Lomonosov Moscow State Universtiy
Faculty of Applied Mathematics and Cybernetics
Laboratory of Mathematical Methods of Recognition

Khismatullin Vladimir

Interpretable latent spaces for multivariate time series

Moscow, 2023

Contents

1 Introduction

In the real world, objects possess many characteristics understandable to humans: color, shape, size. Modern deep learning models, known as autoencoders, can learn to represent high-dimensional objects in a lower-dimensional latent space with minimal loss. However, such representations convey information about the natural characteristics of objects only indirectly. In other words, although these models achieve high levels of data compression, they act as black boxes resulting from the optimization of specific objective functions. The challenge of obtaining interpretable low-dimensional representations lies in constructing representations that not only effectively describe real-world objects but also carry explicit semantic meanings understandable to humans.

Solving this problem is highly relevant in many areas involving generative models. On the one hand, it allows for generating data through the controlled variation of semantic meanings in an interpretable latent space. On the other, it serves as a form of regularization for models trained on top of latent representations [1]. Existing models of interpretable spaces in image generation already show excellent performance in tasks like style transfer. For example, in [2], style transfer is applied to clothing, hairstyles, and gestures in images. Another model [3] allows for the transfer of facial features.

For sequences, there are solutions that build interpretable representations for each time step. For instance, representations are built for video frames to separate static and dynamic parts, enabling the prediction of future frames [4], [5]. In text style transfer, representations are divided into components responsible for style and content, enabling continuous changes in tone from negative to positive and vice versa [6]. The work in [7] introduces a probabilistic model for sequences of latent representations, achieving interpretable results in various style transfer tasks for sequences.

All of these models assume that the dimensionality of the objects—sequence elements—is much higher than that of the projected multivariate time series. Moreover, they focus on obtaining interpretable representations without the use of concept labels during training. Both aspects highlight the limited applicability of existing approaches to multivariate time series.

Thus, the goal of this study is to develop a model for obtaining efficient and interpretable representations specifically for multivariate time series.

2 Problem Formulation

Let us introduce some basic notation:

- $\theta_1, \theta_2, \dots, \theta_K$ — interpretable concepts, each taking values from Ω_k. Typically, $\Omega_k \subset \mathbb{N}$ or $\Omega_k \subset \mathcal{R}$.
- $(X_i, y_i^1, y_i^2, \dots, y_i^K)_{i=1}^N$ — a dataset of time series and their corresponding concept values, where $X_i \in \mathbb{X} \subset \mathcal{R}^{L \times m}$, with L as the length of the time series and m as the dimensionality of the series projection. $y_i^k \in \Omega_k, , k = \overline{1, K}$.

- $z_i, \tilde{z}_i \in \mathcal{R}^h$ — latent and interpretable representations corresponding to object X_i. As mentioned earlier, the latent representation is a low-dimensional ($h << m \cdot L$) encoding of X_i, while the interpretable representation carries additional semantic meaning, as defined in this section.

As noted, the task of extracting interpretable representations involves identifying a specific space and the transformation associated with it. This section provides a formal definition of these concepts and the formulation of the tasks needed to achieve the study's objective.

2.1 Key Definitions

There are two main approaches to defining the interpretability of latent representations. The first approach involves a qualitative definition of interpretability through a probabilistic framework. The second introduces metrics that quantitatively assess interpretability. The classical intuitive definition is as follows [1]:

Definition 2.1. General Formulation:
Interpretable latent representations are latent encodings of objects that disentangle various independent and informative factors underlying data variations. Each latent variable should correspond to a single factor, and each factor should be associated with only one set of variables, excluding cases where latent variables are interdependent.

This first possible formalization treats real-world sequences as parametric functions, which works well for processes such as particle motion in space. For example, one interpretable variation parameter could be color, directly tied to wavelength.

Definition 2.2. Functional Formulation:
Given a parametric mapping $f = f(t|\theta_1, \theta_2, \ldots, \theta_K) : X \rightarrow \Omega$, the task of uncovering interpretable latent representations involves reconstructing the dependency $f(\cdot|\theta_1, \theta_2, \ldots, \theta_K)$ from a sample of function values, the argument t, and the concepts.

In practice, the factors of variation can be divided into two groups: the first group unambiguously defines the object, while the second introduces minor variations, which can be treated as noise. For example, in the case of shapes, the color of the shape is a significant concept, while the age of the metal it is made from is insignificant. This leads to the following definition:

Definition 2.3. Probabilistic Formulation: It is assumed that the observed objects — time series $X \in \mathbb{X} \subset \mathbb{R}^{T \times M}$ — are drawn from a certain distribution $X \sim p(X|\theta_1, \ldots, \theta_K)$, where the interpretable parameters $\theta_1, \ldots, \theta_K$ are the factors of variation. The task of obtaining interpretable latent representations involves finding an invertible mapping $\phi : \mathbb{X} \subset \mathbb{R}^{L \times m} \rightarrow \mathbb{R}^h, , \tilde{z} = \phi(X)$, such that the marginal distribution $p(\tilde{z}|\theta_1, \ldots, \theta_K)$ has a predefined form $p_0(\tilde{z}|\theta_1, \ldots, \theta_K)$.

It is obvious that this formulation is equivalent to the functional formulation in the case of a finite set of concepts, as it involves reconstructing the distribution of the data

$P(X|\theta_1,\ldots,\theta_K)$ using some latent representations. Indeed, if the required mapping is constructed, the original probability function can be recovered as $P(X|\theta_1,\ldots,\theta_K) = p_0(\phi^{-1}(\tilde{z})|\theta_1,\ldots,\theta_K)$.

Having defined the concept of interpretable representation, we now present the method for achieving the goal of this work:

Definition 2.4. Solution to the Problem of Interpretable Representations: The solution consists of three parts:

1. **Choosing the marginal distribution** $p_0(\tilde{z}|\theta_1,\ldots,\theta_K)$ that defines the interpretable space. At this stage, the theoretical quality of the distribution must be evaluated against objective metrics, along with its applicability to practical tasks, such as object generation.
2. **Designing and training an autoencoder** — an algorithm consisting of an encoder that efficiently maps the original time series to a latent representation: $E : \mathbb{X} \subset \mathbb{R}^{L\times m} \to \mathbb{R}^h$, and a decoder that reconstructs the original series from the latent representation with minimal loss: $D : \mathbb{R}^h \to \mathbb{X} \subset \mathbb{R}^{L\times m}$, subject to the constraint $\|D(E(X)) - X\|^2 \to \min$.
3. **Creating and optimizing a bijective normalizing mapping** that transforms the latent space into an interpretable latent space: $T : Z \to \tilde{Z} : p(T(z)|\theta_1,\ldots,\theta_K) = p_0(\tilde{z}|\theta_1,\ldots,\theta_K)$, thus implementing the marginal distribution by either variable transformation or latent space optimization.

The image 1 illustrates the proximity of objects with different concepts across various spaces. Proximity is determined by constructing a probabilistic model using t-SNE for each of the representations. In the original space, the objects are somewhat mixed: there are many clusters composed of several classes. When mapped to the latent space, despite a 20-fold reduction in dimensionality, the proximity of objects does not imply similarity in an interpretable sense. Only in the interpretable space is it clearly visible that the objects cluster according to the concepts introduced in the marginal distribution — by type of actions, indicated by color, and by type of viewpoints, indicated by the shape of the points.

Following the probabilistic formulation and the definition of the problem's solution introduced earlier, the separation of concepts into components is achieved by modeling their relationships through a distribution. Specifically, it is assumed that although the distributions $p(X|\overline{\theta})$ and $p(z|\overline{\theta})$ are unknown, the components are transformed to a known distribution $p_0(\tilde{z}|\overline{\theta})$ using the mapping $T : z \to \tilde{z}$. Thus, the process of obtaining new interpretable representations from latent ones is fully determined by the distribution model p_0 and the transformation algorithm T.

It is important to note that it is impossible to model all potential factors of environmental variation. Therefore, a new concept, θ_0, is added to the primary set of concepts to account for all factors not included in the training set. For example, if the observed objects are cars, in addition to obvious concepts such as model, shape, and color, there are various modifications — tinting, decals, etc. Although these additional

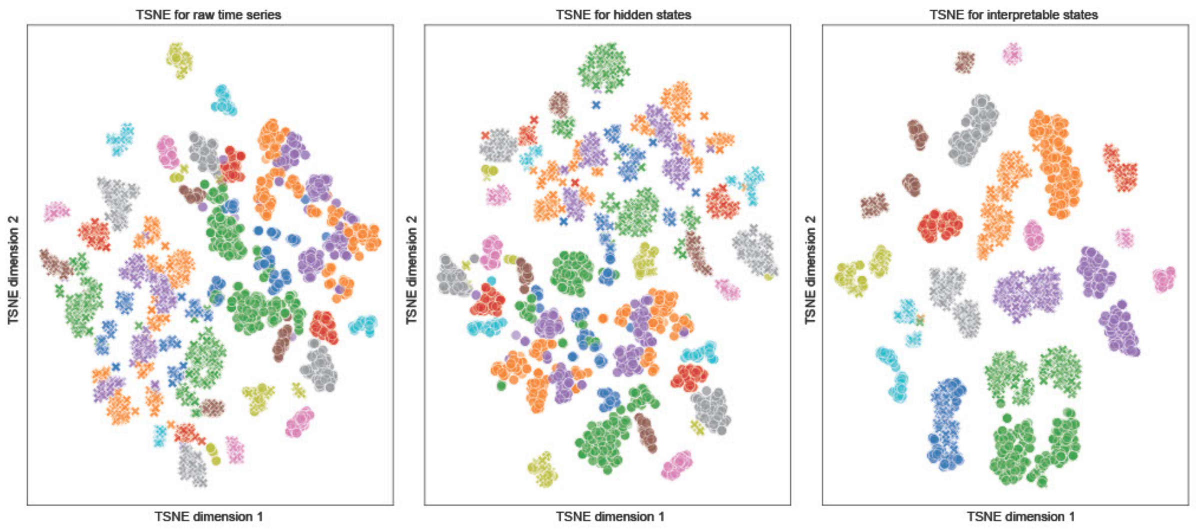

Figure 1: Comparison of object proximity in the training set across three representations: original (time series $X \in \mathcal{R}^{60\times26}$), intermediate (latent representation of the autoencoder $z \in \mathcal{R}^{64}$), and interpretable (normalized using the CENNF method $\tilde{z} \in \mathcal{R}^{64}$). Colors represent different types of actions, while shapes denote different viewpoints.

concepts are meaningful, they are unannotated in the original dataset and are referred to as residual factors.

