MODARN-GENT — GENERATIVE MODULAR AUTOREGRESSIVE NETWORKS FOR MEDICAL TIME SERIES

Mihai David
MSc optional project report
July 2023
EPFL
mihai.david@epfl.ch

Mary-Anne Hartley & Hojat Karami
Supervision
Intelligent Global Health Research Group
EPFL
mary-anne.hartley@epfl.ch
hojat.karami@epfl.ch

ABSTRACT

BACKGROUND. In medicine, patients are frequently observed over fine-grained time frames, and variations in repeated measures are crucial to determining how well the patient is responding to therapy. However, it might be difficult to gather and manage granular time series data representing a patient stay in the hospital, not only because confidentiality limits data exchange between institutions or third parties, but also because of the data's persistent missingness. Targeted monitoring, in which the presence or intensity of measurements is predictive of specific outcomes like severity, produces this missingness known as MNAR (missingness-not-at-random). Although various techniques have been created to create synthetic times series, many of them fail to get beyond this crucial limitation.
AIM. This work proposes MoDNARN-GenT, a modular neural network framework able to learn from time series data that suffers from systematic missingness with the aim of generating simulated time series data in the form of EHRs (electronic health records).
METHODS/FINDINGS. We use both a custom toy dataset with 400 records and 5 time series features and a subset of the most popular publicly available EHRs (electronic health records) dataset, subset comprising of 6926 records and 14 time series features. We redesign the existing MoDN[1] (Modular Clinical Decision Support Networks) architecture to take as input and generate time series data. Through multiple versions of the toy dataset we investigate how different levels of missingness may hinder the network from learning and generating data and present its current limitations for working with a particularly complex real dataset.
CONCLUSION. We show the power of MoDARN-GenT to work with time series data as it yields good results in both learning and generating time series from data without missingness. Even though the modularization allows the model to learn from a flexible number and combination of inputs without being biased by the systematic missingness, we conclude that the amount of missing data has a negative impact on the overall learning performance.

Keywords Modular Decision Networks · Autoregressive Model · Timeseries · Electronic Health Records

1 Background

Medical data is a scarce data source as it is rarely publicly available or difficult to collect from institutions or third-party medical partners. Even if there is access to medical records, the concerns over privacy, ethical and legal constraints, ownership, intellectual property, and the difficulty of guaranteeing anonymization arise [2].

Moreover, medical data suffers from biased missingness known as MNAR (missingness-not-at-random). This is a particular issue for time series data, where more severe patients are subject to more intense monitoring and, hence, data acquisition. If this data is used in traditional machine learning models, it may learn to use missingness patterns to predict specific outcomes. This can lead to major failure when data is missing for independent reasons, such as resource limitations, which is a highly probable scenario. Therefore, the need of generating privacy preserving synthetic medical data arises.

A lot of work has been put into developing methods that create artificial medical datasets by learning from existing data, especially in the past 5 years [3]. From statistical and probabilistic models that attempt to simulate the real data, to more advanced machine learning and deep learning techniques like logistic regression or auto-encoders, the input data usually comprises of time-agnostic categorical or numerical variables. The latest data generators are based on Generative Adversarial Networks (GANs), with PATE-GAN [4], DP-GAN [5], actGAN [6] and SynTEG [7] being only a few of the promising solutions. Even though some of them ([5], [7]) can deal with time series as input, they are limited by the need for large amounts of data and show difficulties in handling data missingness.

MoDN[1] was proposed as a robust solution to the systematic data missingness. The modular network sequentially encodes individual features into shared vector representations, ignoring the lack of data for specific patients, while being able to make multiple predictions of diagnoses that are updated at each step of a consultation. However, MoDN was designed to work with tabular, time-agnostic data which may not fully reflect the evolution of a patient health status throughout an ICU stay with a low enough granularity.

To address this issue, we introduce ModARN-GenT, a medical time series data generator based on an adapted version of Modular Clinical Support Decision Networks (MoDN). By adapting the original implementation of MoDN, we show the network ability to learn different time series patterns in order to generate synthetic time series data, while dealing with systematic data missingness. We provide an insight into the network limitations regarding computational complexity and high amount of missing data.

2 Aim and Objectives

The aim of the project is to generate synthetic time dependent electronic health records using a custom model based on MoDN and assess the synthetic data quality. We split the work in the following objectives:

1. **Objective 1.** Review the literature on medical time series generation (Section 3)
2. **Objective 2.** Create a toy dataset (Section 4.1)
3. **Objective 3.** Adapt MoDN to work with time series data (Section 4.4)
4. **Objective 4.** Test the model capacity to learn and generate data from toy datasets with various missingness patterns (Section 5.1)
5. **Objective 4.** Use a preprocessing pipeline to build a subset from the MIMIC[8] dataset (Section 4.2)
6. **Objective 5.** Test the model capacity to learn and generate data from real world data (Section 5.2)

3 Related work

3.1 Existing data generators

The best comprehensive guide for existing synthetic data generators for tabular health records is done by Hernandez et al. in their work [3]. After a careful selection and filtering, they include 34 publications in their systematic review. The publications cover a very wide range of methods from baseline or statistical methods and machine learning models to more sophisticated deep learning approaches like auto-encoders (AE), Generative Adversarial Networks (GANs) or Ensembles. They even cover some procedures or frameworks that are composed of several steps or modules, including Aten Framework [9] and Synthea [10].

3.2 Processing pipeline for MIMIC

MIMIC is a popular, public, and free EHR dataset in a raw format that has been used in numerous studies. However, due to its cumbersome complexity, the absence of standardized preprocessing pipeline can be a significant barrier to tackle this rare resource. Fortunately, Gupta et al. [11], developed a greatly customizable pipeline to extract, clean, and preprocess the data available in the fourth version of the MIMIC dataset (MIMIC-IV). The MIMIC subset used in our work will be created using a custom version of this data pipeline.

3.3 Contributions

This work proposes to adapt MoDN, which can handle data missingness by design and does not need any type of imputation thanks to its modularity. Thus, we decompose time series data into multiple time blocks that include a varied number of available features that are fed sequentially into neural network encoders modules. Additionally, the network is able to make multiple predictions of diagnoses and features after each time block, values which will be incorporated into the shared vector representation and used in generating the data for the next time step, in the manner of autoregressive networks.

Our contribution includes:

- Redesigning MoDN to work with time series
- Exploring the ability of the model to learn meaningful representation from time series data.
- Testing the model capacity to generate time series data from toy datasets with various patterns of missingness.
- Applying the model to a real world benchmark dataset, MIMIC, which has notoriously challenging missingness patterns and biased time series.

4 Methods

4.1 Toy Dataset

Firstly, we build a toy dataset that consists of 400 records. Each record is characterized by 3 static features, 5 time series features with 10 time steps, and a binary label. Static features include gender (Male or Female), insurance (3 categories) and age (integer value between 18 and 99). The time series features are depicted in Figure 1, and their patterns were chosen after inspecting the trend of the features in the MIMIC dataset presented in the next section.

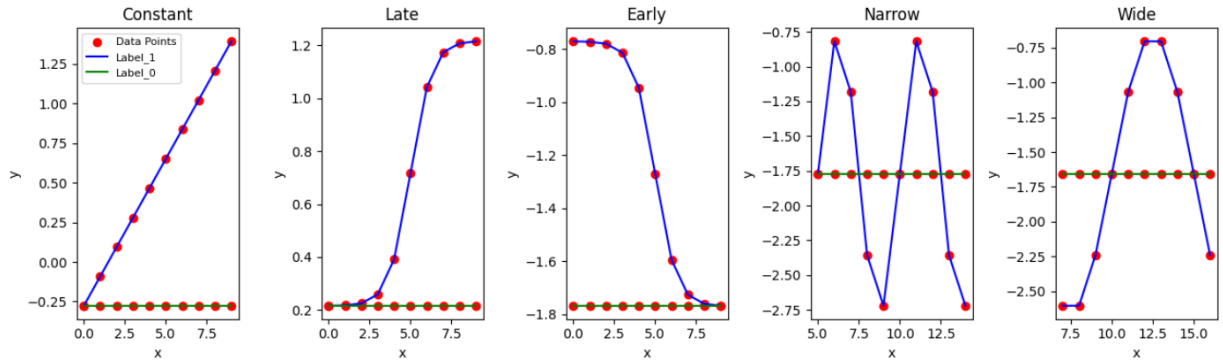


Figure 1: Time series variables of the toy dataset

For each of the time series we keep a constant value for the negative label (0) and a specific pattern for the positive label (1), as shown above. Next we will describe the characteristics of each time series:

- *Constant feature*: We employ a specific slope for the records with a positive label. For each patient we randomize the starting point on the y axis and the slope of the line of positive patients. Moreover, we make this feature, only, dependent on gender, meaning that for males we generate data starting at a lower starting point (between -2 and 0) and with a lower slope (between 0 and 0.5), while for females we generate data at a higher starting point (between 0 and 2) and a higher slope in the case of positive labels (between 1.5 and 2).

