Synthetic hypnogram generation using Generative Adversarial Network

Tamás Bartos (202102242)

06 June 2023

Contents

1 Abstract

Sleeping is an essential part of everyday life and general human health. The discovery of electrical activity in the brain facilitated a significant advancement in sleep medicine[23, 21]. The examination of patterns of brain activity during sleep led to the classification of these cycles, called sleeping stages and the creation of hypnograms, which is the representation of the sleeping stages in a sleep sequence. This aided researchers to identify and define sleeping disorders and in the last couple of years got an extra boost from advancements in artificial intelligence. This thesis explores the potential application of a time-series generative adversarial network (TimeGAN) [25], leveraging its temporal data capture capabilities and the ability to generate realistic synthetic data.

Here we introduce a model trained on a large collection of annotated hypnograms, capable of learning the distribution of sleep stages and the underlying patterns.We discovered, that the presented model can provide high-quality generated data, that aligns with the characteristics and temporal dependencies of real-world sleep data. In addition, it can be used as a validation tool for sleep data as its discriminator component has the ability to distinguish between realistic and non-realistic data. Synthetic hypnogram can offers several things, they have the potential to augment existing datasets, address data scarcity issues and can further facilitate the evolution of sleep analysis algorithms.

2 Introduction

Sleep analysis is a vital part of modern sleep medicine as it plays an significant role in understanding patterns in sleep cycles and their indication for health and wellbeing. It requires sequences of sleep stages, which is the classification of the electroencephalogram (EEG) patterns of the brain during sleeping. Hypnograms represent the sequence and duration of sleep stages. They are essential tools for any sleep research, but the availability of large-scale annotated datasets can be a limitation for further research as machine learning algorithms are gaining space in this field. Training and evaluating them requires a significant amount of data. To overcome this adversary, generative neural networks can provide solutions by augmenting or replace existing datasets. In recent years generative adversarial networks(GANs) attracted attention as a reliable generative model. This master's thesis provides insight to time-series GANs and their application as a generative model for synthetic hypnogram generation.

Since Goodfellow et al.(2014) [7] introduced GANs, as a versatile generative network, capable of generating high quality synthetic data, its popularity has been constantly rising. It is made out of two main component, a generator, responsible for generating synthetic samples and a discriminator, responsible to evaluate the authenticity of the samples. The networks are trained together in an adversarial setting, hence the name, where the generator tries to generate samples, indistinguishable from the real ones, while the discriminator strives to classify every samples correctly. During the training process, GANs learn to capture underlying data distribution.

The versatility of the GANs lies in their capability to model complex, high-dimensional data across various domains, such as images, texts and time. It has been successfully implemented in numerous fields, like in image synthesis and natural language processing. Using GAN for hypnograms shows great promises, as capturing the temporal dynamics and the statistical characteristics of the data is a complex problem that GANs excel in. However, traditional GANs might struggle capturing the temporal dependencies in time series data, where it plays a crucial role. To overcome this limitation, Jinsung Yoon et al. (2019) [25] introduce a novel framework, time series GAN (TimeGAN), tailored for time series data, with accurate temporal dependencies. In addition to the regular components, TimeGAN introduces an embedding function, that maps the samples from

the feature space to the latent space. This latent space representation enables high quality comparison of the samples, facilitating the measurements of metrics. Using the latent space, the model is capable of capturing the essential characteristics and the temporal dependencies of the original data, enhancing the fidelity of the generated sequences.

To provide mapping between the latent space and the feature space, a recovery function has also been added. It ensures that statistical properties of synthetic data are aligning with the real data, not only on sequence level, but at every individual time steps.

The primary goal of this thesis is to introduce a methodology for generating synthetic hypnograms, that can expand or even replace real world datasets. To achieve realistic distribution on the generated data, the model has to be capable of finding the characteristics and temporal dynamics of real world hypnograms. To accomplish the required model characteristics, it has to learn the underlying patterns and distribution of sleep stages by training on a large, diverse, annotated dataset, and effectively utilize it during generative process. The aim is to find a set of hyperparameters, that can make-up a TimeGAN model, capable of generating high quality, diverse hypnograms with a reasonable training time (considering the possibilities of this thesis), which are ideal to use as training data for sleep data analysis.

To ensure the diversity of the training data, three distinct datasets were used. These are all accessed from the free to use National Sleep Research Resource (NSRR) [26] database, which was specifically made for sleep research, thus containing diverse datasets. The main evaluation metrics are the predictive and discriminative scores. Both of these metrics utilize a post-hoc RNN model, to score the generator capability of capturing conditional distribution in form of predictive characteristics, and to score the fidelity of the generated data by using the off-the-shelf RNN model as a discriminator, trained in a supervised manner.

Training a TimeGAN, or any GANs in general, is a complex problem. In case of a TimeGAN model, three different losses, a reconstruction loss, a supervised loss and an adversarial loss have to be balanced to reach optimal working state. To ensure, our data gets mapped to the latent space and then back to the feature space, the embedding and the recovery functions have to be trained first. Doing so we ensure seamless transition between the spaces. To encourage the model to capture the stepwise conditional distribution, a

supervised training follows. It utilizes the original data as the supervisor to ensure, that the generated sequences align with the statistical properties of the real data. The final training phase is a joint training process, where the goal is to optimize the entire system as a whole. It trains the generator, the discriminator and the embedding function at the same time, allowing them to improve collectively.

The expected outcome of this thesis is a base model, that can be the stepping stone for high quality, diverse and highly realistic synthetic hypnograms, but on its current state still capable of generating high quality, realistic synthetic data. The significance of synthetic hypnograms lies in the need of high quantity, high quality, diverse hypnograms, which is not present, as data scarcity is a huge problem in a lot of sleep related diseases.

During my thesis, I will present a soft introduction of hypnograms and the used datasets, then show what a GAN is and what special about TimeGAN is and what metrics are being used to evaluate the model performance. The training strategy will be introduced, as training GANs, especially TimeGANs is a complex problem. In the end, metric results, and comparisons to non-TimeGAN model generated hypnograms will be presented.

3 Methods

3.1 Data sets

Sleep stage annotations play a crucial role in understanding and analyzing sleep patterns and disorders. The most common way to represent these stages is a hypnogram. It is a form of polysomnography(PSG), the graphical representation of sleeping stages over time. Figure 1 shows a recording of hypnogram of a healthy, normal adult. The sleeping stages are based on recordings of brain wave activity from electroencephalogram (EEG). There are 5 different stages of sleeping [16]:

- Wakefulness: The state of being awake and alert.
- Rapid Eye Movement (REM):Vivid dreams and heightened brain activity, similar to Wakefulness describes this state.
- Non-Rapid Eye Movement Stage 1 (N1): Transition stage between wakefulness and sleep, also know as light sleep.
- Non-Rapid Eye Movement Stage 2 (N2): Deeper sleep, which makes up a huge portion of the sleep cycle.
- Non-Rapid Eye Movement Stage 3 (N3): Deep sleep stage associated with restful sleep.

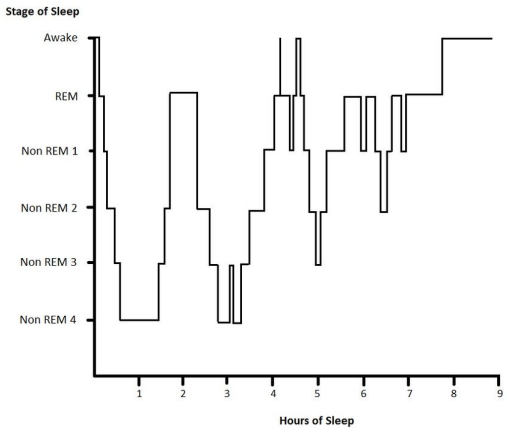


Figure 1: Example hypnogram [5]

Figure 1 shows an example hypnogram of a normal, healthy adult. It shows that sleep is cyclic, and an average human goes through four to six sleeping cycle [13]. Dr. Vyas et al.(2023) [22] gives a good explanation about the cyclic nature of sleeping stages in her article. To gather enough useable data, the National Sleep Research Resource (NSRR) [26] database was use. It is a free, web-based resource created by the National Institutes of Health (NIH) to promote open science and data sharing in sleep research. It offers datasets, tools and research resources to assist research in the field of sleep medicine.

NSRR is intended to aid research in a variety of fields including sleep and circadian science by supporting secondary data analysis, algorithmic development and signal processing through high-quality data sets[26]. The datasets can be used by researchers to examine and evaluate large scale datasets on sleep disorders, using standardized tools and procedures.

Its open science strategy supports collaboration and data sharing, accelerating progress in the field of sleep research and increasing our understanding of sleep disorders and their impact on health. This makes NSRR an essential resource for anyone interested in sleep research, offering a wealth of data and tools to support research. We used 3 different datasets in order to obtain hypnograms of people with various health background.

