regarding their physical properties. As exact labeling on similarities is rarely available, we use both direct and indirect approaches to estimate the quality of obtained models.

To train the model, we focus on the well-linking problem [2]: we consider a pair of intervals as positive if they belong to the same well (target value for this pair is equal to 1) while a pair of intervals is negative if they are from two different wells – target value, in this case, is equal to 0. So, we do not require any labels for training for such labeling. See the problem statement for model training in Figure 1. The scope of application of the models trained via this approach is wider than distinguishing intervals from different wells, as we hope to obtain representations useful for other problems.

2) NEURAL NETWORK ARCHITECTURES

We use two loss functions in our experiments: the Siamese and Triplet losses depicted in Figures 2 and 3, respectively. The Siamese loss requires a pair of intervals with a label indicating whether they belong to the same well or not, while the Triplet loss takes as an input a triplet of intervals: an anchor interval, a positive (an interval from the same well as the anchor), and a negative (an interval belong to a different).

[2]https://confluence.ecmwf.int/display/CKB/ERA5%3A+data+documentation
[3]https://www.kaggle.com/datasets/szrlee/stock-time-series-20050101-to-20171231

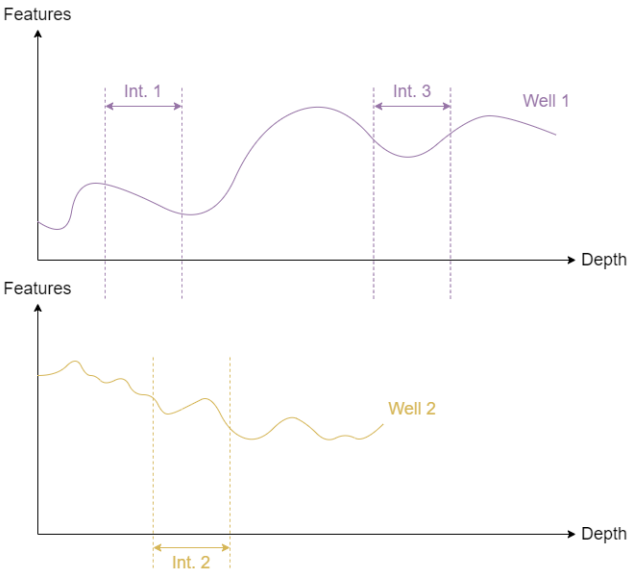


FIGURE 1. Well-linking problem statement. Interval (Int.) 1 and Int. 3 are considered similar, so the prediction for this pair should be as close to one as possible. Intervals 1 and 2 are from different wells, and their similarity prediction should equal zero.

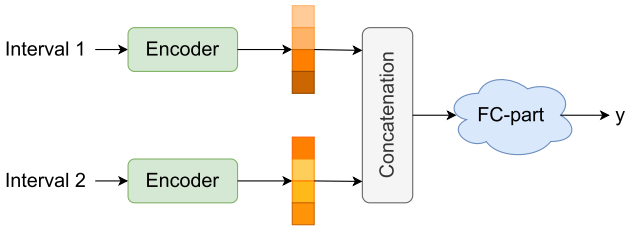


FIGURE 2. Siamese loss function. It can use an additional fully-connected part (FC-part) to predict the target similarity for a pair of intervals or Euclidean or cosine distance instead.

In our case, the Siamese loss is represented by the binary cross entropy (BCE) between the target and the predictions:

$$BCE = -\frac{1}{n}\left(\sum_{i=1}^{n} y_i \log p_i + (1 - y_i) \log (1 - p_i)\right), \quad (1)$$

where y_i are the target values, p_i are the predicted probabilities, n is the number of objects in the data.

For the Triplet loss, we use the standard formula with the Euclidean distance:

$$L = \max\left(||\mathbf{a}_i - \mathbf{p}_i||_2 - ||\mathbf{a}_i - \mathbf{n}_i||_2 + \text{margin}, 0\right), \quad (2)$$

where $\mathbf{a}_i$, $\mathbf{p}_i$, and $\mathbf{n}_i$ are embeddings of anchor, positive, and negative well-intervals, respectively. Following the research [2], we set the margin to 1.75 for all datasets.

As encoders for these architectures, we propose to use different variants of Transformers: our Reguformer with different regularization strategies and baselines such as the basic Transformer, Informer, Performer, DropDim, and LRformer. Details on the architectures are provided below.

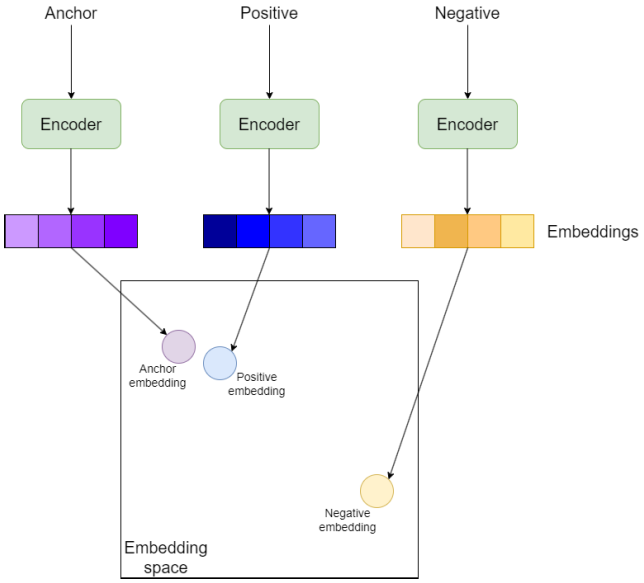


FIGURE 3. Triplet architecture with the desired distribution of embeddings in the embedding space.

a: TRANSFORMER

It is a neural network architecture introduced for NLP tasks [3]. Transformers outperform alternative recurrent and convolution models in many problems. Besides recursion avoidance that speeds up calculations via parallel computation during both training and inference steps, the authors introduced a self-attention mechanism in the encoder part, which we utilize as a backbone for the classification task.

The model's input X is a sequence of length L described by the d_0 features ($X \in \mathbb{R}^{L\times d_0}$). Transformer processes it via N layers: the input to the first layer is X. Inputs to the following layers are outputs of respective previous layers. The input sequence length is L for all layers. Each layer consists of an attention block and a position-wise feed-forward block (or feed-forward network, FFN) and is followed by layer normalization with a skip connection.

The attention block is a Query-Key-Value (QKV) model $\mathcal{A}(Q, K, V)$, which helps utilize the connections between different points in a sequence:

$$\mathcal{A}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V, \tag{3}$$

where $Q, K, V \in \mathbb{R}^{L\times d}$ are the query, key, and value matrices, respectively, and d is the embeddings' dimension. In the encoder, $Q = XW^Q$, $K = XW^K$, $V = XW^V$, where X is the output from the previous layer. Matrix multiplication of Q and K followed by softmax gives the attention matrix A. It consists of weights for V matrix elements that share information between different words (or log data from different well levels). Thus, the elements of the attention matrix provide information about relations between different parts of a sequence. Higher attention results from closer correspondence between a key and a query, and, thus, reflects closer correspondence between different elements of a sequence. The division by $\sqrt{d}$ mitigates the vanishing gradient problem. We can rewrite the formula for the i^{th} query. According to [49], a kernel smoother in probability form defines the i^{th} query's attention matrix [8]:

$$\mathcal{A}(\mathbf{q}_i, K, V) = \sum_j \frac{k(\mathbf{q}_i, \mathbf{k}_j)}{\sum_l k(\mathbf{q}_i, \mathbf{k}_l)}\mathbf{v}_j = \sum_j p(\mathbf{k}_j|\mathbf{q}_i), \tag{4}$$

where $\mathbf{k}_j$ is the j^{th} column of the keys matrix K, $k(\mathbf{q}_i, \mathbf{k}_j) = \exp\left(\frac{\mathbf{q}_i\mathbf{k}_j^T}{\sqrt{d}}\right)$ is the asymmetric exponential kernel and $p(\mathbf{k}_j|\mathbf{q}_i) = \frac{k(\mathbf{q}_i,\mathbf{k}_j)}{\sum_l k(\mathbf{q}_i,\mathbf{k}_l)}$.

However, it is beneficial to utilize not one attention function but several with different projection matrices to retrieve different kinds of information from the data, so MultiHead attention was introduced:

$$\mathcal{MA}(Q, K, V) = \text{concat}(\text{head}_1, \ldots, \text{head}_\text{H})W^O, \tag{5}$$

where $\text{head}_\text{i} = \mathcal{A}(XW_i^Q, X\ W_i^K, X\ W_i^V)$.

Processing the sequence as a whole avoids the problem of forgetting past information, which is typical for RNNs.

The position-wise FFN is a fully-connected neural network, which is applied to each element of the sequence separately and uniformly:

$$\text{FFN}(x) = \max(0, x\ W_1 + b_1)W_2 + b_2. \tag{6}$$

b: INFORMER

The main novelty of the Informer [8] encoder part is the sparse self-attention mechanism *ProbSparse*. The idea is to use only the queries (4) for which $p(\mathbf{k}_j|\mathbf{q}_i)$ is far from zero. Again, let Q, K and V be the query, key and value matrices respectively, $\mathbf{q}_i$, $\mathbf{k}_i$, $\mathbf{v}_i$ be the i^{th} rows of these matrices. The authors propose to use the following sparsity measure to select top queries $\bar{Q}$:

$$\mathcal{M}(\mathbf{q}_i, K) = \log\sum_j k(\mathbf{q}_i, \mathbf{k}_j) - \frac{1}{L}\sum_j k(\mathbf{q}_i, \mathbf{k}_j). \tag{7}$$

Using query sparsity measurement (7), we can get top queries and define self-attention similar to the original work: *ProbSparse* self-attention is

$$\mathcal{A}(Q, K, V) = \text{softmax}\left(\frac{\bar{Q}K^\top}{\sqrt{d}}\right)V, \tag{8}$$

where $\bar{Q}$ is created only from top queries. So, for a single head, we obtain a sparse attention matrix. As we have multiple heads, we avoid severe loss of information via *ProbSparse* self-attention.

The following equation determines the number of selected rows in the query matrix:

$$\sqcap = c\ln L, \tag{9}$$

where L is a sequence length and c is a constant sampling factor, which is a hyperparameter.

TABLE 1. The eight Reguformer modifications. The last column refers to the attention block formula; each regularization requires its own C for the general formula provided in the column heading. $\bar{\cdot}$ and $\widetilde{\cdot}$ denotes top and random elements selection, respectively. The regularization implements both for train and inference.