- *Late feature*: We employ a sigmoid like fixed function, that has lower values in earlier time steps and higher values for later time steps. We randomize the starting point on the y axis, keeping everything else fixed. This feature does not depend on other static features.
- *Early feature*: We employ an inverse sigmoid like fixed function, that has lower values in later time steps and higher values in earlier time steps. We randomize the starting point on the y axis, keeping everything else fixed. This feature does not depend on other static features.
- *Narrow feature*: We employ a sinusoidal function with a higher frequency and a fixed amplitude for the positive cases. We randomize the phase of the signal and the starting point on the y axis (between -2 and 2) and sample 10 time steps from it. This feature does not depend on other static features.
- *Wide feature*: This feature differs from the Narrow feature only by a lower frequency signal.

The dataset is perfectly balanced in terms of label and gender, meaning that we have 200 positive records for which there are 100 males and 100 females and 200 negative records for which there are 100 males and 100 females. All the other static features are assigned uniformly at random for each record.

4.1.1 Data representation for training

We store the data under a dataframe formatted as in Figure 2. An entry in the dataset is considered to be a *consultation* of a patient. Each consultation has 11 time blocks: 10 for the time series and 1 for the static variables (annotated with a -1). Each time-block consists of multiple *questions* (or features) and their *answers* (or values) are stored in each cell of the resulting dataframe. Each answer of a patient will be accessed by a multi-index column key, composed from the time-block index and question name.

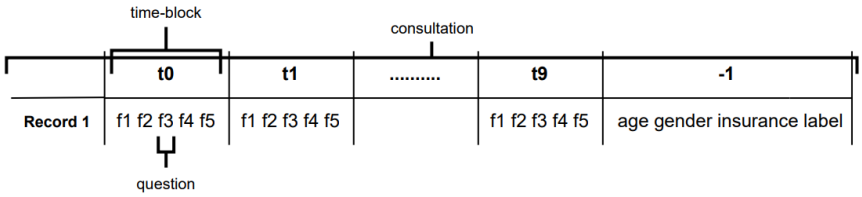


Figure 2: Data representation for the toy dataset

4.1.2 Preprocessing

The only preprocessing we apply is standardization of each continuous feature by using the statistics across all available time steps.

4.2 MIMIC dataset

MIMIC (Medical Information Mart for Intensive Care) [8] is the biggest, most popular, public, and free EHR dataset in a raw format that has been used in many recent studies as a benchmark dataset. It comprises de-identified EHRs from the patients admitted at Beth Israel Deaconess Medical Center in Boston, MA, USA, from the years 2008 to 2019. The data stored in over 30 tables contains information for each patient regarding hospital stays, including laboratory measurements, medications administered, and, vital signs. It consists of data for over 257,000 distinct patients, yielding 524,000 admission records. We will use the fourth version of this dataset as it is the most actual.

4.2.1 Preprocessing pipeline and subset definition

Due to MIMIC's size and complexity, we resort to using a custom preprocessing pipeline based on [11] for creating a meaningful subset. The main steps for preprocessing are presented in Figure 3.

We choose to work with ICU data from sub-version v1.0 of the dataset. We choose mortality of patients in the next 48 hours as the general binary label or outcome. We do not filter patients based on a specific condition and consider all records, regardless of previous diseases. One limitation is that with the current data pipeline, one can only choose one specific outcome/disease as the label. We consider as features only laboratory measurements (*lab events*), which include hematology measurements, blood gases, chemistry panels, and less common tests such as genetic assays, and demographics. An important note is that a row represents a patient stay in the hospital (or consultation, as we depict in Figure 2) which does not directly map to a patient admission as one could be hospitalized many times. However, we

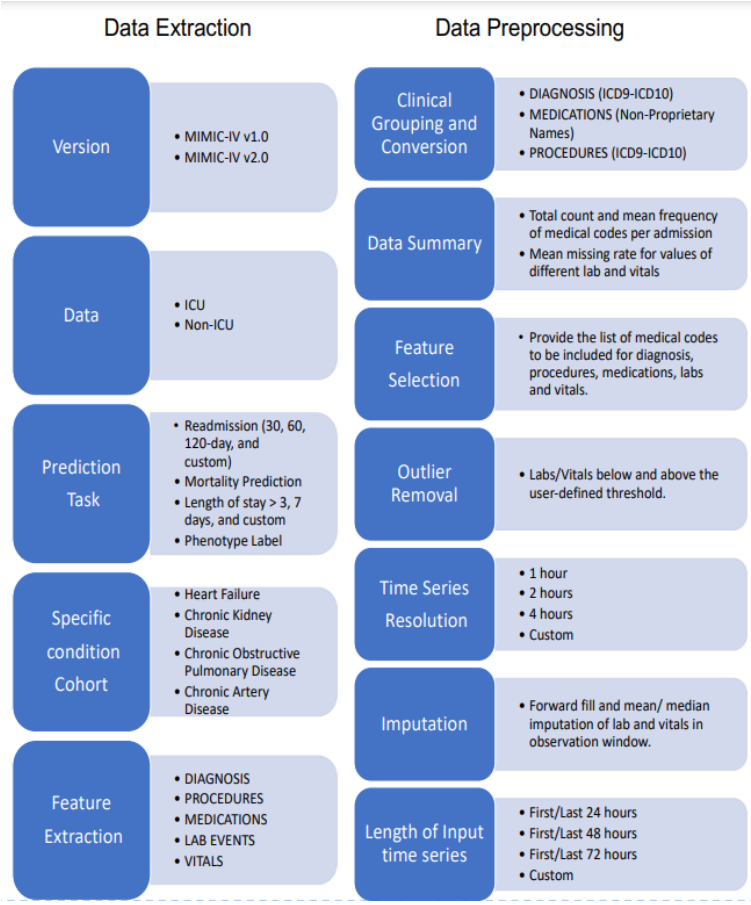


Figure 3: MIMIC preprocessing steps [11]

consider a new stay as a new patient. We perform outlier removal and only keep data that fits between 2nd and 98th percentiles.

Next, we choose our time series representation. We consider a time frame of the last 48 hours of a patient stay in the hospital. The time series resolution is fixed to 1 hour and we do not use any imputation techniques or further time-window aggregations. We also perform undersampling of the majority class as there are enough records to not affect the dataset size, resulting in a perfectly balanced dataset in terms of label.

In addition to the main data pipeline, we filter out a feature if there are more then 95% patients missing it. At this point we are left with 108 continuous features and 6941 stays or rows. Finally, we keep only 14 continuous features that have a lower missingness and 4 demographics representing static features (age, insurance, gender, ethnicity) whose distributions can be seen in Figure 4. We also remove malformed rows that only contain missing data, remaining with 6926 records.

We standardize each of the continuous features by using the statistics across all available time steps, as in the toy dataset.

4.2.2 Data representation for training

The data is stored in a dataframe exactly in the same format as for the toy dataset (see Figure 2). The only difference is that there will be 48 time blocks with 14 time series features each, and 4 static variables together with the label in the last block annotated with -1.

4.2.3 Data missingness

Even though we managed to extract a meaningful subset from the original MIMIC dataset, we are still facing with a very high degree of missingness, a major drawback of EHRs. In Figure 5 we present the missingness for the *Chloride(serum)* feature across the 48 time steps. The average missingness for this feature is 91%. All the other features have a very

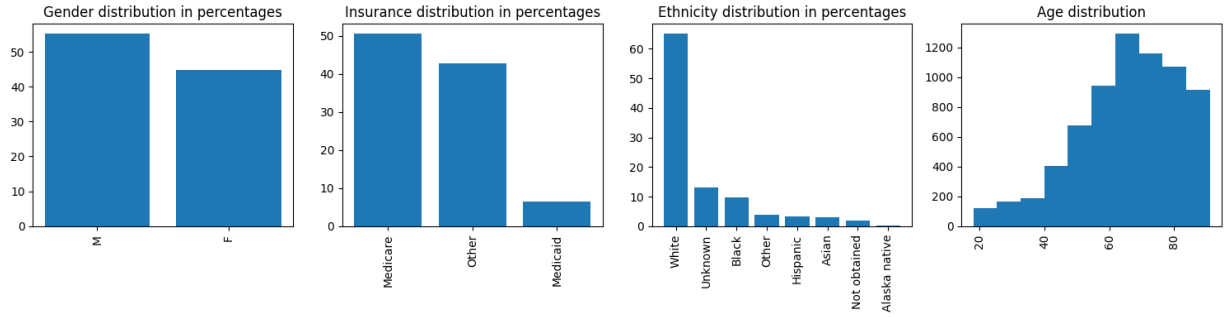


Figure 4: Static features distributions for the balanced MIMIC subset

similar pattern for the missing data across time and an average missingness between 91% and 94%. Overall, there is a 92% missingness across all time steps and all time series features in this MIMIC subset. In Figure 6 we show the average missingness across time, for all the features, per label. This plot confirms that negative cases have a higher missingness as they require lower attention, thus less features measured during a stay.