3.1.1 Apnea, Bariatric Surgery, and CPAP study

The objective of the Apnea, Bariatric surgery, and CPAP (ABC) [1] study was to compare the effectiveness of bariatric surgery to continuous positive airway pressure (CPAP) therapy and weight loss counseling in the treatment of patients with class II obesity and severe obstructive sleep apnea (OSA).

The ABC dataset consists of 3 rounds of tests. First, a baseline test was done with 49 individuals, which was followed by a second round of assessment after 9 months with 43 individuals, and after 18 months a final round of testing was done with 40 individuals. The hypnograms obtained during the study provide detailed information about sleep architecture, apnea severity, and related parameters.

3.1.2 Childhood Adenotonsillectomy Trial

The objective of Childhood Adenotonsillectomy Trial [12] was to test if children between the ages of 5 and 9.9 years, with mild to moderate obstructive sleep apnea randomized

to early adenotonsillectomy (eAT) will show greater levels of neurocognitive functioning compared to children randomized to watchful waiting and supportive care after a 7 months long observation period.

Standardized full polysomnography with central scoring was done at baseline and after 7 months. In total, 1447 children had screening polysomnographs,while 464 were given randomized treatment. Out of these tests, 3 groups of test were made, the baseline assessment created 453 polysomnograms and the follow-up assessment was conducted with 407 individuals and 779 test from a non-randomized set is also included.

3.1.3 Sleep Heart Health Study

The Sleep Heart Health Study (SHHS) [18] is a multi-center cohort study aimed at investigating the relationship between sleep disorders, cardiovascular disease and other health outcomes. The study was implemented by the National Heart Lung & Blood Institute, originally ran between November 1,1995 and January 1,1998 and a second polysomnogram was obtained during exam cycle 3 between January 2001 and June 2003.

6441 man and woman aged 40 years and older were enrolled for the first visit. The second polysomnogram was obtained in 3295 participants, allowing for a follow-up assessment of sleep parameters and their association with cardiovascular health.

The SHHS data set includes comprehensive information on sleep, respiratory variables, cardiovascular measurements, and health outcomes collected from participants. This extensive dataset permits a comprehensive examination of the relationships between sleep-disordered respiration, nocturnal hypoxemia, sleep architecture, and various cardiovascular risk factors.

3.2 Generative Adversarial Network

The network is made of 2 elements, a generator and a discriminator. While the generator is responsible for generating synthetic data, the discriminator serves as an evaluator, trying to distinguish between real and synthetic data. Both of them are neural networks and competing with each other during training, where the generator tries to generate synthetic data, that can fool the discriminator while the discriminator strives to classify them accurately.

Both the discriminator and the generator are refined by contending against each other

during the iterative training process. The generator attempts to enhance its realistic data generation capabilities, while the discriminator strives to improve its discriminatory prowess. Through this iterative feedback cycle, GANs improve their performance over time, resulting in the production of highly plausible synthetic data. In short, the generative model tries to maximize the probability of the discriminator making a mistake, while the discriminator tries to minimize mistakes. As a minmax game, the discriminator tries to maximize the probability of assigning the correct label to both examples and simultaneously train the generator to minimize (log(1-D(G(z))), which means they are playing a two-player minmax game with a value function V(G,D) [7]:

$$V(D,G) = E_{x\ p_{data}}(x)[logD(x)] + E_{z\ p_z}(z)[log(1 - D(G(z)))] \tag{1}$$

where

- G: generator
- D: discriminator
- P_{data}(x): real data distribution
- P(z): generator distribution
- x: sample from P_{data}(x)
- Z: sample from P(z)
- G(z): generator network
- D(x): discriminator network

During the training process of the generator network, random seed gets inserted as input, then this random noise propagate through the generator and outputs synthetic data. Then this synthetic data gets passed to the discriminator for classification and the result is being used to calculate the generator loss. In the end, backpropagation is being used to adjust the generator weights.

During the discriminator training, the generator weights are constant and real and fake data is being fed to the model, to classify them. The loss is being calculated to adjust the discriminator weights.

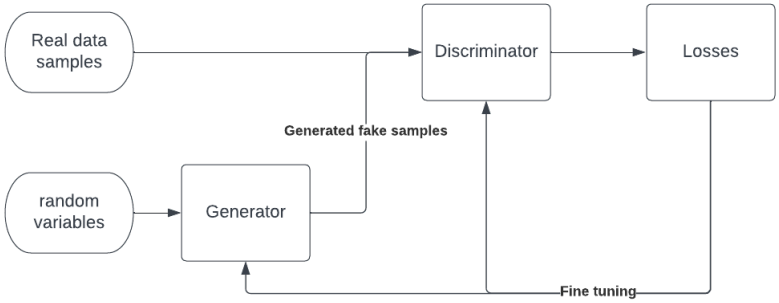


Figure 2: GAN architecture

Since the introduction by Goodfellow et al.(2014) [7], numerous variants of GAN emerged as highly successful networks, such as Conditional GAN (CGAN) [14], Deep Convolutional GAN (DCGAN) [19], Super Resolution GAN (SRGAN) [9] and Time series GAN (TimeGAN) [25].

GANs have numerous advantages, such as high-quality results, the ability of being trained with unlabeled data and versatility, as they can be applied for wide variety of tasks, including image-to-image translation, text-to-image synthesis, data augmentation and many more. However, it also has disadvantages, such as training instability, as it can be difficult to train with the risk on instability, mode collapse and convergence failure. The training process also requires a lot of computational resources, while it is also prone to overfit the training data, while biases and unfairness of training data can lead to biased synthetic data. As GANs are complex networks, it can be opaque and difficult to interpret, making it challenging to ensure accountability, transparency and fairness in their applications.

3.3 Time series GAN

Time series data is ubiquitous across various domains, characterized by sequential patterns and temporal dependencies. To successfully model time-series data, a model must be able to capture the datasets feature distributions within every moment, but should be able to capture the complex dynamics of the features across time. While GANs have shown promise in generating diverse and realistic data, the direct application of such model to time series data is non-trivial due to the inherent temporal dependencies. To address this challenge, a special GAN framework, TimeGAN has been proposed by Jin-

sung Yoon and Daniel Jarret in 2019 [25], designed for generating time series data, by combining the flexibility of unsupervised nature of GANs, in addition with the control, afforded by a supervised segment, which allows the model to capture time conditional distribution within the data by using the real data as supervisor. The framework also introduces an embedding and recovery network as an auto-encoder model.

TimeGAN enhances existing methodologies for time series data by incorporating specialized components that are tailored to the unique features of such data. In contrast to fundamental GAN frameworks, TimeGAN incorporates the additional auxiliary functions, embedding and recovery, to enhance its capabilities. The embedding network optimizes the dimensionality of the adversarial learning space by facilitating a reversible mapping between the features of the time series and a latent space. In reverse, the recovery network facilitates the transformation from the latent space to the initial feature space.

Implemented in the proposed framework are three losses shown at Figure 3: an unsupervised loss, known from the fundamental GAN framework, a supervised loss, and a reconstruction loss. The unsupervised adversarial loss operates on both real and synthetic data, the step-wise supervised loss further improves the model performance by utilizing the real data as a supervisory signal, explicitly guiding the model to reproduce the data step-wise conditional distribution, while the reconstruction loss improves the recovery functions reconstruction capabilities.

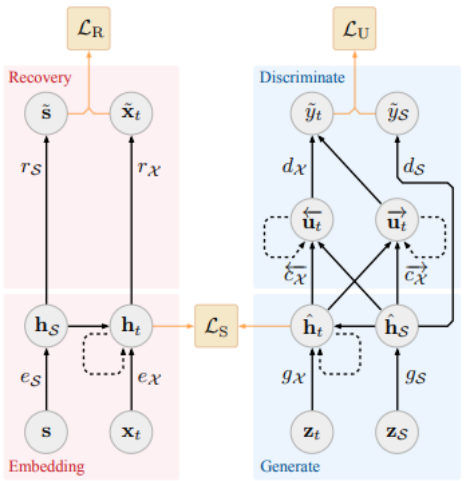


Figure 3: TimeGAN initiated with RNNs [25]. The solid lines shows forward propagation of the data, while the dashed line shows gradient backpropagation

3.3.1 Embedder and Recovery

The embedding and recovery functions ensure the mapping between the feature and the latent space, allowing the adversarial network to learn the underlying temporal dynamics of the data. The embedding network optimizes the dimensionality of the adversarial learning space by facilitating a reversible mapping between the features of the time series and a latent space. In reverse, the recovery network facilitates the transformation from the latent space to the initial feature space. These mapping functions (embedding and recovery) are respectively implemented by a recurrent neural network and a feed-forward network.

