Temporalizing Static Graph Autoencoders to Handle Temporal Networks

Mounir Haddad, Cécile Bothorel, Philippe Lenca, Dominique Bedart

Temporalizing Static Graph Autoencoders to Handle Temporal Networks

Mounir Haddad
Lab-STICC, CNRS UMR 6285
IMT Atlantique, Brest, France
DSI Global Services
Plessis-Robinson, France
mounir.haddad@imt-atlantique.fr

Cécile Bothorel
Lab-STICC, CNRS UMR 6285
IMT Atlantique, Brest, France
cecile.bothorel@imt-atlantique.fr

Philippe Lenca
Lab-STICC, CNRS UMR 6285
IMT Atlantique, Brest, France
philippe.lenca@imt-atlantique.fr

Dominique Bedart
DSI Global Services
Plessis-Robinson, France
dominique.bedart@dsi-globalservices.fr

***Abstract*—Graph autoencoders (GAE), also known as graph embedding methods, learn latent representations of the nodes of a graph in a low-dimensional space where the structural information is preserved. While real-world graphs are generally dynamic, only a few embedding methods handle the temporal dimension: Even though they have proven their reliability, the majority of the embedding techniques address the case of static networks and present poor performances when applied to temporal ones.**

In this paper, we present a generic method to *temporalize* static graph autoencoders, i.e. adapt different static graph embedding methods to the case of temporal networks. This is made possible by learning optimal connections between timesteps' graphs in order to form a single merged spatio-temporal network. We prove that this highly improves the inference tasks' accuracy of the temporalized methods. We also show that the learned connections are directly related to nodes characteristics and can be used beyond the scope of the embedding they are designed for.

***Index Terms*—Graph autoencoders, Graph embeddings, Temporal networks, Node classification, Edge reconstruction and prediction**

I. Introduction

Many real-world phenomena consist of interactions between entities, generally represented in graphs. When they are addressed properly, these graphs can reveal important information about their fundamental structures helping to discover local, global or temporal interaction patterns. Such knowledge is very useful to understand how information gets diffused or how epidemics spread. However, in their original form, graphs are not easily exploitable by machine learning models. Therefore, a prerequisite is to build graphs' representations suitable for the downstream inference tasks.

During the last decade, new research approaches known as representation learning techniques aim to encode various data types into low-dimensional representations, called embeddings, in latent vector spaces [2]. Representation learning techniques have been first designed for text mining [17] and have shown very conclusive performances. Thereafter, they have been adapted to other data structures such as graphs. Some approaches are based on random walks [5], [9], [21], matrix factorization [1], [4], [19], or neural networks [14], [20], [26].

Although these methods are well-proven and show remarkable results regarding numerous application scenarios, they are designed to address static graphs and are generally less efficient when they are applied to temporal ones, partly due to the misalignment and the instability of the embeddings [10]. However, the temporal dimension is elementary for evolution patterns appreciation. There are different ways to incorporate this additional data. Some techniques employ the available temporal information to build more efficient global embeddings [18], [29]. In this work, we focus on temporal graph autoencoders, i.e. those which return sequences of embeddings [11], [12], [16], [22], [28]. These techniques enhance the performances on the inference tasks that are sensitive to the temporal dimension. However, they are hard to design and are therefore few comparing to the static ones. In this paper, rather than conceiving another temporal embedding model, we present a generic method to *temporalize* existing static graph autoencoders, provided that they are composed of neural networks taking graphs adjacency matrices as input.

For this purpose, we use and modify the so-called *supra-adjacency* representation of a temporal network [7], [23], [25]. It consists of a mapping between a sequence of graphs (a temporal network in our case) and a static *supra-graph* whose *supra-nodes* are pairs of {node, timestep} of the original temporal graph. The *supra-edges* of this supra-graph are transposed from the original graph: For example, an edge between the nodes v_1 and v_2 at timestep t translates into a supra-edge between the supra-nodes $\{v_1, t\}$ and $\{v_2, t\}$. At this stage, the supra-graph is composed of T disconnected components, where T is the number of timesteps in the original temporal graph. Then, to connect pairs of supra-nodes not belonging to the same component, additional weighted

supra-edges are created. In the rest of the paper, *temporal edges* and *temporal weights* will respectively refer to these additional supra-edges and their weights.

Our main contribution lies in the way these temporal edges are created. Indeed, where other supra-adjacency-based approaches [7], [23], [25] attribute fixed weights to the *supra-edges* connecting the supra-nodes, we assign learned weights, making our method more data-driven. We show in our experiments that the built supra-graph is more suitable to node classification/prediction and link reconstruction/prediction inference tasks in terms of performances. Also, we demonstrate that the learned temporal edges can directly be employed to temporalize other static graph embeddings models that are not suitable for the presented temporalization method, such as skip-gram-based ones [9], [21]. This ensures highly improved inference performances. Finally, we expose some interesting correlations between the characteristics of the nodes and the learned temporal edges' weights.

The remainder of this paper is organized as follows. In section II, we describe our temporalization approach. Section III presents our experimental setting. In section IV, we expose the performed experiments and interpret the results.