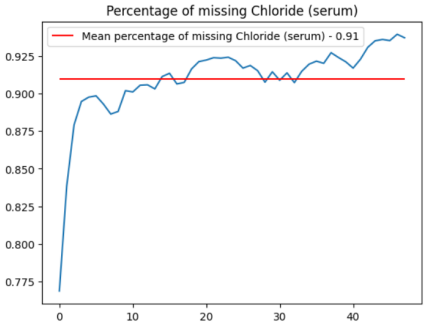


Figure 5: Missingness pattern for Chloride(serum)

Figure 6: Per label class feature missingness

4.3 Additional toy datasets

The initial version of the toy dataset does not contain any missing data. We further create 2 toy datasets that are subject to data missingness. The first version, called *toy_missing_91*, simulates the MIMIC subset missingness in a sense that we employ a similar percentage of missing data per feature. Therefore, we keep a similar pattern of missingness (see Figure 5), but across 10 time steps, with an average of 91% missing data, for each time series feature. The second version of the toy dataset with missing data is called *toy_missing_31* and compared to *toy_missing_91* it has an average of 31% missing time series data across all time steps, while keeping the same pattern of missingness across time.

4.4 Model architecture

4.4.1 Original MoDN with feature decoders

Designed by Trottet et al., Modular Clinical Support Decision Networks [1], is an architecture that allows to simulate a patient medical consultation by making multiple predictions of diagnoses and features, updated at each step of a consultation thanks to answers to the previous questions. The 3 key features of this network are its modularity, sequential processing of inputs an shared vector representation. When trained with feature decoders it can predict any variable from the original training set, whether it as a feature or a label, as all the features can pass through both encoders and decoders. In the original implementation all the feature decoders are applied on the initial state vector and after each of the features gets encoded into the state vector. All the features are static, thus their values are fixed and do not change with time. Figure 7 shows an overview of the network.

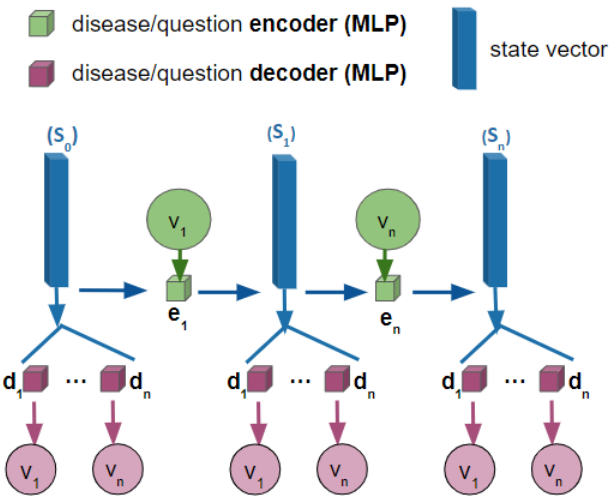


Figure 7: Original MoDN

4.4.2 *MoDN for time-series*

The main difference from the original MoDN implementation is that the we sequentially encode all the available features from a time block, before applying all the feature decoders on the resulted state vector. Thus, we use the entire information from a time block to predict each feature at each time step. An important note is that both the encoder and decoder of a feature are shared among all the time steps at which the feature is available. To increase the number of (value, target) pairs for the time series, during training, the target for a feature decoder will be chosen as the value from the next available time step for that feature. If there are no more available values in the future time steps, the decoder for the specific feature will not be used anymore.

Another key change is that the initial shared state vector is a random variable initialised, for each patient, exactly like the linear layers of the network (see 4.4.4). In the original implementation, the initial state vector was a learnable tensor. However, as the aim is to generate varied EHRs for patients that may have the same static information available, we need a source of randomness. A visual representation for a version that processes consultations with maximum 5 variables, spanning across 3 time steps is shown in Figure8, where v_5 is a static variable like Label or Gender as it is present at every time step.

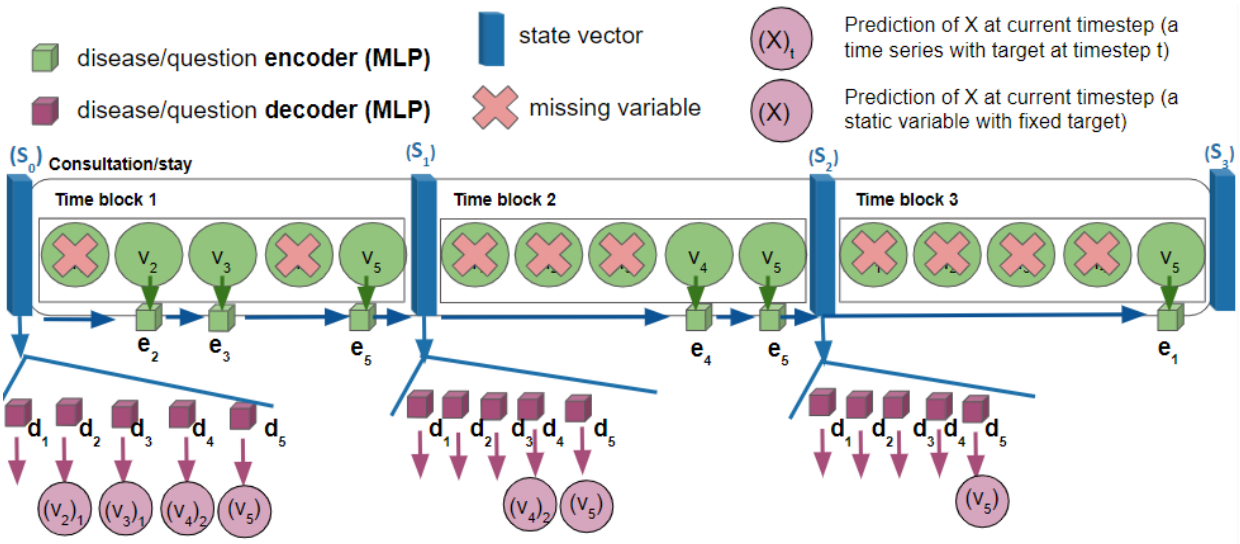


Figure 8: MoDN for time-series

4.4.3 *Autoregressive data generation process*

The data generation process takes place in the following manner: based on already available information about a patient, like demographics or diseases, we generate several synthetic time series measurements, simulating his stay in the hospital. For this reason, we use the static variables of each patient (together with the label) as the initial information. We consider that this information is in fact the available data in time block t_0 and encode it into the shared vector representation. Thus, we will start to generate data from t_1, by applying decoders for all features that are not part of the initial information at the end of each time block. The generated values at a particular time step will be used as actual information of the next time step, and encoded into the state vector. The process continues until data is generated for the last time step of the time series. For the generated static variables, the most frequent value, in case of categorical static variables, and the mean value, in case of continuous static variables (e.g. age) will be considered as final values for each consultations. A process is shown in Figure 9.

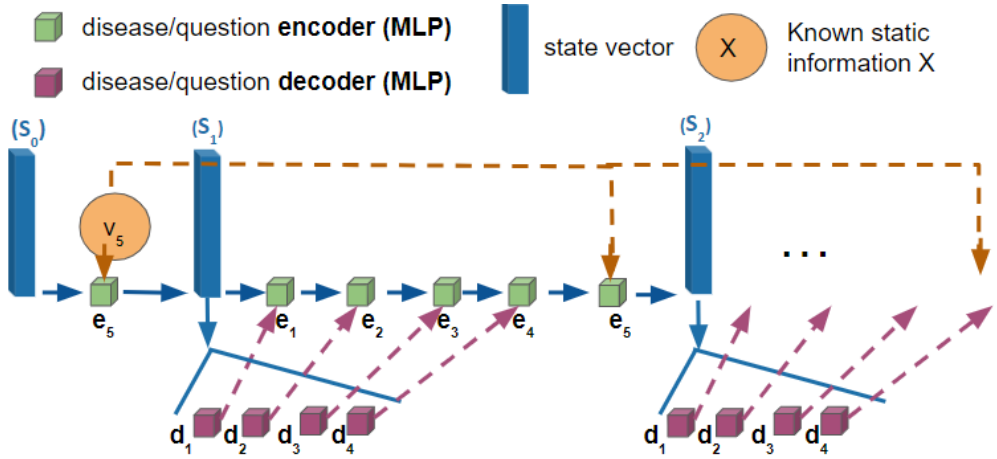


Figure 9: Data generation process

4.4.4 *Fixed hyperparameters*

To make the network trainable with time series as input we had to change the weight initialization of the linear layers from the encoders and decoders. The default initialization for the linear layers is Kaiming Uniform with a negative slope of the rectifier equal to $\sqrt{5}$ (from PyTorch documentation [12]). We chose a negative slope of the rectifier of $\sqrt{55}$ to make the initial weights much smaller and yield better performances. We keep this initialization for all the experiments conducted. Other hyperparameters that we keep fixed during all the experiments are the shared vector state size and learning rates for the encoders and static decoders. We set a state size of 60 and the same fixed learning rate of 1e-2 for all feature encoders and static decoders as in the original MoDN implementation.

