Data Mining and Knowledge Discovery (2021) 35:2141–2167
https://doi.org/10.1007/s10618-021-00777-1

Selego: robust variate selection for accurate time series forecasting

Manoj Tiwaskar[1] · Yash Garg[1] · Xinsheng Li[1] · K. Selçuk Candan[1] · Maria Luisa Sapino[2]

Received: 19 September 2020 / Accepted: 23 June 2021 / Published online: 28 July 2021

Abstract

Naïve extensions of uni-variate prediction techniques lead to an unwelcome increase in the cost of multi-variate model learning and significant deteriorations in the model performance. In this paper, we first argue that (a) one can learn a more accurate forecasting model by leveraging *temporal alignments* among variates to quantify the importance of the recorded variates with respect to a target variate. We further argue that, (b) for this purpose we need to quantify temporal correlation, not in terms of series similarity, but in terms of temporal alignments of key "events" impacting these series. Finally, we argue that (c) while learning a temporal model using recurrence based techniques (such as RNN and LSTM—even when leveraging attention strategies) is difficult and costly, we can achieve better performance by coupling simpler CNNs with an *adaptive variate selection strategy*. Relying on these arguments, we propose a *Selego* framework (Selego is a word of *latin* origin meaning "*selection*") for variate selection and experimentally evaluate the performance of the proposed approach on various forecasting models, such as LSTM, RNN, and CNN, for different top-X% variates and different forecasting time in the future (lead) on multiple real-world

Responsible editor: Annalisa Appice, Sergio Escalera, Jose A. Gamez, Heike Trautmann

✉ Manoj Tiwaskar
mtiwaska@asu.edu

Yash Garg
ygarg@asu.edu

Xinsheng Li
lxinshen@asu.edu

K. Selçuk Candan
candan@asu.edu

Maria Luisa Sapino
mlsapino@di.unito.it

1 Arizona State University, 699 S Mill Ave, Tempe, Arizona, US

2 University of Turin, Via Pessinetto, 12, 10149 Torino, TO, Italy

datasets. Experiments show that the proposed framework can offer significant (90 − 98%) drops in the number of recorded variates that are needed to train predictive models, while simultaneously boosting accuracy.

Keywords Forecasting · Recurrent and convolutional networks · Variate selection

1 Introduction

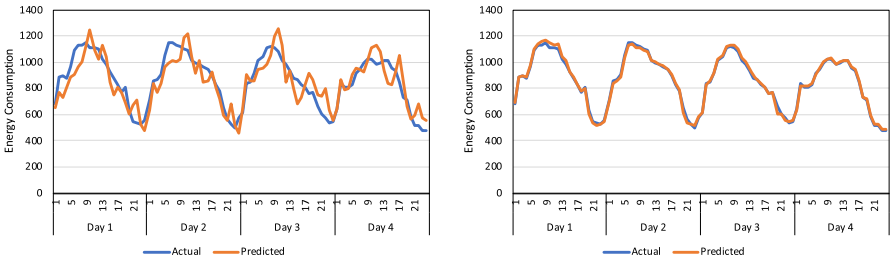
The problem of time series forecasting involves learning a function f that can map the observations from the past $(t_1, t_2, \ldots, t-1)$ to the present (t) or the future. The problem involves a set of recorded variates $\mathbb{X} \in \boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_T$ that drive a set of target variates, $\mathbb{Y}$. In practice, the forecasts are often imprecise due to various reasons and the function f is often learned with an error, ϵ. The ultimate goal of forecasting model learning task, therefore, is to minimize this error. To do so, various statistical and deep models have been proposed. Statistical forecasting models for time series, primarily SVR (Drucker et al. 1997) and ARIMA (Box et al. 2015), have helped reduce the forecasting error in various real-world applications, however, with the increase in the number of variates of the time series, SVR and ARIMA have fallen short in their ability to learn. Neural network-based techniques, such as recurrent neural networks (RNNs) (Rumelhart and Hinton 1986) demonstrated that the short-comings of SVR and ARIMA can be overcome by exploring deep features[1] and their evolution overtime by relying on the $(t-1)^{th}$ state of the network to learn the t^{th} state. Unfortunately, RNNs have proven ineffective on long time series due to catastrophic forgetting. Long-Short Term-Memory networks (LSTM) (Hochreiter and Schmidhuber 1997) have been relatively successful in their ability to remember and model time series, yet they also suffer from model complexity and are susceptible to noise.

1.1 Key observations

Unfortunately, naïve extensions of uni-variate forecasting techniques to multi-variate data lead to both increases in the cost of training these models and, more importantly, deterioration in the model performance, as not all variates may contribute equally to the forecasting performance. In this paper,

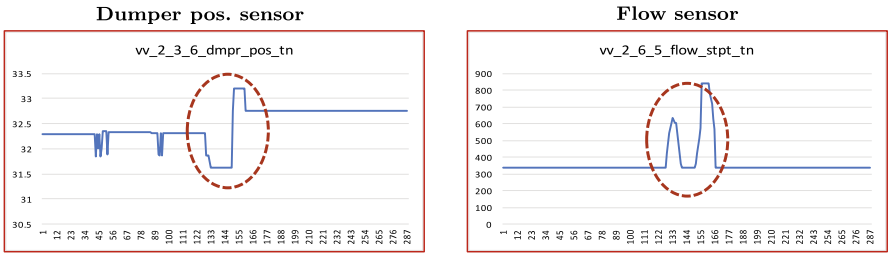
- we first observe that one can learn a more accurate forecasting model by leveraging *temporal correlations* among variates to quantify the importance of the recorded variates with respect to a target variate and using these correlations to help reduce the number of variates needed to train a model (Fig. 1);
- we further observe that traditional time series similarity/distance functions, such as DTW, are fundamentally ill-suited for this purpose as recorded variates relevant for a particular task do not necessarily look similar to each other (Fig. 2); instead,

[1] While the terms "variate" and "feature" are often used interchangeably, in this paper, we make a clear distinction: A "variate" is an input time series describing a time-varying property of the system being observed, whereas a "feature" is a temporal pattern extracted from a given time series and can be used to characterize that series.



(a) CNN with all 513 variates **(b)** CNN with all top-10 variates

Fig. 1 Sample results:CNN based building 1-hour energy consumption prediction results **a** using all 513 variates (100%) versus **b** using only the top-10 (2%) variates (see Sect. 3 for details): both models benefit from variate selection



Two different types of sensors (differently) registering impacts from the same two events

Fig. 2 Dumper and flow sensors that have *aligned but dis-similar* temporal structures; note that in these scenarios, the two temporal structures would be judged to be very different from each other under common distance (such as DTW) or similarity (such as Pearson's correlation) functions

the relationship between two series needs to be quantified based on temporal alignments of the "key events" (or local patterns) identified on these series *irrespective of how these key events themselves look*;

– we finally observe that, while trying to learn a temporal model for a multi-variate time series using recurrence based techniques (such as RNN and LSTM) is difficult and costly (*even when they are leveraging attention strategies*), we can achieve better performance by coupling simpler CNN based models with an *adaptive variate selection strategy* that captures the temporal evolution of the pairwise relationships among the variates.

1.2 Our contribution: the Selego framework

Relying on these observations, in this paper, we propose a *Selego* framework for variate selection: traditional variate selection mechanisms either require a one-to-one alignment of data points or rely heavily on series similarity. Selego ranks variates based on the co-occurrence of key temporal features/events to select variates that have high impact on forecasting of a target variable.

Figure 3 provides an overview of the steps (and substeps) of the Selego framework. As we see here, Selego performs feature extraction from the series and uses these

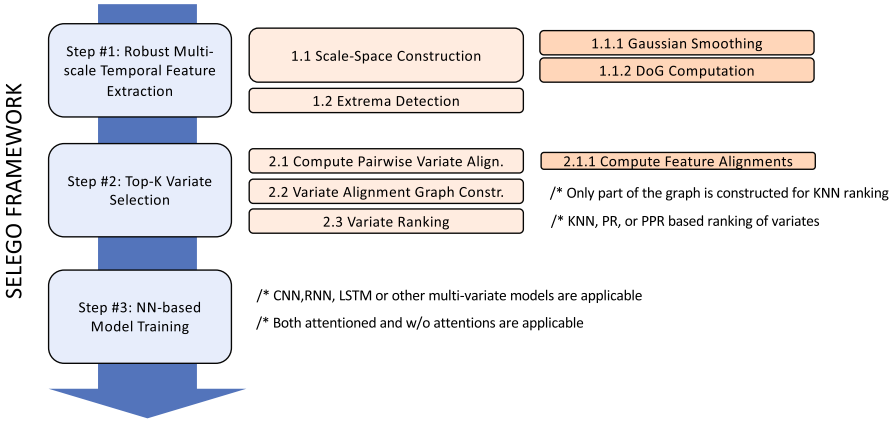


Fig. 3 A schematic overview of the Selego framework

features to support variate selection. In the final step, the predictive model is trained on these reduced subset of variates. The key result of this paper is that the models can be very accurate if they are trained on a small properly selected fraction of the original variates – the accuracies of such models can in fact be better than significantly over models trained on full data set, even when attention based techniques are used to perform in-neural-network feature weighing.

We experimentally validate the key contributions of Selego in Sect. 3 and observe that Selego is able to identify impactful recorded variates for the target variate, and apply to various domains such as, building energy optimization (Bianco et al. 2009), fuel consumption (Goodwin et al. 2004), stock price prediction (Qin et al. 2017), and brain signals(Fernandez-Fraga and Aceves-Fernandez 2018). Experiments show that the proposed framework can offer significant (90−98%) drop in the number of variates that are needed to train predictive models, while also boosting model accuracies.

1.3 Related works

1.3.1 Time series modeling

The increase in diversity and complexity of recorded data led to the need to differentiate relevant aspects of the data from those aspects that are irrelevant. For conventional, tabular data, decomposition based latent feature extraction and dynamic topic modeling techniques (Tucker 1966; Pearson 1901; Blei and Lafferty 2006), have been the long standing go-to method for dimensionality reduction. However, time-agnostic reduction techniques are not generally suited for the task of forecasting.