II. Autoencoders temporalization

The key idea behind our approach is the way we adapt and slightly modify autoencoders' structure to learn the optimal weights to be assigned to the temporal edges.

A. Additional input layer

Given a static neural network-based embedding method taking an adjacency matrix as input, we first insert a preliminary layer connected to the input data, which role is to form the supra-adjacency matrix. Concretely, this layer builds a block diagonal matrix of shape $(|V|\cdot T)\times(|V|\cdot T)$ from the sequence of T input $|V| \times |V|$ adjacency matrices, where V is the set of the original graph nodes. Then, additional trainable entries, corresponding to the weights of temporal edges, are filled in the block diagonal matrix, as figure 1a shows.

For the sake of model simplicity, we do not consider the possibility of creating temporal edges between any supra-nodes. The temporal edges we allow are of shape: $\{v,\, t_i\}$, $\{v,\, t_j\}$ (figures 1b and 1c), i.e. supra-edges between a node and itself at different timesteps. Otherwise, the number of trainable weights would be much larger, increasing training time.

We consider different variants of temporal edges:

1) Directed/undirected temporal edges: Once the supra-adjacency matrix is built, all its entries have the same nature, irrespective of whether they correspond to learned temporal edges or supra-edges derived from the original graph. This means that, beyond their latent character, temporal edges may also have a practical signification similar to the other edges: If so, a temporal edge between $\{v,\, t_i\}$ and $\{v,\, t_j\}$ would mean that there is a mutual influence between the states of the node v at t_i and t_j. Having said that, a node that would impact itself in previous timesteps doesn't seem to be realistic. Therefore, one possibility to avoid this situation is to use directed temporal edges, for example a directed edge from $\{v, t_i\}$ to $\{v, t_j\}$ with $t_i < t_j$. This translates into an asymmetric supra-adjacency matrix where only the lower triangular part of the matrix is filled with the learned temporal weights. Otherwise, one can still consider symmetric supra-adjacency matrices. In such a case, an undirected connection between $\{v, t_i\}$ and $\{v, t_j\}$ can be interpreted as a temporal smoothing constraint that forces the continuity of a node's embeddings over time, rather than a mutual influence between two different timesteps states of a node. We define the hyperparameter s that controls the two possibilities: s equals 0 or 1 when we respectively impose directed/undirected temporal edges (resp. figures 1c and 1d).

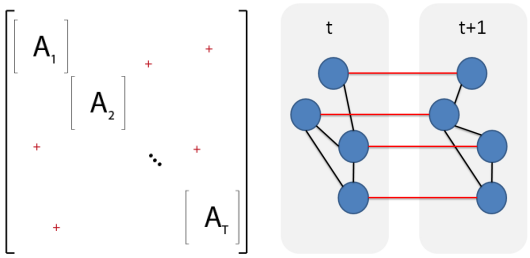
2) Temporal window: The main intuition behind the presented autoencoder temporalization method lies in the fact that a node embedding should be conditioned by its interactions as well as by its previous states. The most straightforward way to achieve this is to create temporal edges between each pair of supra-nodes $\{v,\, t\}$, $\{v,\, t+1\}$ (with $(v, t) \in V \times [\![1,\, T-1]\!]$). However, it is possible to let the temporalized autoencoder build more sophisticated and complex temporal evolution patterns by allowing the temporal edges to cover longer time intervals. To that end, we consider a hyper-parameter called the temporal window w. For a given value of w, the set of temporal edges to add to the supra-adjacency matrix is: $\{v, t\}$, $\{v, t+i\}$ for $(v, i, t) \in V \times [\![1, w]\!] \times [\![1, T-1]\!]$ with $t+i \leqslant T$ (figure 1e). Also, it is possible to combine undirected temporal edges with a temporal window $w > 1$ as shown in figure 1f.

B. Output layer modification

In order to obtain reliable embeddings, the last layer of graph autoencoders generally consists of a comparison between the input adjacency matrix and a pairwise similarity measure of nodes' embeddings, often the dot product. In our case, it is not relevant to compare the whole supra-adjacency matrix to the similarities of all supra-nodes pairs. As a matter of fact, the reliability of the embeddings is determined by the ability to preserve the input graph structures. Thus, as the temporal edges are additional artifacts, they shouldn't be taken into account within the last layer. Consequently, in the last layer, the comparison is made between the supra-adjacency matrix with no temporal edges and the similarities of the supra-nodes pairs that belong to the same timestep. This is equivalent to comparing the sequence of the T input adjacency matrices and the pairwise similarities of nodes for each timestep.

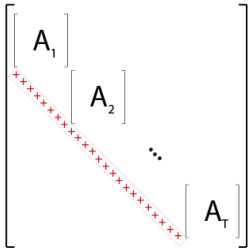
C. Temporal embeddings

A temporalized autoencoder learns latent representations of supra-nodes. This means that, given a temporal graph composed of $|V|$ nodes on T timesteps, the temporalized autoencoder returns $|V| \cdot T$ embedding vectors, one for each supra-node. Then, it is easily possible to reshape this embedding matrix into T matrices of shape $|V| \times d$, where d is the embedding dimension. Each one of these T matrices represents the embeddings of the original graph nodes' at a certain timestep.