2.2 Quantitative Assessment of Interpretability

To further formalize the interpretability of latent representations, metrics are introduced to evaluate the quality of interpretability. The literature [8] provides multiple examples of heuristic indicators that effectively compare latent representations for interpretability. In this work, we use a metric considered the best for the key concepts discussed in [8]. These concepts are directly linked to an alternative definition of interpretability provided in the same review.

Alternative Definition of Interpretability An independent alternative to the probabilistic formulation is a task based on an invertible mapping $T : \mathcal{R}^h \to \mathcal{R}^h$ and a given set of objects $\mathbf{X} = X_1, X_2, \dots, X_N$, their representations $\mathbf{Z} = (z_1, z_2, \dots, z_N)$, and a set of observed concepts $\overline{\theta} = (\theta_1, \theta_2, \dots, \theta_K)$. The goal is to obtain a representation $\tilde{z} = T(z)$ in which clearly identifiable interpretable components exist.

The interpretable components of the vector representation $\tilde{z} \in \mathcal{R}^h$ are defined as sets of indices:

$$J_k = \{i_1, i_2 \dots i_p\}, k = \overline{1, K} : \cup_{k=0}^{K} J_k = \{1, 2, \dots, h\},\ J_k \cap J_j = \emptyset,\ j \neq k.$$

The interpretability of a component indexed by k means that it is associated with the concept θ_k. For example, if $J_1 = 7, 8, 9$, then the components $\tilde{z}_7, \tilde{z}_8, \tilde{z}_9$ must correspond to the concept θ_1.

Furthermore, the correspondence of a concept to a component J_k implies that this component is determined solely by the value of the concept θ_k. Moreover, it must be informative, meaning that there should exist a mapping $\tau_k : \mathcal{R}^{|J_k|} \to \Omega_k$ that reconstructs the concept value from the interpretable representation. These three properties are called modularity, compactness, and informativeness. Based on these properties, the DCI metric [9] is constructed, which is recognized by the authors of two reviews [8], [10] as the best by several criteria.

DCI Metric To quantitatively evaluate the relationship between latent variables and factors of variation, a matrix of relative importance $R \in \mathcal{R}+^{h\times K}$ is introduced. This matrix reflects the importance of each component in predicting a given concept. A high value of Ri, k indicates that z_i plays a significant role in predicting the value of θ_k. We use the MDI algorithm — mean decrease impurity — with a random forest to construct this matrix:

$$R_{i,k} = \sum_{T \in \mathrm{RF}_k} \sum_{j \in T} w_j \cdot [h_j = i]$$

Here, RF_k is the random forest used to predict θ_k; T is a tree from this forest; h_j is the index of the variable used at the j-th node of the tree; and w_j is the weight of the node, determined by the number of objects split at that node.

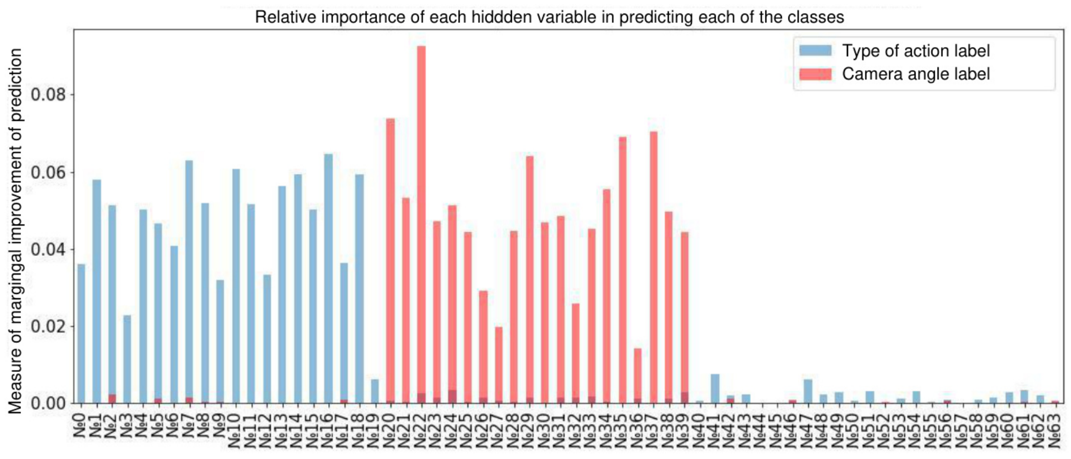

Figure 2: Importance of each component of the interpretable latent representation $\tilde{z}$ in predicting the class label θ_k. Random forest MDI algorithm.

The graph shows an interpretable representation: the first 20 coordinates of the vector correspond to the first concept — the type of human movement in the video, while coordinates 20–40 correspond to the second concept — the camera angle. Thus, the requirement for modularity is satisfied for this representation. However, compactness is violated because each concept is associated with 20 variables rather than one. The property of informativeness holds if the true values of the type of movement and camera angle can be determined using the 64 variables.

After defining the matrix of relative importance, the DCI evaluation approach consists of three parts:

1. **Disentanglement** — disentanglement, also known as modularity. This evaluates how many different parameters — concepts θ_k — on average need to be considered to determine the value of the variable $\tilde{z}_i$. In formula:

$$D = \frac{1}{h}\sum_{i=1}^{h} D_i = \frac{1}{h}\sum_{i=1}^{h}(1-\hat{H}_i), \text{ where } \hat{H}_i = -\sum_{k=1}^{K} P_{i,k}\log_K P_{i,k};\ P_{i,k} = \frac{R_{i,k}}{\sum_{k=1}^{K} R_{i,k}}$$

clarify: when all concepts $\theta_1, \theta_2, \ldots, \theta_K$ contribute equally to the latent variable $\tilde{z}_i$, the normalized entropy value equals $-K\frac{1}{K}\log_K\frac{1}{K} = 1$. Thus, the value of the metric $D_i = 1 - \hat{H}_i = 1 - 1 = 0$ is minimal for this variable.
2. **Completeness** — completeness, also referred to as compactness. This evaluates how many different variables $\tilde{z}_i$ need to be considered, on average, to determine the value of the concept θ_k. In formula:

$$C = \frac{1}{K}\sum_{k=1}^{K} C_k = \frac{1}{K}\sum_{i=1}^{K}(1-\hat{H}_k), \text{ where } \hat{H}_k = -\sum_{i=1}^{h} P_{i,k}\log_h P_{i,k};\ P_{i,k} = \frac{R_{i,k}}{\sum_{i=1}^{h} R_{i,k}}$$

Similarly, if all variables $\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_{m_I}$ contribute equally to the prediction of θ_k, the corresponding metric $C_k = 1 - \hat{H}_k = 0$ will be minimal for this concept.
3. **Informativeness** — informativeness, also called clarity. This measures the total amount of information about the observed concepts contained in the interpretable representations. Let $\mathcal{L}_k(y, \hat{y})$ be a loss function, for example, $\mathcal{L}_k(y, \hat{y}) = [y \neq \hat{y}]$, and let $\psi_k(\tilde{z})$ be an algorithm predicting the value of the concept θ_k based on the interpretable representation $\tilde{z}$. Then the informativeness measure is expressed as the average loss over the holdout set:

$$I = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{N_t}\sum_{n=1}^{N_t}\mathcal{L}(y_n^k, \psi_k(\tilde{z}_n)),$$

where $\{\tilde{z}_n, y_n^1, y_n^2, \ldots, y_n^K\}_{n=1}^{N_t}$ is the holdout set in which the objects are mapped to the latent interpretable space.

Having defined the key factors of interpretability and the ability to quantify them, we now consider approaches to constructing the marginal distribution that allow for both easy generation of new objects and suitability for evaluation using the discussed metric.

2.3 Approaches to Extracting Interpretability

There are two fundamentally different approaches to constructing latent representations. The first is an unsupervised learning task, which involves identifying concepts based on patterns in the data without explicitly providing the model with information about these concepts during training. The second involves constructing latent representations based on information about the concepts present in the training set.

The first approach has seen broader development since it is less powerful, and constructing high-quality solutions using this approach requires more complex modeling.

Moreover, the authors of [11] argue that modeling interpretable dependencies without supervision is theoretically impossible. However, if an algorithm can identify groups of variables $J_1, \dots, J_K$ corresponding to different concepts, then by training a classification algorithm on top of these representations and using the matrix of relative importance, it is possible to establish a correspondence between these groups of variables and the concepts $\theta_1, \theta_2, \dots, \theta_K$.

According to the introduced definition 2.3, the approach to extracting interpretability is directly related only to the chosen model for the distribution of interpretable variables with known concepts $p_0(\tilde{z}|\theta_1, \theta_2, \dots, \theta_K)$. Therefore, to define a specific approach, it is sufficient to write out the marginal distribution considered for it.