In this context, deep networks, such as RNNs and LSTMs, have proven particularly successful through the use of combinations of linear and non-linear operations (Clevert et al. 2016). RNNs' success is often limited to the length of the time series as it suffers from the problem of catastrophic forgetting (Bahdanau et al. 2015). Long-short Term Memory (LSTM) NNs were proposed as a solution to the RNNs' shortcoming which

introduced a cell state in addition to the hidden state to remember the past patterns (Hochreiter and Schmidhuber 1997). While LSTM has shown success, such as in speech translation, voice recognition, and video processing (LSTM-CNN), it is still vulnerable to learning noisy models in the presence of large number of input features which can deteriorate the model performance. (He et al. 2016) has shown that after a certain depth and width the performance of the deep network degrades.

1.3.2 Variate selection

Generally speaking, there are two approaches for reducing the dimensionality of a multi-variate time series: (a) (latent) feature selection and (b) variate-selection. A potential solution to feature selection is the use of attention mechanisms employed at hidden layers in the network (Bahdanau et al. 2015; Qin et al. 2017; Wang et al. 2017), aimed to help determine and focus on the subset of important input features during network training. However, the success of attention mechanisms heavily relies on the design of the network architecture. Search for high-performing network architecture in itself is a complex process (Bergstra and Bengio 2012; Zoph and Le 2017; Garg and Candan 2019). Additionally, works such as Garg and Candan (2021a, b) proposed to leverage multi-scale feature withing the network itself, to learn informative deep representation of the time series.

As further discussed in Sect. 2.7.2, the variate selection approach often relies on the creation of a variate-variate alignment graph on which node ranking techniques, such as KNN, PR, and PPR (Tong et al. 2006), are used for ranking the variates. (Roffo et al. 2015), for example, in Inf-FS creates a variate-variate relationship matrix that maximizes the separation between the variates and then uses a random-walk based technique for selecting a diverse set of variates. When creating a variate-variate alignment graph, it may be often necessary to compare sequences against a target sequence based on their underlying patterns. For systems where observations are binary, Multivariate Hawkes Process (MHP) based analysis techniques have been proposed to discover the underlying temporal dependencies (Linderman and Adams 2014; Yuan et al. 2019). These dependencies can be used to select variates when systems can be described as point processes, but are not applicable when the variates do not correspond to point process time series. Euclidean distance and, more generally $L_p - norm$ measures, were among the first used to determine the similarity between two numeric time series. Euclidean distance and others, such as cosine and correlation similarity (Salton and McGill 1983; Shatkay and Zdonik 1996), assume a strict synchrony among time series and are not suitable when two time series can have different speeds or are shifted in time. FRESH (Christ et al. 2018) avoids this problem by extracting global temporal features of a given series, but this fails to account for the specific patterns or co-occurrence of events across series when they are being compared. Dynamic time warping (DTW) is a widely used technique to find an optimal alignment between two given sequences under certain restrictions(Sakoe and Chiba 1978; Chen and Ng 2004; Keogh and Ratanamahatana 2005). Yet, as we see in this paper, since DTW relies on global time series similarity, rather than accounting for the distribution of significant local patterns in the series, it may not perform well in applications where forecasting must rely on dissimilar patterns across series. Symbolic aggregated approximation

(SAX) (Lin et al. 2002), in contrast, creates a symbolic vector representation for each variate, where each symbol corresponds to a local pattern in the input series and where the vector representation accounts for each and every pattern in the series of a given length. (Garg and Candan 2019) demonstrated that **salient** localized features extracted *before* training a NN-based model, which have been largely ignored by many works in the domain, can improve accuracy by highlighting key insights in the data. In this paper, we build upon a similar idea and propose a variate selection mechanism to help improve the forecasting accuracy (reduce error) by intelligently quantifying the inter-variate relationships as a function of **salient** local temporal features extracted from individual variates.

2 Selego: robust variate selection for time series forecasting

In this section, we present the *Selego framework* which leverages salient localized temporal events to select subset of variates from a multi-variate time series.

2.1 Uni- and multi-variate time series

A uni-variate time series (UVTS) is a sequence of ordered pairs of observations and time at which observations were recorded for a given attribute (variate), $\boldsymbol{T} = [(\mathrm{v}_1, \mathrm{t}_1), (\mathrm{v}_2, \mathrm{t}_2), \ldots, (\mathrm{v}_T, \mathrm{t}_T)]$. While in general the temporal separation between two consecutive timestamps can be non-periodic, in this paper we assume that timestamps recorded in a UVTS are periodic in nature. We denote the prefix of $\boldsymbol{T}$ until time t as $\boldsymbol{T}_{[t]}$, whereas we denote the value of $\boldsymbol{T}$ at time t as $\boldsymbol{T}_{(t)}$. A multi-variate time series (MVTS), $\mathbb{T}$, is a set of uni-variate time series, s.t. $\mathbb{T} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, \ldots, \boldsymbol{T}_K\}$ where, K is the number of variates, $\mathbb{T} \in \mathbb{R}^{K \times T}$, and $\boldsymbol{T}_i \in \mathbb{R}^{1 \times T}$.

2.2 Time series forecasting problem

Time series forecasting involves learning a function f that can map historical observations at time $1, 2, \ldots, t$ to the future observations at time $t + l$; we refer to the value of l as the "*lead time*". The problem involves a set of source variates $\mathbb{X} \subseteq \mathbb{T}$ that drive a set of target variates, $\mathbb{Y} \subseteq \mathbb{T}$; i.e. $f : \mathbb{X}_{[t]} \rightarrow \mathbb{Y}_{(t+l)}$.

2.3 NN-based forecasting models

While Selegohas wide applicability, in this paper, we explore its use within the context of neural-network (NN) based forecasting models:

Convolutional neural models (CNNs): Modern neural networks leverage *depth* and *width* of their models to learn complex patterns in the data in the form of deep features (Szegedy et al. 2015). CNN achieves this by repeatedly applying convolution operations (complemented with non-linear activation functions and pooling operations that scale the data) to identify multi-scale patterns of different complexities. In the case

of a time series *prediction* problem, where the goal is to discover a function of the form $f : \mathbb{X}_{[t]} \rightarrow \mathbb{Y}_{(t+l)}$ the model training would be carried out by providing as input prefixes, $\mathbb{X}_{[t]}$, of the input series up to time t and as output the values, $\mathbb{Y}_{(t+l)}$, at time $t + l$ of the target series.

Recurrent neural models (RNNs and LSTMs): CNN lack the ability to memorize temporal patterns over time. To counter this, recurrent networks (RNN) introduced a memorization block in form of a recurrent connection to remember the pattern at time $t - 1$ to help inform the network at time t (Rumelhart and Hinton 1986). More specifically, thanks to repeated convolution and pooling operations, in CNNs, each time instant is informed by the entire temporal length of the series (with weights reflecting temporal neighborhood), whereas in recurrent models the learning process proceeds one timestamp at a time, while leveraging hidden state from the previous timestamp. LSTM (Hochreiter and Schmidhuber 1997) extends RNNs with the ability to forget and has been shown to be more effective than the conventional feed-forward neural networks and recurrent neural networks.

Attention mechanisms: One difficulty with neural network based inference is that a large number of model parameters need to be learned from data. This is especially problematic for sparse and noisy data sets where it is difficult to learn these model parameters for accurate inference. Recent research (Bahdanau et al. 2015; Qin et al. 2017) has shown that *attention mechanisms*, which help the neural network to focus on different aspects of the data at different stages of inference, have the potential to alleviate this difficulty to some degree. The challenge with such attention mechanisms, however, is that the attention model itself needs to be constructed carefully from data to ensure that the model is able to learn to focus on the most relevant patterns, without ignoring patterns critical for inference.

This motivates the need for careful variate selection: as we have seen in Fig. 1 (and as we experimentally validate in Sect. 3), variate selection can boost the predictive model accuracies. Yet, as we also see in Fig. 2, the subset of the variates that help predict a time series do not necessarily look like the target series. Instead, the subset of the variates to be used must present evidence of impact from events that drive the shape of the target series.

2.4 Robust localized temporal patterns/features

Localized patterns (a notable example being SIFT(Lowe 2004)) have been shown to be effective for image retrieval and object detection applications, as well as neural network hyper-parameter search (Garg and Candan 2019). (Candan et al. 2012) has proposed a SIFT-like approach, called SDTW, to detect significant temporal events on time series, and has shown that localized temporal patterns can be used to speed up expensive time series operations, such as DTW computations. Both SIFT (for images) and SDTW (for time series), however, extract and rely on feature descriptors for object comparisons. In this paper, however, we propose a Selego framework which extracts and uses robust features for computing variate alignments purely based on temporal alignment, without considering feature descriptors – in fact, that one should

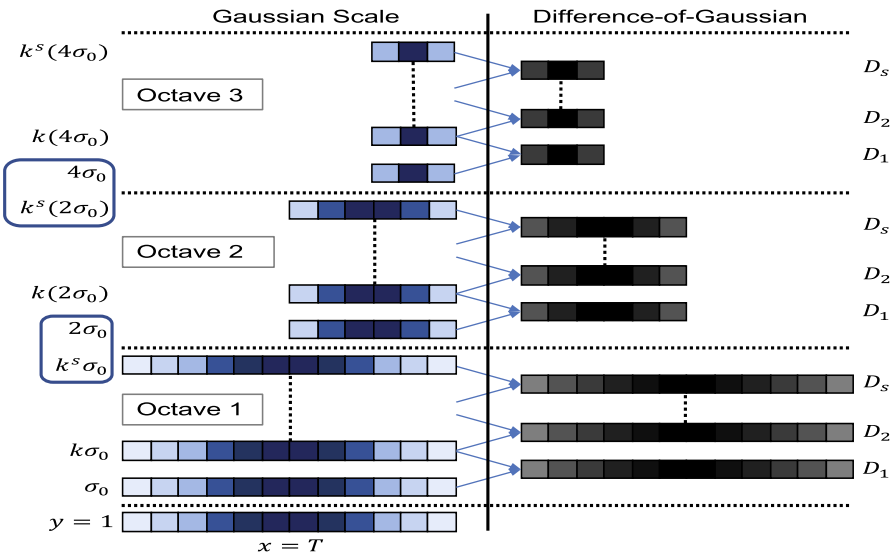


Fig. 4 Generating DoG for a single variate ($y = 1$) taken from a multi-variate time series. The length of the input time series is T

ignore pattern similarities that can be measured by comparing feature descriptors, but considering temporal alignments among key events is one of the major differentiators of Selego from the prior work. Here, we describe the localized feature extraction process (consisting of "scale-space generation" and "extrema detection" steps) used by Selego to identify "robust localized temporal patterns" in the individual variates.