4.5 Experiments

Prediction on 3 toy datasets We test the network's ability to learn and predict different custom-generated time series from 3 toy datasets with various degrees of missingness, more exactly 91%, 31% and 0% missing data. We use the process presented in section 4.4.2.

We randomly split 80% of the data to be the training set and the other 20% to be the validation set. A very important note is that the algorithm is trained on CPU only, while using the Adam optimizer with a full-SGD step, meaning that the parameters of the network get updated only after all patients were seen once.

During training we shuffle not only the patient but the features inside each time block. We keep the time window of the time series equal to the time series length. We tune the learning rates of each continuous decoder, ending up with 6 specific values (Age and 5 time series features). We let the network train for 2000 epochs using a step decay of the learning rate at 250 epochs. We use an early stopping mechanism with a patience of 1/4 the number of epochs. A crucial step in the experiment setup was to apply gradient norm clipping with a maximum norm of 1. We keep the same setup for all 3 versions of the toy dataset. The results of this experiment can be found in section 5.1.1.

Generation for toy dataset To generate data, we use the process presented in section 4.4.3. We load an already trained model and use it to generate patients similar to those in the validation set used during training. We define as

known information for each patient the 3 static variables present in the validation set. In this way we ensure that the network has the same initial information for each patient. The results of this experiment can be found in section 5.1.2.

Prediction on MIMIC subset We test the network's ability to learn and predict time-series data corresponding to real world data EHRs. This experiment setup is similar to the prediction experiment for the toy datasets.

One difference is that we restricted the maximum time window length to 24 time steps instead of the original 48 time steps available for this dataset. If a consultation includes data in more than 24 time blocks, we randomly choose maximum 24 blocks while keeping their order. This tweak ensured a more stable loss and a lower computational time. We let the network train for 3000 epochs using a step decay of the learning rate at 300 epoch. Even though we experimented with tuning the learning rates of the continuous decoders, we did not see any improvement due to the high amount of data missingness. Therefore, we finally kept all the learning rates of the time series decoders to a constant of 1e-9 in the code. We also clip the gradients norm to a maximum of 1. Observations about the results can be found at section 5.2.1.

Generation for MIMIC subset This experiment setup is exactly the same as for the toy dataset. The only difference is that we will use as known information for each patient, the 4 static features from the validation set of this dataset. Observations about the results can be found at section 5.2.2.

5 Results

5.1 Results for the toy datasets

We will show the model's capacity to learn, predict (5.1.1) and generate (5.1.2) time series for 3 types of toy datasets with various data missingness amounts. Please consult the appendix A for metrics comparison between the 3 datasets.

5.1.1 Prediction results

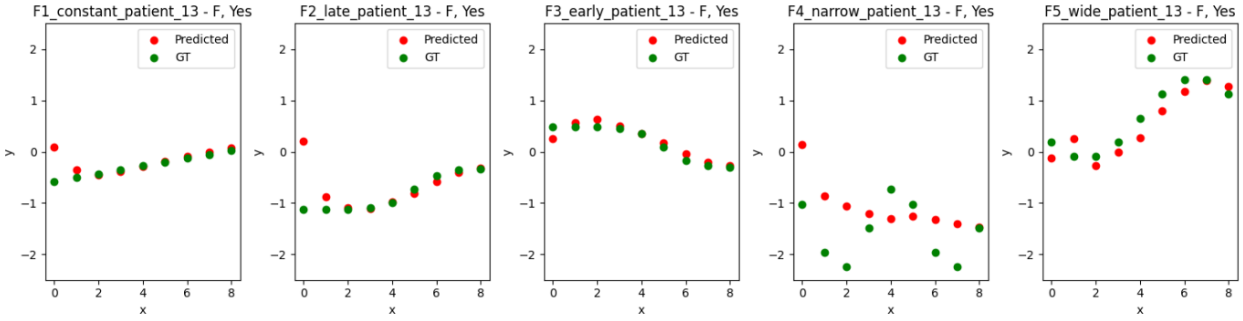
In Figure 10 we show the model's performance on the *toy* dataset. We sample 3 patients form the validation set and show the predicted versus ground-truth values for each time-series. At each time step, we use information encoded up until that point in time to predict the next value. An important note is that for time step 0, there is only random information from the initial state taken into consideration, therefore, the poor results for this time step. Among the static decoders at each time step there is the label decoder too. We plot blue dots for time steps where the ground-truth label differs from the predicted label. Usually the network mistakes the label only at t_0, or very rarely at t_1 Besides the Narrow feature for the positive cases, the model is able to capture the continuous patterns very well.

In Figure 11 we present the results for the *toy_missing_31* dataset for the same patients. Compared to the full toy dataset, the model begins to make worse predictions even for the constant patterns of the negative cases. Even if there is a good amount of missing data which can be seen in the plots, the model still captures the time series patterns for all the features, besides Narrow, relatively well.

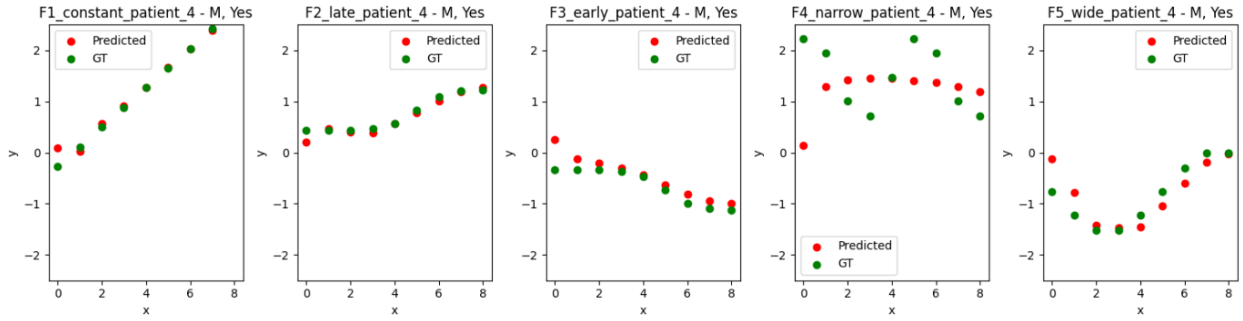
Due to a very high degree of missingness, the ground truth versus prediction plots won't be relevant for the *toy_missing_91* dataset. Therefore, in Section A of the appendix we present per time step metrics (RMSE or Macro-F1 score, depending on the feature) for each feature, at validation time. We also compare the metrics for this dataset with the other two datasets in order to show the performance decrease with more data missingness.

5.1.2 Generation results

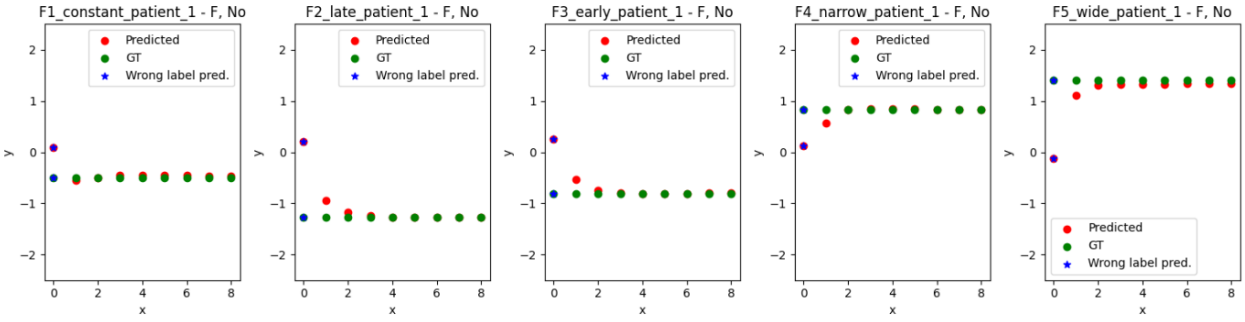
In Figure 12 and Figure 13 we show the data generation process results for the toy dataset. We restrict ourselves to this dataset only, as it provides a satisfactory overview of the generation process. Each time series is shown for all the 80 records in the validation set, the mean being represented in bold. The Constant feature is both label and gender dependent. Even if the network is able to predict well the different patterns for males and females 5.1.1, we observe, in plot (d), that the it encounters difficulties in generating the corresponding pattern for males. The model yields good performances for the Late and Early features. However, even if the initial state is randomised from patient to patient to ensure variance among generated EHRs, the generated data has a much lower variance than the ground truth data for all features. We observe the model's difficulties to generate more complex oscillations of the Narrow and Wide features. A solution might be to condition the generation process for each feature with prior (learnable) information to force the network to break the small variance barrier.