The embedding function can be written as: $e : S \times \prod_t X \rightarrow H_S \times \prod_t H_X$,where H_S and H_X denotes the latent vector spaces corresponding to S and X feature spaces and it takes the features to their latent code $h_s, h_{1:T} = e(s, X_{1:T})$. In this case (both in the original paper [25] and this thesis) **e** is implemented as a recurrent network,

$$H_S = e_S(s) \qquad\qquad h_t = e_X(h_s, h_{t-1}, x_t) \tag{2}$$

where e_s is network for static features and e_X is a recurrent network for temporal features. To transform the static and temporal features back to the feature representation $\tilde{S}, \tilde{X}_{1:T} = r(h_s, h_{1:T})$, the recovery function $r : H_S \times \prod_t H_X \rightarrow S \times \prod_t X$ has to be used. The **r** is implemented as a feed-forward network,

$$\tilde{s} = r_S(h_S) \qquad\qquad \tilde{X_t} = r_X(H_t) \tag{3}$$

where r_S is the recovery function for static and r_X is the recovery function for temporal embeddings. The embedding and recovery functions can be configured and parameterized by any architecture; the only requirement is that they have to be autoregressive and adhere to causal ordering.

3.3.2 Generator and Discriminator

Unlike the generator presented by Goodfellow et al. (2014) [7], the TimeGAN generator creates synthetic output to the latent space, not the feature space. The generator function $g : Z_s \times \prod_t Z_X \rightarrow H_S \times \prod_t H_X$, where Z_S and ZX are vector spaces which defines known distributions used to drawn random inputs from for generating output int H_S and

H_X. The generator function takes static and temporal random vectors to latent space $\hat{h}_S, \hat{h}_{1:T} = g(z_S, z_{1:T})$. The **g** generator function, similarly to **e** embedding function 2 and **r** recovery function 3, is being implemented as a recurrent network,

$$\hat{h}_S = g_S(z_S) \qquad \hat{h}_t = g_X(\hat{h}_S, \hat{h}_{t-1}, z_t) \tag{4}$$

where g_S is a network for static features and g_X is generator for temporal features. Z_s is a random sample from any given distribution, while z_t follows a stochastic process.

Similarly to the generator, the discriminator also uses inputs from the latent space. The discriminator function $d : H_S \times \prod_t H_X \to [0,1] \times \prod_t [0,1]$ gets the static and temporal information and returns a classification result $\tilde{y}_S, \tilde{y}_{1:T} = d(\tilde{h}_S, \tilde{h}_{1:T})$. The discriminator is implemented as a bidirectional recurrent network with feedforward output layer:

$$\tilde{y}_S = d_S(\tilde{h}_S) \qquad \tilde{y}_t = d_X(\overleftarrow{u}_t, \overrightarrow{u}_t) \tag{5}$$

where $\overrightarrow{u}_t$ denotes forward hidden states, while $\overleftarrow{u}_t$ denotes backward hidden states and d_x and d_s are output classification functions.

3.4 Metrics

Different metrics were used to measure the performance of the trained model(s). They show different aspects of the model and their performance. The metrics can be divided into two groups, training performance metrics and trained model performance metrics. Training model metrics are the different training losses, while trained model performance metrics are predictive score, discriminative score, embedder-recovery functions performance and a train-on-synthetic, test-on-real evaluation and a set of histograms, that shows the distribution of different sleeping stages at a given moment in a batch of hypnograms. Comparing the histograms of the generated data and the real data can give us insight of the similarities of the data distribution through a batch, showing us if the synthetic hypnograms have similar distribution as the real data. In short, we want to show the *diversity (1)*, the *fidelity (2)* and the *usefulness (3)* of the generated data.

Losses: Figure 4 shows the the losses, present in the model. In the first two phases, the reconstruction loss, then the supervised loss will be present and be used as evaluation metric, while in the last phase, all three of the losses will be present. However, the visualized losses will be the losses, that affects the generator, the discriminator and the

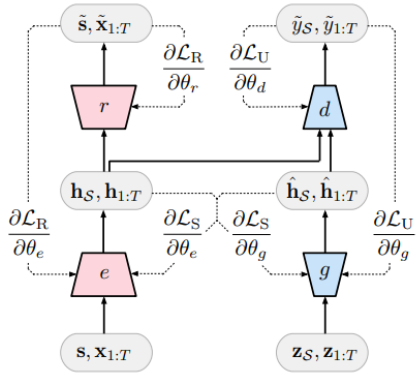
embedder, and the connection between these losses and the model loss can be seen on Figure 4b. Losses are essential metrics in any neural networks, as they are directly guide the learning process towards achieving optimal parameters by minimizing their corresponding value.

Discriminative score: To measure the *fidelity (2)* of our dataset, a post-hoc time-series classification model was trained to distinguish between original and synthetic datasets. Each original sequence is initially designated as real, while each generated sequence is labeled as fake. Then, as a standard supervised task, an off-the-shelf (RNN) classifier is trained to distinguish between the classes. The classification error on the test set is then reported, providing a quantitative evaluation of fidelity. In this implementation, if the score is 0, that means the model was able to predict the labels perfectly.

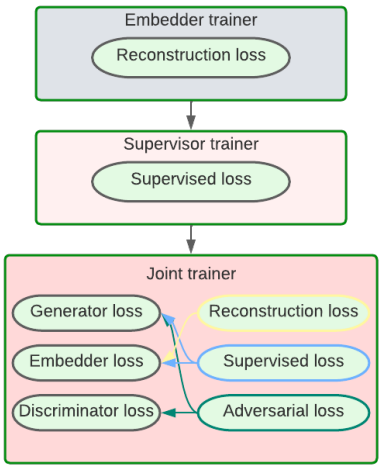
Predictive score: As a metric, predictive score gives feedback on how well our network is capable of transferring the predictive characteristics of the original data to the generated one. We expect TimeGAN to excel in capturing conditional distributions [25]. To measure this, the same method from the original paper was used. It uses the generated dataset to train a post-hoc sequence-prediction model to predict next-step temporal vectors over each input sequence. We us this model to evaluate the dataset containing the original data. The performance is measured by mean absolute error (MAE). This shows *usefulness (3)* of our generated dataset.

Distribution: Showing the data distribution of time series data is not an easy task to accomplish. The original paper uses t-SNE [10] and PCA [4] to visualize the high dimensional data it generates [25]. We follow the same route, but also going to add a time step wise distribution through a big batch of generated and real data. Three different parts of the hypnograms were evaluated: the start of the hypnograms, the middle of the hypnograms and the point, where most of the hypnograms are starting to become padding values.

4 Training strategy



(a) Training scheme of TimeGAN framework [25]



(b) Used losses during training cycles [6]

Figure 4: (a) Training scheme of a TimeGAN model; Dashed arrows indicate backpropagation, while normal arrows indicate forward propagation. (b) Shows the losses used during each training phases. Each rectangle is a different phase, while the last block shows the connection between the losses and the model element losses.

The training strategy of a TimeGAN model has 3 distinct parts. First, the embedder and the recovery have to be trained to be able to provide lossless transfer between feature and latent space. Secondly, the supervisor function has to be trained and lastly a joint trainer trains the autoencoding components with the adversarial components. This means, that TimeGAN learns to encode features, generate representations and iterate across time at the same time. In the framework, proposed by Jinsung Yoon et al.(2019) [25] all of the training segments are trained under the same number of iterations, as the phases has equal weighting, however, Dannenberg et al. (2021) [6] propose improvements in his thesis regarding the weights and iterations of the framework, improving the resource requirements of the model training.

4.1 Embedder-recovery training

The first part of the training is the training of the embedding and recovery functions. As it is a reversible mapping between the latent and the feature spaces, the two network have to be able to produce accurate reconstruction of s and $x_{1:T}$ original data from the

latent space to $\tilde{s}$,$\tilde{X}_{1:T}$ reconstructed data. The objective function in this training segment is the reconstruction loss, show on Figure 4.

$$\mathcal{L}_R = \mathbb{E}_{s,x_{1:T}\sim p}[\| s - \tilde{s} \|_2 + \Sigma_t \| x_t - \tilde{x}_t \|_2] \tag{6}$$

4.2 Supervisor training

The second segment of training is the supervisor training, where the generator is trained with the supervised loss only. This segment is used to improve the ability of the generator to capture step-wise conditional distribution in the data. To achieve this, the generator is being trained in alternating fashion. It is also being trained in open-loop mode, where the input is real data embeddings $h_{1:t-1}$ to generate the next latent vector. It is now possible to compute gradients on a loss that encapsulates the difference between distributions $p(H_t|H_S, H_{1:t-1}$ and $\hat{p}(H_t|H_S, H_{1:t-1}$. The supervised loss is calculated by maximum likelihood,

$$\mathcal{L}_s = \mathbb{E}_{s,x_{1:T}\sim p}[\Sigma_t \| h_t - g_x(h_S, h_{t-1}, z_t) \|_2] \tag{7}$$

where $g_x(h_S, h_{t-1}, z_t)$ approximates $\mathbb{E}_{z_t\sim\mathcal{N}}[\hat{p}(H_t|H_S, H_{1:t-1}, z_t]$ expected probability distribution with a single sample of z_t [25].