Unsupervised Approach The first approach to training interpretable representations is unsupervised learning. In this case, a general distribution is defined. There is no class division, but it is possible to divide the coordinates into $\hat{K}$ groups based on their proximity.

$$p_0(\tilde{z}) = p(\mathbf{z}J_1, \mathbf{z}J_1, \dots, \mathbf{z}J_K) = \prod k = 1^K p_k(\mathbf{z}J_k) = \prod k = 1^K p_k(z_{i_1}, z_{i_2}, \dots, z_{i_{j_k}}) \quad (1)$$

For the distribution within each group, a simple distribution is typically chosen [1], such as a factorized normal distribution:

$$p_k(z_{i_1}, z_{i_2}, \dots, z_{i_{j_k}}) = \prod_{i \in J_k} \mathcal{N}(z_i|\mu_i, \sigma_i^2) \quad (2)$$

Pairwise Object Comparison An intermediate approach between supervised and unsupervised learning is based on the equality of concepts for objects. That is, this approach is used when the dataset does not contain complete information about the values of the concepts $\theta_1, \theta_2, \dots, \theta_K$, but only information about whether the concepts are equal or different for different objects in the dataset. In this case, a similarity parameter $\sigma_{\text{sim}} < 1$ is defined, and the following types of distributions are considered [2]:

$$\begin{aligned} p_k(\mathbf{z}^1_{J_k}, \mathbf{z}^2_{J_k}) = p_k(\mathbf{z}^1_{J_k}) p_k(\mathbf{z}^2_{J_k}) &= \prod_{i \in J_k} \mathcal{N}(z_i^1|\mu_i, \sigma_i^2) \prod_{i \in J_k} \mathcal{N}(z_i^2|\mu_i, \sigma_i^2) &, y_k^1 \neq y_k^2 \\ &= \prod_{i \in J_k} \mathcal{N}(z_i^1|\mu_i, \sigma_i^2) \prod_{i \in J_k} \mathcal{N}(z_i^2|z_i^1 \sigma_{\text{sim}} + \mu_i(1 - \sigma_{\text{sim}}), \sigma_i^2(1 - \sigma_{\text{sim}}^2)) &, \text{otherwise} \end{aligned}$$

Thus, when the values of the concept θ_k do not match for a pair of objects, the corresponding latent variables are assumed to be independent. If they match, the variables are correlated with the parameter σ_{sim}. In the simplest case, when $\mu_i = 0, , \sigma_i = 1, , i = \overline{1, K}$, the formula takes a more understandable form:

$$\begin{aligned} p_k(Z^1_{J_k}, Z^2_{J_k}) &= \prod_{i \in J_k} \mathcal{N}(z_i^1|0, 1) \prod_{i \in J_k} \mathcal{N}(z_i^2|0, 1) &, y_k^1 \neq y_k^2 \\ &= \prod_{i \in J_k} \mathcal{N}(z_i^1|0, 1) \prod_{i \in J_k} \mathcal{N}(z_i^2|z_i^1 \sigma_{\text{sim}}, (1 - \sigma_{\text{sim}}^2)) &, \text{otherwise} \end{aligned} \quad (3)$$

That is, if the k-th concepts of the objects match, the pair of objects is not independently distributed. Specifically, the second object in the J_k coordinates has a mean close to the coordinates of the first object, and the variance is reduced by a factor of $\frac{1}{1-\sigma_{\text{sim}}^2}$.

This method can also be extended to the unsupervised learning case. Assuming some objects are similar, the procedure can be generalized by introducing a similarity function $\sigma_{\text{sim}}(X_1, X_2)$. Similarity can refer to proximity in terms of a semantic factor, such as class, or proximity according to an analytical measure, such as the widely used DTW measure for time series.

Conditional Distribution In practice, the previous two approaches can successfully be used in latent transformations — generating new objects by replacing parts of interpretable representations. For example, if a group of variables J_k corresponds to the concept of skin color θ_k, changing these variables can alter the skin color of a person from the original to another color present in the dataset. Furthermore, this change can be made gradually using a convex combination of latent representations.

However, these methods do not allow for the high-quality generation of new objects conditioned on a class, which requires more powerful supervised approaches. The literature [11] emphasizes that these methods fundamentally differ from the previous ones and focus more on conditional distribution generation than on learning latent concepts. Therefore, this section primarily presents approaches to conditional generation rather than interpretable unfolding, although the task remains the same.

To model a conditional distribution using the same variable factorization formulas 1, the following formula for the conditional distribution of latent representations is introduced:

$$p_k(z_{i_1}, z_{i_2}, \dots, z_{i_{j_k}} | \theta_k) = \prod_{i \in J_k} \mathcal{N}(z_i | \mu_i(\theta_k), \sigma_i^2(\theta_k)) \tag{4}$$

When the set of values θ_k is finite, a center and variation are defined for each class based on the value of θ_k. When the set of values θ_k is infinite, a continuous mapping $g : \Omega_k \to \mathcal{R}^{|J_k|}$ is defined. For example, in the task of learning cosine amplitudes, each latent amplitude θ_k is mapped to a vector $(\theta_k, \theta_k, \dots, \theta_k) \in \mathcal{R}^{|J_k|}$. In general, $\mu_i(\theta_k)$ and $\sigma_i(\theta_k)$ are parametric mappings optimized during the model's training process.

3 Considered Approaches

As noted earlier, to implement interpretable representations, in addition to the interpretable marginal distribution $p_0(\tilde{z}|\theta_1, \theta_2, \ldots, \theta_K)$, it is necessary to define the architecture of the autoencoder — the invertible mapping $E : \mathbf{X} \to Z$ and the architecture that ensures the normalization mapping $T : Z \to \tilde{Z}$.

3.1 Autoencoder Architecture

Let us first define the outer model — the autoencoder. A review of existing approaches for constructing latent representations for sequences has shown that all methods can be divided into two categories.

The first category produces latent representations in the form of sequences $z \in \mathcal{R}^{L \times h}$. In particular, deep recurrent networks and attention mechanisms are effectively used to obtain such sequences [12]. This approach assumes that the dimensionality of the original time series is much higher than the latent dimensionality $(m >> h)$. Although this mechanism is effective for tasks such as machine translation and video processing, it is rarely applicable to other tasks where this dimensional relationship holds.

The second category focuses on constructing fixed-length latent representations — vectors from the space $\mathcal{R}^h$. While data compression with this method significantly outperforms that of sequence-based representations, it comes with its own drawbacks. First, it is inefficient for tasks involving sequences of arbitrary or highly variable lengths. Second, the complexity of decoding such representations is significantly higher than that of sequence-based methods, as it requires constructing the mapping $D : \mathcal{R}^h \to \mathcal{R}^{L \times m}$.

In this work, we focus on methods that operate with fixed vector representations $z \in \mathcal{R}^h$, as most existing approaches to latent representation unfolding work with such representations. Moreover, for multivariate time series, most real-world data consists of sequences with either fixed or limited lengths.

3.1.1 Standard Approaches

Although recurrent networks are the best approach for many time series tasks, models based on LSTM and GRU exhibit unsatisfactory performance when decoding vector representations. For example, in the task of encoding human skeletal actions, such architectures fail to learn the concept of the skeleton as a whole, focusing only on variations of some of its parts. For this reason, autoencoder models based on recurrent neural networks with fixed-size latent representations were not considered.

Another popular model is deep one-dimensional convolutional neural networks. The first type of networks considered were those with one-dimensional convolutions, as exemplified in the work [13]. However, for the same skeletal encoding task, the author was unable to maintain the integrity of the skeleton during decoding.

3.1.2 Time Series Decomposition Models

After reviewing the existing autoencoder models for time series, the best-performing architecture was the one first proposed in the TimeVAE model [14]. The authors suggest using a convolutional network as the encoder and a block-based model for decomposing the series into seasonality and trend components as the decoder. This aligns with the notion that decoding time series is a more complex task than encoding them.

In more detail, the encoder is defined as the mapping $E : \mathbf{X} \to Z$

$$E(x) = F(C_r(C_{r-1}(\dots(C_2(C_1(x))))))$$

where $C_i, i = \overline{1, r}$ are convolutional neural layers, and F is a fully connected network.

To define the decoder blocks, we introduce the trend and seasonality matrices, which consist of polynomials of degree up to p and periodic functions:

$$\begin{aligned} T_{trend} \in \mathcal{R}^{L\times p} &: T_{l,j} = \left(\frac{l}{L}\right)^j \\ S_{season} \in \mathcal{R}^{L\times(2\cdot g-1)} &: S_{l,j} = \cos\left(l\frac{2\pi j}{L}\right), \ j = \overline{1, g} \\ & \quad S_{l,j} = \sin\left(l\frac{2\pi(j-g)}{L}\right), \ j = \overline{g+1, 2\cdot g-1} \end{aligned}$$

Additionally, neural networks are introduced to activate the blocks by generating coefficient matrices for the decomposition of the series into basis functions:

$$F_{trend} : \mathcal{R}^h \to \mathcal{R}^{p\times m} \text{ and } F_{season} : \mathcal{R}^h \to \mathcal{R}^{(2\cdot g-1)\times m}$$

The idea of the decoder D is to approximate the MSTL — decomposition of the time series. Specifically, given the latent representation z, the original time series is reconstructed as:

$$X = D(z) = T_{trend}F_{trend}(z) + S_{season}F_{season}(z)$$

It is also important to note that this procedure is valid in the sense that adding periodic functions with commensurable frequencies $\frac{1\cdot 2\pi}{L}, \frac{2\cdot 2\pi}{L}, \dots, \frac{g\cdot 2\pi}{L}$ results in a periodic function.

3.1.3 Proposed Model

To enhance the capabilities of the autoencoder for time series modeling tasks, a new model is proposed. The assumption of additivity of latent representations is introduced, formulated as a requirement that, on the one hand, the encoder must return the sum of the trend and seasonality estimates. On the other hand, the latent representation must be decoded additively as the sum of the trend and seasonal components. Specifically, if $X = X_{season} + X_{trend}$, then: $E(X) = z = E(X_{season}) + E(X_{trend}) = z_{season} + z_{trend}$. This property, however, is not satisfied in the first model.