Notation	Description	$\mathcal{A}(Q,K,V) = \text{softmax}\left(\frac{C}{\sqrt{d}}\right)V$
randQ	Random Queries	$C = \widetilde{Q}K^\top$
randK	Random Keys	$C = Q\widetilde{K}^\top$
randQ_randK	Random Queries and Keys	$C = \widetilde{Q}\widetilde{K}^\top$
topQ	Top Queries	$C = \bar{Q}K^\top$
topK	Top Keys	$C = Q\bar{K}^\top$
topQ_topK	Top Queries and Keys	$C = \bar{Q}\bar{K}^\top$
topQ_randK	Top Queries Random Keys	$C = \bar{Q}\widetilde{K}^\top$
randQ_topK	Random Queries Top Keys	$C = \widetilde{Q}\bar{K}^\top$

Generally, computation complexity for each query-key lookup decreases from quadratic $O(L^2)$ to $O(L \ln L)$, the layer memory usage from $O(L^2)$ to $O(L \ln L)$.

c: REGULARIZED TRANSFORMER (REGUFORMER)

We propose a generalized class of models, Regularized Transformers, consisting of eight models. All these models inherit the attention mechanism from the Transformer but implement it more efficiently in terms of memory consumption and computational cost. Our variants of Reguformers have the computational and memory complexity equal to either $O(L \ln L)$ in cases of top queries/keys selection or $O(\ln^2 L)$ if we choose queries/keys randomly. Table 1 contains aggregated information about each Reguformer modification. To select queries or/and keys, we, first, calculate the number of elements that should be selected via the formula (9), and, second, use random sampling in random selection case or/and the sparsity measure (7) to select $\sqcap$ objects.

The Reguformer architecture allows the adoption of the dropout of the attention matrix and makes a wide class of models with the simplest regularization as a random choice of queries or/and keys and with a more complicated technique as the sparsity measure. The Informer model is one of the representatives of the Reguformers class. It is worth noting that we consider the regularization of the query and key matrix and omit the regularization of the value matrix, as this procedure is equivalent to implementing dropout ideas to at least one of the other matrices.

The introduction of the sparsification mechanism and adoption of dropout ideas not only decreases the computational and memory complexity but also increases the models' robustness. The dropout technique [50] is proposed initially to cope with neural networks' overfitting problem and handling missing values. Implementing these ideas allows for the wide usage of Reguformers in the industry.

3) ADDITIONAL BASELINES

a: PERFORMER

The Performer architecture [9] utilizes the *Fast Attention Via positive Orthogonal Random features (FAVOR+)* mechanism that provably achieves linear time and space complexity in L of full-rank attention matrix estimation at any precision. Moreover, it does not rely on prior assumptions about matrix structure.

FAVOR+ mechanism can be summarised in the following way. Attention matrix $\mathcal{A}$ may be considered a kernel matrix with a kernel $k(x, y) = \exp(x^\top y)$. This allows it to be estimated by an unbiased approximation with a random feature map ϕ_{trig}:

$$\mathcal{A}(Q, K, V) = \hat{D}^{-1}(Q'((K')^T V)) \tag{10}$$

$$\hat{D} = \text{diag}(Q'((K')^T 1_L) \tag{11}$$

$Q', K' \in \mathbb{R}^{L \times r}$ have rows $\phi(q_i^T)^T$ and $\phi(k_i^T)^T$ respectively. Random feature maps have to be positive, regularized, and orthogonal to achieve a stable estimation of the attention matrix with low variance. This approach results in exponentially small bounds on large deviation probabilities and achieves $O(Ld^2 \log d)$ complexity.

Multiple existing implementations of Performer can be found online.[4,5,6] Nevertheless, the theory presented in the paper [9] may not only be applied to increase the performance of Transformers but also to expand the Transformer architecture class and even go beyond their scope. For example, random feature methods can be used for a broader family of kernels [51], [52].

b: DropDim

To increase the vanilla Transformer's robustness, the authors of [10] integrate the dropout idea into the self-attention mechanism. DropDim eliminates embedding dimensions with a certain probability. The authors present two methods for their elimination: random and span. In random mode, embedding dimensions are dropped independently, while span mode includes randomly choosing the starting dimension and eliminating the predefined number of elements after this starting point. Moreover, the authors show a significant increase in quality when combined with a label smoothing technique [53]. In our experiments, we optimize hyperparameters with (only for Siamese models as Triplet loss does not require any labels) and without label smoothing and optimize the DropDim mode as a hyperparameter.

[4]https://github.com/lucidrains/performer-pytorch
[5]https://github.com/xl402/performer
[6]https://github.com/nawnoes/pytorch-performer

c: LRformer

The vanilla Transformers tend to be overconfident in their predictions. The authors of [11] propose a regularization method based on the Lipschitz bound. The key idea of their modification is in the self-attention calculation: instead of the classical formula presented in (3), they try to approximate it in the way that it satisfies the bi-Lipschitz condition:

$$\mathcal{A}(Q,K,V) = \text{softmax}\left(-\alpha \frac{||Q||^2_{\text{row}} - 2Q^TK + ||K||^2_{\text{col}}{}^T}{||Q||_F||X^T||_{(-\infty,2)}}\right)V, \quad (12)$$

where $||\cdot||_F$ and $||\cdot||_{(-\infty,2)}$ denote the Frobenius and $(-\infty, 2)$-norms, respectively.

4) ATTENTION ANALYSIS

To investigate sensitivity to inputs and identify the most important inputs, we can consider the attention matrix A. We evaluate the attention matrix interpretability using the Transformer model in the following ways:

- Calculate the correlation between the attention weights and the models' gradients. According to [54] and [55], the attention weights seem to be uncorrelated with the gradients. Moreover, as stated in [55], Transformer-based models are resistant to removing random interval parts, while it is more sensitive to deleting parts with the smallest attention weights. However, in these papers, the authors consider only simple architectures, while the interpretability of the larger models based on attention remains an open question.
- Eliminate parts of intervals and examine the models' quality.

  If excluding the most important (according to some criterion) measurements reduces the quality more significantly than a random drop, we can say that the criterion reasonably identifies the most important features. We consider the following criteria for the i-th element of a sequence:

  1) The smallest attention weights $q_i^\top k_i$. To select top-k queries, we consider the diagonal elements of the attention matrix: we find an index of the k smallest elements and drop the row with this index in the original well-interval. We take the sum of attention matrices over all layers and all heads.
  2) The highest gradient with respect to the inputs. We select the top-k model's gradients, find their indices, and delete elements in the original well-interval according to these indices.

  We use zeros and values from the normal distribution as masks for the masking part when we fill gaps with random values instead of dropping them completely.

IV. EXPERIMENTS AND RESULTS

Our experimental evaluation consists of several parts that sequentially answer the following questions:

- Does the usage of models based on transformer architectures improve the similarity models compared to RNNs used previously?
- Does the introduction of the regularization technique improve the models' similarity estimation?
- How efficient and robust is the proposed approach? In what way is it better than recurrent architectures?
- Can embeddings produced by our Reguformers be reused to solve problems other than similarity estimation?
- Does attention add to the interpretability of the model by providing a new way to conduct sensitivity analysis?

In all our experiments, we use the following notation for the regularization techniques of the Reguformer:

$$\text{reg\_type}_Q Q\_\text{reg\_type}_K K, \quad (13)$$

where the first part, $\text{reg\_type}_Q Q$, and the second part, $\text{reg\_type}_K K$, stands for the regularization type of query and keys matrices, respectively. The two variants for both reg_type_Q and reg_type_K are "top" and "rand", which refers to the top rows selection via sparsity measurement proposed in [8] or to random selection, respectively. If the modification includes only query or key matrix regularization, the other part in the regularization notation is skipped.

The code for all our experiments is available at `roguLINA/Reguformer`.[7]

A. MODELS QUALITY

This Section compares the quality of our models for the similarity problem. Following the research [2], for both Siamese and Triplet models, we calculate the Euclidean and cosine distances (Eucl.dist. and Cos.dist.) to obtain the probabilities. In the Siamese case, we also use the three fully-connected layers (3 FC). More details of the models' training can be found in Appendix B.

We consider the area under the receiver operating characteristic (ROC AUC), the area under the precision-recall curve (PR AUC), and the F1-score as the most representative metrics and use them for models' evaluation. The first one was used during the hyperparameter optimization with Optuna.[8] The detailed description of the hyperparameter optimization procedure is presented in Appendix B.

Figure 4 demonstrates the models' quality during the hyperparameter search for the well-linking task. Each point represents a model with a selected hyperparameter set. The variants of our Reguformer show scores comparable to the classical Transformer and to another efficient Transformer modification, the Performer. Moreover, some of the regularization techniques outperform all baselines.

We evaluate the models' quality via cross-validation with five splits. For each split, we train our model with 25000 pairs in the case of Siamese-based approach or triplets in the case

[7] https://github.com/roguLINA/Reguformer
[8] https://github.com/optuna/optuna

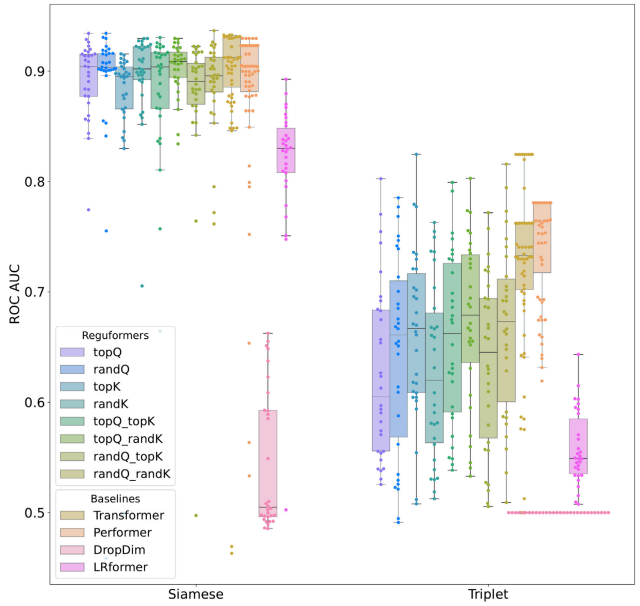


FIGURE 4. ROC AUC scores of Regularized transformers (Reguformers) and the baselines: Transformer, Performer, DropDim, and LRformer in Siamese and triplet architectures during hyperparameter optimization for well-linking problem. Each point represents the model's quality with a selected hyperparameter set. For DropDim, we also implemented label smoothing for the Siamese loss function. However, we omit this boxplot due to its poorer quality than the clean DropDim. We present the DropDim with label smoothing scores in the tables in Appendix C. The scores for triplet DropDim are close to 0.5, illustrated by the horizontal line of points in the boxplot. The order of models' boxplots from left to right is the same as in the legend from top to bottom.