4.3 Joint trainer

The final training sequence is a joint trainer where the adversarial (unsupervised) loss and the previous 2 sequences are trained together, making this the most resource consuming part of the training process. Here, the algorithm learns to encode, iterate and generate at the same time. The autoregressive generator works in pure open-loop mode, receiving synthetic embeddings in order to generate the next synthetic vector. The gradients are calculated on the adversarial loss to allow the discriminator to maximize or the generator to minimize the chance of providing correct classification for the training data and the synthetic data from the generator,

$$\mathcal{L}_U = \mathbb{E}_{S,X_{1:T}\sim p}[\log y_S + \Sigma_t \log y_t] + \mathbb{E}_{S,X_{1:T}\sim \hat{p}}[\log(1 - \hat{y}_S) + \Sigma_t \log(1 - \hat{y}_t)] \tag{8}$$

4.4 Optimization

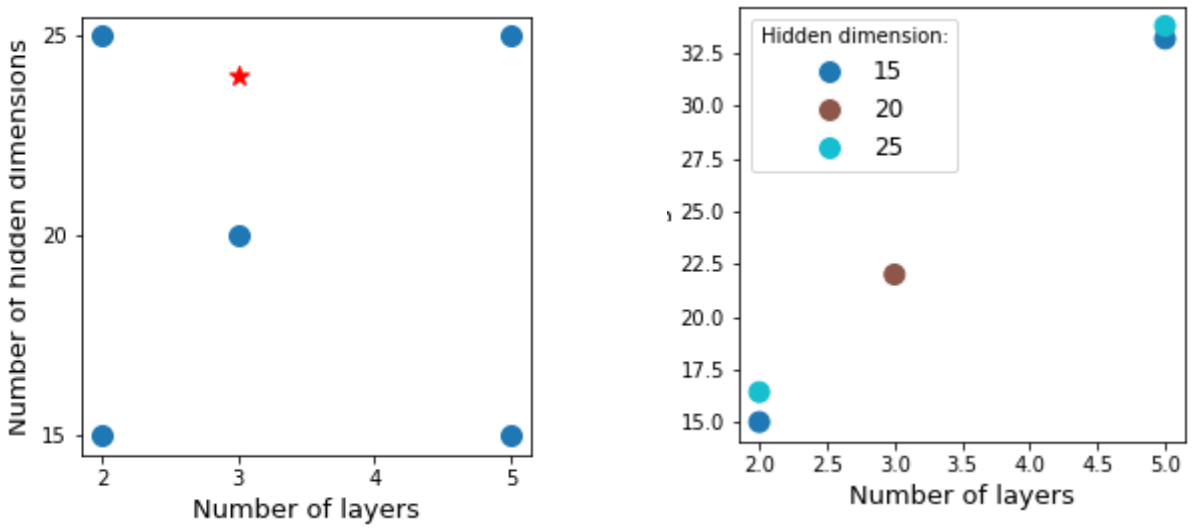
Figure 4 presents the mechanics of the training approach. θ_e,θ_r,θ_g,θ_d denotes the parameters of the different networks. To optimize the parameters, we have to consider

which loss(es) have effect on them. In case of the embedder and recovery parameters, the supervised and the recovery loss have to be considered,

$$\min_{\theta_e,\theta_r}(\lambda\mathcal{L}_S + \mathcal{L}_R) \tag{9}$$

where $\lambda \geq 0$ is a parameter that balances the losses and $\mathcal{L}_S$ is also incorporated in such manner that the embedding process reduces dimensionality of the adversarial learning space, and also aiding the generator in learning temporal relationships. In case of the generator and the discriminator, the objective function is the combination of the supervised and the unsupervised loss. The unsupervised loss is maximized with respect to the discriminator parameters and the generator parameters are minimized with respect to the combined loss, taking the $\eta \geq 0$ balancing hyperparameter into account.

$$\min_{\theta_g}(\eta\mathcal{L}_S + \max_{\theta_d}\mathcal{L}_U) \tag{10}$$



(a) Main hyperparameters the different training setups used

(b) Training time distribution during hyperparameter search

Figure 5: Hyperparameter search. It shows the two most significant hyperparameters of the models on which a small grid search was used on.

4.5 Hyperparameters

Hyperparameters in machine learning algorithms are notoriously hard to change, let alone optimize without a lot of computational resource [11].There are a variety of methods for

finding the optimal hyperparameters for a specific data collection [2]. Using manual setting has advantages, such as it allows checks on the effects of hyperparameter changes, by re-evaluating model after each iteration. This method's biggest drawback is the huge time requirements to adjust each hyperparameters one-by-one [24]. Recommended hyperparameter settings in packages are based on prior experiences; therefor they can be a good starting point for any hyperparameter search, although they might not yield the optimum overall accuracy. There are also methods for optimizing hyperparameters. These are optimization algorithms that use the available data to reduce the generalization error of the machine learning model for any given set of hyperparameters [8].

In the original proposal of the TimeGAN framework [25], greedy search algorithm [27] was used to find optimal solution as it is capable of finding an optimal hyperparameter set. The initial hyperparameters used in this project were based on the original TimeGAN paper [25] and the recommended parameters of the PyTorch implementation [3] with some fine tuning during the training phases. The 3 most important hyperparameter, proven by the original TimeGAN paper and by the thesis paper written by Dannenberg et al. (2021) [6] that recommend improvements. The most influential hyperparameters are the hidden dimensions, which are the number of features of the Long Short-Term Memory(LSTM) model [20, 17], that are the core network elements. The number of hidden dimensions also defines the dimensionality of the latent space, while the number of layers parameter defines the depth of the LSTM. The last parameter is the batch size, which is the size of the data batches, that are given to the model at one time. Based on the result of different training setup, Figure 5b shows, that the depth of the LSTM models influences the training time the most, considering they were trained on the same hardware and software. To optimize training time, Dannenberg (2021) [6] provides optimization options, such as separating the training epoch numbers of each training phase, as based on his measurements, training phase 1 and 2 occupy a significantly lower portion of the training time compared to the joint training phase.

5 Results

In this section, the results of the different metrics will be introduced. In the beginning, the different models will be compared based on their predictive and discriminative scores, then the best performing model will be evaluated more deeply, such as their loss functions and their stage distribution on different time periods. Finally, the discriminator will be tested on hypnograms from different sources or ones, that were transformed in unrealistic ways, such as one, that goes back in time or large language model (LLM), such as ChatGPT [15] generated ones.

5.1 Metric results

To measure predictive and discriminative score, an off-the-shelf RNN model was trained, first as a post-hoc sequence prediction model for the predictive score, then as a classification model for the discriminative score. In addition, the discriminator of the model was tested on a portion of the original dataset and a set of synthetic data, generated based on the lengths of the original dataset. In order to have a basis of comparison, we also measured the scores of the other trained model, presented on Figure 5a. In addition we also trained 2 models on generated data, one where the length information is the same as the real data and the other one has similar distribution of length information as the real data. In addition to the recommended tests, an additional discriminator test was added, which tests the quality of the models own discriminator against their generated data, showing how successful the training process was. For this test, a batch of 64 hypnograms were used from each models generated data and real datasets, and was fed through the discriminator, giving a discriminator score for each hypnogram, based on how real it thinks the data is. Taking the average of that score gives us an average real and average fake scores. These values are between 0 and 1, where 0 means the discriminator was perfectly confident, that that batch of data was fake, while 1 means that it thought the data was real. Using this score with a visualized generated hypnogram shows us how well the discriminator and the generator works.

Table 1 shows the results of the predictive and the discriminative metric, completed by the two generated data trained model and the result of my discriminator test. The naming of the model is based on their hidden dimension and number of layers hyperparameters.

Model name	Predictive score	Discriminative score	Avg real score	Avg fake score
2-15	0.0928	0	0.9996	0.9997
2-25	0.0926	0	0.4498	0.4735
3-20	0.1929	0.4887	0.9278	0.9181
5-15	0.0927	0	0.9668	0.0392
5-25	0.0927	0	0.9659	0.0467

Table 1: Predictive and discriminative metric results for different model setups. The bold model name indicates the most optimal model from our set.

First of all, the results of the discriminative score clearly show, that the classifier model was unable to capture any pattern or characteristics from the real data, so it classifies everything as real data, thus the score 0. However, the predictive score also gives us a small value, unlike the discriminative score, it can make sense. The low score means, that the model was unable to capture temporal dynamics and dependencies properly. The additional discriminator score can also be misleading if someone only look at the score. For example, the model with 2 layers and 15 dimensions (2-15, row 1) has a close to perfect score, but in reality, the generated hypnograms are highly and unrealistically cyclic. Figure 6 shows an example generated signal from that model setup. Based on the scores for this particular setup, we may draw the conclusion that in this setup, the model was not able to find a good set of weights. The hyperparameter set of (2,25) has similar problems, although in that case, the discriminator does a bad job at discriminating signals, as the discriminator test shows, that its discriminator gives away basically random results, as 0.5 value means, that the discriminator has no real idea, what is real data and what is not, meaning it was not able to capture information about the real dataset. Based on these two trained models, we can argue that the training process requires longer training or the LSTM model requires more than two layers to capture temporal information successfully.