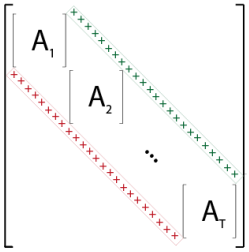


(a) Supra-adjacency matrix structure

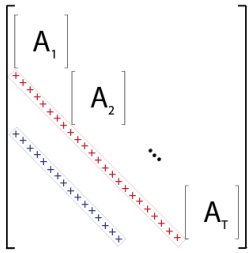
(b) Temporal graph basic temporalization



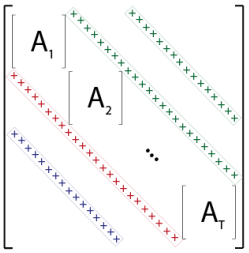
(c) Temporalization with $s = 0$ and $w = 1$



(d) Temporalization with $s = 1$ and $w = 1$



(e) Temporalization with $s = 0$ and $w = 2$



(f) Temporalization with $s = 1$ and $w = 2$

Fig. 1: Examples of supra-adjacency matrices. A_i represents the i-th timestep adjacency and the symbols $+$ mark the locations of the temporal weights entries

III. Experimental setting

A. Temporalized autoencoders

As stated in section II, the embedding methods that are suitable to our temporalization technique must meet some criteria: They should consist of neural networks taking an adjacency matrix in input. We consider 5 models, presented in the following papers:

- SDNE [26]: This model learns embeddings by training a neural network aiming to preserve jointly the first-order and the second-order proximity of graphs nodes'. Apart from the embedding dimension d, SDNE has two hyperparameters α and ν as well as the number of hidden layers and their respective sizes. In our experiments, we use popular default values for α and ν and we consider only one hidden layer, with a size equal to $2d$.
- GAE/VGAE [14]: This work describes two different methods. The main idea behind both is employing a graph convolutional network [13] as the encoder and an embedding pairwise inner product as the decoder. VGAE differs from GAE as it maps the input data to a distribution rather than a vector. The embeddings are obtained by a random sample of the learned distribution. In our experiments, we use the default values for the two models' hyperparameters, i.e. the learning rate, the decay, and the dropout. For the hidden layer, we set its size to double the embedding dimension.
- ARGAE/ARGVAE [20]: In this work, the variational graph autoencoder approach models have been taken over and modified using adversarial regularization [8] to enforce latent embeddings to match a prior distribution. In a similar way to [14], two variants are designed, ARGAE and its variational version ARGVAE. In our experiments, we keep the same hyperparameters' values used in GAE/VGAE.

For the temporalization purpose, two additional hyperparameters s and w, are required as described in section II-A. In our experiments, we test a grid search over $(s, w) \in \{0, 1\} \times \{1, 2, 3\}$. In the rest of the paper, a temporalized autoencoder will be noted *TT* (trained temporalization), e.g. SDNE_TT or GAE_TT.

B. Baseline methods

To evaluate our approach, we compare the temporalized autoencoders' performances to other static and temporal embedding models. As static ones, we consider deepwalk (DW) [21], node2vec (N2V) [9] as well as the original static autoencoders we temporalize, i.e. SDNE, GAE, VGAE, ARGAE, and ARGVAE.

Also, to challenge the step where we learn the temporal weights, we temporalize each one of the static embedding methods using fixed (non-trainable) temporal edges' weights, in a similar way to the other supra-adjacency-based methods [7], [15], [23], [25]. In this context, we consider different strategies for assigning the fixed temporal weights: the maximum/average value of all the temporal network weights, or, for each node, the maximum/average value of its edges' weights. In the rest of the paper, this *fixed temporalization* method will be noted *FT*, e.g. SDNE_FT or GAE_FT.

For the temporal embedding baseline methods, we employ dynamicTriad (DT) [28] and temporalNode2vec (TN2V) [11].

Below, the hyperparameters sets tested for the different baseline methods considered in the comparison:

- DW: with wl and ws representing respectively the walk length and the window size, a grid search over $(wl, ws) \in \{40, 80, 120\} \times \{3, 5, 7\}$.
- N2V: we keep the values of wl and ws giving the best performances in DW, then we perform a grid search over $(p, q) \in \{0.5, 1, 2\}^2$.
- DT: a grid search over $(\beta_0, \beta_1) \in \{0.01, 0.1, 1, 10\}^2$ where β_0 and β_1 respectively stand for the triad closure process weight and the temporal smoothing parameter.
- TN2V: This model has 8 hyperparameters. For our experiments, we employ authors' tested values.

C. Datasets

To compare the different algorithms' performances, we gathered 3 real-world temporal networks, for which we have metadata about the nodes' labels, i.e. the ground-truth communities the nodes belong to.