(a) Predictions for positive, female patient

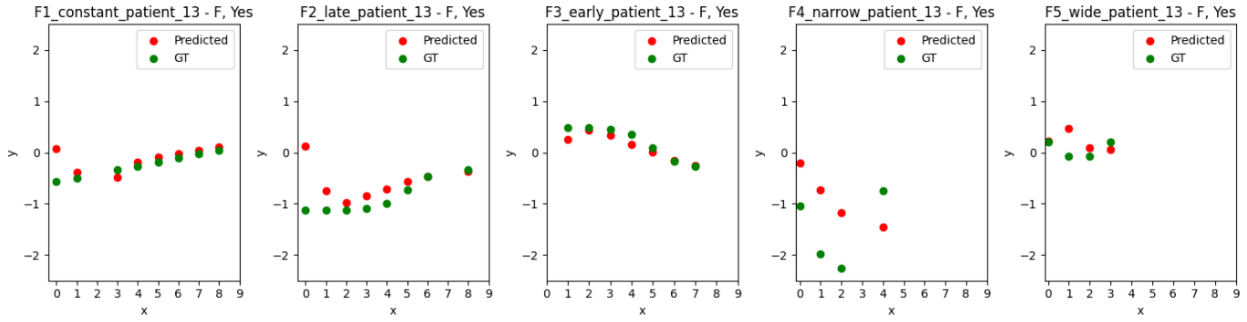


(b) Predictions for positive, male patient

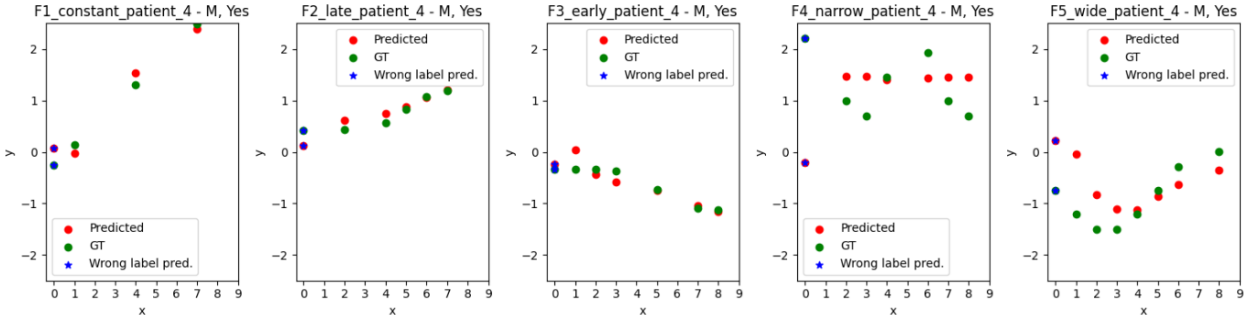


(c) Predictions for negative, female patient

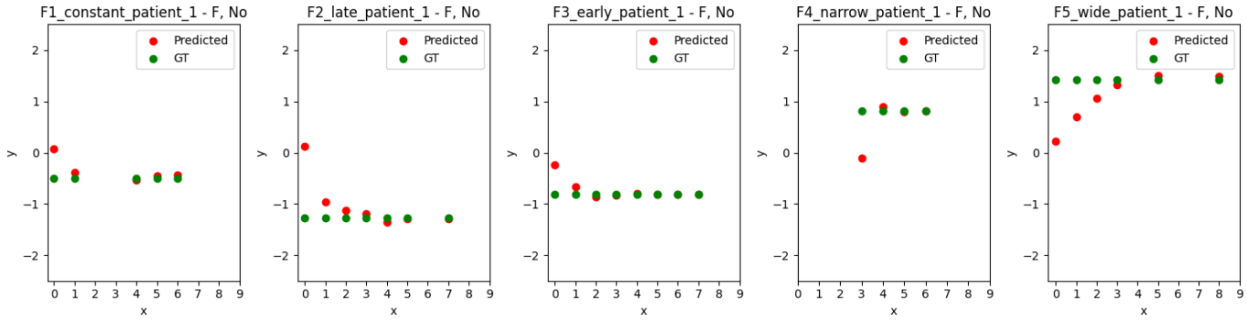
Figure 10: Time-series predictions vs. groud truth for the toy dataset



(a) Predictions for positive, female patient

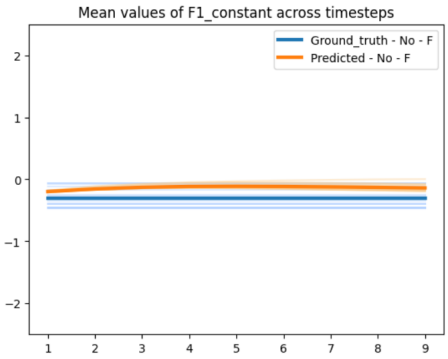


(b) Predictions for positive, male patient

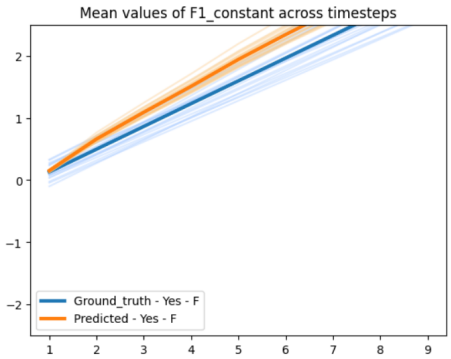


(c) Predictions for negative, female patient

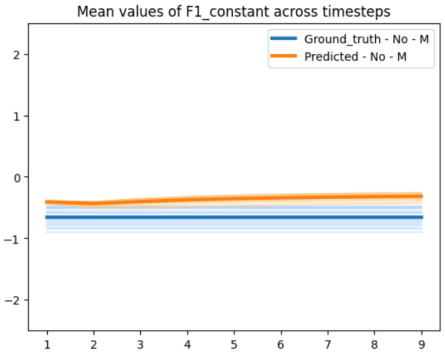
Figure 11: Time-series predictions vs. groud truth for the toy_missing_31 dataset



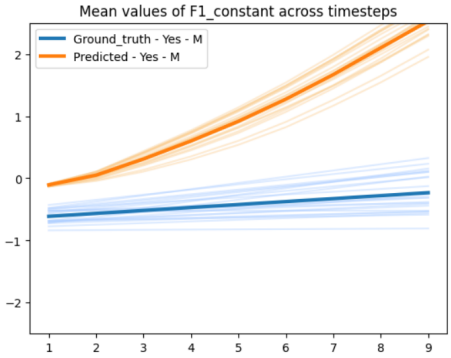
(a) Negative Female - Constant feature



(b) Positive Female - Constant feature



(c) Negative Male - Constant feature



(d) Positive Male - Constant feature

Figure 12: Entire ground truth data versus generated data for Constant feature of toy dataset (this feature is also gender dependent, not only label dependent, thus plots (a), (b), (c), (d) correspond to all possible combinations)

5.2 Results for the MIMIC subset

5.2.1 Prediction results

Due to a very high percentage of data misingness (for a consultation, only 1.2 features on average are available each time block) the ground truth data versus predicted data plots are not meaningful. However, in Section B of the appendix, we provide an insight into the network performance by plotting the metrics across time obtained on the validation set for 3 selected features. We conclude that a too high degree of missingness present in a real world EHRs dataset hinders the network to learn any meaningful information.