In the other end of test parameters, the model with 5 layers and 15 dimensions run into the same problem as the other models and receives 0 score for discriminative score. The discriminator score on the other hand shows, especially with the help of a generated hypnogram, that the generator produces terrible quality data, an example can be shown on Figure 7. This means, that we can not really evaluate the discriminator on its generated data. To get a bit better understanding of the quality of the discriminator,

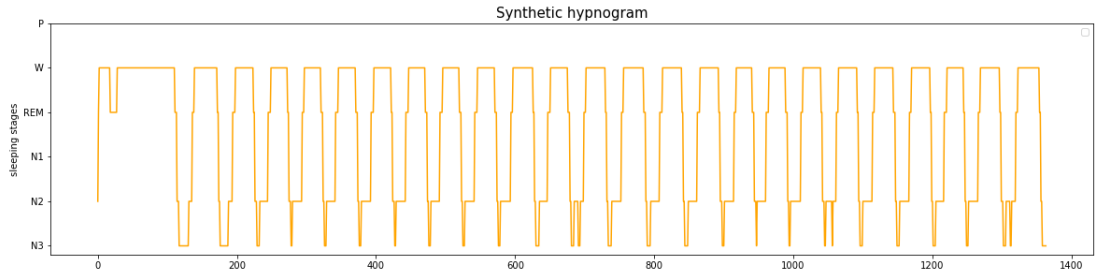


Figure 6: A random generated hypnogram from the model with 2 layers and 15 hidden dimensions. It is a highly periodic signal, which is not typical for hypnograms.

we can use the generated data from a different dataset, such as the 3 layers, 20 dimensions model, which has overall the best (or least worse) scores. Using that dataset, the average score of a batch of synthetic data is 0.9594, which means the discriminator is unable to differentiate the generated data from the real one. The other model with the same number of LSTM layers has similar results. Testing the discriminator with the main generated dataset, the average score is 0.9835. If we compare the 4 already introduced

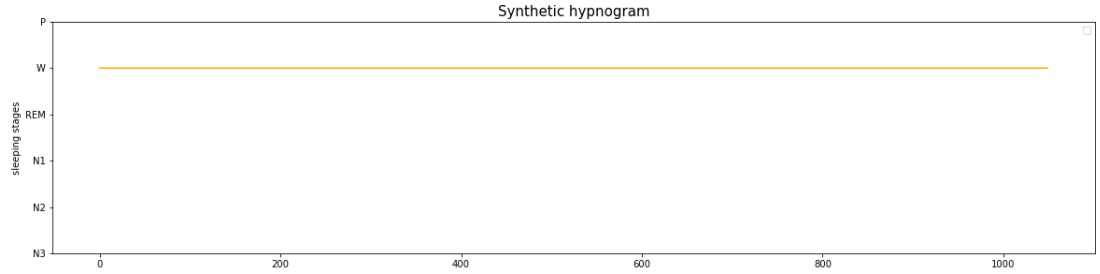


Figure 7: A random generated hypnogram from the model with 5 layers and 15 hidden dimensions. It shows, that the model was unable to capture temporal dynamics.

models, we can see, that the biggest difference maker in term of temporal data capture is the model depth. In case of 2, the model was unable to catch deeper characteristics, so its generator generates periodic data. On the other hand, the high layer number might increase the chance of mode collapse, regardless of processes implemented into TimeGAN to prevent it. Another issue might be the relatively low training epochs, so increasing it might yield better results.

Compared to the already analyzed models, the one with 3 layers and 20 dimensions produced reasonable scores, except for predictive score, which is also low like the other models. This means, that it might also have problems with properly capturing depen-

dencies, but out of all tested model, it scored the highest, meaning it is the most capable model out of the 5. Its discriminative score is also around 0.5, meaning the ad-hoc discriminator has a lot harder time discriminating between real and syntactic hypnograms, and the 0.5 score means that it is basically randomly labeling the signals and does not have a clear way to distinguish between the real and synthetic signal. Looking at the table we can see, that the discriminator score is quite high, which means the generator of this parameter setup can provide high enough quality synthetic hypnograms, that the discriminator can not differentiate from real ones. Figure 8 shows an example of a real and a generated hypnograms. The most noticeable thing on the figure is that the generator was able to learn the cyclic nature of a hypnogram, without becoming highly periodical like Figure 6. However it still has obvious deficiencies like in the beginning for some reason it always starts from deep sleep state (NREM 3), instead of the wakefulness state. Figure 16a shows that it is the case in nearly every generated hypnogram. A similar problem, that is not visible on this example, or on the distribution histograms is the fact, that not every hypnogram converges to wakefulness state.

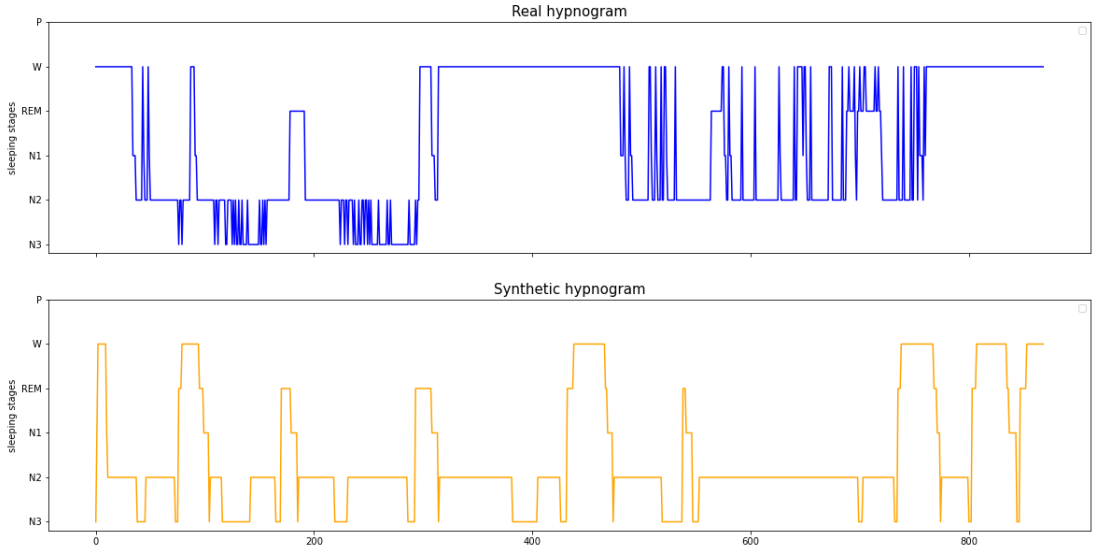


Figure 8: A pair of synthetic and real hypnograms. This example shows, that the generator started to pick up characteristics of real data, such as the cyclic but not periodical nature of the signal.

Another example that can show the quality of the discriminator and the generator is if we compare the discriminator score of a batch of real and fake hypnograms, where the compared pairs have the same length, meaning the discriminator can not rely on length

based knowledge. Figure 9 shows the result of that comparison. The figure shows, that for most of the time, the synthetic data stays between the 0.8-1 range, which is really good result for the generator and it means that currently it is comfortably winning against the discriminator in their adversarial game. As it was mentioned before in subsection 4.5 the

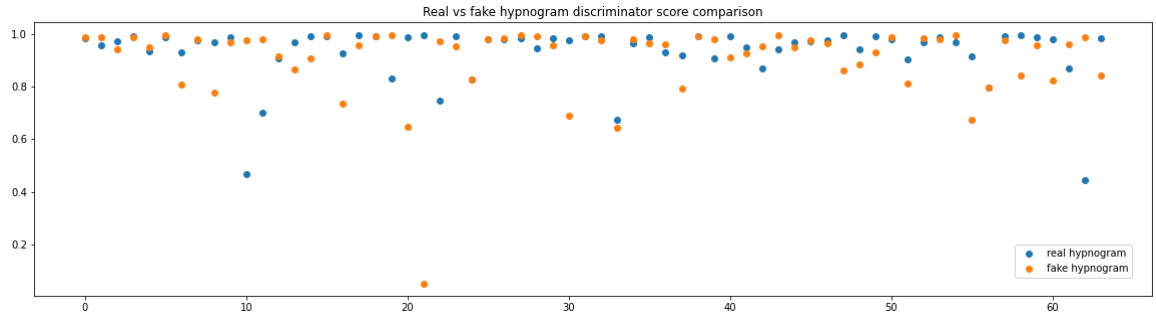


Figure 9: Discriminator score of a batch of real and fake hypnograms (higher is better for the generator).