- AMiner [24]: This dataset[1] consists of 11371 coauthor relationships (edges) between 2385 researchers (nodes), divided into 8 timestamped weighted graphs, one per year, where weights refer to the number of common articles between two authors. Regarding the research domains their articles address, researchers are mapped to research fields (labels).
- Yelp: This dataset[2] traces the timestamped comments web users made on businesses (malls, restaurants...). Upon this data, we build a temporal graph divided into 7 timesteps of equal durations, where the 2445 users and businesses are the nodes and the 2839 comments are the edges. As businesses are assigned with categories, we map users to the same categories, regarding the kind of businesses they usually comment on.
- Tmall: This dataset is extracted from the sales at Tmall[3] during the period preceding the "Double Eleven Day" event. It traces the interactions online shoppers had with products. Based on this data, we form a temporal graph of 8 timesteps (with equal durations), 2586 nodes (shoppers and products) and 4152 edges (interactions). Similarly to Yelp, we assign labels to the nodes based on the products' categories.

As the three considered datasets have a number of nodes of the same order of magnitude (and approximately a dozen of ground-truth communities), we consider an embedding dimension $d = 16$ for all of them[4].

D. Application scenarios

To compare the different models' performances, we consider 4 inference tasks.

- Node classification/prediction: Based on the output embeddings, the goal is to find nodes' labels using a classifier: the current timestep's ones for the node classification and the next timestep's ones for the node prediction.
- Edge reconstruction/prediction: In these two tasks, a classifier is trained on the distances between pairs of nodes' embeddings to determine the pairs that are connected by an edge (in the current timestep for the reconstruction and in the next one for the prediction).

For the different inference tasks, we use logistic regression as the classifier and the F1 score as the evaluation metric.

[1]We use an extract of the original AMiner dataset

[2]An extract of Yelp challenge dataset: https://www.yelp.com/dataset

[3]https://tianchi.aliyun.com/competition/entrance/231576/information

[4]The embedding dimension to choose depends mostly on the number of the ground-truth communities, but also on the number of nodes [3], [11].

IV. Experiments and results analysis

A. Temporalization contribution

First, we compare the inference scores of the original autoencoders and their fixed and trained temporalization. Figure 2 shows the results of this experiment where we represent the best scores for each model (best hyperparameters for TT and best fixing weights strategy for FT), inference task, and dataset.

As expected, the original autoencoders are less efficient than the temporalized ones. Also, we can see a noticeable difference in models performances ranking between the time-dependent tasks (i.e. node prediction and edge prediction) and the time-agnostic tasks (node classification and edge reconstruction). As a matter of fact, for the edge prediction task and, to a lesser extent, for the node class prediction task, the trained temporization presents better performances comparing to the fixed temporization. The original static autoencoder is outperformed by both temporalization variants. First, this means that temporalizing autoencoders, in a fixed or trained fashion, improves the embeddings efficiency. Secondly, learning the temporal weights brings additional enhancement to the temporalization process. This can be explained by the ability of the trained temporalization of capturing both spatial structures and temporal evolution patterns of the input temporal graph. On the other hand, the improvement of temporalization is minor and unsystematic when it comes to node classification and edge reconstruction.

Also, we notice that the different autoencoders globally react in the same way to the temporalization process. The improvements made on the tasks sensitive to the temporal dimension concern all the considered embedding methods and do not seem to change models ranking in F1 score.

B. Learned temporal weights reuse

Next, we look at the possibility of reusing the temporal weights learned within the autoencoders temporalization process. The aim is to figure out whether these temporal weights are specific to the autoencoder they have been conceived for, or they might be used beyond this scope. To that end, we conceive temporalized (fixed and trained) versions of DW and N2V as follows:

- DW_FT and N2V_FT: fixed temporalization, in a similar fashion to the fixed temporalization described in III-B.
- DW_TT and N2V_TT: fixed (non-trainable) temporalization where we employ the temporal weights learned within the temporalized autoencoder that gives the best F1 score for each inference task and dataset.

Figure 3 exposes the obtained results. Overall, temporalizing DW and N2V highly increases the inference performances for both time-dependent and time-independent inference tasks. Furthermore, DW_TT (resp. N2V_TT) presents an improvement, often very small, but yet systematic, comparing to DW_FT (resp. N2V_FT). This confirms our prior intuition stating that the learned temporal weights capture information about the temporal evolution patterns of nodes and can thus be