2.4.1 Temporal scale-space generation

Temporal features of interest can be of different lengths. Based on the argument that the interesting events will be maximally different from the overall pattern in their local neighborhoods, Selego searches for those points that have largest variations with respect to both time and scale. Therefore, the first step of the process is to create a scale-space consisting of multiple smoothed versions of a given series – each resulting series is then subtracted from the series in the adjacent temporal scale to obtain what we refer to as the difference-of-Gaussian (DoG) series. Intuitively, the smoothing process can be seen as generating a multi-scale representation of the given series and thus the differences between smoothed versions of a given series correspond to differences between the same series at different scales. Let $\boldsymbol{T}_v$ represent a univariate time series, s.t. $\boldsymbol{T}_v \in \mathbb{T}[v, *]$, and $\boldsymbol{T}_v^{(t,\sigma)}$ represents the smoothed version of $\boldsymbol{T}_v$ through convolution with the Gaussian function along the temporal dimension: $\mathbf{G}(\mathrm{t}, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-t^2}{2\sigma^2}}$ such that $\boldsymbol{T}_v^{(\mathrm{t},\sigma)} = \mathbf{G}(\mathrm{t}, \sigma) * \boldsymbol{T}_v$. Gaussian smoothing is used to create a multi-scale representation of a given series, $\boldsymbol{T}_v$: As shown in Fig. 4, the scale space is created by first applying an initial smoothing with parameter σ_0 and then adding L layers of smoothing, where the s^{th} smoothing layer is Gaussian smoothed

Fig. 5 A candidate feature point, $\mathcal{F}$, (black) and its neighbors in adjacent scales "$s+1$" (red) and "$s-1$" (yellow) and in time "$t-1$" (blue) and "$t+1$" (green) (Color figure online)

at level $\kappa^s \times \sigma_0$, where κ is a constant multiplicative factor). Intuitively, repeated application of Gaussian smoothing at multiple-layers enables details to disappear. Consequently, while extrema of DoG correspond to fine grain details in lower layers, they correspond to large patterns in higher layers. As shown in the figure, for efficiency purposes, we organize the scales into octaves with increasingly shorter lengths by sub-sampling the series when the amount of smoothing applied on the series is such that the series length can be halved without loss of details. Once the scale space is constructed, the search for features is performed by comparing the immediate neighbors. As shown in Fig. 4, to support this search, we simultaneously create a Difference-of Gaussian (DoG) representation, $\mathbf{D}_v^{(\mathrm{t},\sigma)} = \boldsymbol{T}_v^{(\mathrm{t},\kappa\sigma)} - \boldsymbol{T}_v^{(\mathrm{t},\sigma)}$, of the input series, $\boldsymbol{T}_v$. The overall process has computational complexity of $O(\mathbf{LWT})$, where $\mathbf{L} = L$ is the number of scales created, $\mathbf{W}$ is the length of the Gaussian window used for time series smoothing, and $\mathbf{T} = T$, is the length of the time series.

2.4.2 Extrema detection

In this step, we search for points of interest, $\langle t, s\rangle$ across multiple scales of the given time series, v, by searching over multiple scales and locations of the given series (here s denotes the corresponding scale), with overall computational complexity of $O(\mathbf{LT})$. The search of local extrema (features) is performed by comparing the immediate neighbors (see Fig. 5) along both time and scale in the Difference-of Gaussian (DoG) representation, $\mathbf{D}_v^{(\mathrm{t},\sigma)}$, of the input series created in the previous step. This enables the algorithm to prune features that are similar to their local neighborhood both in scale and time and, thus, highlight regions of the time series that are distinct from their local neighborhood. More specifically, a pair $\langle t, s\rangle$ on variate v, is an extremum if it is maximum or minimum across its *eight* neighbors – *three* per each neighboring scales ($s-1$ and $s+1$) and *two* temporal neighbors of t (i.e. $t-1$ and $t+1$):

$$extrema \begin{pmatrix} \boldsymbol{D}_v^{t-1,\kappa^{s+1}\sigma} & \boldsymbol{D}_v^{t,\kappa^{s+1}\sigma} & \boldsymbol{D}_v^{t+1,\kappa^{s+1}\sigma} \\ \boldsymbol{D}_v^{t-1,\kappa^{s}\sigma} & \boldsymbol{D}_v^{t,\kappa^{s}\sigma} & \boldsymbol{D}_v^{t+1,\kappa^{s}\sigma} \\ \boldsymbol{D}_v^{t-1,\kappa^{s-1}\sigma} & \boldsymbol{D}_v^{t,\kappa^{s-1}\sigma} & \boldsymbol{D}_v^{t+1,\kappa^{s-1}\sigma} \end{pmatrix}. \tag{1}$$

In other words, $\langle t, s\rangle$ is designated as an extremum if it is greater than $\Theta\%$ of the maximum of its *8* scale-time neighbors in DoG ($\boldsymbol{D}$).

Note that each identified feature has an associated temporal feature scope, defined by the temporal scale (s) in which it is located. Since under Gaussian smoothing *three* standard deviation would cover $\sim 99.73\%$ of the original temporal points that have

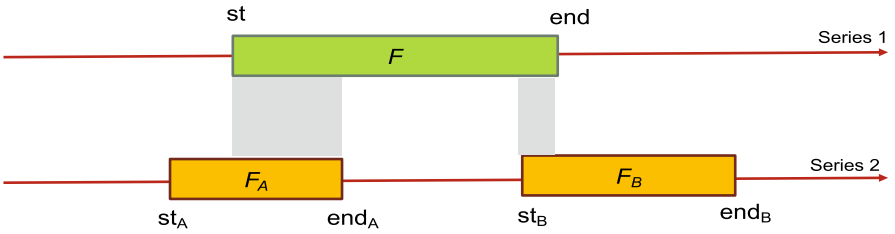


Fig. 6 The temporal alignment between two features depends on degree of overlap between their scope: in this example, the feature $\mathcal{F}$ on Series #1 (highlighted in green) is better aligned with feature $\mathcal{F}_A$ than with feature $\mathcal{F}_B$ on Series #2 (Color figure online)

contributed to the feature, the radius of the feature is set to 3σ: each key temporal feature, $\mathcal{F}$, can be written as triple, $\langle v, t, s\rangle$ and would cover a time interval on variate v, s.t. $t\_scope(\langle v, t, s\rangle) = [t - 3\kappa^s\sigma_0, t + 3\kappa^s\sigma_0)]$.

2.5 Measuring feature alignment

Once these key features are extracted, the Selego framework relies on the co-occurrence of salient temporal features to quantify the degree of temporal alignment among variates (Fig. 6). Therefore, we first propose a feature alignment measure, *interval alignment*, that measures the temporal overlap (feature co-occurrence) between the features on different variates in the same multi-variate time series. Let $\mathcal{F}_1\langle v_1, t_1, s_1\rangle$ and $\mathcal{F}_2\langle v_2, t_2, s_2\rangle$, be two features; the *interval alignment* (IA) between two features is defined as follows:

$$\texttt{IA}(\mathcal{F}_1, \mathcal{F}_2) = \begin{cases} overlap(\mathcal{F}_1, \mathcal{F}_2), & overlap(\mathcal{F}_1, \mathcal{F}_2) \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $overlap(\mathcal{F}_1, \mathcal{F}_2) = min(t_{end,1}, t_{end,2}) - max(t_{start,1}, t_{start,2})$. Here, $t_{start,i}$ and $t_{end,i}$ represents the start and end time of the feature, $\mathcal{F}_i$, respectively.

It is important to note that the magnitude of IA is likely to be larger for feature pairs that are identified in higher scales, since the $overlap()$ function measures the absolute (not relative) amount of overlap between two feature intervals and since the features at larger octaves/scales have larger scopes. This choice reflects the fact that a large overlap between two features with large scopes is a clearer evidence of temporal alignment between the corresponding variates. Note that this overlap based feature alignment measure relies on a weak assumption that there will be some degree of temporal co-location among events across series that are related or impact each other. As we see in Sect. 3, this assumption led to high model accuracy in the data sets we have considered. However, in applications where related events can have significant temporal lag, it is possible to replace this definition with an alternative that measure alignment of two series based on their their temporal distance, instead of temporal overlap. This flexibility in the definition if alignment is one of the advantages of the Selego framework.

2.6 Measuring variate alignment

Given two variates, $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$, of a multi-variate time series, $\mathbb{T}$, and the features sets, $\mathbb{F}_i = \{\mathcal{F}_{i,1}, \mathcal{F}_{i,2}, \ldots, \mathcal{F}_{i,|\mathbb{F}_i|}\}$ and $\mathbb{F}_j = \{\mathcal{F}_{j,1}, \mathcal{F}_{j,2}, \ldots, \mathcal{F}_{j,|\mathbb{F}_j|}\}$ respectively, we define the temporal alignment of variate $\boldsymbol{T}_i$ against variate $\boldsymbol{T}_j$ as follows:

$$\texttt{TA}(\boldsymbol{T}_i|\boldsymbol{T}_j) = \frac{\sum_{m=1}^{|\mathbb{F}_i|} max_{n\in(1,\ldots,|\mathbb{F}_j|)}\, \texttt{IA}(\mathcal{F}_{i,m}, \mathcal{F}_{j,n})}{|\mathbb{F}_i|}. \tag{3}$$

Given this, we then define the variate alignment (VA) between the two variates as $\texttt{VA}(\boldsymbol{T}_i, \boldsymbol{T}_j) = \texttt{TA}(\boldsymbol{T}_i|\boldsymbol{T}_j) + \texttt{TA}(\boldsymbol{T}_j|\boldsymbol{T}_i)$, with computational complexity of $O(|\mathbb{F}_i|.|\mathbb{F}_j|)$ – note that this is a worst case complexity and in practice only pairs of features that temporally overlap need the be considered, making the complexity closer to $O(|\mathbb{F}_i| + |\mathbb{F}_j|)$ in practice. It is also important to note that, while this measure seeks maximal temporal alignment between features of the variates $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$, this does not imply that the time series will actually be similar – this is because, the variate alignment function, VA, and its various components do not consider how the individual features/patterns look; instead, they focus only on whether the features/patterns are temporally aligned or not.