5.2.2 Generation results

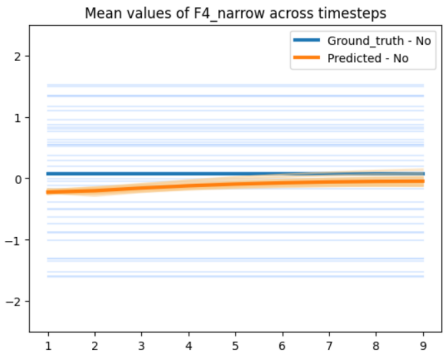
Unfortunately, all the generated time series were almost constant for all time series and label values as the network was not able to correctly learn and predict the patterns in the previous step due to a very high degree of missingness.

6 Discussion

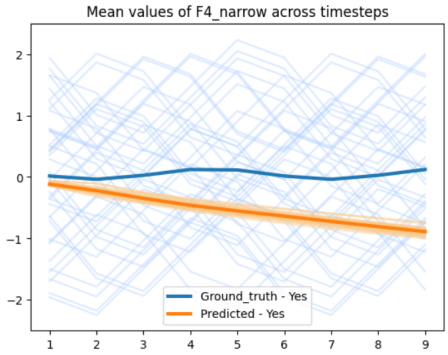
6.1 Limitations

For the current implementation, which was based on a simplified version of MoDN, the biggest drawback is the computational time. As the network uses only the CPU for computations, the training time for larger datasets, like the MIMIC subset, is very long. Even though we restricted the time window to maximum 24 time steps and used a remote cluster with increased CPU resources, we did not manage to fully train the network with the MIMIC subset, as it took more than a week. Breaking the full SGD step used in the current implementaion into a mini batch GD step, while using the parallelization features of the GPU, would solve this limitation.

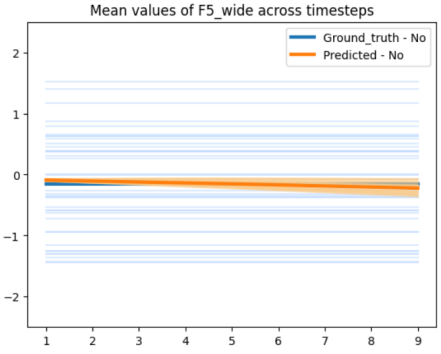
The hyperparameters space is quite wide and is directly proportional to the number of features. For example, the learning rates of each feature decoder need careful tuning so that the network yields the best performances. The more



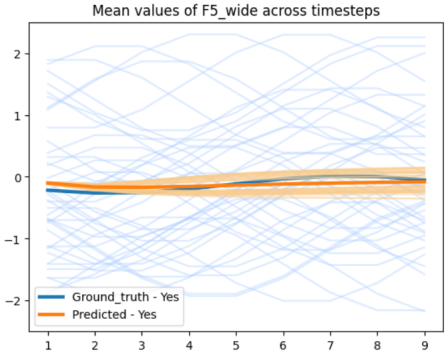
(a) Negative - Narrow feature



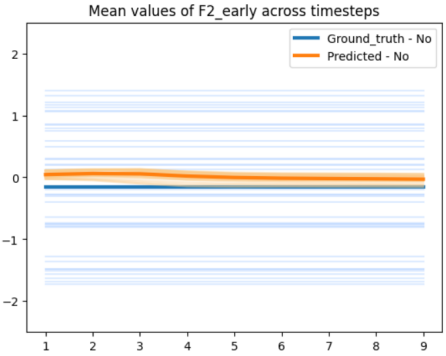
(b) Positive - Narrow feature



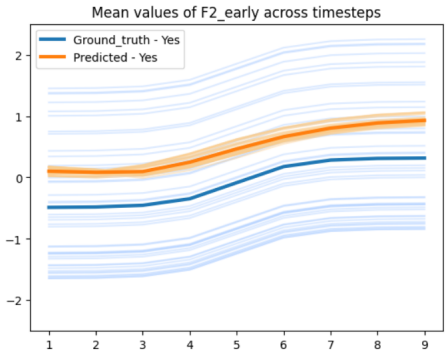
(c) Negative - Narrow feature



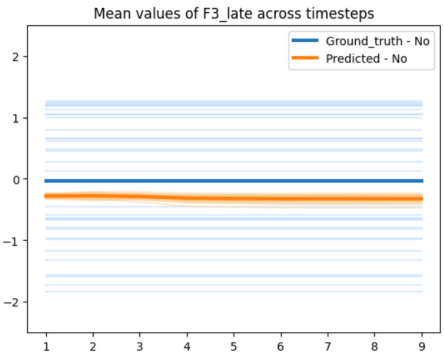
(d) Positive - Narrow feature



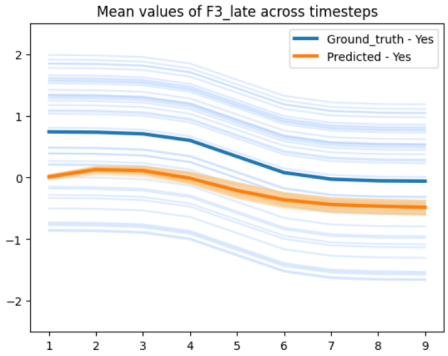
(e) Negative - Late feature



(f) Positive - Late feature



(g) Negative - Early feature



(h) Positive - Early feature

Figure 13: Entire ground truth data versus generated data for Late, Early, Narrow and Wide features of the toy dataset

features in the dataset, the more hyperparameters to tune. Even though we performed several experiments for choosing the most suitable hyperparameters for each dataset, an extensive tuning is definitely required to obtain better results.

The best model during training is currently chosen as the one that yields the lowest overall loss. Because the metrics are not necessarily correlated with the loss or because there may be a focus on the performance of a particular part of the network (e.g. the time series feature decoders), exploring other criteria for choosing the best model is advisable.

The entire data generation pipeline needs improvements. The rather simple autoregressive manner in which data is generated presents some limitations. To begin with, the worst feature predictions are made in the first time steps due to lack of information or a larger time window size. Also, the model uses the predicted values from the current time step, encodes them into the shared vector state, and use the resulting state to predict the features from the next time step. Because of the unreliable predictions in the initial time steps, the network may not be able to follow the real pattern of the time-series in the following time steps. Moreover, a robust evaluation of the generated data is not implemented yet.

6.2 Future work

As stated in Section 6.1, a GPU based implementation is needed. This will allow us to speed up the training through the parallelization used for the mini-batch gradient descent updates.

Thanks to the size of the original MIMIC dataset we allowed ourselves to perform an undersampling step of the majority class and have a perfectly balanced dataset. However, testing how the network performs with highly imbalanced datasets is part of the future work.

For evaluating the synthetic generated data we should compare the results from the network trained with synthetic data and tested on real data, with the results by training and testing on real data. Moreover, we should include privacy metrics when evaluating the synthetic data as it will be used both for enriching or complementing the available data and enabling secure and private sharing of the data. SynthCity [13] project offers a suite of metrics for this type of evaluation.

The modularity of the network provides the possibility of adding any type of new feature into the architecture of the network by adding a new specific encoder and the corresponding decoder. Thus, fine-tuning it with new data is a highly valuable feature. However, we did not implement the the fine-tuning mechanism in the code yet, as the features are pre-defined and fixed for each dataset.

Finally, we look forward to compare the synthetic data obtained by the current state-of-the-art GAN-based data generators with the synthetic data generated by ModARN-GenT by assessing their quality through various statistical tests and metrics available in the SynthCity project.

6.3 Conclusion

In this paper, we proposed ModARN-GenT, a MoDN based modular neural network framework that can deal with any type of input, including time series data, with the aim of learning the data patterns and generating synthetic time series data in the form of electronic health records. We explored its potential to deal with systematic data missingness but also showed its limitations in learning from data with a very high percentage of missingness. We proposed an autoregressive generative pipeline that yields satisfactory results, especially when there is no missing data. In conclusion, further development is definitely worth it for this high potential idea of building a medical data generator that is also able to deal with data missingness.

Acknowledgments

I want to express my gratitude to my two supervisors, Hojat Karami for his technical advice and Mary-Anne Hartley for her mentoring and encouragement during the project. I also like to thank Estelle Richard, who assisted me in getting started on this project because she has previously worked with MoDN to generate time series data.

References

[1] Cécile Trottet, Thijs Vogels, Kristina Keitel, Alexandra V Kulunkina, Rainer Tan, Ludovico Cobuccio, Martin Jaggi, and Mary-Anne Hartley. Modular clinical decision support networks (modn)—updatable, interpretable, and portable predictions for evolving clinical environments. *medRxiv*, 2022.

[2] Sharyl J Nass, Laura A Levit, and Lawrence O Gostin. *2 - The Value and Importance of Health Information Privacy*. National Academies Press (US), Washington (DC), 2009.

[3] Mikel Hernandez, Gorka Epelde, Ane Alberdi, Rodrigo Cilla, and Debbie Rankin. Synthetic data generation for tabular health records: A systematic review. *Neurocomputing*, 493:28–45, 2022.