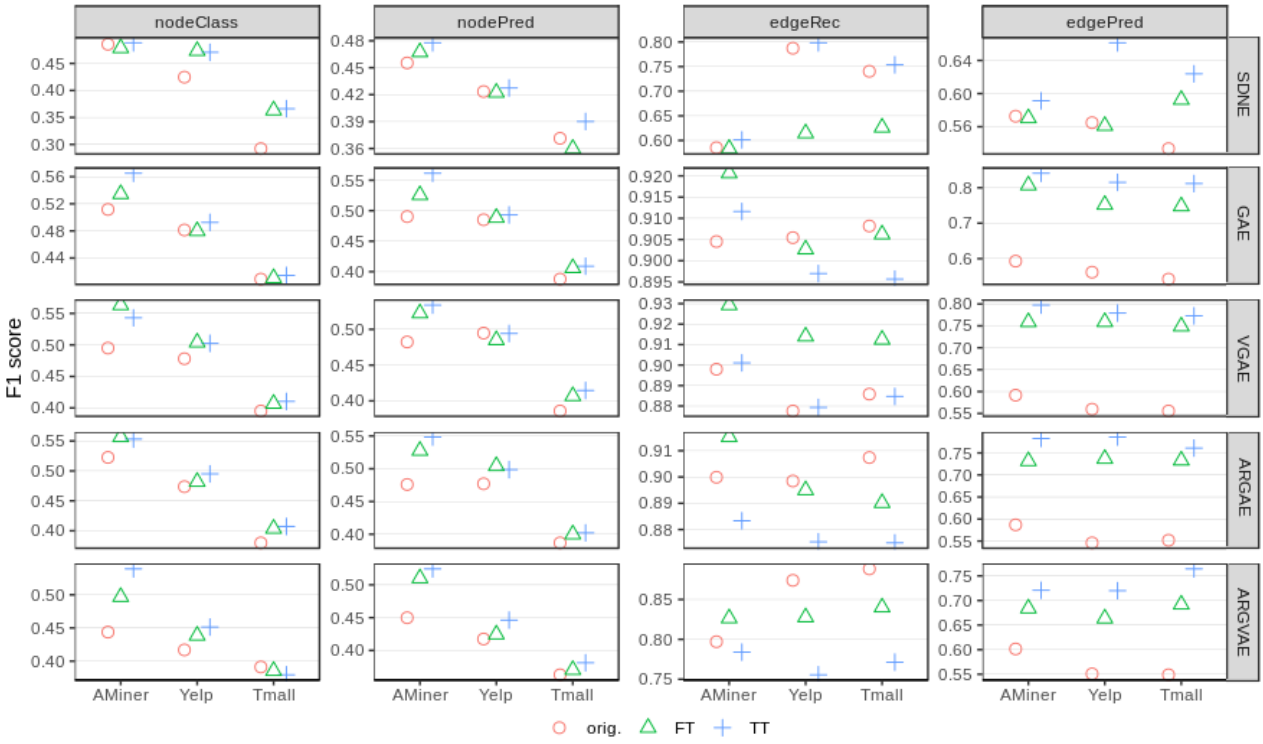


Fig. 2: Improvements induced by the autoencoders temporalization

used for other purposes than their conception scope. Also, one possible explanation for the small difference between the FT and TT variants performances is that the F1 score is already high (generally above 0.9 for edge-related tasks).

C. Comparison to temporal baseline methods

In order to challenge our temporalization method, we compare the performances of the temporalized autoencoders to the temporal embedding approaches according to the setting described in section III-B. Table I summarises the obtained results: for each method, dataset, and inference task, the embeddings giving the best performances are reported.

The most important remark concerns the temporalized version of N2V. Indeed, N2V_TT outperforms all the other methods (including the temporal ones), for all the datasets and tasks. DW_TT has also good performances, especially for AMiner and Yelp datasets. This corroborates the observations presented in section IV-B concerning the effectiveness of the learned temporal weights. On another note, the temporalized autoencoders have dissimilar results: For example, GAE_TT gives superior results comparing to DT on node-related tasks and better scores than TN2V on edge-related tasks. On the other hand, SDNE_TT's results are relatively low, although temporalization has enhanced them.

It should be noted that, as described in the experimental setting in section III-A, we use the popular default values for the autoencoders hyperparameters, contrary to DT, TN2V, DW_TT, and N2V_TT: For the sake of simplicity, only the