2.7 Top-*k* variate selection

Let $\mathbb{T}$ be, as described in Sect. 2.1, a multi-variate time series, s.t. $\mathbb{T} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, \ldots, \boldsymbol{T}_K\}$, where K is the number of variates. As formulated in Sect. 2.2, let the task be to learn a function $f : \mathbb{X}_{[t]} \rightarrow \mathbb{Y}_{(t+l)}$ to forecast, with lead time l, a set of target variates, $\mathbb{Y} \subseteq \mathbb{T}$ using a set of source variates $\mathbb{X} \subseteq \mathbb{T}$. To help select the top-k variates in $\mathbb{X}$ to be used for training a predictive model, Selego considers (but not necessarily fully creates) a lead-l *variate alignment graph*, $G_{\mathbb{X},\mathbb{Y},l}(V, E, w_l)$.

2.7.1 Lead-*l* variate alignment graph

As formalized in Sect. 2.2, our goal is to design a model for predicting event with l units of lead time. To achieve this, we construct a lead-l variate alignment graph, $G_{\mathbb{X},\mathbb{Y},l}(V, E, w_l)$, which is a weighted graph where

- $V = \mathbb{X} \cup \mathbb{Y}$ and $E = E_{XX} \cup E_{YY} \cup E_{XY}$, where $E_{XX} = \{\langle \boldsymbol{T}_n, \boldsymbol{T}_m\rangle \mid \boldsymbol{T}_n, \boldsymbol{T}_m \in \mathbb{X}\}$, $E_{YY} = \{\langle \boldsymbol{T}_n, \boldsymbol{T}_m\rangle \mid \boldsymbol{T}_n, \boldsymbol{T}_m \in \mathbb{Y}\}$, $E_{XY} = \{\langle \boldsymbol{T}_n, \boldsymbol{T}_m\rangle \mid \boldsymbol{T}_n \in \mathbb{X}, \boldsymbol{T}_m \in \mathbb{Y}\}$,
- for all $\langle \boldsymbol{T}_n, \boldsymbol{T}_m\rangle \in E_{XX} \cup E_{YY}$, the edge weight is computed as $w_l(\langle \boldsymbol{T}_n, \boldsymbol{T}_m\rangle) = \texttt{VA}(\boldsymbol{T}_n, \boldsymbol{T}_m)$, and
- for all $\langle \boldsymbol{T}_n, \boldsymbol{T}_m\rangle \in E_{XY}$, the edge weight is computed as $w_l(\langle \boldsymbol{T}_n, \boldsymbol{T}_m\rangle) = \texttt{VA}(\boldsymbol{T}_n^{\{l\}}, \boldsymbol{T}_m^{\langle l\rangle})$; here $\boldsymbol{T}^{\langle l\rangle} = [(\mathrm{v}_{l+1}, \mathrm{t}_1), (\mathrm{v}_{l+2}, \mathrm{t}_2), \ldots, (\mathrm{v}_T, \mathrm{t}_{T-l})]$ is the l-step back-shifted version of $\boldsymbol{T}$, whereas $\boldsymbol{T}^{\{l\}} = \big[(\mathrm{v}_1, \mathrm{t}_1), (\mathrm{v}_2, \mathrm{t}_2), \ldots, (\mathrm{v}_{T-l}, \mathrm{t}_{T-l})\big]$ is the l-step shortened version of $\boldsymbol{T}$.

Above, E_{XX} are the edges among the source variates, E_{YY} are the edges among the target variates, and E_{XY} are the edges from the source to target variates. Intuitively, the

weight $w_l(\langle \boldsymbol{T}_n, \boldsymbol{T}_m \rangle)$ for an edge crossing the source and target variates represents the temporal alignment among series where the target variates are shifted l steps backwards (this enables us to compare source and target series with l-step lag). Note that, the graph $G_{\mathbb{X},\mathbb{Y},l}(V, E, w_l)$ represents lead-l alignments between $O(\|X \cup Y\|^2)$ variate pairs. As we see below, however, in practice, we do not need to enumerate the entire variate alignment graph.

2.7.2 Variate selection

Given the lead-l variate alignment graph, $G_{\mathbb{X},\mathbb{Y},l}(V, E, w_l)$, the k source variates, $\hat{\mathbb{X}}$, to be used for training can be selected using various node selection strategies, including random walk based techniques, such as Personalized PageRank (Tong et al. 2006), a commonly used node ranking scheme which ranks the nodes in a graph with respect to a given seed node set (the target nodes $\mathbb{Y}$ in this case) through a random walk that emphasizes those nodes that are quickly reachable from the seed nodes over a large number of paths. (Roffo et al. 2015) proposed an inf-FS method to generate a variate-variate relationship matrix that maximizes the separation between the variates, followed by a random-walk over the variates to rank each variate based on their overall importance to the multi-variate time series. In order to prevent the specific graph centrality measure from clouding the results and to assess the general applicability of the variate selection approach to multi-variate time series forecasting, in the experiments reported in Sect. 3, as default, we use a much simpler (and cheaper) k-nearest neighbor strategy, where we only consider the edge set, E_{XY}, between source/target pairs and rank the target variates in $\mathbb{X}$ according to their average edge weights towards the source variates in $\mathbb{Y}$ to select the top-k target variates, $\hat{\mathbb{X}}$. Experiments show that this alone is a highly effective strategy in variate selection.

2.8 Lead-l model training

Once the top-k subset, $\hat{\mathbb{X}}$, of source variates are selected, to learn a function $f : \mathbb{X}_{[t]} \rightarrow \mathbb{Y}_{(t+l)}$ which forecasts with lead time l, we train a model (CNN, RNN, or LSTM) of the target variables, $\mathbb{Y}$, using only the selected source variables, $\hat{\mathbb{X}}$. More specifically, the training process seeks the function $f_l : \hat{\mathbb{X}}^{\{l\}}_{[t]} \rightarrow \mathbb{Y}^{\langle l \rangle}_{(t)}$, where $\mathbb{Y}^{\langle l \rangle}$ is the l-step back-shifted version of the target variates[2] in $\mathbb{Y}$ and $\hat{\mathbb{X}}^{\{l\}}$ is the l-step shortened version of the k source variates in $\hat{\mathbb{X}}$. The specific training processes used in our experimental evaluations are detailed in Sect. 3.3.

3 Experiments

In this section, we experimentally evaluate the validity of the key arguments presented in Sect. 1.2 and assess the effectiveness of the Selego framework against alternative

[2] Without loss of generality, in the experiments reported in Sect. 3, we consider target sets each with a single variate (i.e., $|\mathbb{Y}| = 1$).

Table 1 Overview of multi-variate time series datasets

Datasets	NASDAQ	EEG (O1, O2)	FC	BE
# of Variates	81	80 (=5× 16)	157	390
# of Timestamps	210	19	212	24
Lead (l)	1, 5, 10, 50	1, 5, 10	1, 5, 10, 50	1, 5, 10
Top % Variates	10, 50, 90, 100	10, 50, 90, 100	5, 10, 50, 90, 100	2, 10, 50, 90, 100
Time Unit	1 minute	5 seconds	1 minute	1 hour

variate selection strategies, for various data sets and forecasting models[3]. We implemented Selego in Python environment (3.5.2) using Keras Deep Learning Library (2.2.4-tf) with TensorFlow Backend (1.14.0) (Abadi et al. 2016). All models were trained on an Intel Xeon E5-2670 2.3 GHz Quad-Core Processor with 32GB RAM equipped with Nvidia Tesla P100 GPU with 16 GiB GDDR5 RAM with CUDA-10.0 and cuDNN v7.6.4[4]. The variate selection processes were executed on MATLAB R2018b U5 (9.5.0.1178774) on MacOS 10.14.6 with 2.9Hz Intel Core i5 equipped with NVIDIA GeForce GT 750M 1GB graphics card.

3.1 Datasets

As we summarize in Table 1, to evaluate the application of the proposed Selego framework, we consider four real-world datasets from a variety of domains:

NASDAQ (S&P and APPL): (Qin et al. 2017), comprises of prominent NASDAQ stocks; stock prices and index are recorded for 105 days from July 26, 2016 to Dec 22, 2016. We explore two targets for this dataset, the S&P Index and APPL.

EEG-BCI: (Fernandez-Fraga and Aceves-Fernandez 2018) records brain signals, using the BCI System, for 30 subjects while they are performing visual activities.

There are 16 EEG sensors placed on the subjects. The time domain signal from each sensor is transformed into 5 frequency bands, leading into a total of 80 variates. Among these, we consider the observed responses from the left and right *occipital lobes (O1,O2)* of the subjects as to 10 ($= 2 \times 5$) target variates.

Fuel Consumption (FC): This is a proprietary dataset, comprising of $\sim$ 500 variates for various flights averaging for 3.5 hours from takeoff to landing. Here, we forecast the fuel consumption for the flights using $\sim$ 157 (non-categorical) variates that are not directly measuring aspects of fuel consumption.

Building Energy (BE): This is a proprietary dataset with 512 variates recording various indoor (e.g. heating, cooling, airflow) and outdoor sensor readings for 30 consecutive days at a resolutions of 1 hour. For this data set, we consider 390 non-categorical variates and select as the target variable the overall power consumption.

[3] Our source codes and the public data sets used in these experiments are available .

[4] Results presented in this paper were obtained using NSF testbed: "*Chameleon: A Large-Scale Reconfigurable Experimental Environment for Cloud Research*"

3.2 Alternative feature and variate selection methods

In addition to Selego, we consider the following feature selection strategies:

DTW (Berndt and Clifford 1994) is a widely-used elastic distance measure which accounts for differences in speed of patterns across two time series. Top-k variates are selected by inversely sorting the variates based on their DTW distances to the target variate. Note that, unlike Selego(which emphasizes temporal alignments of key events), DTW gives precedence to variates that have similar shapes.
PCA, compares variates in a latent space: we first create a variate-variate co-variance matrix, C, which is then decomposed into $C = USU^T$ using PCA (Pearson 1901) based decomposition. Here, U is a factor matrix, where the rows correspond to source and target variates and columns correspond to latent basis vectors. The top k variates are selected by computing the dot product of the rows of U corresponding to source variates with the row corresponding to the target variate.
Inf-FS (Roffo et al. 2015) is a feature selection strategy which ranks input variates based on a random-walk on their transition graph representing the inverse (Spearman) correlation between the variates.
FRESH (Christ et al. 2018) is a state of the art global temporal feature extraction technique, commonly used for regression tasks. The authors provide a tsfresh package, which we use to extract global features from each variate. We, then rank the input variates with respect to the target variate by measuring the closeness of variate's features with the target's features using Euclidean distance[5].
SAX (Lin et al. 2002)

We also considered the symbolic aggregated approximation (SAX) features proposed in (Lin et al. 2002).