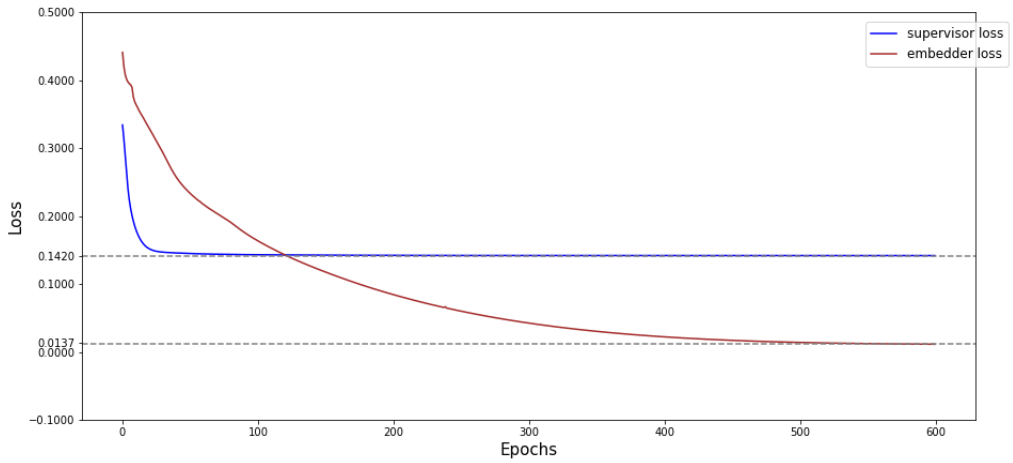


Figure 10: Embedded and supervisor loss during training. It shows, that the supervisor training required significantly less epochs to find an optimal parameter set.

model giving the most optimal results in term of predictive and discriminative score, while still keeping the training time within acceptable limits considering our computational resources has a latent space dimensionality of 20 and has 3 LSTMs layered on each other. Additionally, the other most significant variable based on the TimeGAN paper [25] and related papers [6] is the batch size, which was chosen to be 64.

Figure 10 shows the loss change during the embedder training loop and the supervisor training loop. The supervisor was able to find its optimum in a very short time, while the reconstruction loss needed significantly more iteration. Figure 11 compares the generator,

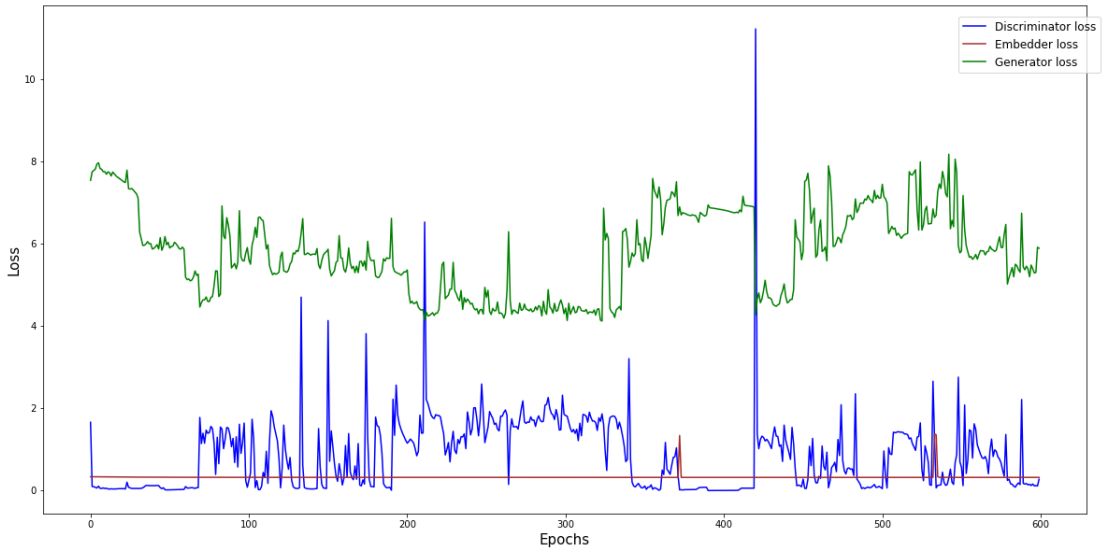
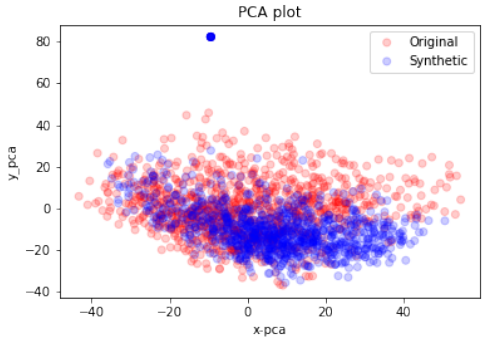


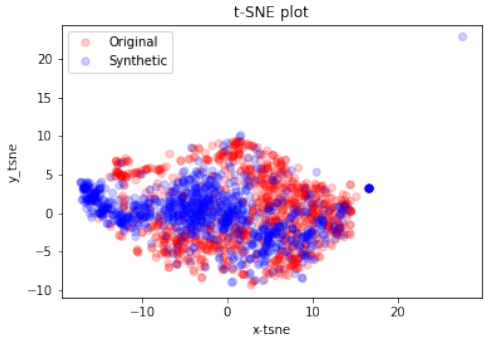
Figure 11: Discriminator, generator and embedder loss during training.

the discriminator and the reconstruction loss. These losses are the combinations of the supervised loss Equation 7, Equation 8 and Equation 6 and the connection between them can be seen on Figure 4b. Based on the nature of GANs, the discriminator and the generator respond to changes in each others parameters. Figure 11 gives us insight in that when either of the adversaries find an improved weight set, that makes the other loss change in the other direction. This phenomenon proves, that during the third training process, the discriminator and the generator are truly adversaries and effect each other as they should.

To conclude our metrics, the diversity of our data has to be measured. t-SNE, PCA and a set of histograms gives qualitative assessment of the data distribution. Figure 12 shows that the synthetic data has a similar distribution as the real data, far from being completely similar, but for the scope of our project it is a promising result, as it means that the model was able to generate reasonably diverse data. It suggests, that the generator was able to capture underlying structure, statistical properties and relationships, and able to generate data based on those findings. However, it is important to note, that this visual similarity does not guarantee accurate representation of real distribution as specific feature distributions may not be captured by PCA and t-SNE. Another method to visualize distribution, that was used is histograms at some specific timestamps. It does not show as high quality qualitative results as PCA or t-SNE, however it can show some bugs and error our generator generates. Figure 16 shows these histograms in a way, that



(a) PCA visualization of the generated and the original datasets



(b) t-SNE visualization of the generated and the original datasets

Figure 12: PCA and t-SNE visualization of the generated and the original datasets. Red denotes original and blue denotes generated.

the top 3 rows, are the histograms of the synthetic data, while the bottom 3 represents the original data. The 2 sets of histograms have the same logic behind them; the first row represents the beginning, the second represents the middle and the third represent a sequence where most of the hypnograms are close to the end of their measurement, thus getting padding state. We can see on the second and third rows, that the data sleeping stage distribution through a single time point is similar, supporting the idea what the PCA and the t-SNE visualization presents. However we can see on the first row, that the generator has problems generating starting value as basically each hypnogram start in non-REM state and in the next few time-steps have the exact same sleeping states. That is obviously an error, that has to be taken care of in the future as it shows on the real dataset, that the majority of its hypnograms have a Wakefulness starting stage.

5.2 Tests on other hypnograms

In order to have a bit more comparison, different generated hypnograms can be used to test the model. First of all, we can use hypnograms that are going back in time, meaning they still have the same cyclic nature of a hypnogram, but the relationship between neighbouring data points are different. Our expectation from a flawless model would be that it classifies these datasets as fake hypnograms, however it is fair to assume, that the discriminator of the model is not at the state yet, where it can confidently know this small differences in hypnograms. Figure 13 shows expected results as for most of

the hypnograms, the discriminator was unable to deduce the fact, that the signals were fed backwards, however it was still able to label a few signals as not real. During this measurement the average score for the flipped real data set was 0.968 while the average for the flipped synthetic was 0.9232, fairly similar to the results of the non transformed datasets.