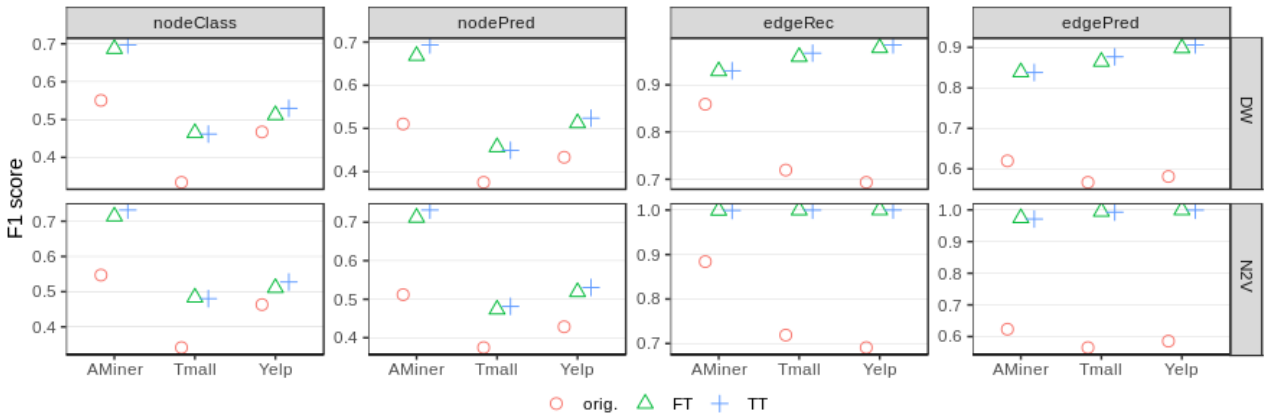


Fig. 3: DW and N2V temporalization using trained and fixed temporal weights

Method	AMiner				Yelp				Tmall			
	nc	np	er	ep	nc	np	er	ep	nc	np	er	ep
SDNE_TT	0.49	0.48	0.601	0.591	0.47	0.43	0.798	0.661	0.37	0.39	0.754	0.624
GAE_TT	0.57	0.56	0.912	0.841	0.49	0.49	0.897	0.815	0.41	0.41	0.896	0.812
VGAE_TT	0.54	0.53	0.901	0.797	0.50	0.49	0.879	0.779	0.41	0.41	0.885	0.773
ARGAE_TT	0.55	0.55	0.883	0.783	0.50	0.50	0.875	0.786	0.41	0.40	0.875	0.761
ARGVAE_TT	0.54	0.52	0.784	0.721	0.45	0.45	0.756	0.720	0.38	0.38	0.771	0.764
DT	0.56	0.54	**0.997**	**0.927**	0.48	0.45	0.979	**0.957**	0.37	0.37	**0.988**	**0.931**
TN2V	0.61	0.60	0.886	0.747	**0.53**	0.49	0.871	0.813	**0.52**	**0.48**	0.883	0.800
DW_TT	**0.70**	**0.69**	0.930	0.838	**0.53**	**0.52**	**0.985**	0.906	0.46	0.45	0.967	0.877
N2V_TT	**0.73**	**0.73**	**0.999**	**0.971**	**0.53**	**0.53**	**0.999**	**0.999**	**0.48**	**0.48**	**0.999**	**0.993**

TABLE I: Temporalized autoencoders vs. temporal embedding models. nc, np, er, and ep respectively stand for node classification, node class prediction, edge reconstruction, and edge prediction.

temporalization hyperparameters s and w are varied. Consequently, it is likely to improve the temporalized autoencoders scores by varying their hyperparameters.

D. Temporalization hyperparameters analysis

Thereafter, we analyze the impact of the temporalization hyperparameters on the inference scores (figure 4). Concerning the hyperparameter s controlling the supra-adjacency matrix symmetry, we can see that, apart from the edge reconstruction task, considering undirected temporal edges is more advantageous. One possible reason justifying this finding may be the need of forcing strong temporal continuity to achieve better performances, as explained in section II-A1. On another note, regarding the temporal window, we notice that, overall, larger w values lead to improved inference scores. This might be due to the possibility of building sophisticated temporal evolution patterns with large values of w, as shown in section II-A2.

E. Temporal weights analysis

Finally, we take interest in the interpretation of the temporal weights. The idea is to explain the learned value of a temporal weight regarding the characteristics of the supra-nodes it connects. Given a supra-edge between $\{v,\ t_i\}$ and $\{v,\ t_j\}$, we define 3 explanatory features:

- Adjacency change ($|\Delta A|$): The euclidean distance between the adjacency vector of v at the timestep t_i and the one at t_j. $|\Delta A| = ||Adj(v_j) - Adj(v_i)||$
- Sum weights change (ΔSW): The difference between the sum of the weights of v edges at the timesteps t_i and t_j. $\Delta SW = \sum weights(v_j) - \sum weights(v_i)$
- Sum weights sum (SWS): The sum of the weights of all the edges v has in the timesteps t_i and t_j. $SWS = \sum weights(v_j) + \sum weights(v_i)$

That being set, we compute the Spearman partial correlation between the target variable (i.e. the learned temporal weights) and the explanatory features. Figure 5 shows the results of this experiment where we represent the correlations as well as their corresponding p-values. There are different observations one can make. First, we notice that the correlations are different