In the proposed model, the decoder works in two stages and, unlike other autoencoder models, depends on the encoder. The complete flow of the improved model can be described in several steps:

1. Assume an encoder $E : \mathcal{R}^{L\times m} \to \mathcal{R}^h$ is given, and the latent representation $z = E(x) \in \mathcal{R}^h$ is obtained.
2. The trend component of the reconstructed series is computed as $\hat{x}trend = StrendsT_{trend}(z) \in \mathcal{R}^{l\times m}$, where $T_{trend}(z) : \mathcal{R}^h \to \mathcal{R}^{p\times m}$ is a deep neural network, and $S_{trends} \in \mathcal{R}^{l\times p}$ are polynomials of degree $0, 1, 2, \dots, p-1$.
3. The latent representation, excluding the trend, is obtained as: $z_{season} = z - E(\hat{x}_{trend}) = E(x) - E(T_{trend}T(E(z)))$.
4. The seasonal components are computed based on the latent representation, excluding the trend: $\hat{x}_{season} = S_{seasons}T_{season}(z_{season}) \in \mathcal{R}^{L\times m}$, where $T_{season}(z_{season}) : \mathcal{R}^h \to \mathcal{R}^{N\times m}$ is a deep neural network, and $S_{seasons} \in \mathcal{R}^{L\times N}$ represents the seasonal components of the series with length L. For $N = 2g$, it includes all cosines and sines with frequencies $\frac{2\pi k}{L}, k = \overline{1, g}$.
5. The final reconstructed series is given by: $\hat{x} = \hat{x}_{trend} + \hat{x}_{season}$.

The key difference in the proposed decoder model lies in step 3, where the latent representation of the recovered trend is subtracted from the latent representation, and the seasonal component is computed based on the result.

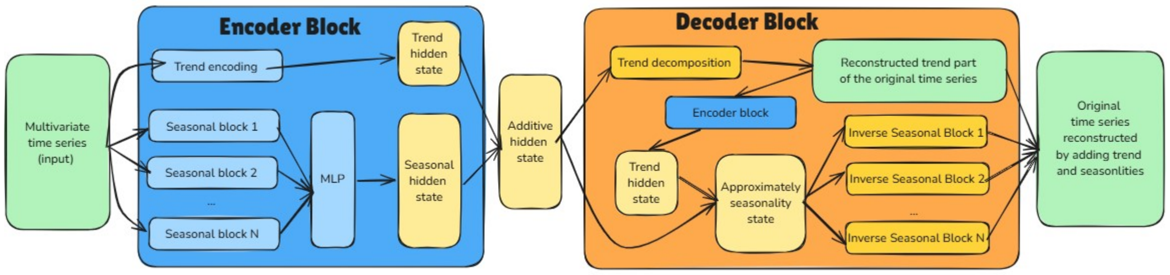

Figure 3: The proposed autoencoder architecture for multivariate time series. The encoder is based on trend and seasonality blocks, with the encoder used to extract the trend. Main idea is to impose an approximate additivity of the encoder:
$z = E(X_{\text{trend}} + X_{\text{season}}) \approx E(X_{\text{trend}}) + E(X_{\text{season}}) = z_{\text{trend}} + z_{\text{season}}$

In the proposed model, the encoder uses the reverse idea by generating representations as functions of the coefficients of decomposition into trend and seasonal components:

$$z = E_{season}\left(\langle X, S_{season}\rangle\right) + E_{trend}\left(\langle X, T_{trend}\rangle\right)$$

3.2 Architecture of the Normalizing Mapping

Regardless of the approaches used to obtain latent representations and their normalization, it is necessary to construct a mapping $T : z \to \tilde{z}$ that ensures the normalization of latent representations. A review of architectures performing this task identified two methods.

3.2.1 Normalizing Flows

The first approach explicitly maps the latent representation z to a new representation $\tilde{z}$. The transition from the original distribution $p(z)$ to the marginal distribution $p_0(\tilde{z})$ is achieved by transforming variables using the architecture of normalizing flows. A normalizing flow [15] is defined as an invertible mapping $T(z)$, represented as a composition of invertible transformations:

$$\tilde{z} = T(z) = T_\tau \circ T_{\tau-1} \circ \dots T_2 \circ T_1(z) \tag{5}$$

$$z = T^{-1}(T(z)) = T_1^{-1} \circ T_2^{-1} \circ \dots T_{\tau-1}^{-1} \circ T_\tau^{-1} \circ T_\tau \circ T_{\tau-1} \circ \dots T_2 \circ T_1(z) \tag{6}$$

The model is optimized by maximizing the likelihood of the latent (non-interpretable) representation $p(z)$.

$$\begin{gathered} p(z) = p_0(T(z)) \left|\det\left(\frac{\partial T(z)}{\partial z^T}\right)\right| \iff p(z)dz = p(\tilde{z})d\tilde{z} \\ \log\left(p(z)\right) = \log(p_0(T(z))) + \log\left|\det\left(\frac{\partial T(z)}{\partial z^T}\right)\right| \\ \log\left|\det\left(\frac{\partial T(z)}{\partial z^T}\right)\right| = \log\prod_{t=1}^{T}\left|\det\left(\frac{\partial T_t(\cdot)}{\partial T_{t-1}(\cdot)^T}\right)\right| = \sum_{t=1}^{T}\log\left|\det\left(\frac{\partial T_t(\cdot)}{\partial T_{t-1}(\cdot)^T}\right)\right| \end{gathered} \tag{7}$$

Given that $\tilde{z}$ is normally distributed: $\tilde{z} \equiv \mathcal{N}(\tilde{z}|0, I)$, the optimized expression takes a simple form:

$$\mathcal{L} := \log p(z) = -\frac{z^T z}{2} + \log\left(\frac{1}{2\pi}\right) - \log|T'(z)|$$

To simplify the calculation of the transition Jacobians,$\mathcal{J}_t = \left|\det\left(\frac{\partial T_t(\cdot)}{\partial T_{t-1}(\cdot)^T}\right)\right|$ the transformations T_i are chosen from a restricted class with easily computable transition Jacobians $\mathcal{J}_t$.

The coupling block described in [16] is used as the transformation T_t in the considered architecture.

$$\begin{gathered} \hat{z} = (\hat{z}_1, \hat{z}_2) = ((z_{1,1}, z_{1,2}^1, \dots, z_{1,h_1}^1), (z_{2,1}, z_{2,2}, \dots, z_{2,h_2}^2)) \\ T_t(z) = ((C_t(\hat{z}_2) + \exp\{H_t(\hat{z}_2)\} \odot \hat{z}_1), \hat{z}_2), \text{ where } C_t, H_t : \mathcal{R}^{h_2} \to \mathcal{R}^{h_1} \end{gathered}$$

This block first shuffles the variables by randomly splitting them into two groups, $\hat{z}_1$ and $\hat{z}_2$, and then applies a nonlinear invertible transformation to them. The Jacobian matrix is easy to compute:

$$\left|\det\left(\frac{\partial T_t(z)}{\partial z^T}\right)\right| = \prod_{i=1}^{h_1}\exp\{h_t(z_2)\}_i = \exp\left\{\sum_{i=1}^{h_1}(h_t(z_2))_i\right\}$$

A significant drawback of this model is the instability of the computations. Due to the large number of transformations, errors accumulate rapidly, which disrupts the invertibility of the flow. In practice, compositions of $T \geq 30$ such blocks fail to train. For this reason, additional activation normalization blocks are introduced at each step. The final block of the model can be expressed as follows:

$$\hat{T}_t(z) = T_i(\alpha_t(z - \beta_t)), \text{ where } \alpha_t \in \mathcal{R}_+,\ \beta_t \in \mathcal{R};\ T_t \text{ —previously defined network}$$

Finally, likelihood maximization $\log p(z)$ is equivalent to maximizing:

$$\log p_0(\tilde{z}) + \sum_{t=1}^{T} \log \left((\alpha_t)^h \exp \left\{ \sum_{i=1}^{h_1} (h_t(\alpha_t(z_2 - \beta_t)))_i \right\} \right) =$$
$$\log p_0(\tilde{z}) + \sum_{t=1}^{T} \left(h \log \alpha_t + \sum_{i=1}^{h_1} (h_t(\alpha_t(z_2 - \beta_t)))_i \right)$$

3.2.2 Variational Autoencoders

The key difference between this approach and the previous one is the simultaneous training of the autoencoder and the normalization of the latent representation. On the one hand, this method requires fewer parameters to normalize the representation. On the other hand, it is applied directly to objects and generally cannot be used with pre-trained latent representations.

The main idea is to approximate the latent representation $z = E(x)$, $z \sim q(z|x)$ to the marginal distribution $p_0(z)$ while minimizing the deviation of the reconstructed value $D(E(x))$ from the true value x. The variational lower bound for an unconditional variational autoencoder, which maximizes the unconditional likelihood $\log p_\phi(x)$, is derived as follows:

$$\log p_\phi(x) = \mathbb{E}_{z\sim q(z|x)} \log p_\phi(x) = \mathbb{E}_{z\sim q(z|x)} \log \frac{p_\phi(x,z)q(z|x)}{q(z|x)p_\phi(z|x)} =$$
$$= \mathbb{E}_{z\sim q(z|x)} \log \frac{p_\phi(x,z)}{q(z|x)} + KL(q(z|x)||p_\phi(z|x)) \geq \mathbb{E}_{z\sim q(z|x)} \log \frac{p(x,z)}{q(z|x)} = \text{VLB}(q,\phi)$$
$$\text{VLB}(q,\phi) = \text{const} + \mathbb{E}_{z\sim q(z|x)} \log \frac{p_0(z)}{q(z|x)} + \mathbb{E}_{z\sim q(z|x)} \log p_\phi(x|z) =$$
$$= \text{const} + \mathbb{E}_{z\sim q(z|x)} \log p_\phi(x|z) - KL(q(z|x)||p_0(z)) \quad (8)$$

A conditional autoencoder is also considered, which models the conditional distribution $\log p_\phi(x|\overline{\theta})$. Similarly, the estimate for the conditional case is derived as:

$$\log p_{\phi,\psi}(x|\overline{\theta}) \geq \text{VLB}(q,\phi,\psi) = \mathbb{E}_{z\sim q(z|x,\overline{\theta})} \log p_\phi(x|z,\overline{\theta}) - KL(q(z|x,\overline{\theta})||p_{0,\psi}(z|\overline{\theta})) \quad (9)$$

To fully derive the optimized functional, it is necessary to introduce the intermediate distribution $q(z|x,\overline{y})$ and the likelihood of the object. The distribution $p_0(z)$ is determined by the interpretability approach chosen earlier, in the first step of the problem solution 1.