We create a symbolic vector representation for each variate, we then rank the input variates with respect to target variate by computing the closeness of two SAX features in MINDIST (Lin et al. 2002)

We considered different configurations of SAX (window size {3, 16, 24, 32} and dictionary size {7, 10} to create a symbolic vector representation for each variate.

The reported results are the best pair of window and dictionary sizes for each dataset based on the highest average DCG score amongst all the possible pairs.

Given the above feature selection strategies, we consider three variate selection methods: **KNN** (default) where we rank all variates based on their temporal alignments to target series; **PPR** where we also take into account the topology of the resulting variate alignment graph through personalized pagerank (with 85% emphasis given on graph topology and 15% emphasis on the target series) (Tong et al. 2006); and **PR** where the target is ignored and the variates are selected solely based on variate centrality as computed by pagerank. Note that the Inf-FS method (Roffo et al. 2015), by design, relies on a PR based for variate selection.

[5] Since the components of the FRESH feature vector are of potentially of very different scales, each component has been re-scaled to between 0 and 1 to prevent large valued components from having undue bias in the final ranking.

3.3 Neural network based models

As described in Sect. 2.3, variate selection can be used within the context of various neural-network (NN) based models. In this section, we consider CNN, RNN, and LSTM-based models (both attentioned and without attention) RNN and LSTM relies on recurrence to model temporal patterns. CNN does not rely on recurrence, but aims to capture multi-scale patterns by relying multiple layers of convolution and polling operations. Consequently, RNN and LSTM are used as (relatively complex) models that are time aware; whereas (1D) CNN is used as a (relatively simple) non-recurrent model. The hyper-parameters of the NN models have been empirically selected as reported below:

Recurrent neural models: We consider two widely-used recurrent models, LSTM (Hochreiter and Schmidhuber 1997) and RNN (Rumelhart and Hinton 1986). As the default model architecture, we consider a model with 1 hidden layer with 200 computational units (LSTM, RNN) – the hidden activations were "*tanh*" and "*hard sigmoid*" for RNN and LSTM respectively. '*Linear*'" activation was used as output activation for all models. Models were trained for 200 epochs with batch size of 1, using mean absolute error ("*mae*") and "RMSProp" as model loss and optimizer[6].

Convolutional neural models: In addition to recurrent models, we also consider convolutional kernels as simple (non-sequential) model. In particular, CNN sees the entire temporal length at any given instance opposed to recurrent models where only one time instance (in sequence) is available at a time. To ensure fair comparison against LSTM and RNN experiments, we considered a CNN model with 1 hidden layer with 200 computational units, with linear activation function. The model was trained for 200 epochs with batch size of 1, using "*mae*" and "RMSProp" as model loss and optimizer.

Attentioned models (Bahdanau et al. 2015): We also considered attentioned versions of the CNN, RNN, and LSTM models. In particular, we applied the (Bahdanau et al. 2015) encoder-decoder based attention module, which used encoder to map an input sequence $(\boldsymbol{T}^1 \dots \boldsymbol{T}^T)$ to a sequence of continuous representation $\boldsymbol{Z} = (\boldsymbol{Z}^1 \dots \boldsymbol{Z}^T)$ and decoder generates an output sequence $\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{Y}}^1 \dots \hat{\boldsymbol{Y}}^T)$ one element at a time, i.e. applying *fine-grain* attention.

3.4 Data normalization

In the experiments, we considered three alternative normalization strategies:

- **No normalization:** In this case, we use the input time series as is.
- **Min-Max normalization:** In this case, each uni-variate time series is independently scaled such that the minimum value is equal to 0.0 and the maximum value is equal to 100.0.
- **Z-normalization:** In this case, we use the well-known Z-normalization strategy (Mueen and Keogh 2016) to normalize each uni-variate time series.

[6] We report the best model performance across 200 epochs.

Note that the variate selection and the NN-based model training steps *do not necessarily need to rely on the same normalization strategy*.

3.5 Experiment parameters

To assess the variate selection strategies in different settings, we explored various top-X% of variates selections and different temporal "*lead*" conditions. We varied the ratio of the selected source variates from 2% to 100% of variates in the data set (excluding the target variable) to demonstrate how Selego performs with different numbers of variates – note that $k = \lceil num\_variates \times (X/100) \rceil$. We trained forecasting models for varying leads from $l = 1$ to $l = 50$. To extract the key patterns using Selego, we set σ_0 to 0.5, the maximum number of scales to 9, and κ to $\sqrt[3]{2}$ – this leads local features of sizes $3 = (6 \times 0.5)$ time units to $24 = (6 \times ((\sqrt[3]{2})^9 \times 0.5))$ time units for computing variate alignments. These numbers have been selected to make sure that the lengths of the extracted features are compatible with the lengths of all data sets considered. We use 70% of the available samples for training, 10% for validation, and 20% for testing.

3.6 Evaluation metrics

We measure accuracy using *mean absolute error* (MAE): $MAE(\boldsymbol{Y}_{true}, \boldsymbol{Y}_{pred}) = \frac{1}{T} \sum_{t=1}^{T} \left| \boldsymbol{Y}_{true}[t] - \boldsymbol{Y}_{pred}[t] \right|$; here, $\boldsymbol{Y}_{true}$ and $\boldsymbol{Y}_{pred}$ are the true and predicted values of the target variable and T is the length of the time series. Note that, if the data is normalized, we bring the data back to the original value range before computing the MAE. We use the resulting MAE values in two different ways:

- For comparing the accuracy performance for a given approach under various problem settings, we compute and report the *average MAE* for all testing instances for each configuration.
- For comparing the alternative variate selection strategies we compute and report $DCG_{avg,S}(D) = \frac{1}{l} \times \sum_l DCG_S(D, l)$, where S is a variate selection strategy, D is a data set, l is the forecasting lead, and $DCG_S(D, l)$ is defined as $DCG_S(D, l) = \sum_{i=1..|S|} \frac{rank\_count(D,S,l,i)}{log_2(i+1)}$. Here $rank\_count(D, S, l, i)$ is the number of problem configurations (model, number of variates etc.) in which the variate selection strategy provides the i^{th} best (i.e., lowest) MAE among all available strategies. Intuitively, the higher the $DCG_{avg,S}(D)$ value is, the better performing is the variate selection strategy S for data set D.

3.7 Results and discussions

As we discussed in Sect. 3.4, the variate selection and the NN-based model training steps do not necessarily need to rely on the same normalization strategy. Therefore, before investigating the impact of variate selection strategies on forecasting accuracies, in Table 2, we first consider the impact of data normalization on model accuracy when no variate selection is applied. As we see in the table, the Z-normalization strategy leads

Table 2 Average MAE scores under different normalization strategies (w/o variate selection, lead-1 prediction): Z-normalization leads to the best overall accuracy across NN-models and data set (note that the normalization has been applied during both feature extraction for variate alignment and model training)

		Prediction Error (MAE) as a Function of the Normalization Strategy (w/o Variate Selection)											
		LSTM			RNN			CNN			Overall Average MAE		
		No-Norm	Min-Max	Z-Norm	No-Norm	Min-Max	Z-Norm	No-Norm	Min-Max	Z-Norm	No-Norm	Min-Max	Z-Norm
100% Variates Lead-1	Building	659.24	45.20	57.89	211.96	48.79	55.60	44.60	53.51	79.97	305.3	49.2	64.5
	Aviage	1329.09	157.50	221.32	1189.97	705.24	276.81	315.17	323.36	232.46	944.7	395.4	243.5
	Nasdaq	4312.62	8.70	14.20	2039.96	17.51	15.11	18.72	42.75	8.92	2123.8	23.0	12.7
	Apple	6.68	1.56	1.51	6.45	1.35	1.60	2.13	1.69	0.77	5.1	1.5	1.3
	EEG-O1	1516.36	17.17	14.59	911.36	15.84	15.42	23.67	11.33	2.54	817.1	14.8	10.9
	EEG-O2	1525.88	16.89	15.65	912.30	12.70	16.74	25.44	11.96	2.60	821.2	13.9	11.7
										Average	836.2	82.9	57.4

to the best overall accuracy accross NN-models and data sets (even for the building energy data where min-max normalization provides better result, the difference is relatively minor). Therefore, in the rest of this section, we will train NN-models on Z-normalized data by default (although variate selection process may be applied on three considered normalization strategies).

3.7.1 Impact of variate selection on forecasting accuracy

In Table 3, we present average MAE values for different degrees of variate selection, learning models, data normalization strategies, and forecasting leads (the MAE scores included in this table are averages of MAEs for the six variate selection strategies). From this table, we see that *CNN with tight variate selection provides the best overall results*: It is interesting to note that, even though it is not often the best option when considering all 100% of the variates, CNN-based models become highly effective when we are able to select and focus only the relevant variates through the variate selection strategy; this confirms our argument that, when coupled with variate selection, CNNs could be more effective than sequence-aware recurrent networks (such as RNN and LSTM) that attempt to learn temporal patterns (through recurrence) but have difficulties in achieving this task in practice.

As expected, when using all 100% variates, attention technique may be used to help reduce MAE, but its impact on accuracy is limited and in some cases (especially when aiming forecasting with large leads) attention can actually reduce accuracy; in contrast, variate selection is significantly more effective in eliminating noise and unnecessary data and, thus, consistently provides significantly large reductions in MAE. In the experiments, the only noticeable exception is for NASDAQ-APPLE data set with lead times ≥ 5, using LSTM model with very tight (10%) variate selection – but even for that data set and lead times, RNN and CNN both provide significant accuracy gains using only 10% selected variates.