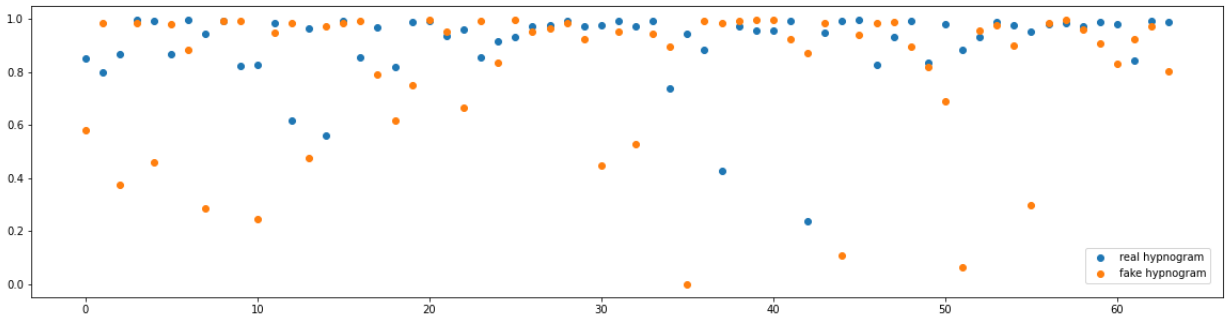


Figure 13: Discriminator results of a batch of real and synthetic sequence if their x axis has been flipped. The x axis represents each elements of the batch, while the y axis shows the discriminator output of each signal.

Another transformation we can do is flipping the y axis of the dataset. This case we can see a lot more success from the discriminator as Figure 14 shows that the discriminator had a reasonable amount of success deducing that the datasets are not realistic. During this measurement the average score for the flipped real data set was 0.3432 while the average for the flipped synthetic was 0.3986, both of the mare significantly worse, than the discriminator scores of the normal dataset presented on Table 1.

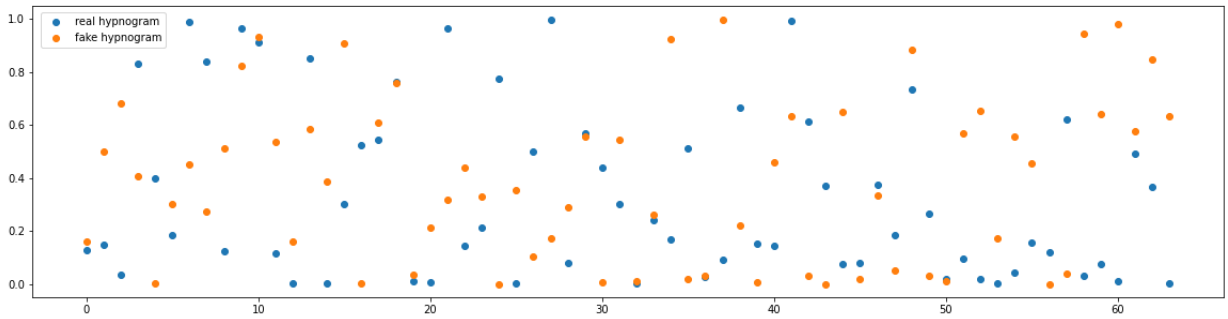


Figure 14: Discriminator results of a batch of real and synthetic sequence if their y axis has been flipped. The x axis represents each elements of the batch, while the y axis shows the discriminator output of each signal.

The last experiment was using the built in generator capabilities of ChatGPT and its

understanding of hypnograms. Figure 15 shows as two hypnograms generated by ChatGPT. Both shows, that ChatGPT does not understand the deeper relationship between points, however its knowledge of cyclic data generation provides it with a good enough generative capabilities that it can fool our discriminator, as it has really hard time realizing that a cyclic signal is not necessary a hypnogram. That was the case with these two generated signals too, as the first signal had a discriminator score of 0.908 while the other one had a score of 0.9155, meaning the discriminator had a high confidence that they are real hypnograms.

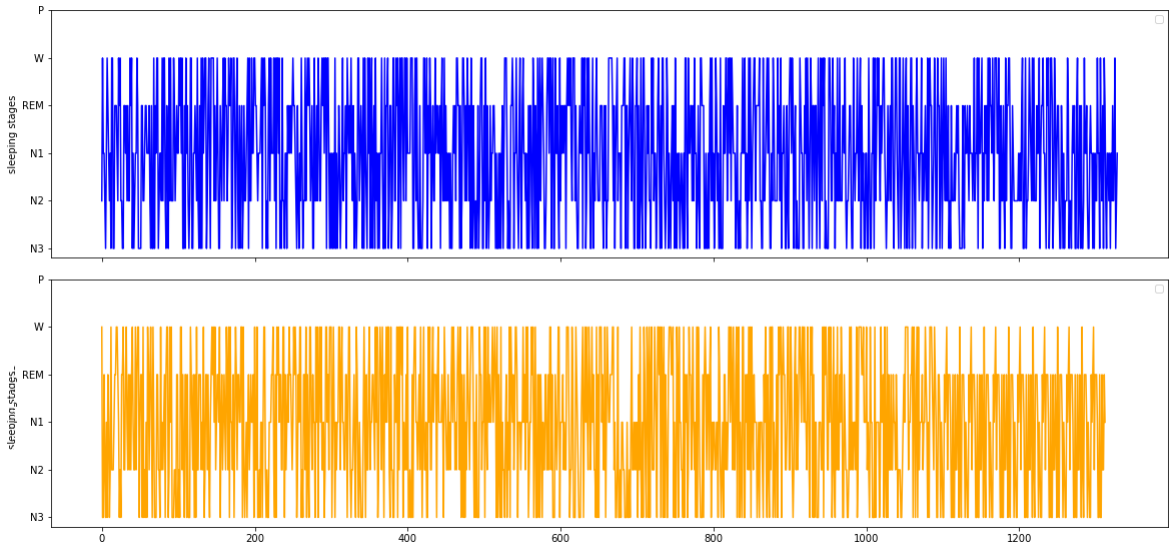


Figure 15: Hypnograms generated by ChatGPT. It is a highly random but cyclic dataset, without any realdistribution.

6 Discussion and conclusion

To conclude, we found that TimeGAN [25] is capable of learning temporal dynamics of time-series data, such as hypnograms. Our goal was to introduce a model, that can be a base for a well functioning TimeGAN model, capable of generating good quality hypnograms and identifying bad quality hypnograms using its discriminator. It showed that it can consider the distribution of the signals, can find their characteristics and apply them during the generative process. Based on the results presented in this thesis, it is fair to say, that the presented model is far from being a high quality one, but it can show the direction, can be the starting point towards a higher quality one.

But to achieve that, there are numerous problems it has to tackle. For example, its inability to label high frequency periodic signals (such as the one generated by ChatGPT) as fake is concerning. Also the jumps in the beginning of every generated data means, that its not just the discriminator, but the generator also has a lot more to learn. Based on the predictive score, it is fair to say, it can find characteristics and underlying connection, but not proficient at doing it.But these issues might get solved with a longer, more optimized training routine. Further research could be conducted on the different components of the model, as changing them might yield improved results. Also implementing improved methods presented in the bachelor thesis based on TimeGAN [6] could improve both the model performance and the training times.

However if the model gets through these issues, it has numerous ways that can further improve. Based on the capabilities of TimeGAN, static features, such as age, gender can also be added, to further diversify the generated data. And if all of these updates are added to it, it will be able to generate real diverse data, that can push science away from its biggest deficiency, scarce datasets.

Acronyms

ABC Apnea, Bariatric Surgery, and CPAP study. 8

CGAN Conditional GAN. 11

CPAP Continuous positive airway pressure. 8

DCGAN Deep Convolutional GAN. 11

eAT Early Adenotonsillectomy. 9

EEG Electroencephalogram. 4, 7

GAN Generative Adversarial Network. 4–6, 10–12, 25

LLM Large Language Model. 20

LSTM Long Short-Term Memory. 19, 21, 22, 24

MAE Mean absolute error. 15

NIH National Institutes of Health. 8

NREM Non-Rapid Eye Movement. 23

NSRR National Sleep Research Resource. 5, 8

OSA Obstructive sleep apnea. 8

PCA Principal component analysis. 15, 25, 26

PSG Polysomnography. 7

REM Rapid Eye Movemen. 7, 26

RNN Recurrent neural network. 5, 12, 15, 20

SHHS Sleep Heart Health Study. 9

SRGAN Super Resolution GAN. 11

t-SNE t-distributed stochastic neighbor embedding. 15, 25, 26

TimeGAN Time series GAN. 3–6, 11–13, 15, 16, 19, 22, 24, 29

References

[1] Jessie P Bakker et al. "Gastric Banding Surgery versus Continuous Positive Airway Pressure for Obstructive Sleep Apnea: A Randomized Controlled Trial". en. In: *Am J Respir Crit Care Med* 197.8 (Apr. 2018), pp. 1080–1083.

[2] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization". In: *J. Mach. Learn. Res.* 13.null (Feb. 2012), pp. 281–305. ISSN: 1532-4435.

[3] birdx0810. *TimeGAN pytorch implementation*. timegan-pytorch. 2022. URL: https://github.com/birdx0810/timegan-pytorch.

[4] Fred Bryant and Paul Yarnold. "Principal-component analysis and exploratory and confirmatory factor analysis". In: (Jan. 2001).