Assuming the likelihood is normal with a mean at the decoded object, $p_\phi(x|z,\overline{y}) = \mathcal{N}(x|D(z),I)$, and the intermediate distribution is normal $q(z|x,\overline{y}) = \mathcal{N}(z|E(z),\Sigma(x))$, the complete functional can be derived analytically. The expression for the first term is the sum of squared deviations between the decoded object and the true value. For the KL divergence, if the marginal distribution is multivariate normal, the following holds:

$$\mathrm{KL}(p(\tilde{z})||p_0(\tilde{z})) = \mathbb{E}_{p(\tilde{z})} \log \frac{p(\tilde{z})}{p_0(\tilde{z})} = \mathbb{E}_{p(\tilde{z})} \log \frac{\sqrt{|\Sigma_0|} \exp\{-\frac{1}{2} (\tilde{z} - \mu(z))^T \Sigma(z) (\tilde{z} - \mu(z))\}}{\sqrt{|\Sigma(z)|} \exp\{-\frac{1}{2} (\tilde{z} - \mu_0)^T \Sigma_0 (\tilde{z} - \mu_0)\}} =$$

$$= \mathbb{E}_{p(\tilde{z})} \left(\frac{1}{2} \log \frac{|\Sigma_0|}{|\Sigma(z)|} - \frac{1}{2} (\tilde{z} - \mu(z))^T \Sigma(z) (\tilde{z} - \mu(z)) + \frac{1}{2} (\tilde{z} - \mu_0)^T \Sigma_0 (\tilde{z} - \mu_0) \right)$$

By substituting the given formulas into 8 or 9, we obtain the maximized functionals for the tasks of unconditional or conditional autoencoders, respectively.

In practice, more complex loss functions are used, but all of them are based on approximating the log-likelihood through the variational lower bound [17]. In the task of unfolding neural networks — constructing interpretable latent spaces without supervision — the β-VAE model [18] is typically chosen as the baseline. The only difference between it and the unconditional variational autoencoder lies in a heuristic modification of the variational lower bound functional:

$$\log p_\phi(x) \geq \mathrm{VLB}_\beta(q, \phi) = \mathbb{E}_{z \sim q(z|x)} \log p_\phi(x|z) - \beta KL(q(z|x)||p_0(z)) \tag{10}$$

The hyperparameter $\beta > 1$ is introduced to increase the independence of the components of $q(z|x)$. Indeed, in the limiting case $\beta \to \infty$, the distribution degenerates into $p_0(z)$, which is assumed to be normal with independent components. Consequently, the components of $z = E(x) \sim q(z|x)$ become independent. A significant improvement to this model is the idea of investigating [3]:

$$\log p_\phi(x) \geq \mathrm{VLB}_\beta(q, \phi) = \mathbb{E}_{z \sim q(z|x)} \log p_\phi(x|z) - \beta \left| KL(q(z|x)||p_{0\psi(z)}) - \Gamma \right| \tag{11}$$

As the authors show, a linear increase in the hyperparameter Γ with the growth of the epoch number first makes the components independent and then gradually tunes the model to recover the original data. According to them, this approach allows for controlling the amount of information lost during the minimization of the divergence between distributions.

Proposed Improvement It is also worth noting that no heuristics have been found to improve the interpretability of representations for the supervised learning case. Therefore, a specialized architecture for a conditional variational autoencoder is proposed for the case of multiple concepts. Let us compare it with the standard architecture of a conditional variational autoencoder—CVAE:

$$p_0(z|x, \overline{\theta}) = \mathcal{N}(z|\mu(\overline{\theta}), \Sigma(\overline{\theta})) = \prod_{i=1}^{h} \mathcal{N}(z|\mu(\overline{\theta})_i, \sigma(\overline{\theta})_i) \tag{12}$$

In which $\mu(\overline{\theta}), \sigma(\overline{\theta})$ are deep neural networks $\mu, \sigma : \mathcal{R}^K \to \mathcal{R}^h$.

The proposed method relies on the introduced concepts of partial block distribution 1, 4. The key difference lies in constructing independent networks for predicting the centers of different parameters θ_k. Specifically:

$$p_0(z|x,\overline{\theta}) = \prod_{k=0}^{K} p_k(z_{J_k}|\theta_k) = \mathcal{N}(z_{J_0}|0,I)\prod_{k=1}^{K}\mathcal{N}(z_{J_K}|\mu_k(\theta_k),\Sigma_k(\theta_k)) =$$
$$= \mathcal{N}(z_{J_0}|0,I)\prod_{k=1}^{K}\prod_{i\in J_k}\mathcal{N}(z_i|\mu_k(\theta_k)_i,\Sigma_k(\theta_k)_i) \tag{13}$$

Its potential advantage lies in the independence of networks across different classes. For example, if two features θ_1, θ_2 are correlated for some reason in the dataset, the CVAE model will learn this dependency, making it less likely to accurately interpret pairs (θ_1, θ_2) that were not encountered in the training set.

The proposed approach, however, avoids these drawbacks, as the coordinate blocks J_1, J_2 corresponding to the variables θ_1, θ_2 are independent by design.

Thus, with proper model training—i.e., by minimizing the KL divergence between the latent and marginal distributions, $KL(q(z|x,\overline{\theta})||p_{0,\psi}(z|\overline{\theta}))$, the interpretable components will approximately satisfy $z \sim q(z|x,\overline{\theta}) \approx p_{0,\psi}(z|\overline{\theta})$, which corresponds to the key property of high-quality interpretable representations—modularity.

4 Results

This section compares all three components of the interpretable representation model. It addresses questions about the quality of different approaches for constructing latent representations, the performance of interpretability architectures based on key metrics, and the ability to generate unique series.

4.1 Considered Datasets

To evaluate the quality of the autoencoder models and the unfolding of interpretable representations, three datasets were selected:

Sine As a simple model, synthetic data representing the realizations of certain functions on a discrete time grid is considered, generated according to the following principle: First, random amplitudes, shifts, and frequencies of cosines are generated:

$$\alpha_i \sim \mathbb{U}(0.3, 5),\ \psi_i \sim \mathbb{U}(0, 2\pi),\ \phi_i \sim \mathbb{U}\left(0.1, \frac{\pi}{2}\right),\ i = \overline{1, 4}$$

The cosine series are then reproduced on a discrete grid:

$$f_i(t) = \alpha_i \cos(\phi(t + \psi_i)),\ i = \overline{1, 4},\ t = \overline{1, 50}$$

To make the task more challenging, the resulting time series has $6 = C_4^2$ components, formed by summing the series with indices from the set of index pairs $1, 2, 3, 4$ without repetitions. Thus, the j-th channel of the time series can be written as:

$$X_j = f_{c_{j,1}}(t) + f_{c_{j,2}}(t) + \varepsilon_{t,j},\ \varepsilon_{t,j} \sim \mathcal{N}(\varepsilon|0, 0.03),\ j = \overline{1, 6},\ t = \overline{1, 50}$$
$$\text{here } c_j = ((1,2),(1,3),(1,4),(2,3),(2,4),(3,4))_j$$

In this case, the interpretable concepts are the true amplitudes $\alpha_i, i = \overline{1, 4}$ and frequencies $\phi_i, , i = \overline{1, 4}$ of the cosines. It is important to note that index permutations are selected to generate six new periodic functions using the sets from the original four. This attempts to train the model to capture the true latent dependency.

StockV The second dataset consists of standard financial time series provided by the authors in [19]. A notable feature of this dataset is the significant variation in both the means and volatilities of the components. After normalization, the components are on the order of 10^{-1}: the first three components exhibit short-term deviations on the order of 10^{-4}, the next two have deviations on the order of 10^{-3}, and the final sixth component—representing total trading volume—has a magnitude on the order of 10^{-2}.

Interpretable representations are not explicitly highlighted in this task. However, it is possible to define certain concepts, such as the direction of the short-term trend, the trading season, or to use technical analysis tools to extract domain-specific concepts. Nevertheless, in this work, the dataset is used solely to evaluate the performance of autoencoders when dealing with noisy data.

UPENN There are limited publicly available datasets of large multivariate time series with clearly identifiable parameters. As a real-world example of time series with discernible concepts, we consider the UPENN Action dataset [20]. This dataset contains 2,500 videos with annotated human skeleton time series. The time series consist of 13 key points of the skeleton, each defined by x and y coordinates relative to the camera, forming a multivariate time series of dimensionality $L = 60, m = 26$.

As part of preprocessing, the data was cleaned to remove sequences where parts of the skeleton were not observed for various reasons. An example of such problematic instances is footage of a tennis player whose legs were out of the camera frame. As a result, fewer than 1,500 objects remained in the dataset. To ensure proper model training, the training set was augmented with standard normal noise.

The interpretable concepts selected are:

The action performed by the person: pitching and receiving in baseball, bowling, golf swings, pull-ups, push-ups, squats, and other physical exercises. The camera angle on the person: facing the person, from behind, from the left, and from the right.

4.2 Comparison of Autoencoder Models

For each of the previously described autoencoder models and for each dataset, 15 training iterations were performed, and the quality was evaluated on the holdout set. The standard deviation of the results is also reported.

Model / Data	BASE	TV-TV	TV-MY	MY-TV	MY-MY
UPENN	0.098 ± 0.028	0.054 ± 0.013	0.050 ± 0.012	0.052 ± 0.013	0.049 ± 0.012
Sine	0.109 ± 0.038	0.073 ± 0.020	0.061 ± 0.018	0.064 ± 0.017	0.054 ± 0.015
Stockv	0.017 ± 0.012	0.011 ± 0.012	0.010 ± 0.010	0.011 ± 0.010	0.010 ± 0.008

Table 1: The RMSE performance of the considered autoencoder models is as follows: BASE — an architecture based on a fully connected neural network; TV — a component of the TimeVAE model; MY — a component of the proposed modification. A hyphen separates the designation of the encoder and decoder parts. For example, TV-MY has the encoder from the TimeVAE model and the decoder from the proposed model.

Tables 1 and 2 confirm the hypothesis that the proposed method is a significant improvement over the existing TimeVAE architecture. Although the comparison of average performance and deviation does not clearly demonstrate the significance of the results, the statistical tests indicate that significant improvements were achieved across all datasets compared to both the TimeVAE model and the baseline model.