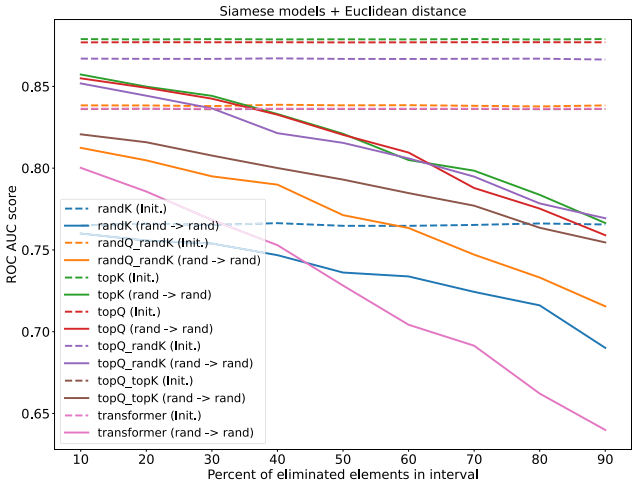


FIGURE 5. The robustness of Reguformer with random keys (randK), Reguformer with random queries and keys (randQ_randK), Reguformer with top keys (topK), Reguformer with top queries (topQ) – the informer's analog, Reguformer with top queries and keys (topQ_topK), Reguformer with top queries and random keys (topQ_randK), and the classical transformer model. For each model, the initial ROC AUC scores are presented (Init.) and ROC AUC during the increase of eliminated random parts of well-intervals with white noise (rand -> rand).

of Triplet-based approach. For testing, we use 5000 pairs or triplets from other wells.

The detailed tables with the comparison of the ROC AUC scores for each of the four datasets are presented in Appendix C.

Table 2 shows the aggregated information about the results: the ROC AUC scores of all models for each dataset. It shows that the highest scores for most of the datasets belong to Reguformer with top queries and keys. Moreover, the Transformer shows comparable quality to our Reguformer. Additionally, the results demonstrate no advantage of the Top Queries approach from the Informer model, despite higher metrics on the well and crimes datasets. The results can be justified considering the nature of the data. Both the wells and crimes datasets initially contain a significant number of missing values and may consist of the noised data due to their specificity. Moreover, the Top Queries Top Keys applies stronger regularization as it utilizes two matrices, key and query, which is excessive for these data. On the other hand, the advantage of the Top Queries approach in these cases is not as significant as the benefit seen with the Top Queries Top Keys method when applied to the weather and stocks datasets.

In the following Sections, we examine Reguformers in more detail and compare them with the Informer and the classical Transformer model. The other models, such as Performer, DropDim, and LRformer, have different ideas behind them, and they were used as the baselines in the similarity tasks.

B. ROBUSTNESS

One explanation for the obtained results for well-linking similarity estimation is that Reguformers are more robust to errors in data because they use sparse attention that utilizes only a part of the data. As it is crucial to the oil&gas industry – an area characterized by many missing values and noise in the data, we conduct the experiment by replacing some data elements with white noise and zeros to prove the statement about Reguformers' robustness.

To analyze the models' robustness, we generate 5000 well-intervals and study the dependence of the ROC AUC scores on the percentage of changed data for the models achieving top results on well-linking tasks, namely, Reguformer with random keys (randK), Reguformer with random queries and keys (randQ_randK), Reguformer with top keys (topK), Reguformer with top queries (topQ) – the Informer's analog, Reguformer with top queries and keys (topQ_topK), Reguformer with top queries and random keys (topQ_randK), and the classical Transformer. Figure 5 demonstrates that most of our Reguformers' quality decreases slower than the Transformer's. Transformer outperforms only Reguformer with random keys and only when the percentage of noised elements is less than 45%. However, even if most parts of the intervals are noised, the performance of all models remains acceptable. Moreover, the Reguformer modification with top keys (topK) is more robust than the Informer model as it illustrates better quality along the whole percentages of changed intervals.

TABLE 2. Comparison of the ROC AUC scores of the best models for all datasets. TOP-1 values for each dataset are highlighted with bold font, and TOP-2 values are underlined.

Model	Wells	Crimes	Weather	Stocks
LSTM	0.934 ± 0.047	-	-	-
Transformer	0.967 ± 0.013	0.991 ± 0.018	0.85 ± 0.016	0.942 ± 0.025
Informer	**0.978 ± 0.006**	**1.0 ± 0.001**	0.789 ± 0.017	0.937 ± 0.028
Performer	0.949 ± 0.017	0.999 ± 0.001	0.725 ± 0.01	0.805 ± 0.025
Lrformer	0.955 ± 0.012	0.911 ± 0.092	0.634 ± 0.042	0.858 ± 0.044
DropDim	0.777 ± 0.014	0.813 ± 0.059	0.717 ± 0.025	0.532 ± 0.019
Top Queries Top Keys	0.97 ± 0.019	0.959 ± 0.052	**0.896 ± 0.018**	**0.944 ± 0.017**

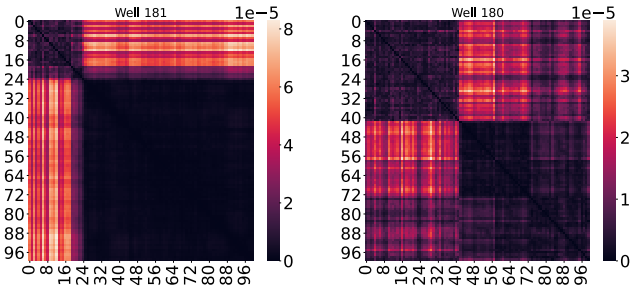


FIGURE 6. Calculated distances between intervals from one well. The two heatmaps refer to the two separate wells: well number 181 (left) and well number 180 (right).

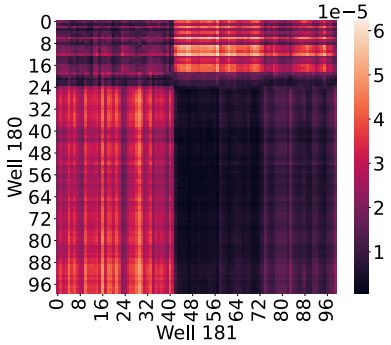


FIGURE 7. The visualization of the cosine distance between embeddings from two different wells.

These results are also confirmed by the same experiment with the replacement with zeros and testing these imitations of missing and noisy data on other datasets in Appendix D.

Considering the oil&gas application, another explanation for the well-linking task results comes from the data's uniform structure, with only one or two rock types in the interval. Thus, we do not need many points to make conclusions. Instead, the model should be robust to errors and missing values. To prove this, we visualize the cosine distance between well-intervals belonging to one well and well-intervals from two different wells. Figure 6 shows that parts can be easily distinguished. These changes indicate different rock types, and each rock type's data is uniform.

The same effect is illustrated in Figure 7 for cross-distances for intervals from different wells. This type of heatmap can be used to compare the rock types of two wells and identify similarities between parts of the wells.

TABLE 3. Comparison of the embeddings' clustering quality. TOP-1 values are highlighted with bold font, and TOP-2 best values are underlined. The upper part of the table contains reference results from [2].

Model	ARI	
	Siamese	Triplet
XGBoost [2]	0.258 ± 0.173	
LSTM [2]	0.569 ± 0.162	0.553 ± 0.122
Transformer	0.793 ± 0.09	**0.981 ± 0.037**
Performer	0.721 ± 0.09	0.634 ± 0.058
Top Queries	0.933 ± 0.076	0.927 ± 0.06
Random Queries	0.812 ± 0.122	0.875 ± 0.105
Top Keys	0.85 ± 0.117	0.964 ± 0.061
Random Keys	0.919 ± 0.053	0.969 ± 0.062
Top Queries Top Keys	0.852 ± 0.097	0.92 ± 0.104
Top Queries Random Keys	**0.948 ± 0.054**	0.941 ± 0.069
Random Queries Top Keys	0.842 ± 0.079	0.952 ± 0.057
Random Queries Random Keys	0.851 ± 0.103	0.967 ± 0.053

C. EMBEDDINGS QUALITY

In this Section, we assess the quality of the well-interval embeddings. The high quality of well-intervals' embeddings is crucial for the oil&gas industry as good well-interval representations will allow to solve downstream tasks.

After training the models, we obtain 5000 embeddings and cluster them with Agglomerative clustering. To get the embeddings, we take the output of a Transformer-based encoder, flatten it, and aggregate it with a linear layer. The obtained clustering is compared to "true" clustering with cluster labels being well names. There are 28 of them in total. All models' Adjusted Rand Index (ARI) scores are presented in Table 3. Judging by the ARI, the closest score to the classical Transformer model is demonstrated by the Reguformer with random keys (randK). However, all other regularization techniques also show a high quality of their embeddings. In this clustering experiment, Triplet models demonstrate better results than the Siamese due to the logic behind the Triplet loss function to place similar objects in the representation space closer than dissimilar ones. Further in this Section, we visualize the embeddings of different Triplet models.

We also try to solve a multiclass classification problem with obtained embeddings. Class labels here are labels for particular wells.

The procedure for training a downstream classifier for embeddings is the following. We generate data intervals from the original data series (wells), which are subsequently transformed and classified. In our case, 5000 well-intervals are generated.

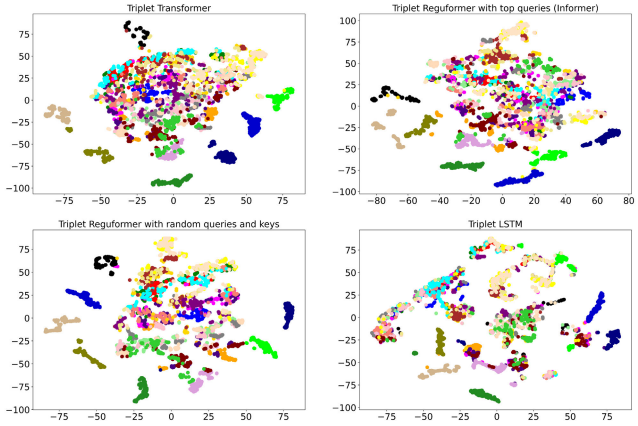


FIGURE 8. Visualization of t-SNE representation for Well-intervals embeddings. Each point corresponds to a single well-interval. Each well has its own color. The LSTM model correctly and precisely distinguishes 6 – 8 wells, transformer – 7 – 8 wells, its efficient modification Informer – 8 – 9 wells, and our Reguformer with random queries and keys – 10 – 12 wells.

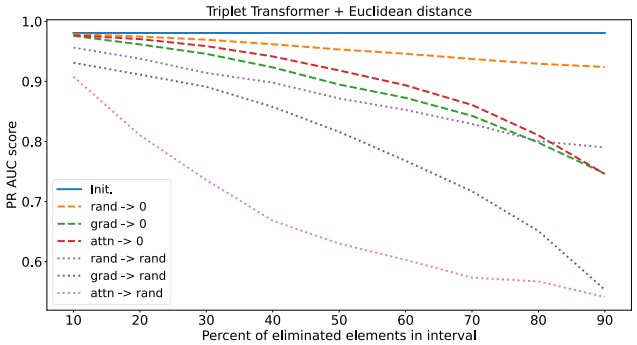


FIGURE 9. PR AUC score during the increase of the percentage of eliminated interval's parts for different criteria for elimination. A faster decrease means that we better identify important measurements. The notation refers to <the criterion for elements elimination> → <the values that substitute the original ones>.