[5] Wikipedia contributors. *Hypnogram*. https://en.wikipedia.org/wiki/Hypnogram. Accessed May 17, 2023. Apr. 2023.

[6] Jan-Mark Dannenberg et al. "Time Series Synthesis using Generative Adversarial Networks". Bachelor thesis. TU Delft Electrical Engineering, Mathematics and Computer Science; TU Delft Software Technology, July 2021. URL: http://resolver.tudelft.nl/uuid:fe2e735d-35fa-4787-9a28-bfbf3a276146.

[7] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[8] Md Riyad Hossain, Douglas Timmer, and Hiram Moya. "Machine learning model optimization with hyper-parameter tuning approach". In: Aug. 2021.

[9] Christian Ledig et al. *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*. 2017. arXiv: 1609.04802 [cs.CV].

[10] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[11] Rafael G. Mantovani et al. "Effectiveness of Random Search in SVM hyper-parameter tuning". In: *2015 International Joint Conference on Neural Networks (IJCNN)*. 2015, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280664.

[12] Carole L Marcus et al. “A randomized trial of adenotonsillectomy for childhood sleep apnea”. en. In: *N. Engl. J. Med.* 368.25 (June 2013), pp. 2366–2376.

[13] Pejman Memar and Farhad Faradji. “A Novel Multi-Class EEG-Based Sleep Stage Classification System”. en. In: *IEEE Trans Neural Syst Rehabil Eng* 26.1 (Jan. 2018), pp. 84–95.

[14] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets*. 2014. arXiv: 1411.1784 [cs.LG].

[15] OpenAI. *OpenAI*. https://openai.com/about. Accessed 2023.

[16] Aakash K Patel et al. *Physiology, Sleep Stages*. StatPearls Publishing, Sept. 2022.

[17] PyTorch. *torch.nn.LSTM - PyTorch Documentation*. Accessed on May 17, 2023. 2023. URL: https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html.

[18] S F Quan et al. “The Sleep Heart Health Study: design, rationale, and methods”. en. In: *Sleep* 20.12 (Dec. 1997), pp. 1077–1085.

[19] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].

[20] Haşim Sak, Andrew Senior, and Françoise Beaufays. *Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition*. 2014. arXiv: 1402.1128 [cs.NE].

[21] John W Shepard Jr et al. “History of the development of sleep medicine in the United States”. en. In: *J Clin Sleep Med* 1.1 (Jan. 2005), pp. 61–82.

[22] Sleep Foundation. *Stages of Sleep*. Sleep Foundation. Accessed: May 17, 2023. URL: %7Bhttps://www.sleepfoundation.org/stages-of-sleep%7D.

[23] Susan L Worley. “The Extraordinary Importance of Sleep: The Detrimental Effects of Inadequate Sleep on Health and Public Safety Drive an Explosion of Sleep Research”. en. In: *P T* 43.12 (Dec. 2018), pp. 758–763.

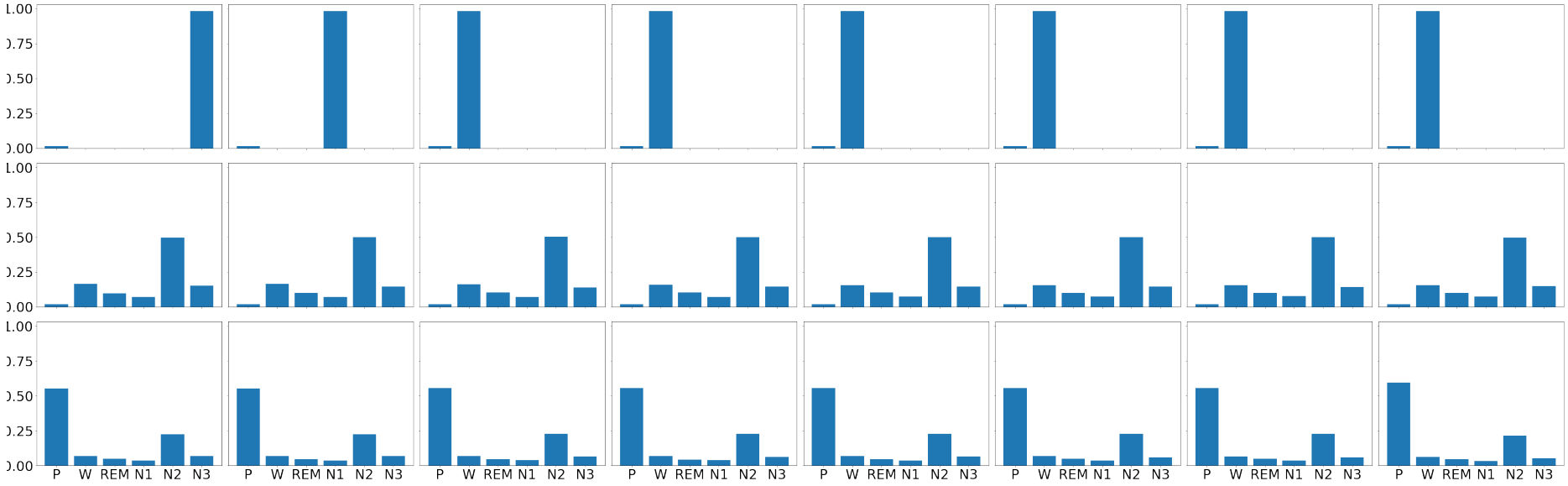
[24] Li Yang and Abdallah Shami. "On hyperparameter optimization of machine learning algorithms: Theory and practice". In: *Neurocomputing* 415 (2020), pp. 295–316. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.07.061. URL: https://www.sciencedirect.com/science/article/pii/S0925231220311693.

[25] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. "Time-series Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

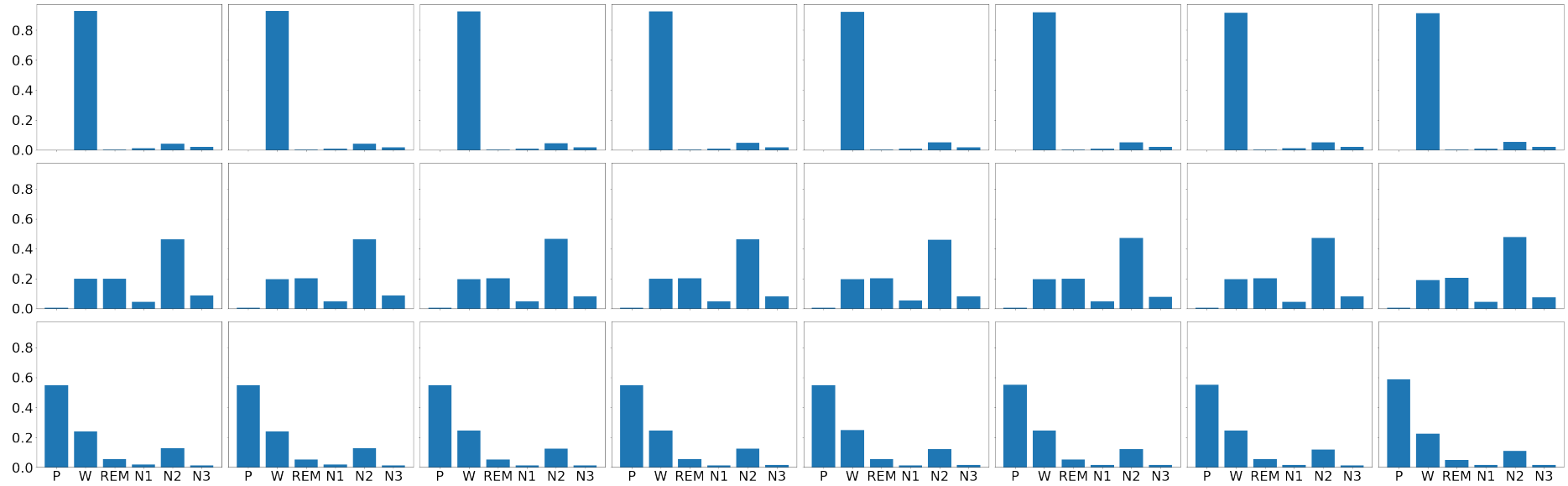
[26] Guo-Qiang Zhang et al. "The National Sleep Research Resource: towards a sleep data commons". en. In: *J Am Med Inform Assoc* 25.10 (Oct. 2018), pp. 1351–1358.

[27] Federico Zocco et al. *Lazy FSCA for Unsupervised Variable Selection*. 2022. arXiv: 2103.02687 [cs.LG].

Appendix



(a) Histograms for synthetic data distribution during a small time sequence



(b) Histograms for real data distribution during a small time sequence

Figure 16: Distribution of the different sleeping stages.The first row show the distribution of stage in the first 8 data point, the second row shows 8 data point from the middle, while the last row shows data points close to the end, where hypnograms starts to get padding value as a state.