It is worth noting that the number of parameters in the proposed structure is approximately half the number of parameters in the BASE model and 1.5 times fewer than in TimeVAE.

Architectures:	BASE VS. MY		TimeVae VS. MY	
test / data	Wilcoxon	Permutation	Wilcoxon	Permutation
UPENN	$5.9 \cdot 10^{-46}$	$< 1 \cdot 10^{-8}$	0.001	0.008
Sine	$< 1 \cdot 10^{-400}$	$< 1 \cdot 10^{-8}$	$1.7 \cdot 10^{-303}$	$< 1 \cdot 10^{-8}$
Stockv	$6.1 \cdot 10^{-49}$	$< 1 \cdot 10^{-8}$	$2.9 \cdot 10^{-4}$	0.005

Table 2: Testing the hypothesis of equal average RMSE performance for the considered autoencoder models. Pairs of architectures compared: the baseline and the proposed model, and TimeVAE and the proposed model. Criteria for paired samples are used.

The most significant performance improvement is observed for the sine dataset, which aligns with the architecture's concept of handling series that can be effectively decomposed into trend and seasonality.

Another interesting aspect of the proposed model is its similarity to the non-differentiable MSTL decomposition. In a way, the proposed architecture serves as a differentiable solution to the MSTL problem—decomposing a series into the sum of a trend and several seasonal components.

4.3 Comparison of Interpretable Representations

We fix the autoencoder architecture to the proposed one. Let us consider different approaches to constructing interpretable representations:

1. UNF — a model described by the standard normalizing flow embedded in the autoencoder, as described in 3.2.1. The marginal distribution is $p_0(\tilde{z}) = \mathcal{N}(\tilde{z}|0, I)$.
2. β-VAE — a model described by the expression 11. Grid search was used to determine the optimal parameters: $\beta = 2.1$ and $\Gamma = 0.1 + 1.2t$, where t is the epoch number. The marginal distribution is $p_0(\tilde{z}) = \mathcal{N}(\tilde{z}|0, I)$.
3. LIKNF — a model with an embedded normalizing flow 3.2.1, implementing the pairwise similarity representation 2.3. The limit distribution is given by the equality for pairwise similarities 3.
4. CENNF — a model with an embedded normalizing flow 3.2.1, implementing the partial marginal distribution defined by the equality 4. The parameters of the marginal distribution μ_i are initialized by a uniform distribution on the unit sphere.
5. CVAE — a conditional variational autoencoder model defined by the system of equations 9, 12.
6. SCVAE — an improved conditional variational autoencoder model, defined by 9, 13, and described in 3.2.2.

According to the results, the CVAE, LIKNF, and SCVAE models are capable of producing interpretable representations with improvements in informativeness, modularity, and completeness. Unsupervised learning methods did not demonstrate satisfactory performance, which may be attributed to the small dataset size. Most successful cases of

Model	BASE	AE	β-VAE	UNF	LIKNF	CVAE	CENNF	SCVAE
Disent.	0.38 $\pm 1 \cdot 10^{-2}$	0.09 $\pm 1 \cdot 10^{-2}$	0.09 $\pm 8 \cdot 10^{-3}$	0.03 $\pm 4 \cdot 10^{-3}$	0.31 $\pm 1 \cdot 10^{-2}$	0.21 $\pm 1 \cdot 10^{-2}$	0.46 $\pm 1 \cdot 10^{-2}$	0.91 $\pm 1 \cdot 10^{-2}$
Comp.	0.08 $\pm 3 \cdot 10^{-3}$	0.03 $\pm 3 \cdot 10^{-3}$	0.03 $\pm 2 \cdot 10^{-3}$	0.04 $\pm 2 \cdot 10^{-3}$	0.14 $\pm 3 \cdot 10^{-3}$	0.07 $\pm 3 \cdot 10^{-3}$	0.13 $\pm 2 \cdot 10^{-3}$	0.26 $\pm 2 \cdot 10^{-3}$
Inform.	0.89 $\pm 2 \cdot 10^{-2}$	0.85 $\pm 2 \cdot 10^{-2}$	0.86 $\pm 2 \cdot 10^{-2}$	0.84 $\pm 2 \cdot 10^{-2}$	0.96 $\pm 1 \cdot 10^{-2}$	0.94 $\pm 1 \cdot 10^{-2}$	0.98 $\pm 2 \cdot 10^{-2}$	1.00 $\pm 3 \cdot 10^{-3}$

Table 3: Performance of the considered normalization models, UPENN data.
BASE — representation X directly from the time series;
AE — representation z based on the autoencoder, without unfolding;
β-VAE and UNF do not receive information about potential factors during training;
LIKNF is based on comparing factors for different objects in the dataset;
CVAE, CENNF, SCVAE — trained with conditions on factors of variation.

training β-VAE and UNF models, which use vectors as latent representations, are achieved on datasets with sizes on the order of 10^5. For example, in [2], the results for the UNF model were obtained on the MNIST dataset (around 60,000 training objects, without augmentations).

As expected in the proposed architecture, the SCVAE model exhibits very high modularity. The idea of introducing independence not only in the zero distribution but also in the model that implements it enables a high-quality division into independent interpretable components $J_1, J_2, \ldots, J_k$ in the vector $\tilde{z}$. It is worth noting that the improvement from artificially introducing component independence in the architecture is significant. First, this is the only difference between the CENNF and SCVAE models, with the latter showing significantly higher performance. Second, although the β-VAE and UNF models are designed to extract independent components, they fail to do so due to the complexity of this task.

4.4 Example of Use in a Generation Task

Following the example in [14], to further evaluate the interpretability of the representations, we compare the ability of various approaches to generate new, unique, yet plausible instances.

As a quantitative metric, discriminator error is used. In the experiment, subsets of the original dataset, sized at 100%, 50%, 20%, and 5% of the original, along with generated data of the same size, are used to train a model that distinguishes between real and artificial data. The model's performance is then evaluated on the holdout set and newly generated instances. In an ideal case of data generation, the discriminator's accuracy on the holdout data should reach 0.5, while a level of 1.0 indicates that the real and artificial data are easily distinguishable.

This approach ensures that, on the one hand, the new data must resemble the original, and on the other hand, the generated objects must be diverse enough that the discriminator does not easily memorize them.

model / α	β-VAE	LIKNF	CVAE	SCVAE
1.00	$[0.999, 0.999]$	$[0.988, 0.990]$	$[0.999, 0.999]$	$[0.985, 0.988]$
0.50	$[0.996, 0.997]$	$[0.904, 0.910]$	$[0.995, 0.996]$	$[0.983, 0.987]$
0.20	$[0.977, 0.982]$	$[0.718, 0.728]$	$[0.947, 0.954]$	$[0.888, 0.898]$
0.05	$[0.813, 0.825]$	$[0.582, 0.593]$	$[0.563, 0.579]$	$[0.616, 0.631]$

Table 4: Performance of the considered models for the unconditional generation task, UPENN data. α corresponds to the proportion of the training dataset used to train the discriminator. Confidence intervals for the discriminator's accuracy are presented at a 0.05 significance level.

As shown in Table 4, the β-VAE model demonstrates poor generation quality, with objects generated by this method being easily distinguishable from real ones, even with small subsets of real objects. Another noteworthy result is the strong performance of the LIKNF algorithm: an accuracy score of 0.7 for $\alpha = 0.25$ indicates significantly good generation quality [14].

The unsatisfactory performance of the other models may be due to the effective memorization of the dataset by the discriminator, which had around $4 \cdot 10^6$ parameters.

Overall, the SCVAE model is capable of generating significantly better instances than the CVAE model, which aligns with the superior performance of the former according to the DCI metrics. This result follows from the fact that the ability of models to generate objects can be hindered by dependencies between factors in the modeled distribution, a limitation that the SCVAE model aims to address.

Generation Experiment №1 The LIKNF model demonstrated the best generation performance. Unlike, for example, variational autoencoders [21], this model allows effective use of latent transformations and smooth data modification. Let us attempt to apply it.

To do so, we fix the time series №1 of a human skeleton 5 with the concepts 'baseball pitch' and 'camera angle'. For this series X_1, the interpretable representation $\tilde{z}_1$ is obtained using LIKNF. Similarly, for the second object 7, with the concepts 'tennis serve' and 'right angle', we extract the representation $\tilde{z}_2$. Next, we replace the coordinates in the original vector $\tilde{z}_1$ corresponding to the 'angle' concept with the components from the vector $\tilde{z}_2$.

$$\tilde{z}_3^i = \begin{cases} \tilde{z}_1^i, \text{ if } i \notin J_2, \text{ where } J_2 \text{ — coordinates representing the angle} \\ \tilde{z}_2^i, \text{ otherwise} \end{cases}$$

The resulting time series $X_3 = D(T^{-1}(\tilde{z}_3))$ of the skeleton, as shown in the plot 8, depicts a human skeleton performing a baseball pitch, with the camera angle from the right. This can be inferred from the swing direction, which moves from left to right.

This example demonstrates the high quality of the representations obtained by the model. Additionally, in the task of interpreting cosine amplitudes, this method allowed the generation of cosines with amplitudes set to $\alpha = 10, 20$. Remarkably, these amplitude values were not part of the training set, meaning the model was able to correctly interpret a more general functional dependency.

Generation Experiment №2 The SCVAE model receives four pairs of concepts to generate the corresponding time series: $(y_1^1, y_1^2) = (\text{Golf swing}, \text{from behind})$, $(y_2^1, y_2^2) = (\text{Golf swing}, \text{from the right})$, $(y_3^1, y_3^2) = (\text{Bowling pitch}, \text{from behind})$, $(y_4^1, y_4^2) = (\text{Bowling pitch}, \text{from the front})$.

Naturally, it is not the text strings that are fed into the model, but rather their numerical codes representing all possible values of these concepts within the training set. As shown in 9, 10, 11, and 12, all four requests generated high-quality series that align with the embedded concepts. This alignment can be verified by observing the direction of the serve. The left part of the human skeleton (relative to the person, not the observer) is highlighted in blue, and the right part in red.