The resulting embeddings are used as inputs to common machine learning models (downstream classifiers):

1) XGBoost gradient boosting classifier with default hyperparameters;
2) Logistic regression equivalent to a neural network with one linear layer;
3) Neural network with three linear layers connected by the ReLU activation function. The dimension of the first layer equals the dimension of embedding, the dimension of the second layer equals 64, and the dimension of the third layer equals 128.

Since our tasks involve classification, we use the following metrics: ROC AUC, PR AUC, and F1-score. We present the comparison in Table 4. As we see from the table, the quality of the models is high. This fact is true even though the classification part is simple. Also, observing all metrics, we can conclude that the Transformer's embeddings are classified quite well, especially given that there are 28 classes in total with the best values provided by the Siamese Reguformer with top queries and random keys, top queries, and random queries. However, all other regularization techniques demonstrate comparable results.

We can also compare the well-intervals' representations visualization. We compress 5000 embeddings obtained via the Triplet models with t-SNE [56] to get two coordinates for each well-interval and plot them. We consider Reguformer with random queries and keys (randQ_randK), which shows the most promising results in terms of the well-linking problem, embeddings quality, and the model's robustness, and compare its embeddings with the Triplet Transformer, the LSTM from [2], and the Reguformer with top queries (topQ), which is equivalent to Informer. In Figure 8, we can distinguish the precise areas dedicated to each well for our Reguformers. This is not the case for LSTM and the classical Transformer. Moreover, the visualization of some wells is better if the Reguformer with random queries and keys (randQ_randK) is used.

D. ATTENTION ANALYSIS

This Section aims to answer the question about the interpretability of the attention scores and their relation to the model's gradients. Thus we conduct the sensitivity analysis of Transformers using well logs. We look at the attention matrix for the pretrained Transformers with the sensitivity analysis in mind. We conduct two experiments:

1) Estimation of correlation between Transformer attentions and models' gradients;
2) For the second experiment, we repeatedly replace a part of an interval with the lowest attention or a feature with the biggest gradient with zeros or numbers from the random normal distribution and calculate the initial and the obtained model's accuracy. Each sequence element's attention is the corresponding $\mathbf{q}_i^T \mathbf{k}_i$ averaged over all layers and heads.

The idea is the same as the authors of [57] proposed: the lower the attention scores, the higher the uncertainty, and consequently, the higher the importance of the part of the interval is.

We generate 5000 pairs of intervals, obtain gradients and attention weights, and calculate the correlation between them. We also examine the dependence of Triplet Transformer accuracy on the percentage of changed elements.

The correlation value is $corr = 0.02 \pm 0.08$. Thus, there is no direct relation between gradients and attention values.

Although the correlation is not high enough, Figure 9 demonstrates that masking of intervals with small attention weights strongly influences the models' quality, mainly when we mask them with the elements from the normal distribution. We also note that Attention-based masking works differently; it is most successful when we replace the given values with random ones. The obtained results prove the statement that "Attention is not not Explanation" [58]. Thus, we get additional evidence that our model is more robust to noise than others.

TABLE 4. Mean values of quality metrics on multiclass classification task of well-intervals' embeddings, TOP-1 values are highlighted with bold font and TOP-2 best values are underlined.

Downstream classifier	Loss type	Regularization type	ROC AUC	PR AUC	F1
XGBoost	Siamese	LSTM	0.978	0.849	0.814
		Transformer	0.919	0.918	0.951
		Top Queries	**0.997**	**0.971**	0.944
		Random Queries	0.994	0.917	0.875
		Top Keys	0.931	0.929	0.958
		Random Keys	0.911	0.909	0.940
		Top Queries Top Keys	0.930	0.929	0.959
		Top Queries Random Keys	0.959	0.958	**0.984**
		Random Queries Top Keys	0.923	0.922	0.960
		Random Queries Random Keys	0.933	0.933	0.965
	Triplet	LSTM	0.976	0.833	0.798
		Transformer	0.931	**0.931**	**0.962**
		Top Queries	0.993	0.916	0.879
		Random Queries	**0.994**	0.926	0.886
		Top Keys	0.895	0.893	0.927
		Random Keys	0.899	0.898	0.933
		Top Queries Top Keys	0.886	0.887	0.922
		Top Queries Random Keys	0.893	0.891	0.916
		Random Queries Top Keys	0.910	0.909	0.946
		Random Queries Random Keys	0.913	0.913	0.944
Linear layer	Siamese	LSTM	0.959	0.571	0.578
		Transformer	0.747	0.765	0.766
		Top Queries	**0.997**	0.935	0.894
		Random Queries	0.986	0.745	0.718
		Top Keys	0.835	0.831	0.875
		Random Keys	0.834	0.835	0.862
		Top Queries Top Keys	0.839	0.838	0.872
		Top Queries Random Keys	0.939	**0.938**	**0.963**
		Random Queries Top Keys	0.831	0.825	0.857
		Random Queries Random Keys	0.841	0.841	0.876
	Triplet	LSTM	0.943	0.528	0.505
		Transformer	0.857	**0.856**	0.879
		Top Queries	**0.986**	0.766	0.735
		Random Queries	**0.986**	0.79	0.761
		Top Keys	0.818	0.820	0.861
		Random Keys	0.828	0.830	0.869
		Top Queries Top Keys	0.793	0.791	0.821
		Top Queries Random Keys	0.839	0.835	0.876
		Random Queries Top Keys	0.786	0.787	0.832
		Random Queries Random Keys	0.856	**0.856**	**0.898**
FC NN	Siamese	LSTM	0.979	0.757	0.736
		Transformer	0.867	0.840	0.895
		Top Queries	**0.999**	**0.972**	0.945
		Random Queries	0.994	0.873	0.818
		Top Keys	0.921	0.914	0.957
		Random Keys	0.903	0.901	0.935
		Top Queries Top Keys	0.923	0.921	0.950
		Top Queries Random Keys	0.964	0.962	**0.981**
		Random Queries Top Keys	0.907	0.900	0.932
		Random Queries Random Keys	0.924	0.920	0.951
	Triplet	LSTM	0.972	0.715	0.684
		Transformer	0.914	**0.913**	0.942
		Top Queries	0.995	0.887	0.834
		Random Queries	**0.996**	0.909	0.867
		Top Keys	0.879	0.872	0.916
		Random Keys	0.905	0.900	0.944
		Top Queries Top Keys	0.855	0.852	0.901
		Top Queries Random Keys	0.889	0.887	0.927
		Random Queries Top Keys	0.900	0.899	0.935
		Random Queries Random Keys	0.908	0.902	**0.948**

E. INFERENCE TIME

We compare the Transformer's modifications regarding GPU inference time and the ROC AUC score. For this experiment, we used Nvidia Titan RTX GPU and oil wells data. We consider a batch of 64 well-intervals, and we measure the GPU inference time on 300 iterations, with 20 warmup iterations. Figure 10 shows that changing the Informer's regularization strategy with top queries (topQ) to the one with random queries (randQ) or, particularly, random queries and keys (randQ_randK) not only reduces the inference time

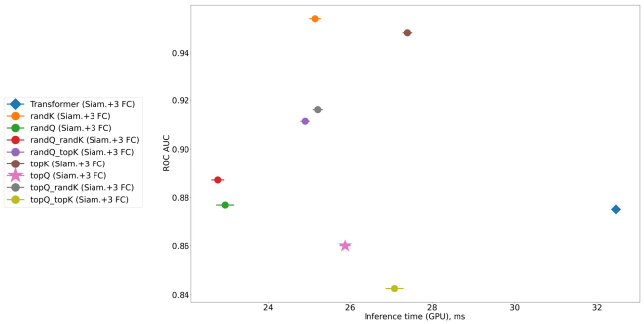


FIGURE 10. The dependence of the ROC AUC metric on the GPU inference time of Siamese models. The Informer's regularization strategy with top queries is highlighted with the star marker, the vanilla transformer – the rhombus.

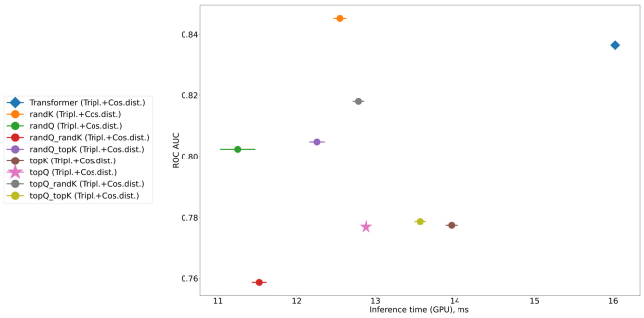


FIGURE 11. The dependence of ROC AUC on the GPU inference time of Triplet models. The Informer's regularization strategy with top queries is highlighted with the star marker, the vanilla transformer – the rhombus.

but also significantly enhances the model's quality. Almost all other regularizations improve the models' performance with similar inference times, except for Reguformer with top queries and keys (topQ_topK), which has a comparable ROC AUC score and inference time to the Informer. Figure 11 demonstrates a similar effect: regularizations with random queries (randQ) or random queries and keys (randQ_randK) accelerate the model's inference with a slight decrease in the ROC AUC score for the latter strategy. Other models' inference times are comparable to the Informer's, while the quality is significantly higher. The performance of Reguformers with top queries and keys (topQ_topK) and top keys (topK) are similar to the Informer's. In general, the ROC AUC score of all Reguformers is high.

V. CONCLUSION AND DISCUSSION

We have successfully applied Transformer-based architectures to oil&gas logging data collected during drilling. Our Reguformer models can accurately solve a similarity problem between intervals of oil wells. These models are better in terms of quality and efficiency, can solve diverse downstream tasks, and are more interpretable and robust than previously used.

The experimental results show an increase in quality: the ROC AUC score of our Reguformer equals 0.97, which outperforms the classical Transformer 0.967 and is close to the Informer 0.978, while previous models only reach 0.934. We note that such improvement takes place for several variations of the Reguformer.

The analogous experiments on three additional datasets from different domains confirm our statement about Reguformers' high quality and robustness.

Judging by the embeddings in the clustering experiment, our Reguformers, both in their Siamese and Triplet variations, provide meaningful well-interval representations in terms of geological properties. Our best ARI score is equal to 0.969 for the random keys regularization strategy, which surpasses Informer with the ARI score equal to 0.933. It is closer to the perfect score of 1 and, again, higher than the previously obtained value of 0.569. The ARI score for the original Transformer is not significantly higher, being 0.981.