3.7.2 Selego versus other variate selection strategies

In Table 4, we present the average DCG scores for the six variate selection algorithms and three data normalization strategies (for a total of 18 alternatives). The DCG scores

Table 3 Average MAE values for different degrees of variate selection, learning models, data normalization strategies, and forecasting leads (the MAE scores are averages of MAEs for the six variate selection strategies)

			Average MAE (average for all SIX Variate Selection Strategies)																	
			LSTM				LSTM (Att)	RNN					RNN(Att)	CNN					CNN(Att)	
			10%	50%	90%	100%	100%	10%	50%	90%	100%	100%	10%	50%	90%	100%	100%	
NASDAQ-INDEX	No Norm	Lead-1	11.2	15.0	13.7	14.2	28.4	3.1	18.8	18.2	15.1	13.9	1.8	13.6	7.9	8.9	10.3	
		Lead-5	23.2	17.2	31.4	10.2	34.1	4.9	13.3	11.3	18.2	14.8	2.0	10.7	8.3	12.3	11.1	
		Lead-10	18.3	18.9	37.7	8.3	38.7	2.9	9.5	11.3	18.7	18.4	1.5	6.6	8.0	11.3	11.0	
		Lead-50	21.4	17.2	24.9	12.2	46.7	2.9	6.2	8.8	14.5	18.7	1.4	5.3	8.0	12.1	20.7	
	Z Norm	Lead-1	19.0	14.7	16.6	14.2	28.4	2.7	12.2	10.2	15.1	13.9	1.3	14.2	7.9	8.9	10.3	
		Lead-5	7.2	20.7	25.4	10.2	34.1	2.2	9.5	10.8	18.2	14.8	0.9	11.5	9.6	12.3	11.1	
		Lead-10	13.0	16.1	43.7	8.3	38.7	3.2	9.9	10.3	18.7	18.4	1.4	9.5	8.1	11.3	11.0	
		Lead-50	5.0	13.7	23.7	12.2	46.7	1.7	5.9	11.0	14.5	18.7	1.2	5.9	8.8	12.1	20.7	
			10%	50%	90%	100%	100%	10%	50%	90%	100%	100%	10%	50%	90%	100%	100%	
NASDAQ-APPLE	No Norm	Lead-1	1.8	1.4	1.3	1.5	2.1	0.2	1.4	1.5	1.4	1.2	0.2	0.6	0.9	0.9	0.8	
		Lead-5	3.3	1.6	1.6	1.3	1.6	0.4	1.4	1.8	1.3	1.7	0.2	0.9	1.3	0.9	1.1	
		Lead-10	3.3	2.2	2.0	1.0	2.5	0.2	1.5	1.7	1.6	1.5	0.1	0.8	1.5	0.9	0.7	
		Lead-50	3.2	1.8	2.1	0.9	1.4	0.3	1.2	1.6	1.9	1.4	0.3	0.6	1.1	0.8	1.1	
	Z Norm	Lead-1	1.7	1.6	1.8	1.5	2.1	0.2	1.6	1.7	1.4	1.2	0.1	0.7	1.4	0.9	0.8	
		Lead-5	1.1	1.6	1.6	1.3	1.6	0.2	1.4	1.8	1.3	1.7	0.1	0.9	1.3	0.9	1.1	
		Lead-10	1.1	1.9	1.7	1.0	2.5	0.2	1.4	1.8	1.6	1.5	0.1	0.6	1.3	0.9	0.7	
		Lead-50	1.5	2.4	2.0	0.9	1.4	0.1	1.3	1.6	1.9	1.4	0.1	0.5	1.0	0.8	1.1	
			10%	50%	90%	100%	100%	10%	50%	90%	100%	100%	10%	50%	90%	100%	100%	
EEG-O1	No Norm	Lead-1	13.6	14.3	14.7	14.6	13.6	13.7	15.1	15.4	15.4	14.1	1.4	1.9	2.4	2.5	2.5	
		Lead-5	14.1	14.6	14.9	15.0	14.8	14.2	15.5	16.0	16.1	15.6	0.9	1.3	1.6	1.9	16.3	
		Lead-10	13.2	14.3	14.7	14.8	14.7	13.8	15.2	15.6	15.5	15.1	0.9	1.4	1.8	1.9	16.3	
	Z Norm	Lead-1	13.5	14.1	14.5	14.6	13.6	13.7	15.0	15.4	15.4	14.1	1.4	1.9	2.4	2.5	2.5	
		Lead-5	14.0	14.6	15.0	15.0	14.8	14.1	15.5	16.0	16.1	15.6	0.9	1.3	1.7	1.9	16.3	
		Lead-10	13.5	14.4	14.8	14.8	14.7	14.0	15.1	15.7	15.5	15.1	1.0	1.5	1.8	1.9	16.3	
			10%	50%	90%	100%	100%	10%	50%	90%	100%	100%	10%	50%	90%	100%	100%	
EEG-O2	No Norm	Lead-1	14.6	15.5	15.7	15.6	15.0	14.7	16.3	16.6	16.7	16.4	1.4	1.9	2.4	2.6	2.5	
		Lead-5	15.1	15.4	15.9	16.0	16.0	15.3	16.5	17.1	17.3	17.1	0.9	1.4	1.8	1.9	17.4	
		Lead-10	14.6	15.3	15.6	15.7	15.3	14.9	16.3	16.9	17.0	16.9	1.0	1.4	1.8	2.0	17.4	
	Z Norm	Lead-1	14.7	15.6	15.9	15.6	15.0	14.7	16.1	17.0	16.7	16.4	1.4	1.9	2.3	2.6	2.5	
		Lead-5	15.1	15.3	16.0	16.0	16.0	15.3	16.5	17.2	17.3	17.1	0.8	1.3	1.8	1.9	17.4	
		Lead-10	14.6	15.4	15.6	15.7	15.3	14.8	16.3	16.8	17.0	16.9	0.9	1.4	1.8	2.0	17.4	
			2%	10%	50%	90%	100%	100%	2%	10%	50%	90%	100%	100%	2%	10%	50%	90%	100%	100%
Building Energy	No Norm	Lead-1	37.3	39.9	49.7	55.4	57.9	50.2	34.5	39.5	52.0	55.3	55.6	51.7	6.9	11.6	38.8	59.0	80.0	62.9
		Lead-5	34.4	41.5	59.4	63.9	71.0	49.7	39.8	44.2	61.1	72.7	74.2	55.8	7.3	12.3	40.3	55.9	60.3	92.7
		Lead-10	35.3	38.6	55.2	68.5	73.8	59.5	35.6	41.5	57.0	65.9	73.2	67.9	7.4	13.5	37.0	47.8	56.4	91.6
	Z Norm	Lead-1	36.5	37.7	53.1	58.2	57.9	50.2	35.0	36.4	51.9	57.4	55.6	51.7	5.3	9.8	35.6	60.8	80.0	62.9
		Lead-5	34.5	40.3	53.2	61.1	71.0	49.7	39.1	40.8	62.2	67.6	74.2	55.8	6.4	11.3	36.4	53.8	60.3	92.7
		Lead-10	35.2	39.2	53.3	68.8	73.8	59.5	38.1	40.5	57.8	69.8	73.2	67.9	7.0	11.7	33.9	53.3	56.4	91.6
			5%	10%	50%	90%	100%	100%	5%	10%	50%	90%	100%	100%	5%	10%	50%	90%	100%	100%
Fuel Cons.	No Norm	Lead-1	186.1	197.0	215.5	209.3	221.3	172.1	197.7	215.3	252.7	313.6	276.8	273.7	48.7	70.0	162.2	232.8	232.5	201.6
		Lead-5	187.5	194.1	233.4	216.1	214.1	213.1	204.9	231.2	260.5	301.1	288.5	270.0	61.8	71.5	170.9	234.5	234.7	223.5
		Lead-10	192.1	197.8	225.6	226.9	227.0	208.9	214.1	230.0	260.0	316.8	335.6	321.5	58.5	72.5	148.1	246.9	261.9	399.2
		Lead-50	194.5	197.9	250.9	263.2	268.5	435.2	206.4	224.2	265.2	287.6	290.0	956.9	40.2	46.8	96.4	175.5	211.0	1008.2
	Z Norm	Lead-1	171.3	184.3	189.7	207.0	221.3	172.1	184.0	195.9	219.3	290.0	276.8	273.7	36.4	50.6	152.1	203.3	232.5	201.6
		Lead-5	177.9	196.2	208.7	214.3	214.1	213.1	191.0	193.9	243.1	300.6	288.5	270.0	32.4	52.0	170.6	233.6	234.7	223.5
		Lead-10	183.9	193.9	216.1	227.3	227.0	208.9	194.5	229.0	262.1	313.2	335.6	321.5	38.5	50.2	143.4	219.1	261.9	399.2
		Lead-50	189.8	200.7	227.7	256.2	268.5	435.2	194.8	216.4	264.2	287.5	290.0	956.9	37.2	46.6	119.6	168.8	211.0	1008.2

included in the table are averages of DCG values for all data sets and all variate selection rates reported in Table 1. As we see in this table, under all data normalization strategies, the proposed Selegovariate selection strategy provides good results, indicating its robustness to the shape of the data – the best overall DCG result is obtained with Selego under Z-normalized data. In fact, the second best DCG is also provided by Selego under the original, non-normalized data: since Selegoignores the shapes of the patterns, but relies only on the co-occurrence/alignment of key events in the time series, it is inherently robust and does not require normalization to return accurate predictions. In contrast, variate selection techniques relying on global features (FRESH) and similarity/distance based measures (DTW, SAX and PCA) perform poorly under all normalization strategies: in fact the worst 11 configurations (among all 18 configurations) are obtained using FRESH, DTW, SAX or PCA. This confirms our argument that variates that have high predictive power have better temporal alignment of the local key "events/features" with the target series key events, and do not necessarily look similar to the target variate. While Inf-FS is somewhat competitive against Selego on non-normalized data, its best overall DCG value, 6.32, is significantly lower than the best DCG value, 6.76, achieved by Selego.