[4] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2019.

[5] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network, 2018.

[6] Aki Koivu, Mikko Sairanen, Antti Airola, and Tapio Pahikkala. Synthetic minority oversampling of vital statistics data with generative adversarial networks. *Journal of the American Medical Informatics Association*, 27(11):1667–1674, 2020.

[7] Ziqi Zhang, Chao Yan, Thomas A Lasko, Jimeng Sun, and Bradley A Malin. SynTEG: a framework for temporal structured electronic health data simulation. *Journal of the American Medical Informatics Association*, 28(3):596–604, 11 2020.

[8] A. Johnson, L. Bulgarelli, T. Pollard, S. Horng, L. A. Celi, and R. Mark. Mimic-iv (version 2.2). *PhysioNet: https://doi.org/10.13026/6mm1-ek67*, 2023.

[9] Scott McLachlan, Kudakwashe Dube, Thomas Gallagher, Bridget Daley, and Jason Walonoski. The aten framework for creating the realistic synthetic electronic health record. 01 2018.

[10] Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association*, 25(3):230–238, 08 2017.

[11] Mehak Gupta, Brennan Gallamoza, Nicolas Cutrona, Pranjal Dhakal, Raphael Poulain, and Rahmatollah Beheshti. An Extensive Data Processing Pipeline for MIMIC-IV. In *Proceedings of the 2nd Machine Learning for Health symposium*, volume 193 of *Proceedings of Machine Learning Research*, pages 311–325. PMLR, 28 Nov 2022.

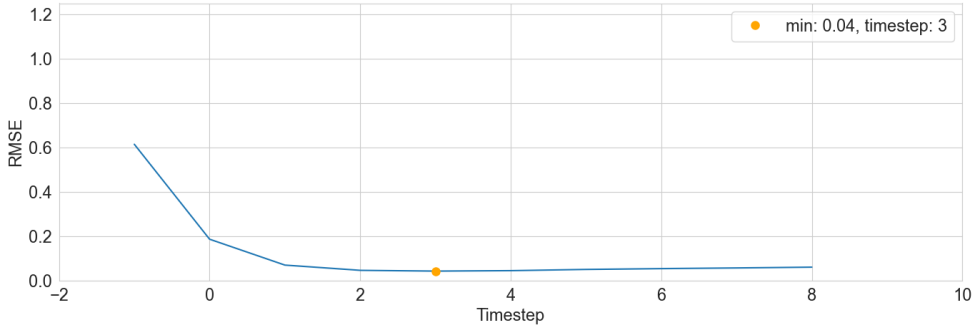
[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[13] Zhaozhi Qian, Bogdan-Constantin Cebere, and Mihaela van der Schaar. Synthcity: facilitating innovative use cases of synthetic data in different data modalities, 2023.

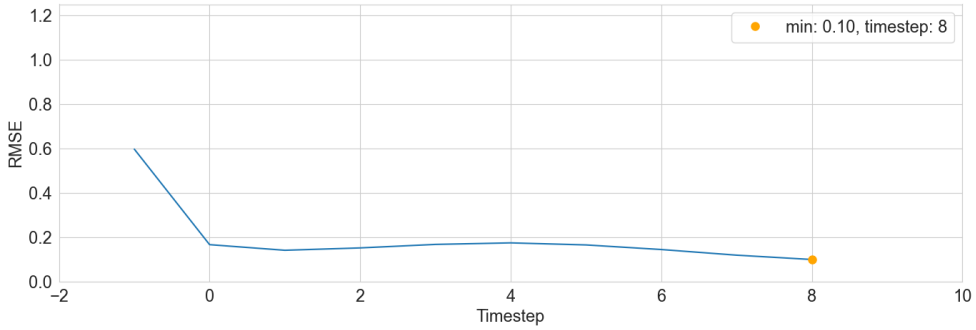
Appendices

A Toy datasets metrics appendix

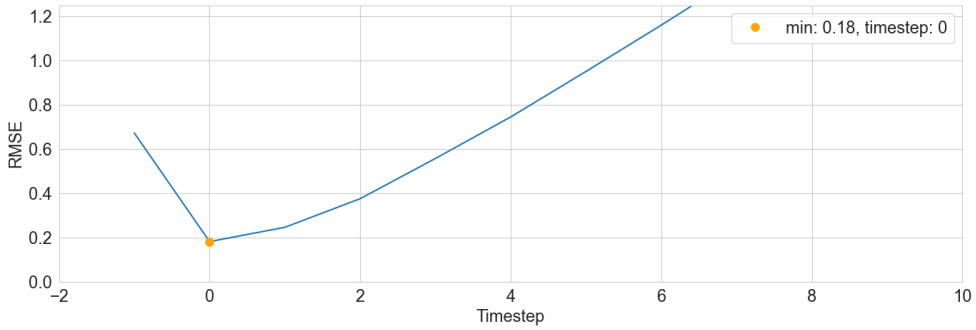
In this section we present the Root Mean Squared Error (RMSE), for continuous variables, and Macro-F1 score, for categorical variables, for 3 selected features, accordingly. The metrics are computed on the standardized validation sets corresponding to each of the 3 versions of the toy datasets. Time step t_{-1} represent the predictions made directly after the initial random state. There are no predictions for t_9, as at this time step there were no ground truth values from a further time step. Figure 14 shows how the RMSE is increasing with more data missingness, yielding very poor results for the toy dataset with an average missingness of 91%. The trend of the RMSE metric is decreasing as the network is supposed to learn better with more information about the time series incorporated into the shared vector representation at each time step.



(a) *Toy* dataset - Constant feature



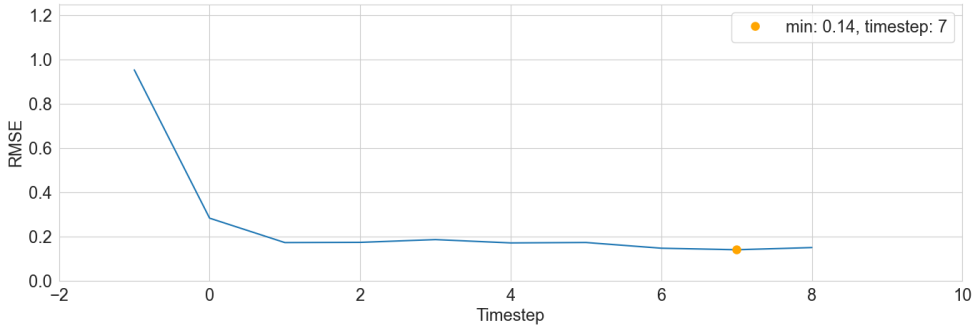
(b) *Toy_missing_31* dataset - Constant feature



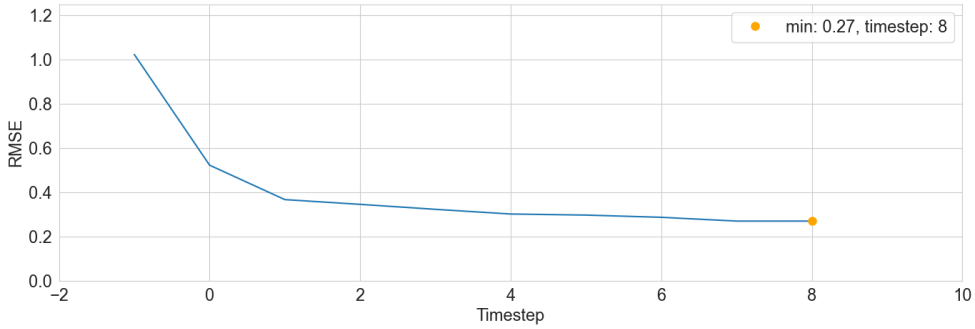
(c) *Toy_missing_91* dataset - Constant feature

Figure 14: Average Constant RMSE at each time step, for each toy validation set

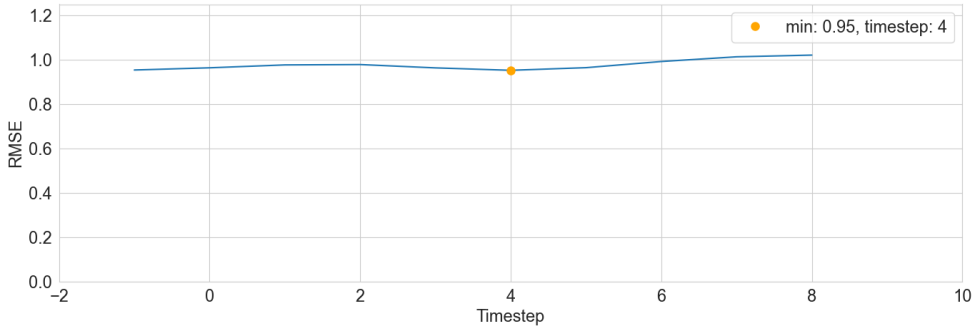
Figure 15 shows the RMSE scores for the Wide feature across the 3 toy datasets. As this feature is more complex than the Constant one, the network yields lower performances for the *toy* and *toy_missing_31* datasets. However, it is not able to learn almost anything from the *toy_missing_91* dataset, as the data is already standardized and the RMSE is close to 1. In the case of the MIMIC subset, the missingness amount is similar to the one of *toy_missing_31* dataset, thus the RMSE results for the time series variables will be very similar B.