Our conclusions about strong embeddings are also supported by the experiments with the embeddings' classification on wells by a simple machine learning algorithm on top of embeddings. This experiment demonstrates the higher quality of embeddings from Reguformers than the ones from the vanilla Transformer. Moreover, we show that attention maps in our models provide interpretability. For our model, lower attention values correspond to more important parts of an interval: a model's quality decreases as the percentage of eliminated parts of intervals with a low attention score increases. Furthermore, we show that the inference time decreases for the best regularization strategies mentioned above: random queries and random queries and keys. Both these techniques allow to improve the models' performance and even to increase the model's quality in the case of the Reguformer with random queries and keys, especially in the Siamese configuration. However, Reguformer with random keys can significantly improve the model's performance without increasing the inference time.

APPENDIX A
DATA PREPROCESSING

A. WELL LOGS

The wells data preprocessing strategy resembles that of [2] and is the following:

1) Eliminate intervals with negative or zero-equal resistivity and cavernous intervals with the difference between calliper and bit size is greater than 0.35.
2) Fill missing values via forward and backward fill.
3) Convert all electrical resistivity data to log-normal scale.
4) Normalize gamma-ray and neutron log data within each well and formation using the standard scaler.
5) Normalize other features by subtracting the mean and dividing by unit variance.

B. CRIMES

To cope with missing values in the columns with latitude and longitude, we change their original -1 values to the most popular latitude and longitude, correspondingly. Then, we apply scaling to these columns: we subtract the minimum

TABLE 5. The hyperparameters search space for all considered models: Reguformers (Reguf), Transformer (Tr), Performer (Perf), DropDim (DD), DropDim with label smoothing (DD+ls), LRformer (LRf). For each regularization type in Reguformer, hyperparameter optimization was conducted separately with the same hyperparameter search space. We use curly brackets for denoting sets, python notation for ranges: start value, excluded stop value and step, and [minimum, maximum] notation for float features optimization in Optuna. Default values, from which Optuna started the optimization process, and the corresponding models for these hyperparameters are in bold.

Parameters	Search space	Models
d_model	{16, 32, **64**}	**Reguf, Tr, DD, DD+ls**
factor	range(3, 12, 2)	Reguf
n_heads	{2, 4, 6, 8} / {2, 4}	Reguf, Tr, DD, DD+ls, LRf[*]/ Perf
dropout	[0.1, 0.5] / [0.1, 0.9]	Reguf, Tr, DD, DD+ls, LRf / Perf
d_ff	{128, 512, 1024}	Reguf, Tr, DD, DD+ls, LRf
e_layers	range(2, 7, 1)	Reguf, Tr, DD, DD+ls, LRf
nb_random_features	range(1, 5, 1)	Perf
p	[0.1, 1]	DD, DD+ls
drop_dim_type	{**"random"**, "span"}	**DD, DD+ls**
alpha[**]	range(**0**, 64, 1)	**DD, DD+ls**, LRf
label_smoothing	[**0**, 1]	**DD+ls**

[*] due to the specifity of LRformer realization, {1, 2, 5} and {1, 5} for weather and stocks data, respectively.
[**] is required in "span" mode in DropDim and in approximated self-attention in LRformer.

TABLE 6. Siamese / Triplet best hyperparameters of baselines. If one value is present, the Siamese and triplet models share this hyperparameter value. DropDim with label smoothing is implemented only in the Siamese models. All the float values are rounded to two decimal places.

Model	Hyperparameter	Wells	Crimes	Weather	Stocks
Transformer	d_model	16 / 64	64 / 16	64 / 23	16 / 64
	n_heads	8 / 2	8 / 4	6 / 2	6
	dropout	0.26 / 0.47	0.31 / 0.11	0.14 / 0.26	0.13
	d_ff	512 / 1024	128 / 1024	128 / 1024	512 / 1024
	e_layers	2 / 3	3 / 5	3 / 6	2 / 4
Performer	n_heads	2 / 4	2 / 4	2	1
	dropout	0.19 / 0.16	0.13	0.18 / 0.29	0.11 / 0.65
	nb_random_features	2 / 4	1 / 3	6 / 8	5 / 4
DropDim	d_model	64	16 / 64	64 / 16	32 / 64
	n_heads	8	6	6 / 2	4 / 8
	dropout	0.1	0.24 / 0.41	0.1 / 0.12	0.22 / 0.41
	d_ff	1024 / 512	512	128	1024
	e_layers	2 / 5	2 / 6	2 / 3	6 / 4
	p	0.503 / -	0.12 / -	0.74 / 0.55	0.81 / -
	drop_dim_type	random / -	span / -	span	span / -
	alpha**	31 / 0	62 / 0	59 / 27	18 / 0
DropDim + label smoothing	d_model	16	16	64	16
	n_heads	4	8	6	2
	dropout	0.17	0.23	0.48	0.15
	d_ff	512	128	512	512
	e_layers	2	2	2	4
	p	0.47	0.27	0.34	0.9
	drop_dim_type	random	random	random	span
	alpha**	50	8	27	36
	label_smoothing	0.58	0.8	0.2	0.71
LRformer	n_heads	1	1 / 4	1 / 2	1
	d_ff	256 / 1536	1536 / 1280	256 / 1280	512 / 768
	dropout	0.11 / 0.17	0.22 / 0.1	0.12 / 0.33	0.19 / 0.11
	e_layers	3 / 1	5 / 6	1 / 5	3 / 4
	alpha**	0.08 / 0.86	0.16 / 0.86	0.61 / 0.08	0.44 / 0.75

value in training samples from each value in each column and divide the result by the difference between maximum and minimum values in training samples in each column.

C. WEATHER

We eliminate weather stations with less than 100 records and normalize other columns by subtracting the mean of the training samples from each value and dividing it by the standard deviation of the training samples.

D. STOCKS

For each company, we fill missing values with forward fill and normalize the data by subtracting the minimum value in training samples from each value in each of these columns and dividing the result by the difference between the maximum and minimum values in training samples in each column.

TABLE 7. Siamese / Triplet best hyperparameters of Reguformers. If one value is present, the Siamese and triplet models share this hyperparameter value. All the float values are rounded to two decimal places.

Model	Hyperparameter	Wells	Crimes	Weather	Stocks
Top Queries	d_model	32 / 64	32	16	16 / 32
	factor	5 / 7	7 / 3	7 / 9	9
	n_heads	8 / 2	4 / 2	8	6
	dropout	0.24/ 0.38	0.15 / 0.18	0.12 / 0.38	0.13 / 0.11
	d_ff	128 / 512	512 / 1024	1024	128 / 512
	e_layers	6 / 5	2 / 6	2 / 6	4 / 3
Random Queries	d_model	64	64 / 16	32	64
	factor	11 / 5	7 / 11	3 / 5	5 / 11
	n_heads	4	4 / 8	4	8 / 4
	dropout	0.28 / 0.11	0.1	0.16 / 0.44	0.14 / 0.23
	d_ff	512 / 128	1024 / 512	128 / 1024	128 / 1024
	e_layers	4 / 5	3 / 5	3 / 5	6 / 5
Top Keys	d_model	64	64 / 16	64 / 32	32 / 64
	factor	9 / 5	9 / 11	5 / 3	11
	n_heads	2 / 6	8	2 / 4	6 / 4
	dropout	0.31 / 0.12	0.18 / 0.13	0.22 / 0.48	0.1 / 0.22
	d_ff	128 / 1024	512	128 / 512	128 / 512
	e_layers	2 / 6	4	6	2 / 3
Random Keys	d_model	32 / 64	16	16	32 / 64
	factor	5 / 11	5 / 9	5 / 9	7 / 5
	n_heads	6 / 8	6 / 4	4 / 6	6
	dropout	0.17 / 0.3	0.15 / 0.1	0.1 / 0.11	0.15 / 0.11
	d_ff	1024 / 512	128 / 1024	128 / 1024	128 / 1024
	e_layers	5 / 2	3 / 4	2 / 6	2 / 5
Top Queries Top Keys	d_model	64	32 / 16	64 / 16	32 / 64
	factor	5 / 3	3 / 5	5 / 3	5 / 3
	n_heads	6 / 2	4	6 / 4	2
	dropout	0.28 / 0.16	0.19 / 0.15	0.11 / 0.21	0.13 / 0.18
	d_ff	512	512 / 1024	128	128
	e_layers	2	5 / 6	5 / 6	2 / 3
Top Queries Random Keys	d_model	64	16	64 / 32	64
	factor	9 / 7	7 / 5	9 / 11	9 / 5
	n_heads	2	8 / 2	8 / 6	6 / 4
	dropout	0.41 / 0.4	0.46 / 0.17	0.34 / 0.13	0.19 / 0.15
	d_ff	128 / 512	128 / 1024	128 / 512	128
	e_layers	2	2 / 5	4 / 5	2 / 5
Random Queries Top Keys	d_model	64	64 / 16	32	16 / 64
	factor	3	3 / 9	7 / 11	5 / 7
	n_heads	8	4	8	8
	dropout	0.21 / 0.15	0.15 / 0.1	0.11 / 0.21	0.12 / 0.15
	d_ff	128	512 / 1024	128	128 / 512
	e_layers	5 / 3	2 / 3	5 / 6	5 / 6
Random Queries Random Keys	d_model	32 / 64	32 / 16	32 / 16	32 / 64
	factor	3 / 7	3 / 5	9	7 / 5
	n_heads	2	6 / 4	4 / 2	2 / 6
	dropout	0.19 / 0.21	0.11	0.12 / 0.24	0.11 / 0.22
	d_ff	512 / 1024	512	128	128 / 512
	e_layers	4 / 6	5 / 4	3 / 6	3 / 4

APPENDIX B
TECHNICAL DETAILS

The code for all the experiments is available in the repository `roguLINA/Reguformer`.[9] The Reguformers' and Informer's realizations are inspired by the code `zhouhaoyi/Informer2020`.[10] For these models, we use the GELU activation function. The code for Performer, DropDim, and LRformer is also available in our repository. The Performer realization is based on `nawnoes/pytorch-performer`.[11]

The architecture of the three fully connected layers for Siamese models is the following: `FC (input_size, hidden_size) + ReLU + Dropout (0.25) +`

[9]https://github.com/roguLINA/Reguformer
[10]https://github.com/zhouhaoyi/Informer2020
[11]https://github.com/nawnoes/pytorch-performer

TABLE 8. Comparison of the ROC AUC scores of models for well-linking problem. TOP-1 values for each model are highlighted with bold font, and TOP-2 best values are underlined. The two best scores among all the models are noted with *italics*. The LSTM reference results are from [2]; all other results are from our experiments.