Table 4 Average DCG scores for the six variate selection algorithms and three data normalization strategies (total 18 alternatives) – the DCG scores are averages of DCG values for all data sets and all six variate selection rates reported in Table 1

Norm Type	Avg. DCG Score (higher the better)		RANK
No Norm	DTW	5.34	9
	PCA	4.92	16
	Inf-FS	6.32	3
	FRESH	4.71	18
	SAX	4.95	15
	Selego	6.76	1
Min-Max Norm	DTW	5.30	10
	PCA	5.15	13
	Inf-FS	5.73	6
	FRESH	4.85	17
	SAX	5.85	5
	Selego	6.17	4
Z Norm	DTW	5.27	11
	PCA	4.95	14
	Inf-FS	5.51	7
	FRESH	5.20	12
	SAX	5.41	8
	Selego	6.71	2

Table 5 Average DCG scores for the six variate selection algorithms under three data normalization strategies for different data sets – the presented DCG scores are averages of DCG values for all variate selection rates reported in Table 1

Norm Type		NASDAQ INDEX	NASDAQ APPLE	EEG-O1	EEG-O2	Building Energy	Fuel Cons.
No Norm	DTW	4.84	4.61	4.93	5.36	5.66	6.63
	PCA	4.61	4.96	3.98	3.85	7.06	5.09
	Inf-FS	5.48	5.80	5.22	5.57	7.90	7.93
	FRESH	4.20	4.38	4.51	4.33	5.12	5.75
	SAX	4.37	4.94	4.40	4.10	5.41	6.46
	Selego	6.24	5.05	6.38	6.54	8.51	7.83
Min-Max Norm	DTW	4.69	4.85	4.77	5.02	6.11	6.37
	PCA	4.04	5.28	3.93	4.09	7.83	5.72
	Inf-FS	4.67	4.26	5.87	5.61	5.64	8.32
	FRESH	4.81	4.59	4.48	4.33	5.61	5.30
	SAX	6.74	5.35	4.70	4.74	5.97	7.59
	Selego	4.80	5.40	5.99	5.96	8.49	6.36
Z Norm	DTW	4.94	4.17	4.62	4.90	6.04	6.93
	PCA	4.37	4.72	4.49	4.30	6.70	5.11
	Inf-FS	5.01	4.32	5.23	5.51	6.84	6.13
	FRESH	5.28	4.81	4.10	4.31	5.48	7.22
	SAX	4.46	5.13	5.39	4.32	6.66	6.51
	Selego	5.68	6.60	5.90	6.40	7.92	7.75

Table 6 Model training times for the Building Energy data set (lead time 5 hours)

		Model Training Time as a Function of the ratio of the Selected Variables (in seconds)														
		LSTM					RNN					CNN				
		2%	10%	50%	90%	100%	2%	10%	50%	90%	100%	2%	10%	50%	90%	100%
Building Energy Lead-5	DTW	111.63	112.11	117.34	121.98	109.03	31.03	33.23	35.24	42.40	67.99	8.07	10.55	18.81	23.00	42.14
	PCA	116.56	120.23	121.51	123.58		29.70	34.20	35.73	42.58		8.94	11.08	17.34	23.49	
	Inf-FS	98.71	102.02	103.82	108.22		30.44	33.22	35.66	40.78		9.43	9.96	18.52	23.01	
	FRESH	98.36	108.52	119.29	123.98		28.40	32.40	31.91	40.18		8.18	11.12	16.17	22.80	
	SAX	96.11	98.63	101.50	103.98		38.57	39.13	37.64	38.46		10.86	12.19	15.72	20.21	
	Selego	98.46	99.32	104.64	107.95		29.05	35.16	36.30	40.96		8.04	10.17	14.58	23.73	

Table 7 Inference times for the Building Energy data set (lead time 5 hours)

		Inference Time as a Function of the ratio of the Selected Variables (in seconds)														
		LSTM					RNN					CNN				
		2%	10%	50%	90%	100%	2%	10%	50%	90%	100%	2%	10%	50%	90%	100%
Building Energy Lead-5	DTW	0.21	0.24	0.26	0.26	0.21	0.12	0.12	0.12	0.12	0.14	0.04	0.04	0.04	0.04	0.06
	PCA	0.24	0.21	0.21	0.24		0.14	0.14	0.13	0.13		0.06	0.04	0.04	0.06	
	Inf-FS	0.21	0.24	0.26	0.26		0.13	0.12	0.12	0.12		0.04	0.04	0.04	0.04	
	FRESH	0.21	0.21	0.22	0.28		0.12	0.14	0.12	0.12		0.05	0.05	0.05	0.05	
	SAX	0.20	0.21	0.20	0.23		0.16	0.18	0.17	0.18		0.07	0.07	0.07	0.07	
	Selego	0.20	0.21	0.21	0.24		0.12	0.12	0.12	0.12		0.04	0.04	0.04	0.06	

Table 8 Variate selection times for the Building energy data set (lead time 5 hours)

		Variate Selection Time (in seconds)		
		Event Extraction	Variate Ranking	Total Time
Building Energy	DTW	N/A	112.57	112.57
	PCA	N/A	0.42	0.42
	Inf-FS	N/A	0.86	0.86
	FRESH	41.17	0.08	41.26
	SAX	N/A	1.34	1.34
	Selego	0.00	0.09	0.09

In Table 5, we take a more detailed look at the DCG scores. In particular, we present average DCG scores separately for each normalization strategy. As we see here, when considering *Z-normalized* data, Selegoprovides the best performance for all data sets/forecasting tasks considered. When considering *non-normalized* data, Selegois superior for 4 out of 6 tasks and for the "Nasdaq Apple" and "Fuel Consumption", Inf-FS provides better performance – note, however, Inf-FS performs poorly under *min-max normalization* and *Z-normalization* strategies for this data set. Note that also when considering *min-max non-normalized* data, Selegois superior most of the tasks: SAX is better on NASDAQ and Inf-FS is better on Fuel Consumption, but neither consistently outperforms Selego. Instead, Selegoproves to be highly robust across data sets and normalization strategies.

3.7.3 Execution times

In Tables 6 through 8, we see the impact of variate selection on the overall computational complexity (due to space limitations, here we only include results for the building energy data set, the results for the other data sets are similar). As we see in Table 6, as would be expected, variate selection tends to reduce the model training

Table 9 Average min-normalized MAE scores for the under three data normalization strategies for different variate ranking strategies (**lower the better**) – the presented scores are averages of CNN-model accuracies for all feature selection strategies and data sets, under the tightest variate selection rates

	PR	PPR	KNN
No Norm	1.12	1.15	1.36
Min-Max Norm	1.23	1.13	1.13
Z Norm	1.10	1.18	1.08

times. The results show that the gains are the most pronounced for the CNN and that Selego provides the highest training time gains. Interestingly, similarity based variate selection strategies (DTW and PCA) hurt the training time under LSTM, which indicates that, if not carried out properly, variate selection can negatively impact training performance.

Table 8, then, presents the execution times for the variate selection process that preceeds model training. As we see here, except for DTW and FRESH, the variate selection times are essentially negligible relative to the model training times reported in Table 6 – DTW takes the most time $\sim$ 112 seconds to compare 389 source variates to one target variate; i.e, $\sim$ 0.3 second on average per comparing a pair of variates. This indicates that Selego based variate selection, not only provides boosts on accuracy, but achieves this without any penalty on the overall time needed to prepare the data for model training.

Table 7 shows that the inference times also slightly improve under variate selection (especially when using Selego, with tight variate budget), but the gains are too slight to be meaningful in the considered application scenarios – though the gains might prove to be significant in other contexts.

3.7.4 Variate ranking strategies

As reported earlier, the above results have been obtained under the KNN-based variate ranking strategy. In Table 9, we also consider alternative PR and PPR-based variate ranking strategies. As we see in this table, KNN-based ranking (which we considered as default) under Z-normalization provides the best overall accuracies among all alternatives. While it does not provide accuracy gains, using the alignment-graph topology information helps avoid worst case accuracy behaviors, with PPR providing the highest robustness, with the tightest accuracy range across different normalization strategies.

4 Conclusions

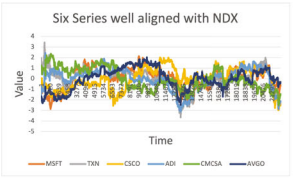
In this paper, we introduced Selegoframework for variate selection to support accurate time series prediction. Selego relies on three key observations: (a) *temporal alignments* among variates can be used to quantify the importance of the recorded variates with respect to a target variate, (b) yet, traditional time series similarity/distance functions, such as DTW, are fundamentally ill-suited for this purpose. Moreover, (c) when coupled with robust variate selection, even simple CNN-based models can potentially be more accurate than complex and costly recurrence based techniques (such as RNN and LSTM). Experiments using LSTM, RNN, and CNN, for different top-X% variates and different forecasting leads on multiple real-world datasets have shown that the proposed framework can offer significant $(90-98\%)$ drops in the number of variates and significantly boost the overall prediction accuracies. Finally, we note that the version of the Selego described in this paper has two limitations: (a) it defines temporal alignment of features based on overlap and (b) it trains the neural network model for a fixed lead time. While we have seen empirically good results under both assumptions, in future work, we will relax these assumptions.

Acknowledgements This work is partially supported by NSF#1827757 "Building Doctor's Medicine Cabinet (BDMC): Data-Driven Services for High Performance and Sustainable Buildings", NSF#1610282 "DataStorm: A Data Enabled System for End-to-End Disaster Planning and Response", NSF#1633381 "BIGDATA: Discovering Context-Sensitive Impact in Complex Systems", NSF#1909555 "pCAR: Discovering and Leveraging Plausibly Causal (p-causal) Relationships to Understand Complex Dynamic Systems", and DOE grant "Securing Grid-interactive Efficient Buildings (GEB) through Cyber Defense and Resilient System (CYDRES)". Part of the research was carried out using the Chameleon testbed supported by the NSF.

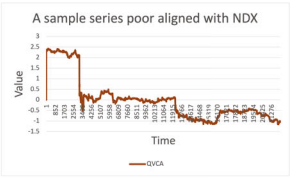
Appendix—sample series and feature distributions

Figures 7 through 9 provide examples of target variables, the best series aligned based on feature distributions, along with a sample for poorly aligned series. In order to better visualize the feature alignments, consecutive series (e.g. the consecutive days in NASDAQ) have been concatenated and the number of feature layers considered in these charts have been raised from the number of layers considered in the experiments. As we see in these figures, temporal alignment of variates does not mean that they must look similar: instead, alignment only means that the two series show evidence of being impacted from the same underlying events. In Fig. 9b, for example, we see six variates that, together, predict the fuel consumption series 9a well. We also see in the figure that these series used for model training are temporally aligned with the target series but are not necessarily similar to it.