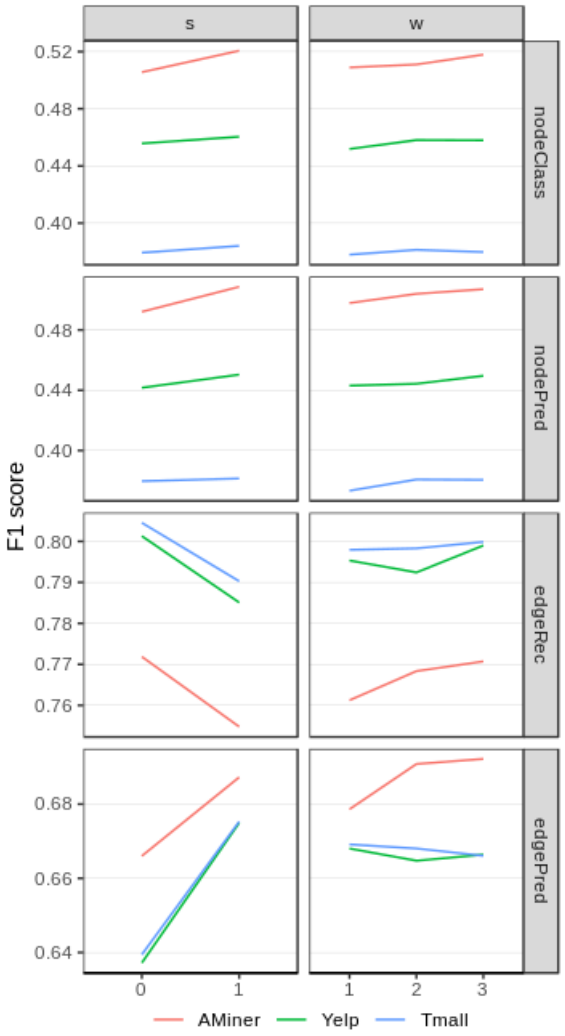


Fig. 4: The impact of the temporalization hyperparameters s and w. For the impact evaluation of s (respectively w), the best inference score for $w \in \{1, 2, 3\}$ (respectively for $s \in \{0, 1\}$) is reported.

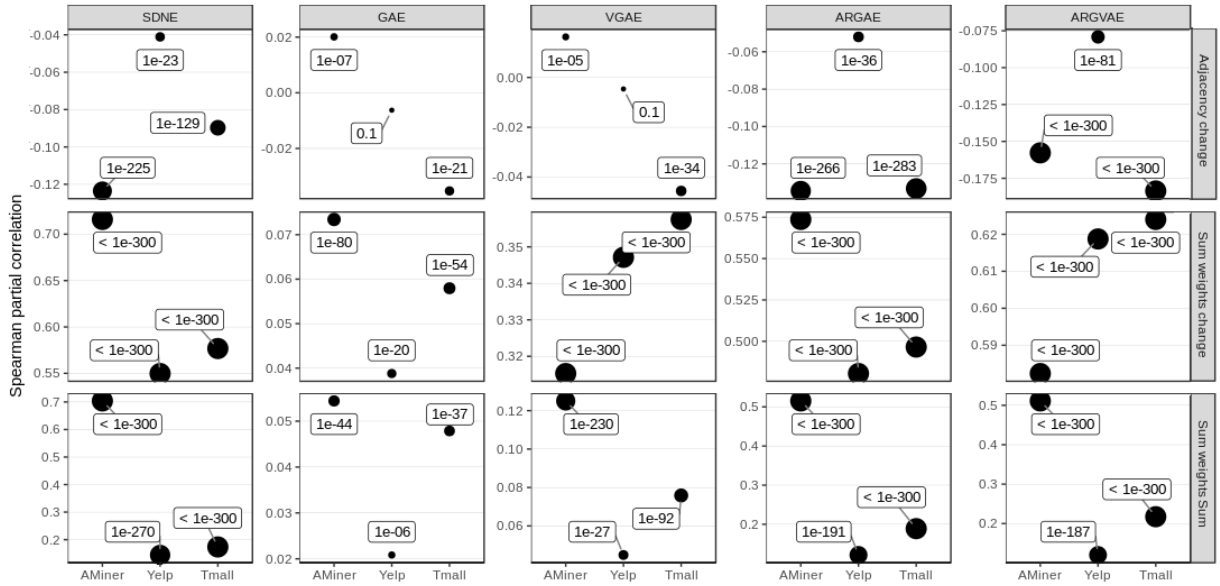
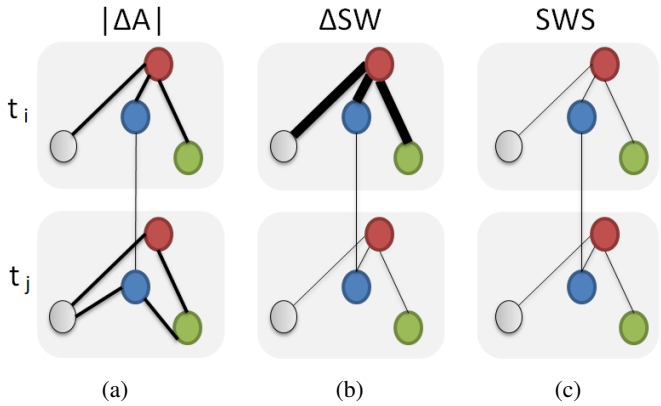


Fig. 5: Temporal weights analysis: Spearman partial correlations with p-values between the learned temporal weights and the defined nodes explanatory features. The smaller the p-values, the larger the size of the points (the values are in boxes).



(a) (b) (c)

Fig. 6: Situations which tend to give low temporal weight (represented by a thin line): (a) neighborhood modifications, (b) node's weights decrease, and (c) node's weights weakness.

between, on one side, SDNE, ARGAE, and ARGVAE, and, on the other, GAE and VGAE.