Model	Siam.+3 FC	Siam.+Eucl.dist.	Siam.+Cos.dist.	Tripl.+Eucl.dist.	Tripl.+Cos.dist.
LSTM [2]	0.934 ± 0.047	**0.814 ± 0.056**	**0.874 ± 0.064**	0.708 ± 0.061	0.770 ± 0.042
Transformer	0.967 ± 0.013	0.721 ± 0.039	0.778 ± 0.049	0.553 ± 0.034	0.837 ± 0.049
Performer	0.949 ± 0.017	0.315 ± 0.108	0.575 ± 0.021	**0.845 ± 0.067**	0.716 ± 0.094
DropDim	0.679 ± 0.033	0.505 ± 0.004	0.514 ± 0.016	0.51 ± 0.015	0.5 ± 0.0
DropDim + label smooting (0.58)	0.597 ± 0.071	0.777 ± 0.014	0.524 ± 0.047	-	-
Lrformer	0.955 ± 0.012	0.677 ± 0.021	0.798 ± 0.047	0.608 ± 0.027	0.573 ± 0.039
Top Queries	***0.978 ± 0.006***	0.524 ± 0.013	0.841 ± 0.04	0.545 ± 0.02	0.787 ± 0.065
Random Queries	0.976 ± 0.013	0.653 ± 0.037	0.849 ± 0.022	0.557 ± 0.029	0.803 ± 0.055
Top Keys	0.975 ± 0.009	0.535 ± 0.012	0.799 ± 0.05	0.535 ± 0.017	0.744 ± 0.052
Random Keys	0.976 ± 0.016	0.632 ± 0.021	0.845 ± 0.038	0.544 ± 0.022	0.633 ± 0.031
Top Queries Top Keys	0.97 ± 0.019	0.535 ± 0.016	0.823 ± 0.054	0.539 ± 0.016	0.789 ± 0.077
Top Queries Random Keys	0.976 ± 0.014	0.503 ± 0.003	0.84 ± 0.057	0.552 ± 0.021	0.742 ± 0.092
Random Queries Top Keys	0.963 ± 0.032	0.516 ± 0.008	0.814 ± 0.021	0.533 ± 0.013	0.753 ± 0.046
Random Queries Random Keys	*0.977 ± 0.008*	0.593 ± 0.028	0.848 ± 0.03	0.56 ± 0.016	**0.848 ± 0.049**

`FC (hidden_size, hidden_size) + ReLU + Dropout (0.25) + FC (hidden_size, output_size) + Sigmoid`. We use `hidden_size` = 64.

For hyperparameters optimization for all models, we use Optuna[12] with 30 iterations, group 2-fold cross-validation with 100 epochs and early stopping with patience 10 for each split and each model. Batch size equals to 64.

We vary the most essential hyperparameters for *all* models:

- `n_heads` – the number of heads for multi-head attention;
- `dropout` – the dropout probability.

In addition, we vary:

- for Reguformer, vanilla Transformer, DropDim, LRformer:
  - `d_ff` – the dimension of the fully-connected layers;
  - `e_layers` – the number of layers in the encoder.
- for Reguformer, vanilla Transformer, DropDim:
  - `d_model` – the model embedding dimension.
- for Reguformer:
  - `factor` – the ProbSparse attention factor.
- for Performer:
  - `nb_random_features` – the number of random features.
- for DropDim:
  - `p` – the probability of dropping the embeddings' dimensions;
  - `drop_dim_type` – the structured dropout method;
  - `alpha` – the pre-defined value determining the maximum length of the consecutive drop during the "span" method.
- for DropDim with label smoothing:
  - `label_smoothing` – the label smoothing factor.
- for LRformer:
  - `alpha` – the scaler factor of the regularization scalar function.

[12]https://github.com/optuna/optuna

The default hyperparameters and hyperparameters' search space for each model are presented in Table 5.

The best hyperparameters for all models are presented in Tables 6 and 7. Zeros for `alpha` DropDim in Triplet configurations in Table 6 means that DropDim decreases the models' quality; they perform better without DropDim.

All these models with the best hyperparameters were tested via group 5-fold cross-validation. It can be seen even from Figure 4 with the hyperparameter optimization process that our Reformers outperform all the baselines.

APPENDIX C
SIMILARITY MODELS QUALITY: DIFFERENT DATASETS

In this Section, we provide the results for the similarity problem statements in different domains, namely, oil&gas, crime incidents, weather, and the stock market. For all datasets, we provide ROC AUC scores as they are the most representative.

According to the ROC AUC values presented in Table 8 for the models with the best hyperparameters, the usage of Transformer-based architectures improves the previous results obtained with RNN. Moreover, the scores of Reguformers are significantly higher than the classical Transformer model's: the best results belong to the Siamese with 3 FC configuration with the mean ROC AUC scores 0.977 – 0.978 achieved by Reguformer with random queries and keys and Reguformer with top queries, respectively, while the Tasnformer's best result is 0.967. This experiment also shows that there is no need to use only the top queries as the Informer model; random queries or keys are enough to reach the comparable quality. In addition, the similar results with ROC AUC equal to 0.976 belong to the Reguformers with random queries, random keys, and top queries and random keys.

Tables 9 and 11 show the best performance of Siamese models with 3 FC. TOP-1 scores for crimes data belong

TABLE 9. Comparison of the ROC AUC scores of models for crimes' districts similarity problem. TOP-1 values for each model are highlighted with bold font, and TOP-2 best values are underlined. The two best scores among all the models are noted with *italics*.

Model	Siam.+3 FC	Siam.+Eucl.dist.	Siam.+Cos.dist.	Tripl.+Eucl.dist.	Tripl.+Cos.dist.
Transformer	0.991 ± 0.018	0.75 ± 0.073	**0.897 ± 0.078**	0.808 ± 0.064	0.718 ± 0.185
Performer	*0.999 ± 0.001*	0.589 ± 0.018	0.549 ± 0.066	0.706 ± 0.078	0.822 ± 0.176
DropDim	0.503 ± 0.026	0.813 ± 0.059	0.5 ± 0.0	0.507 ± 0.009	0.5 ± 0.0
DropDim + label smooting (0.8)	0.503 ± 0.005	0.804 ± 0.053	0.5 ± 0.0	-	-
Lrformer	0.911 ± 0.092	0.809 ± 0.048	0.875 ± 0.112	0.673 ± 0.06	0.713 ± 0.139
Top Queries	***1.0 ± 0.001***	0.791 ± 0.064	0.746 ± 0.166	0.767 ± 0.096	0.773 ± 0.218
Random Queries	***1.0 ± 0.0***	0.668 ± 0.119	0.686 ± 0.183	0.833 ± 0.057	0.846 ± 0.19
Top Keys	0.998 ± 0.002	0.685 ± 0.09	0.655 ± 0.142	0.768 ± 0.093	0.682 ± 0.133
Random Keys	***1.0 ± 0.0***	**0.847 ± 0.089**	0.723 ± 0.212	0.797 ± 0.063	0.873 ± 0.146
Top Queries Top Keys	0.959 ± 0.052	0.783 ± 0.102	0.745 ± 0.186	0.801 ± 0.047	0.85 ± 0.184
Top Queries Random Keys	*0.999 ± 0.001*	0.689 ± 0.069	0.731 ± 0.175	0.779 ± 0.093	**0.893 ± 0.117**
Random Queries Top Keys	***1.0 ± 0.0***	0.701 ± 0.097	0.787 ± 0.156	0.782 ± 0.085	0.844 ± 0.161
Random Queries Random Keys	0.995 ± 0.007	0.833 ± 0.042	0.764 ± 0.208	**0.834 ± 0.065**	0.867 ± 0.194

TABLE 10. Comparison of the ROC AUC scores of models for weather stations similarity problem. TOP-1 values for each model are highlighted with bold font, and TOP-2 best values are underlined. The two best scores among all the models are noted with *italics*.

Model	Siam.+3 FC	Siam.+Eucl.dist.	Siam.+Cos.dist.	Tripl.+Eucl.dist.	Tripl.+Cos.dist.
Transformer	0.499 ± 0.008	0.6 ± 0.062	0.5 ± 0.0	0.85 ± 0.016	0.51 ± 0.017
Performer	0.508 ± 0.015	0.506 ± 0.001	0.502 ± 0.004	0.592 ± 0.027	**0.725 ± 0.01**
DropDim	0.533 ± 0.035	0.5 ± 0.0	0.516 ± 0.008	0.717 ± 0.025	0.5 ± 0.0
DropDim + label smooting (0.12)	0.492 ± 0.006	0.5 ± 0.0	0.5 ± 0.0	-	-
Lrformer	0.536 ± 0.027	0.531 ± 0.012	**0.592 ± 0.019**	0.634 ± 0.042	0.501 ± 0.00
Top Queries	0.533 ± 0.033	0.673 ± 0.065	0.507 ± 0.01	0.789 ± 0.017	0.578 ± 0.053
Random Queries	**0.558 ± 0.041**	0.684 ± 0.015	0.547 ± 0.064	0.712 ± 0.023	0.584 ± 0.047
Top Keys	0.499 ± 0.003	0.503 ± 0.002	0.5 ± 0.0	0.747 ± 0.02	0.619 ± 0.063
Random Keys	0.549 ± 0.027	**0.768 ± 0.033**	0.537 ± 0.04	0.759 ± 0.042	0.503 ± 0.003
Top Queries Top Keys	0.556 ± 0.062	0.735 ± 0.031	0.504 ± 0.005	*0.896 ± 0.018*	0.707 ± 0.034
Top Queries Random Keys	0.506 ± 0.004	0.545 ± 0.061	0.502 ± 0.004	***0.905 ± 0.022***	0.697 ± 0.012
Random Queries Top Keys	0.511 ± 0.014	0.577 ± 0.106	0.502 ± 0.003	0.833 ± 0.019	0.509 ± 0.009
Random Queries Random Keys	0.55 ± 0.036	0.776 ± 0.02	0.535 ± 0.022	0.885 ± 0.013	0.571 ± 0.027

TABLE 11. Comparison of the ROC AUC scores of models for stocks market similarity problem. TOP-1 values for each model are highlighted with bold font, and TOP-2 best values are underlined. The two best scores among all the models are noted with *italics*.