For instance, in the case of the bowling serve from behind — plot 11 — the person swings the ball backward with their right hand and then pushes it forward, away from the camera. This demonstrates that the SCVAE model not only achieves high interpretability but also generates high-quality time series.

The only drawback is the inaccuracy in identifying the camera angle. In plots 11 and 12, the pitch angle might lead an observer to doubt whether the view is truly from behind or from the front, rather than from the right or left. Indeed, there is some angular displacement due to the nature of the training set. The data was collected from recordings of professional matches, where the operators aimed to capture the most visually appealing angles. For this reason, many examples labeled as 'from behind' or 'from the front' were actually recorded with some deviation. Thus, this issue is more related to the characteristics of the dataset itself.

5 Conclusion

This work presents a methodology for obtaining effective interpretable representations of multivariate time series. To achieve this, a review of the formalizations of interpretability was conducted, including accepted definitions and quality metrics.

Since the review did not identify any work that directly addresses our specific task, a formalization of the problem's solution was developed. Several approaches to solving the problem were proposed, based on existing methods from related fields.

A significant improvement to the existing autoencoder model was introduced, leveraging the assumption of decomposing the series into independent trend and seasonality components. A refinement of the CVAE architecture, designated as SCVAE, was proposed, which significantly enhances the interpretability of the model's representations with respect to key metrics.

A comparison of the proposed solutions was carried out—analyzing the capabilities of autoencoders and normalization models on relevant time series examples.

The task of learning interpretable representations was successfully applied to real-world human skeleton movement data. Based on the obtained representations, the possibility of generating new, unique, and plausible time series was demonstrated.

References

[1] X. Wang, H. Chen, S. Tang, Z. Wu, and W. Zhu, "Disentangled representation learning," *arXiv preprint arXiv:2211.11695*, 2022.

[2] P. Esser, R. Rombach, and B. Ommer, "A disentangling invertible interpretation network for explaining latent representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9223–9232, 2020.

[3] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-vae," *arXiv preprint arXiv:1804.03599*, 2018.

[4] E. L. Denton *et al.*, "Unsupervised learning of disentangled representations from video," *Advances in neural information processing systems*, vol. 30, 2017.

[5] J. Bai, W. Wang, Y. Zhou, and C. Xiong, "Representation learning for sequence data with deep autoencoding predictive components," *arXiv preprint arXiv:2010.03135*, 2020.

[6] V. John, L. Mou, H. Bahuleyan, and O. Vechtomova, "Disentangled representation learning for non-parallel text style transfer," *arXiv preprint arXiv:1808.04339*, 2018.

[7] M. Yamada, H. Kim, K. Miyoshi, and H. Yamakawa, "Favae: Sequence disentanglement using information bottleneck principle," *arXiv preprint arXiv:1902.08341*, 2019.

[8] M.-A. Carbonneau, J. Zaidi, J. Boilard, and G. Gagnon, "Measuring disentanglement: A review of metrics," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[9] C. Eastwood and C. K. Williams, "A framework for the quantitative evaluation of disentangled representations," in *International Conference on Learning Representations*, 2018.

[10] C. Eastwood, A. L. Nicolicioiu, J. Von Kügelgen, A. Kekic, F. Träuble, A. Dittadi, and B. Schölkopf, "On the dci framework for evaluating disentangled representations: Extensions and connections to identifiability," in *UAI 2022 Workshop on Causal Representation Learning*, 2022.

[11] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging common assumptions in the unsupervised learning of disentangled representations," in *international conference on machine learning*, pp. 4114–4124, PMLR, 2019.

[12] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," *arXiv preprint arXiv:1704.02971*, 2017.

[13] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, “Temporal convolutional networks for action segmentation and detection,” in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 156–165, 2017.

[14] A. Desai, C. Freeman, Z. Wang, and I. Beaver, “Timevae: A variational auto-encoder for multivariate time series generation,” *arXiv preprint arXiv:2111.08095*, 2021.

[15] I. Kobyzev, S. J. Prince, and M. A. Brubaker, “Normalizing flows: An introduction and review of current methods,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 3964–3979, 2020.

[16] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real nvp,” *arXiv preprint arXiv:1605.08803*, 2016.

[17] R. T. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud, “Isolating sources of disentanglement in variational autoencoders,” *Advances in neural information processing systems*, vol. 31, 2018.

[18] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, “beta-vae: Learning basic visual concepts with a constrained variational framework,” in *International conference on learning representations*, 2017.

[19] J. Yoon, D. Jarrett, and M. Van der Schaar, “Time-series generative adversarial networks,” *Advances in neural information processing systems*, vol. 32, 2019.

[20] W. Zhang, M. Zhu, and K. G. Derpanis, “From actemes to action: A strongly-supervised representation for detailed action understanding,” in *Proceedings of the IEEE international conference on computer vision*, pp. 2248–2255, 2013.

[21] A. Blattmann, T. Milbich, M. Dorkenwald, and B. Ommer, “Behavior-driven synthesis of human dynamics,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12236–12246, 2021.

6 Appendix

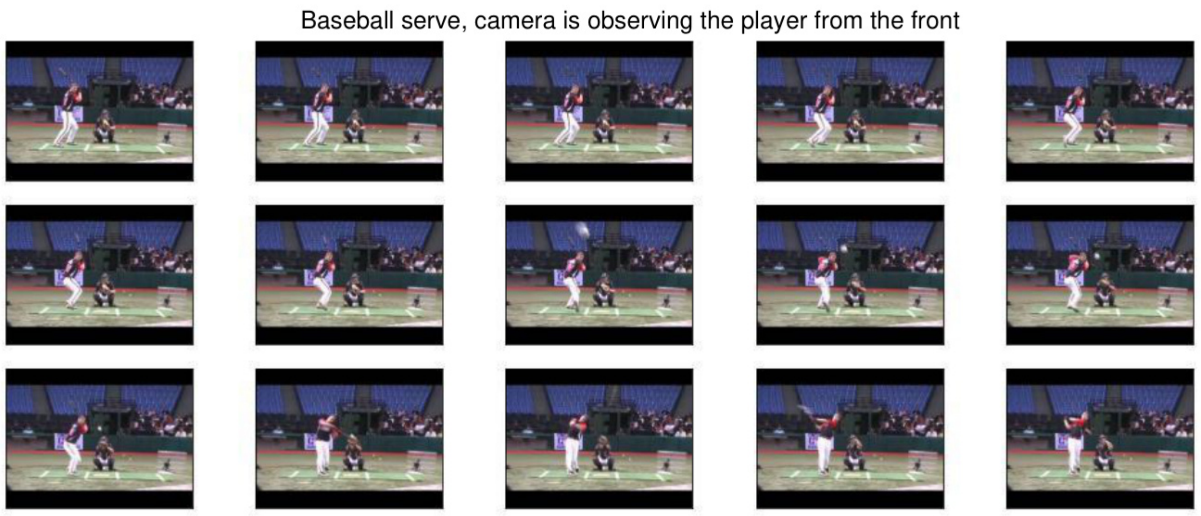
Baseball serve, camera is observing the player from the front