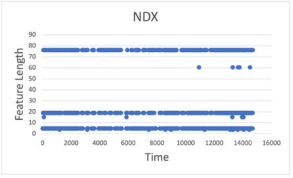
(a) NDX series



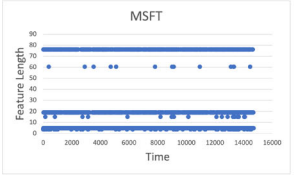
(b) 6 well aligned series



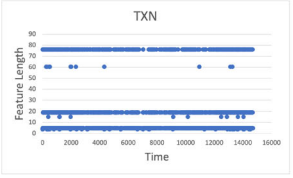
(c) poorly aligned series



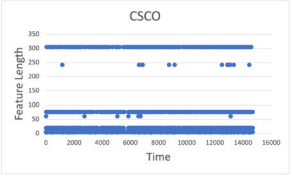
(d) Target variate: NDX



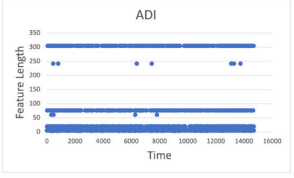
(e) top-1 variate



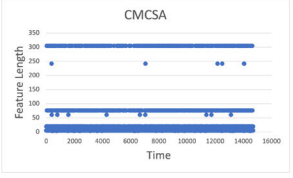
(f) top-2 variate



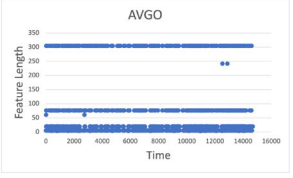
(g) top-3 variate



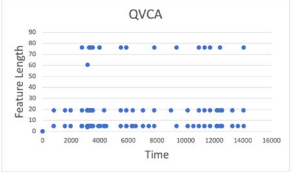
(h) top-4 variate



(i) top-5 variate

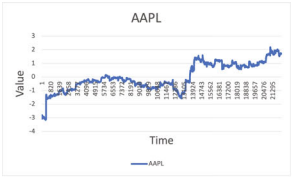


(j) top-6 variate

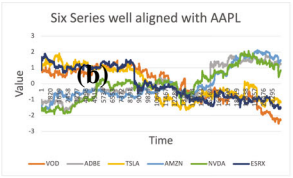


(k) a poorly aligned variate

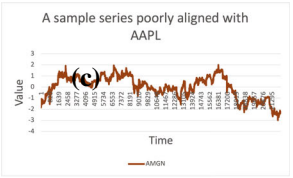
Fig. 7 **a** The target variable NDX (NASDAQ index); **b** the best 6 series aligned with it *(note that alignment of series do not necessarily imply that the series are globally similar – it only means that they show evidence of the same underlying events)*; **c** a poorly aligned series; **d–k** temporal distributions (time and length) of the identified features in these series (here the X-axis denotes the time and the Y-axis identifies the length of the feature identified at a particular point in time)



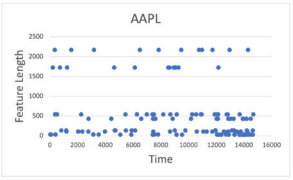
(a) AAPL series



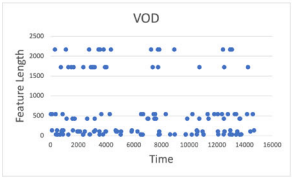
(b) 6 well aligned series



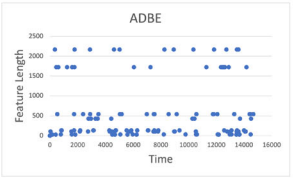
(c) poorly aligned series



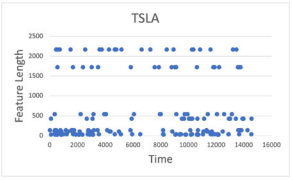
(d) Target variate: APPLE



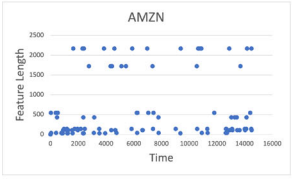
(e) top-1 variate



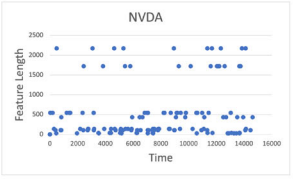
(f) top-2 variate



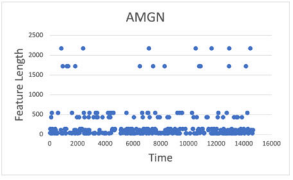
(g) top-3 variate



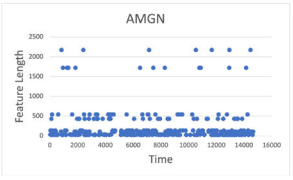
(h) top-4 variate



(i) top-5 variate

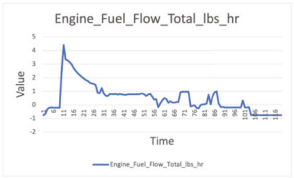


(j) top-6 variate

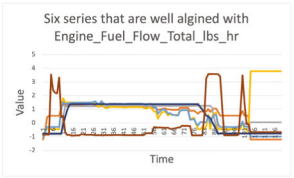


(k) a poorly aligned variate

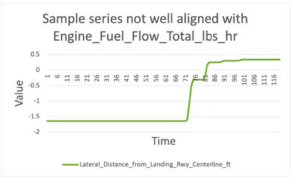
Fig. 8 **a** The target variable AAPL (symbol for the Apple stock); **b** the best 6 series aligned with it*(note that alignment of series do not necessarily imply that the series are globally similar – it only means that they show evidence of the same underlying events)*; **c** a poorly aligned series; **d–k** temporal distributions (time and length) of the identified features in these series (here the X-axis denotes the time and the Y-axis identifies the length of the feature identified at a particular point in time)



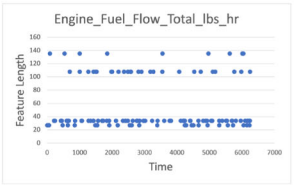
(a) fuel flow series



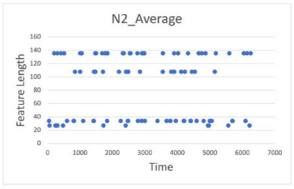
(b) 6 well aligned series



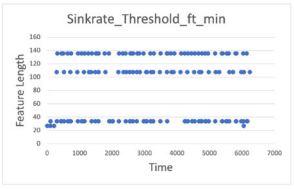
(c) a poorly aligned series



(d) Target variate: fuel flow

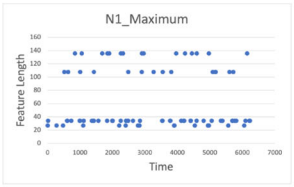


(e) top-1 variate

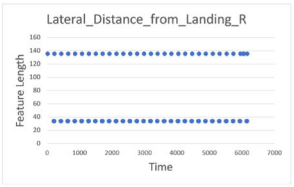


(f) top-2 variate

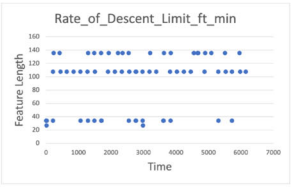
(g) top-3 variate



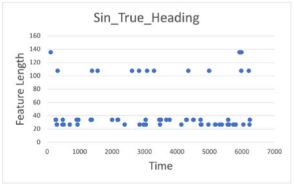
(h) top-4 variate



(i) top-5 variate



(j) top-6 variate



(k) a poorly aligned variate

Fig. 9 **a** The target variable *fuel consumption*; **b** the best 6 series aligned with it *(note that alignment of series do not necessarily imply that the series are globally similar—it only means that they show evidence of the same underlying events)*; **c** a poorly aligned series; **d–k** temporal distributions (time and length) of the identified features in these series (here the X-axis denotes the time and the Y-axis identifies the length of the feature identified at a particular point in time)

References

Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J et al (2016) Tensorflow: A system for large-scale machine learning. In USENIX, OSDI

Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In ICLR

Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. JMLR

Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In KDD workshop

Bianco V, Manca O, Nardini S (2009) Electricity consumption forecasting in italy using linear regression models. Energy 34(9):1413–1421

Blei DM, Lafferty JD (2006) Dynamic topic models. In Proceedings of the 23rd International Conference on Machine Learning (ICML '06), pp 113–120

Box GE, Jenkins GM, Reinsel GC, Ljung GM (2015) Time series analysis, forecasting and control, 5th ed., Wiley

Candan KS, Rossini R, Sapino ML, Wang X (2012) sdtw: Computing dtw distances using locally relevant constraints based on salient feature alignments. VLDB

Chen L, Ng R (2004) On marriage of lp-norms and edit distance. In VLDB

Christ M, Braun N, Neuffer J, Kempa-Liehr A (2018) Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package)

Clevert D-A, Unterthiner T, Hochreiter S (2016) Fast and accurate deep network learning by exponential linear units (elus). ICLR

Drucker H, Burges CJ, Kaufman L, Smola AJ, Vapnik V (1997) Support vector regression machines. In NIPS

Fernandez-Fraga S, Aceves-Fernandez M (2018) Feature extraction of eeg signal upon bci systems based on steady-state visual evoked potentials using the ant colony optimization. Discrete Dynamics in Nature and Society

Garg Y, Candan KS (2019) Racknet: Robust allocation of convolutional kernels in neural networks for image classification. In ICMR

Garg Y, Candan KS (2021a) Sdma: Saliency-driven mutual cross attention for multi-variate time series. In ICPR, IEEE

Garg Y, Candan KS (2021b) Xm2a:multi-scale multi-head attention with cross talk for time series analysis. In MIPR

Goodwin P, Dargay J, Hanly M (2004) Elasticities of road traffic and fuel consumption with respect to price and income: a review. Transport reviews

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In CVPR

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780

Keogh E, Ratanamahatana CA (2005) Exact indexing of dynamic time warping. Knowl Inf Syst 7:358–386

Linderman S, Adams R (2014) Discovering latent network structure in point process data. In ICML'14, PMLR 32(2), pages 1413–1421

Lin J, Keogh E, Lonardi S, Patel P (2002) Finding motifs in time series. In Workshop on Temporal Data Mining

Lowe D (2004) Distinctive image features from scale-invariant keypoints. IJCV

Mueen A, Keogh E (2016) Extracting optimal performance from dynamic time warping. In SIGKDD

Pearson K (1901) Principal components analysis. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science

Qin Y, Song D, Chen H, Cheng W, Jiang G, Cottrell G (2017) A dual-stage attention-based recurrent neural network for time series prediction. IJCAI

Roffo G, Melzi S, Cristani M (2015) Infinite feature selection. In ICCV

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323:533–536

Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. TASSP

Salton G, McGill MJ (1983) Introduction to modern information retrieval. McGraw-Hill, Inc

Shatkay H, Zdonik SB (1996) Approximate queries and representations for large data sequences. In ICDE, IEEE

Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D (2015) Going deeper with convolutions. In CVPR

Tong H, Faloutsos C, Pan J-Y (2006) Fast random walk with restart and its applications. In ICDM, page 613–622
Tucker L (1966) Some mathematical notes on three-mode factor analysis. Psychometrika 31:279–311
Wang F, Jiang M, Qian C, Yang S, Li C, Zhang H, Wang X, Tang X (2017) Residual attention network for image classification. In CVPR
Yuan B, Li H, Bertozzi AL, Brantingham PJ, Porter MA (2019) Multivariate spatiotemporal hawkes processes and network reconstruction. SIAM J. Math. Data Sci. 1(2):356–382
Zoph B, Le QV (2017) Neural architecture search with reinforcement learning. ICLR