Model	Siam.+3 FC	Siam.+Eucl.dist.	Siam.+Cos.dist.	Tripl.+Eucl.dist.	Tripl.+Cos.dist.
Transformer	0.942 ± 0.025	0.713 ± 0.027	0.768 ± 0.045	0.502 ± 0.001	**0.764 ± 0.05**
Performer	0.805 ± 0.025	0.561 ± 0.009	0.503 ± 0.013	0.56 ± 0.015	0.568 ± 0.03
DropDim	0.493 ± 0.01	0.523 ± 0.008	0.5 ± 0.0	0.501 ± 0.001	0.5 ± 0.0
DropDim + label smooting (0.71)	0.497 ± 0.007	0.532 ± 0.019	0.5 ± 0.0	-	-
Lrformer	0.858 ± 0.044	0.625 ± 0.013	0.736 ± 0.056	**0.565 ± 0.007**	0.584 ± 0.046
Top Queries	0.937 ± 0.028	0.682 ± 0.042	0.741 ± 0.077	0.514 ± 0.006	0.73 ± 0.037
Random Queries	0.937 ± 0.028	0.576 ± 0.046	0.77 ± 0.032	0.502 ± 0.002	0.735 ± 0.049
Top Keys	***0.961 ± 0.03***	0.529 ± 0.009	**0.778 ± 0.034**	0.502 ± 0.001	0.749 ± 0.048
Random Keys	*0.947 ± 0.023*	0.513 ± 0.007	0.752 ± 0.029	0.502 ± 0.002	0.75 ± 0.046
Top Queries Top Keys	0.944 ± 0.017	0.527 ± 0.012	0.747 ± 0.023	0.5 ± 0.0	0.737 ± 0.041
Top Queries Random Keys	0.941 ± 0.025	0.503 ± 0.002	0.743 ± 0.04	0.501 ± 0.001	0.735 ± 0.051
Random Queries Top Keys	0.926 ± 0.019	**0.727 ± 0.034**	0.745 ± 0.054	0.505 ± 0.005	0.744 ± 0.066
Random Queries Random Keys	0.939 ± 0.014	0.562 ± 0.025	0.749 ± 0.053	0.501 ± 0.001	0.759 ± 0.047

to Reguformers with top queries (the Informer's analog), random queries, random keys, and random queries and top keys, while Reguformer with top queries and random keys and Performer achieve TOP-2 metrics. Reguformers with top keys and random keys outperform all other models on the stock market data. According to Table 10, combining the top queries with top key or random key matrix regularization with Triplet loss with Euclidean distance improves the original Transformer's quality. All these results prove that, first, the Reguformer performs better on time series similarity tasks, and second, there is no need to select top queries as the other and simpler techniques can outperform the vanilla Transformer. In addition, the Reguformer with random queries and keys demonstrates the comparable quality to the best models for all four considered datasets.

However, there are no strict recommendations about loss function as the most successful models for wells, crimes, and stocks data are Siamese with 3 FC, while for weather – Triplet with Euclidean distance.

APPENDIX D
ROBUSTNESS ON OTHER DATASETS

In this Section, we consider the most successful modifications of the Reguformers and compare their robustness to the

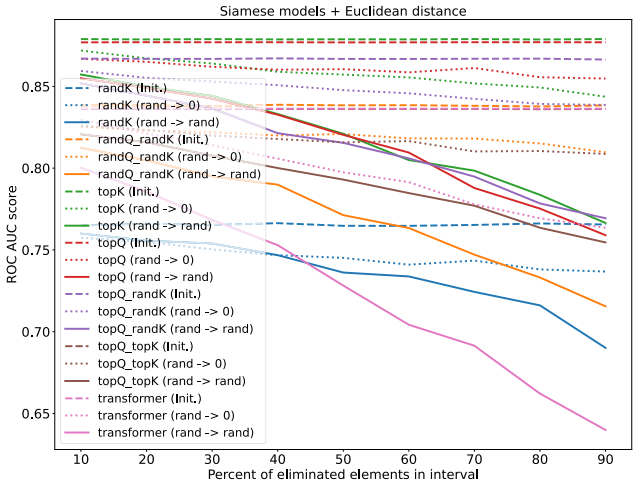


FIGURE 12. The comparison of the robustness of the most promising Reguformers' modifications and the classical Transformer model on the wells data. For each model, the initial ROC AUC scores are presented (Init.) and ROC AUC during the increase of eliminated random parts of wells data intervals with zeros (rand -> 0) and white noise (rand -> rand).

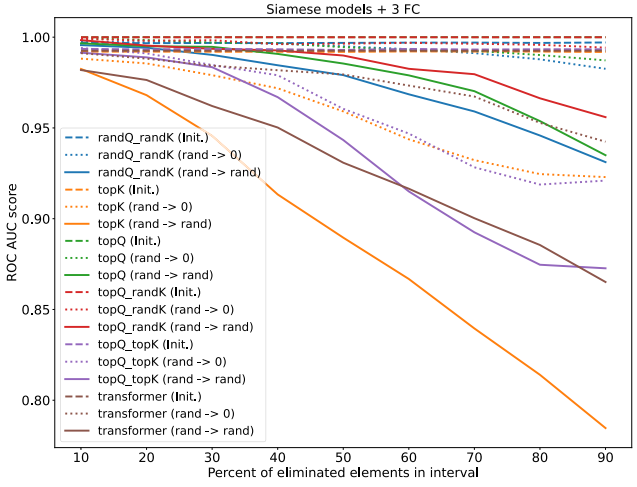


FIGURE 13. The comparison of the robustness of the most promising Reguformers' modifications and the classical transformer model on the crimes data. For each model, the initial ROC AUC scores are presented (Init.) and ROC AUC during the increase of eliminated random parts of crimes data intervals with zeros (rand -> 0) and white noise (rand -> rand).

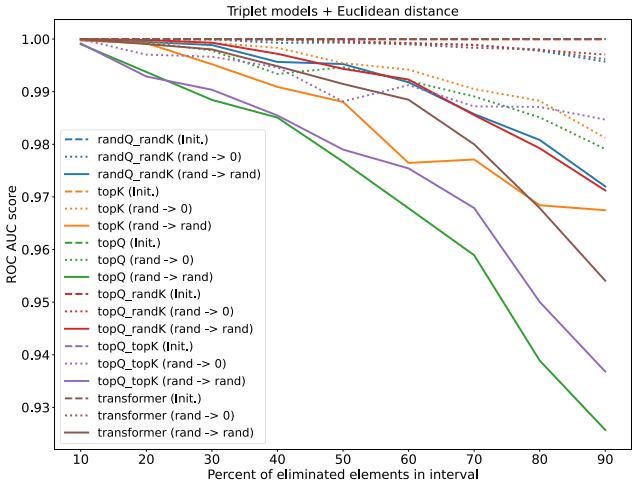


FIGURE 14. The comparison of the robustness of the most promising Reguformers' modifications and the classical transformer model on the weather data. For each model, the initial PR AUC scores are presented (Init.) and PR AUC during the increase of eliminated random parts of weather data intervals with zeros (rand -> 0) and white noise (rand -> rand).

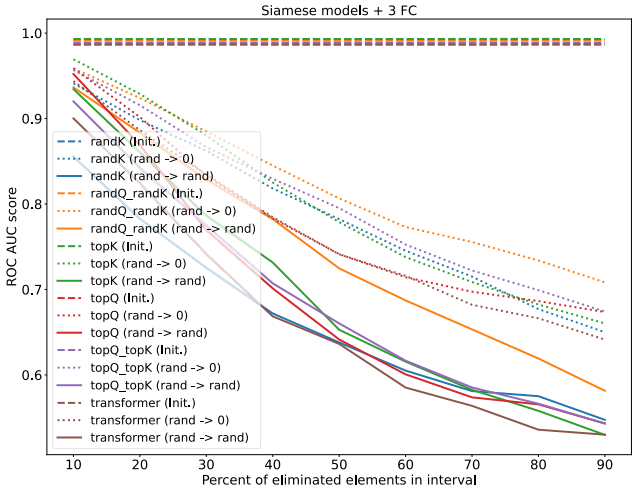


FIGURE 15. The comparison of the robustness of the most promising Reguformers' modifications and the classical transformer model on the stocks data. For each model, the initial PR AUC scores are presented (Init.) and PR AUC during the increase of eliminated random parts of stocks data intervals with zeros (rand -> 0) and white noise (rand -> rand).

Transformer's and the Informer's adaptations. We test the best models in the correspondence with the datasets from Table 2. We provide the results for the simulation of noisy (rand -> rand) and missing values (rand -> 0).

Figures 12, 13, 14, and 15 show that changing the original values to zeros harms the models' quality less than substitution with white noise. For most datasets and elimination techniques, the Transformer demonstrates the worst results, and for the Stock market dataset, its quality reaches even approximately 0.5 (Figure 15). Additionally, for most datasets, Reguformer with top queries and random keys (topQ_randK) and Reguformer with random queries and keys (randQ_randK) show top results and outperform both the classical Transformer and Informer (Reguformer with top queries, topQ).

The results obtained on all datasets confirm the Reguformers' robustness in real-world applications. The experiment setup is similar to industrial cases, dealing with missing or noisy data. Our models successfully cope with the similarity problem even if the percentage of harmed values is high.

ACKNOWLEDGMENT

The authors are grateful to Andrei Volodichev for assisting them with the DropDim baseline preparation and to

Ilya Kuleshov for reviewing the article. They also would like to extend their thanks to Evgenia Romanenkova, Nikolay Stulov, Sergey Egorov, and Narek Gevorgyan, who developed the basic code for the articles [2], [20], which served as the foundation for the deep learning models used in the analysis of well logs.

REFERENCES

[1] M. Aljubran, J. Ramasamy, M. Albassam, and A. Magana-Mora, "Deep learning and time-series analysis for the early detection of lost circulation incidents during drilling operations," *IEEE Access*, vol. 9, pp. 76833–76846, 2021.

[2] E. Romanenkova, A. Rogulina, A. Shakirov, N. Stulov, A. Zaytsev, L. Ismailova, D. Kovalev, K. Katterbauer, and A. AlShehri, "Similarity learning for wells based on logging data," *J. Petroleum Sci. Eng.*, vol. 215, Aug. 2022, Art. no. 110690. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920410522005587

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[4] A. A. Ismail, M. Gunady, L. Pessoa, H. C. Bravo, and S. Feizi, "Input-cell attention reduces vanishing saliency of recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

[5] M. Morvan, N. Nikolaou, K. H. Yip, and I. Waldmann, "Don't pay attention to the noise: Learning self-supervised representations of light curves with a denoising time series transformer," in *Proc. Workshop AI Earth Sci.*, 2022, pp. 1–10.

[6] K. Bagla, A. Kumar, S. Gupta, and A. Gupta, "Noisy text data: Achilles' heel of popular transformer based NLP models," 2021, *arXiv:2110.03353*.

[7] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Comput. Surv.*, vol. 55, no. 6, pp. 1–28, Jun. 2023.

[8] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, vol. 35, no. 12, pp. 11106–11115.

[9] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlös, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, and A. Weller, "Rethinking attention with performers," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–38.

[10] H. Zhang, D. Qu, K. Shao, and X. Yang, "DropDim: A regularization method for transformer networks," *IEEE Signal Process. Lett.*, vol. 29, pp. 474–478, 2022.