(a) *Toy* dataset - Wide feature



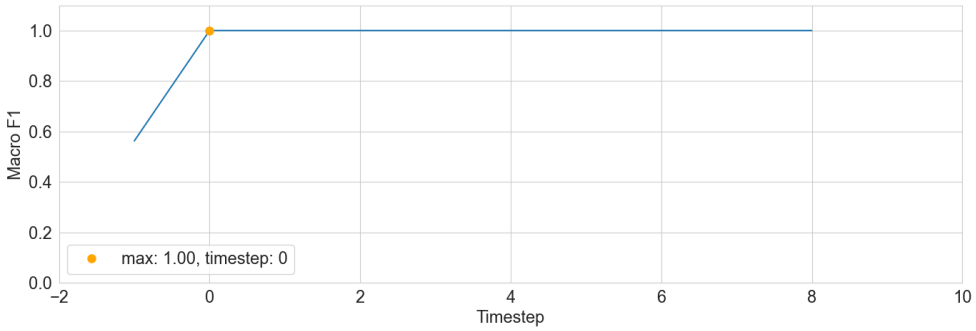
(b) *Toy_missing_31* dataset - Wide feature



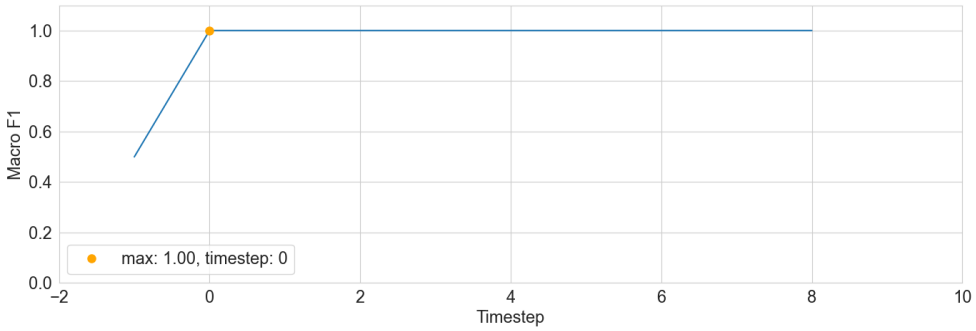
(c) *Toy_missing_91* dataset - Wide feature

Figure 15: Average Wide RMSE at each time step, for each toy validation set

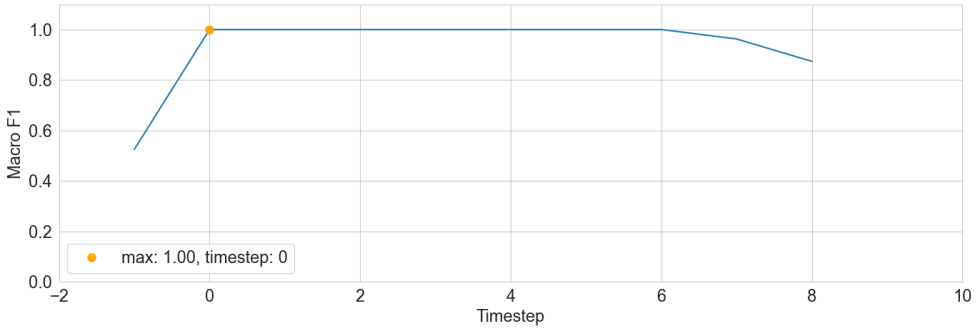
Figure 16 shows the Macro-F1 score for the Label. As the Label is considered as a static variable when training with feature decoders, its value is encoded at each time step, together with other available features and passed over the next time steps. Therefore, when decoding its value, the shared state should already incorporate information about it, thus a theoretically perfect prediction at all time steps besides t_{-1}. However, for the *toy_missing_91* there is a slightly decrease in the Label Macro-F1 score at the last time steps, which we could not explain.



(a) *Toy* dataset - Label



(b) *Toy_missing_31* dataset - Label

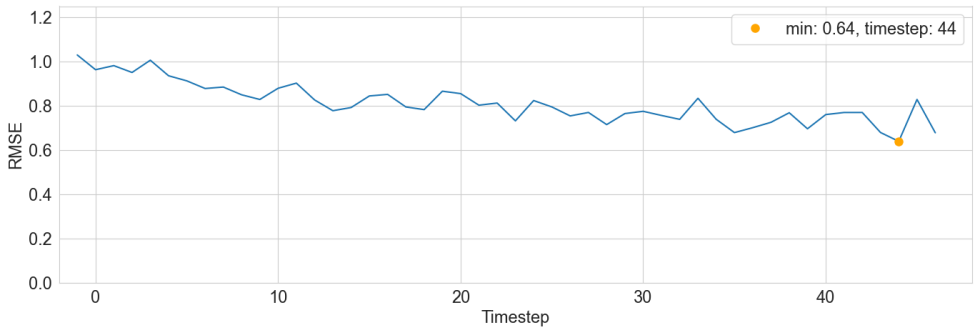


(c) *Toy_missing_91* dataset - Label

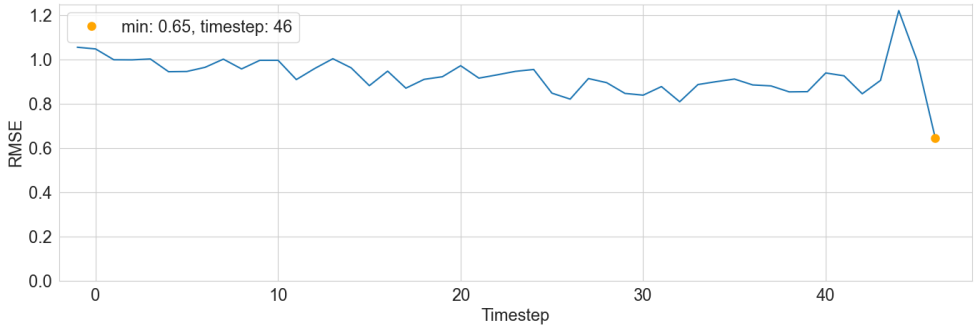
Figure 16: Average Label Macro-F1 score at each time step, for each toy validation set

B MIMIC subset metrics appendix

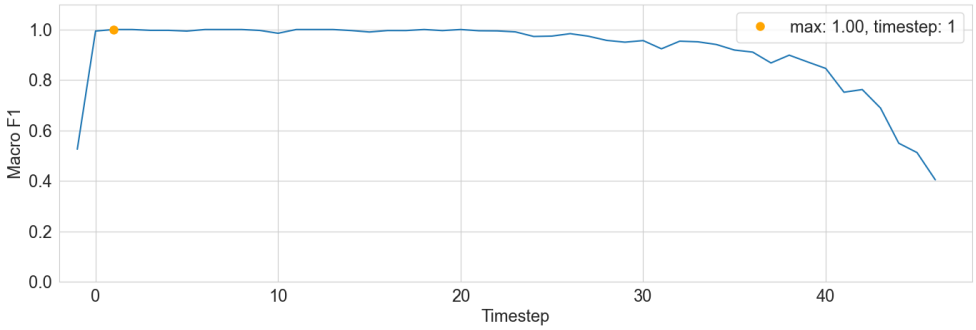
For the MIMIC subset, we display the RMSE and Macro-F1 score metrics for 3 features, in Figure 17. The model has poor performances on the 2 time series (plots (a) and (b)), mostly because of the very high degree of missingness, but also because of the complex pattern of the time series. This poor performance was somewhat expected because we saw a similar result even for the features in the *toy_missing_91* dataset, which has a similar amount of missing data. Regarding the Label Macro-F1 score, we see a similar decrease in the performance in the last time steps, as in the *toy_missing_91* dataset. Although this static outcome gets encoded and decoded after time step the network is still not able to learn it perfectly.



(a) RMSE - Chloride(serum)



(b) RMSE - Anion Gap



(c) Macro F1 score - Label

Figure 17: Average metrics at each time step, for (a) Chloride, (b) Anion Gap and (c) Label features from MIMIC subset