Figure 4: Experiment №1, video of action №1

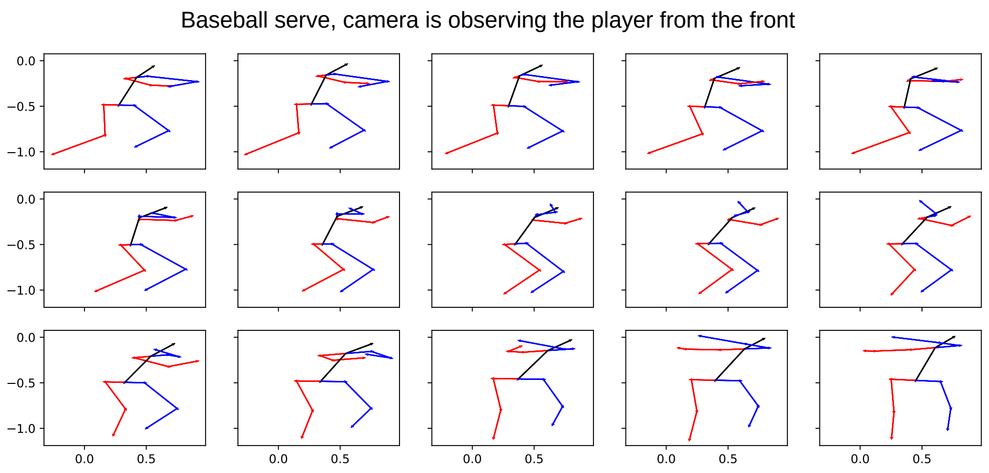

Figure 5: Experiment №1, skeleton of action №1

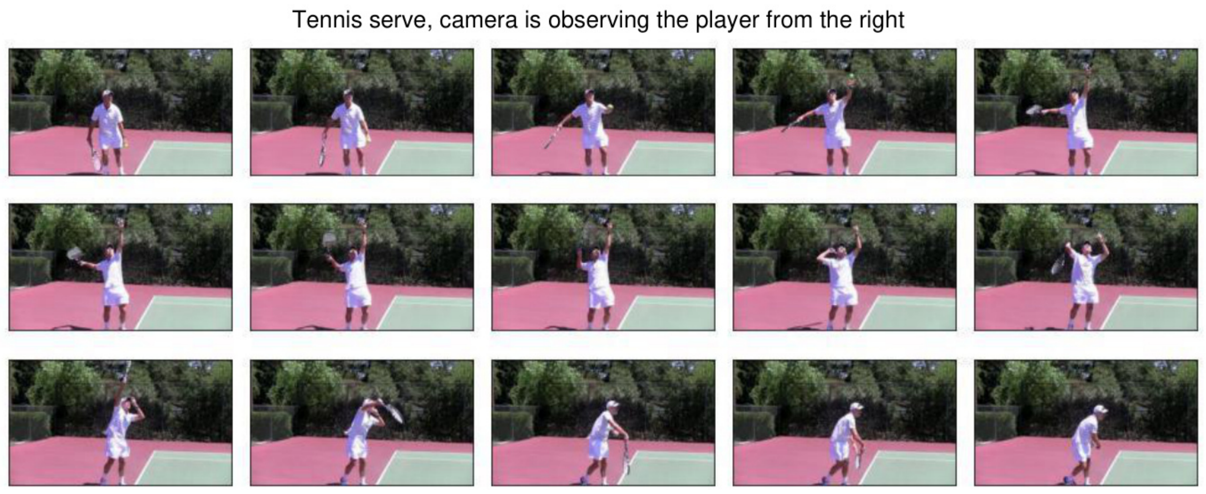

Figure 6: Experiment №1, video of the action №2

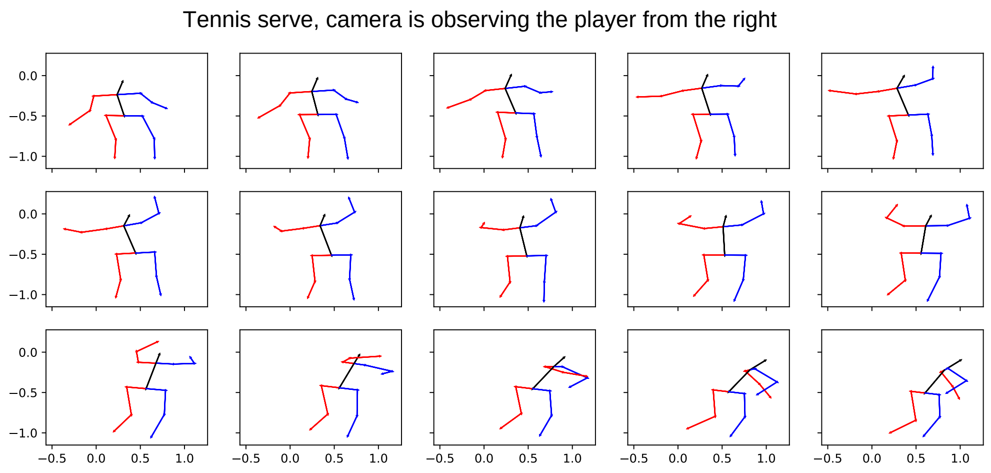

Figure 7: Experiment №1, skeleton of action №2

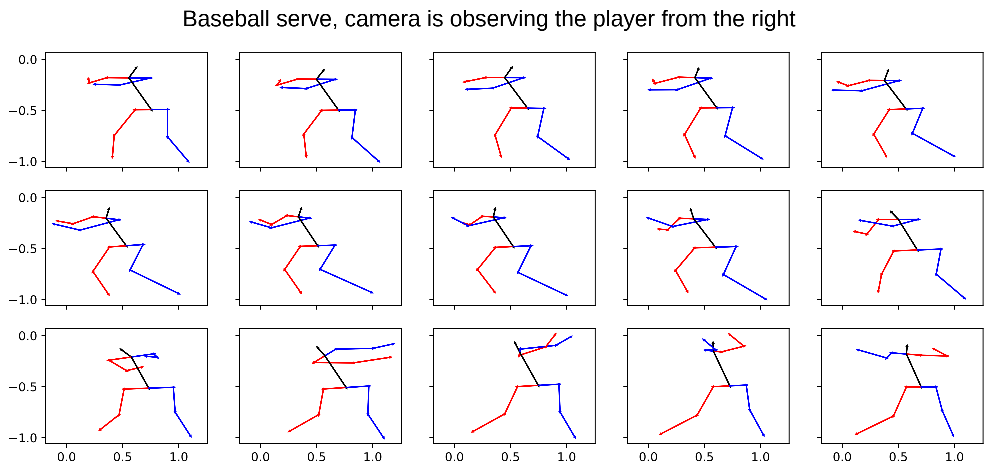

Figure 8: Experiment №1, resulting skeleton (generated by combining two real time series)

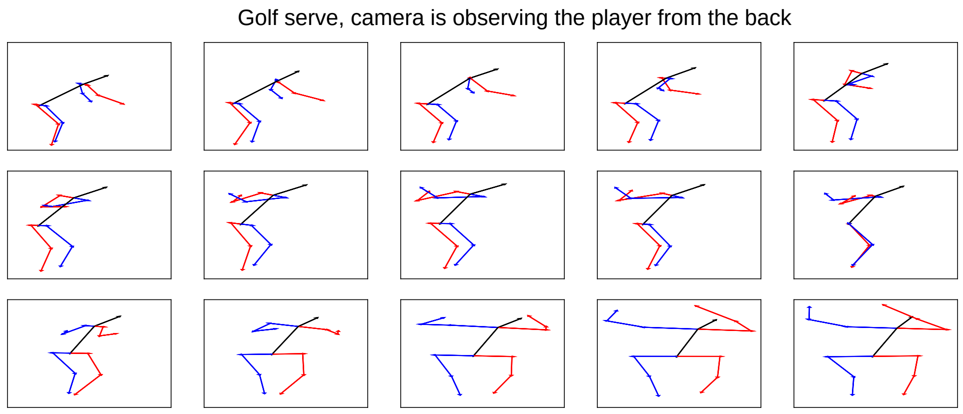

Figure 9: Experiment №2, conditional generation using SCVAE

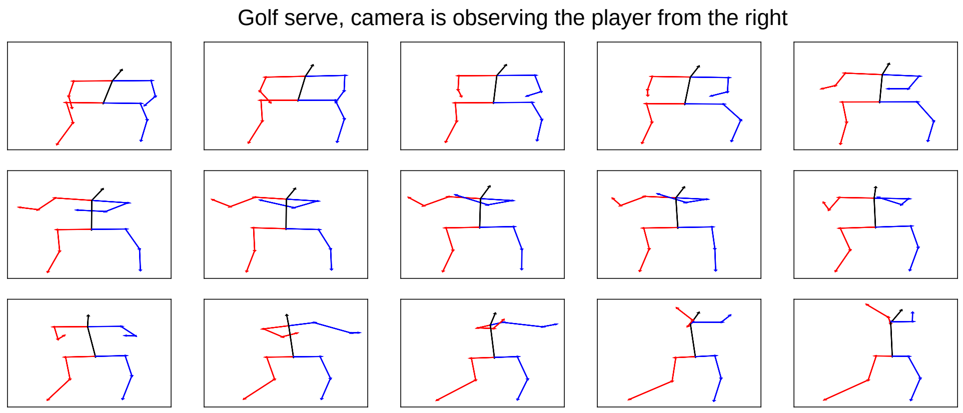

Figure 10: Experiment №2, conditional generation using SCVAE

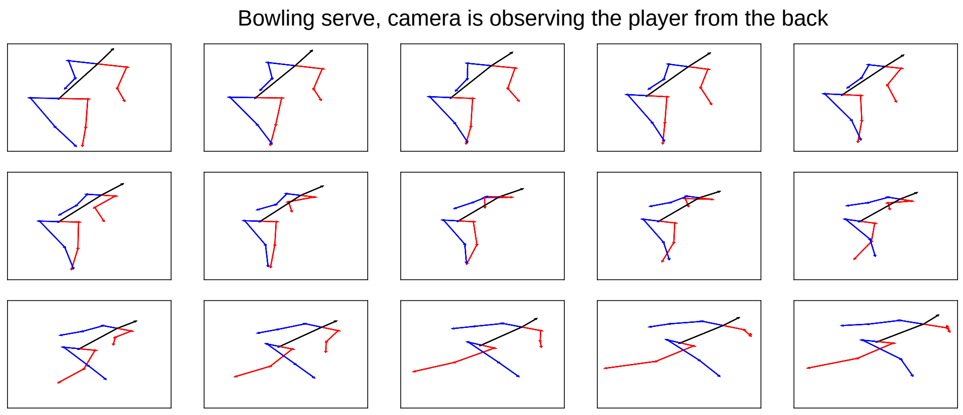

Figure 11: Experiment №2, conditional generation using SCVAE

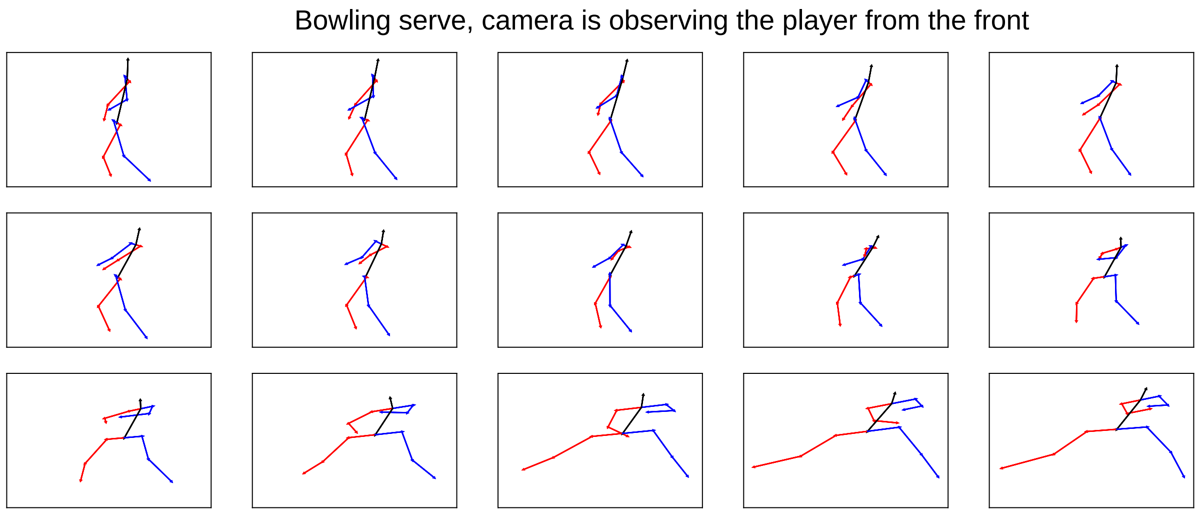

Figure 12: Experiment №2, conditional generation using SCVAE