[11] W. Ye, Y. Ma, X. Cao, and K. Tang, "Mitigating transformer overconfidence via Lipschitz regularization," in *Proc. Uncertainty Artif. Intell.*, 2023, pp. 2422–2432.

[12] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 9, pp. 11121–11128.

[13] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *Proc. Int. Conf. Learn. Represent.*, 2023, pp. 1–24.

[14] A. Rogulina, A. Zaytsev, L. Ismailova, D. Kovalev, K. Katterbauer, and A. Marsala, "Similarity learning for well logs prediction using machine learning algorithms," presented at the Int. Petroleum Technol. Conf., Feb. 2022, Paper D032S158R005.

[15] R. A. Startzman and T.-B. Kuo, "A rule-based system for well log correlation," *SPE Formation Eval.*, vol. 2, no. 3, pp. 311–319, Sep. 1987.

[16] S. Zoraster, R. Paruchuri, and S. Darby, "Curve alignment for well-to-well log correlation," presented at the SPE Annu. Tech. Conf. Exhib., 2004, Paper SPE–90471.

[17] M. Ali, R. Jiang, H. Ma, H. Pan, K. Abbas, U. Ashraf, and J. Ullah, "Machine learning—A novel approach of well logs similarity based on synchronization measures to predict shear sonic logs," *J. Petroleum Sci. Eng.*, vol. 203, Aug. 2021, Art. no. 108602.

[18] A. K. Verma, A. Routray, and W. K. Mohanty, "Assessment of similarity between well logs using synchronization measures," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 12, pp. 2032–2036, Dec. 2014.

[19] A. Kumar, "Transformer-based deep learning models for well log processing and quality control by modelling global dependence of the complex sequences," Presented at the Abu Dhabi Int. Petroleum Exhib. Conf., 2021, Paper D041S108R004

[20] S. Egorov, N. Gevorgyan, and A. Zaytsev, "Self-supervised similarity models based on well-logging data," 2022, *arXiv:2209.12444*.

[21] A. E. Marusov and A. Zaytsev, "Noncontrastive representation learning for intervals from well logs," *IEEE Geosci. Remote Sens. Lett.*, vol. 20, pp. 1–5, 2023.

[22] I. R. Abdrakhmanov, E. A. Kanin, S. A. Boronin, E. V. Burnaev, and A. A. Osiptsov, "Development of deep transformer-based models for long-term prediction of transient production of oil wells," Presented at the SPE Russian Petroleum Technol. Conf., 2021, Paper D021S006R008.

[23] A. R. Behesht Abad, P. S. Tehrani, M. Naveshki, H. Ghorbani, N. Mohamadian, S. Davoodi, S. K.-Y. Aghdam, J. Moghadasi, and H. Saberi, "Predicting oil flow rate through orifice plate with robust machine learning algorithms," *Flow Meas. Instrum.*, vol. 81, Oct. 2021, Art. no. 102047.

[24] A. A. Alakeely and R. N. Horne, "Application of deep learning methods to estimate multiphase flow rate in producing wells using surface measurements," *J. Petroleum Sci. Eng.*, vol. 205, Oct. 2021, Art. no. 108936.

[25] D. E. Rumelhart, "Learning internal representations by error propagation, in parallel distributed processing," *Explor. Microstructure Cognition*, vol. 1, pp. 318–362, 1986.

[26] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds., Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[28] D. Cherniavskii, E. Tulchinskii, V. Mikhailov, I. Proskurina, L. Kushnareva, E. Artemova, S. Barannikov, I. Piontkovskaya, D. Piontkovski, and E. Burnaev, "Acceptability judgements via examining the topology of attention maps," in *Proc. Findings Assoc. Comput. Linguistics*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds., Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 88–107. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.7

[29] Y. Hao, L. Dong, F. Wei, and K. Xu, "Self-attention attribution: Interpreting information interactions inside transformer," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 14, pp. 12963–12971.

[30] J. L. Klein, *Statistical Visions in Time: A History of Time Series Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1997, pp. 1662–1938.

[31] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, "Ts2vec: Towards universal representation of time series," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 8, pp. 8980–8987.

[32] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, and L. Yang, "Big bird: Transformers for longer sequences," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 17283–17297.

[33] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," 2020, *arXiv:2006.04768*.

[34] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," 2020, *arXiv:2004.05150*.

[35] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," 2020, *arXiv:2001.04451*.

[36] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler, "Long range arena : A benchmark for efficient transformers," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–16. [Online]. Available: https://openreview.net/forum?id=qVyeW-grC2k

[37] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 95–104.

[38] I-Research. (2015). *The New Zealand Petroleum & Minerals Online Exploration Database*. Accessed: Aug. 23, 2021. [Online]. Available: https://data.nzpam.govt.nz/GOLD/system/mainframe.asp

[39] G. Science. (2004). *Petlab: New Zealand's National Rock, Mineral and Geoanalytical Database*. Accessed: Aug. 23, 2021. [Online]. Available: http://pet.gns.cri.nz/

[40] A. Boston. (2018). *Crimes in Boston*. Accessed: Nov. 28, 2023. [Online]. Available: https://www.kaggle.com/datasets/AnalyzeBoston/crimes-in-boston

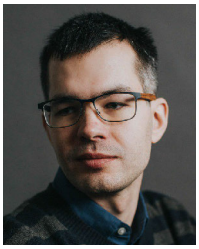
[41] H. K. Sharma, T. Choudhury, and A. Kandwal, "Machine learning based analytical approach for geographical analysis and prediction of Boston city crime using geospatial dataset," *GeoJournal*, vol. 88, pp. 15–27, Aug. 2021.
[42] J. Yin, I. A. Michael, and I. J. Afa, "Machine learning algorithms for visualization and prediction modeling of Boston crime data," Preprints, 2020.
[43] S. Tasnim, P. Sarkar, A. Hossain, and M. A. Ali, "A classification approach to predict severity of crime on Boston city crime data," in *Data Science and SDGs: Challenges, Opportunities and Realities*. Bangladesh: Univ. Rajshahi, Department of Statistics, 2019.
[44] ERA5. (2019). *Using Copernicus Atmosphere Monitoring Service Information*. Accessed: Dec. 7, 2023. [Online]. Available: https://confluence.ecmwf.int/display/CKB/ERA5%3A+data+documentation
[45] Q. He, Z. Shen, M. Wan, and L. Li, "Precipitable water vapor converted from GNSS-ZTD and ERA5 datasets for the monitoring of tropical cyclones," *IEEE Access*, vol. 8, pp. 87275–87290, 2020.
[46] H. Hersbach, B. Bell, P. Berrisford, S. Hirahara, A. Horányi, J. Muñoz-Sabater, J. Nicolas, C. Peubey, R. Radu, and D. Schepers, "The ERA5 global reanalysis," *Quart. J. Roy. Meteorol. Soc.*, vol. 146, no. 730, pp. 1999–2049, Jul. 2020.
[47] Y. Li. (2018). *Djia 30 Stock Time Series*. Accessed: Dec. 13, 2023. [Online]. Available: https://www.kaggle.com/datasets/szrlee/stock-time-series-20050101-to-20171231
[48] M. Fabbri and G. Moro, "Dow Jones trading with deep learning: The unreasonable effectiveness of recurrent neural networks," in *Proc. 7th Int. Conf. Data Sci., Technol. Appl.*, 2018, pp. 142–153.
[49] Y.-H. H. Tsai, S. Bai, M. Yamada, L.-P. Morency, and R. Salakhutdinov, "Transformer dissection: An unified understanding for transformer's attention via the lens of kernel," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong: Association for Computational Linguistics, Nov. 2019, pp. 4344–4353. [Online]. Available: https://aclanthology.org/D19-1443
[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.
[51] H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N. Smith, and L. Kong, "Random feature attention," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–19.
[52] M. Munkhoeva, Y. Kapushev, E. Burnaev, and I. Oseledets, "Quadrature-based features for kernel approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–10.
[53] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[54] S. Serrano and N. A. Smith, "Is attention interpretable?" in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, A. Korhonen, D. Traum, and L. Márquez, Eds., Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2931–2951. [Online]. Available: https://aclanthology.org/P19-1282
[55] S. Jain and B. C. Wallace, "Attention is not explanation," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics*, 2019, pp. 1–16. [Online]. Available: https://api.semanticscholar.org/CorpusID:67855860
[56] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.
[57] R. Kail, K. Fedyanin, N. Muravev, A. Zaytsev, and M. Panov, "ScaleFace: Uncertainty-aware deep metric learning," in *Proc. IEEE 10th Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2023, pp. 1–10. [Online]. Available: https://api.semanticscholar.org/CorpusID:252089949
[58] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Hong Kong: Association for Computational Linguistics, 2019, pp. 11–20.

ALINA ERMILOVA received the B.S. degree in information science and computation technology from the National Research University–Higher School of Economics (HSE University), Moscow, in 2021, and the M.S. degree in data science from Skolkovo Institute of Science and Technology (Skoltech), Moscow, in 2023, where she is currently pursuing the Ph.D. degree in computational and data science and engineering. Since 2020, she has been with the Applied AI Center, Skoltech. Her research interests include time series processing, efficient transformers, models' robustness, and adversarial attacks.

NIKITA BARAMIIA received the B.S. degree in economics from the Faculty of Economics, Lomonosov Moscow State University, Moscow, in 2021, and the M.S. degree in data science from Skolkovo Institute of Science and Technology (Skoltech), Moscow, in 2023. Since 2020, he has gained diverse experience from working in jewelry and bank industries to classifieds IT company Avito as well as research activity with Skoltech. His research interests include self-supervised learning, models robustness, and cost-effective adversarial training.

VALERII KORNILOV received the B.S. degree in economics from the Faculty of Economics, Lomonosov Moscow State University, Moscow, in 2021, and the M.S. degree in data science from Skolkovo Institute of Science and Technology (Skoltech), Moscow, in 2023. His research interests include classic ML and DL approaches for robotics tasks. The primary topic of interest is foundational models for manipulation and navigation.

SERGEY PETRAKOV received the B.S. degree in economics from the Faculty of Economics, Lomonosov Moscow State University, Moscow, Russia, in 2021, and the M.S. degree in data science from Skoltech, Moscow, in 2023, where he is currently pursuing the Ph.D. degree in computational and data science and engineering. His research interests include application of uncertainty estimation methods to NLP tasks and modern transformer-based models.



ALEXEY ZAYTSEV was born in Kharkiv, Ukraine. He received the degree from MIPT, in 2012, and the Ph.D. degree in mathematics from IITP RAS, in 2017. In his master's thesis, he proposed a modification of the Bayesian approach for linear regression that allows an automated feature selection. He is currently an Assistant Professor with Skoltech. His research interests include the development of new methods for sequential data, Bayesian optimization, and embeddings for weakly structured data.