- For SDNE, ARGAE, and ARGVAE, we remark that ΔSW and SWS have globally a strong positive correlation with the learned temporal weights: partial correlations superior to 0.31, with a p-value inferior to 10^{-187}. Also, there is a less pronounced negative correlation between |ΔA|and the temporal weights (around -0.1), with p-values inferior to 10^{-23}. In other words, these correlations mean that, a node v that changes in terms of adjacency (i.e. neighborhood) between two timesteps t_i and t_j, or which weights globally get reduced from t_i to t_j, or which weights are relatively weak at t_i and t_j, will generally have a relatively small temporal weight between its supra-nodes at t_i and t_j. Figure 6 summarises these different scenarios.
- Concerning GAE and VGAE, it seems that the observations above, made on the other temporalized autoencoders, are still appropriate, but with some noticeable anomalies. First, the partial correlations between |ΔA|and the temporal weights on AMiner have an opposite sign as they are positive. Also, for ΔSW and SWS, the partial correlations are significantly smaller compared to the 3 other autoencoders. Moreover, the p-values of the partial correlation for the explanatory features are less marked. Further investigation is needed to explain these anomalies concerning GAE and VGAE partial correlations.

Regarding these findings, an in-depth analysis aiming to discover the correlations between various nodes features and the learned temporal weights could result into a fast training and more generic temporalization method. As a matter of fact, training a temporalized autencoder is a relatively heavy operation as, on the considered datasets, it takes approximately 90 minutes on a 24 cores machine with 64 GB of RAM. Also, if its benefits seem to be clear on the autoencoders that are being temporalized, reusing the temporal weights on other static embedding methods brings minor improvements as shown in section IV-B. Consequently, it is possible to consider designing a user-defined method that creates temporal weights based on nodes' explanatory features, with respect to the discovered partial correlations. Such a process would be a lightweight method to temporalize any static embedding approach (in a similar fashion to DW_TT and N2V_TT) with no constraints on its structure or its input type, unlike our proposed temporalization method. Further work about this possibility is underway.

V. Conclusion and Discussion

In this paper, we presented a graph autoencoder temporalization method that adapts static embedding methods to the

case of temporal networks. To that end, we employ and modify the concept of supra-adjacency matrices to make part of its entries trainable, i.e. the temporal weights, encoding the temporal relationship between a node and itself at different timesteps. We adapt and slightly modify autoencoders' structure to learn the optimal weights to be assigned to the temporal edges. By proceeding in this way, the temporalized autoencoders produce significantly more accurate embeddings regarding different node-related and edge-related inference tasks. We also show that the usefulness of the learned temporal weights goes beyond the scope of the autoencoder they have been conceived for, as they can be employed to temporalize other static embedding methods and greatly enhance their performances. Finally, we examine the impact of the temporalization hyperparameters and analyze the correlations between the learned temporal weights and some nodes' temporal characteristics.

In this work, it is useful to recall that the temporalized autoencoder is in itself to be considered as a parameter of the presented temporalization method. This means that one can adapt our process to any neural network autoencoder that takes adjacency matrices in input. Also, if the considered autoencoder supports some additional node features, then the resulting temporalized version also does. This is for example the case with GAE, VGAE, ARGAE, and ARGVAE. Furthermore, it can handle temporal node features. Finally, it is worthwhile to mention that our temporalization process can be adapted to other graph neural networks that address problems different than representation learning. For example, our method is suitable to temporalize graph generation neural networks such as [6], [27].

REFERENCES

[1] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Nips*, volume 14, pages 585–591, 2001.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[3] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900, 2015.

[5] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[6] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[7] Manlio De Domenico, Albert Solé-Ribalta, Emanuele Cozzo, Mikko Kivelä, Yamir Moreno, Mason A Porter, Sergio Gómez, and Alex Arenas. Mathematical formulation of multilayer networks. *Physical Review X*, 3(4):041022, 2013.

[8] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[9] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[10] Furkan Gürsoy, Mounir Haddad, and Cécile Bothorel. Alignment and stability of embeddings: measurement and inference improvement. *arXiv preprint arXiv:2101.07251*, 2021.

[11] Mounir Haddad, Cécile Bothorel, Philippe Lenca, and Dominique Bedart. Temporalnode2vec: Temporal node embedding in temporal networks. In *International Conference on Complex Networks and Their Applications*, pages 891–902. Springer, 2019.

[12] Chengbin Hou, Han Zhang, Shan He, and Ke Tang. Glodyne: Global topology preserving dynamic network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[13] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[14] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[15] Lin Li and William M Campbell. Matching community structure across online social networks. *arXiv preprint arXiv:1608.01373*, 2016.

[16] Sedigheh Mahdavi, Shima Khoshraftar, and Aijun An. dynnode2vec: Scalable dynamic network embedding. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3762–3765. IEEE, 2018.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[18] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, pages 969–976, 2018.

[19] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114, 2016.

[20] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, 2018.

[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[22] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[23] Koya Sato, Mizuki Oka, Alain Barrat, and Ciro Cattuto. Dyane: dynamics-aware node embedding for temporal networks. *arXiv preprint arXiv:1909.05976*, 2019.

[24] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998, 2008.

[25] Eugenio Valdano, Luca Ferreri, Chiara Poletto, and Vittoria Colizza. Analytical computation of the epidemic threshold on temporal networks. *Physical Review X*, 5(2):021005, 2015.

[26] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[27] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, pages 5708–5717. PMLR, 2018.

[28] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[29] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. Embedding temporal network via neighborhood formation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2857–2866, 2018.