A. Appendix

Table A.1. M3-Mortality prevalence of labels for the binary classification task

	Train	Test	Val
In-hospital deaths	0.135	0.116	0.135

Table A.2. P-Mortality prevalence of labels for the binary classification task

	Train	Test	Val
In-hospital deaths	0.142	0.142	0.142

Table A.3. P-Sepsis prevalence of labels for the online prediction task

	Train	Test	Val
Sepsis occurrence	0.018	0.018	0.018

A.1. Dataset preprocessing

Filtering Due to memory requirements of some of the competitor methods, it was necessary to excluded time series with an extremely large number of measurements. For M3-Mortality, patients with more than 1000 time points were discarded as they contained dramatically different measuring frequencies compared to the rest of the dataset. This led to the exclusion of the following 32 patient records: 73129_2, 48123_2, 76151_2, 41493_1, 65565_1, 55205_1, 41861_1, 58242_4, 54073_1, 46156_1, 55639_1, 89840_1, 43459_1, 10694_2, 51078_2, 90776_1, 89223_1, 12831_2, 80536_1, 78515_1, 62239_2, 58723_1, 40187_1, 79337_1, 51177_1, 70698_1, 48935_1, 54353_2, 19223_2, 58854_1, 80345_1, 48380_1.

In the case of the P-Mortality dataset, some instances did not contain any time series information at all and were thus removed. This led to the exclusion of the following 12 patients: 140501, 150649, 140936, 143656, 141264, 145611, 142998, 147514, 142731,150309, 155655, 156254.

For P-Sepsis some instances did not contain static values or were lacking time series information all together. We thus excluded the following files: p013777.psv, p108796.psv, p115810.psv.

Static variables The datasets often also contain information about static variables, such as age and gender. Table A.4 lists all the static variables for each of them.

Table A.4. Static variables used for each of the datasets in the experiments. Categorical variables are shown in *italics* and were expanded to one-hot encodings.

Dataset	Static Variables
M-Mortality	Height
P-Mortality	Age, *Gender*, Height, *ICUType*
P-Sepsis	Age, *Gender*, HospAdmTime

Time series variables For all datasets, we used all available time series variables including vitals, lab measurements, and interventions. All variables were treated as *continuous*, and no additional transformations were applied.

Splits All datasets were partitioned into three subsets training, validation and testing. For the M-Mortality dataset, the same splits as in (Harutyunyan et al., 2019) were used to ensure comparability of the obtained results. For both Physionet datasets (P-Mortality and P-Sepsis), we did not have access to the held-out test set used in the challenges and thus defined our own splits. For this, the full dataset was split into a training split (80%) and a testing split (20%), while stratifying such that the splits have (approximately) the same class imbalance. This procedure was repeated on the training data to additionally create a validation split. In the case of the online task P-Sepsis, stratification was based on whether the patient develops sepsis or not.

Implementation We provide the complete data preprocessing pipeline including the splits used to generate the results in this work as a separate Python package medical-ts-datasets, which integrates with tensorflow-datasets(TFD). This permits other researchers to *directly* compare to the results in this work. By doing so, we strive to enable more rapid progress in the medical time series community.

A.2. Comparison partners

The following paragraphs give a brief overview of the methods that we used as comparison partners in our experiments.

GRU-simple GRU-SIMPLE (Che et al., 2018) augments the input at time t of a Gated-Recurrent-Unit RNN with a measurement mask m_t^d and a δ_t matrix, which contains the time since the last measurement of the corresponding modality d, such that

$$\delta_t = \begin{cases} s_t - s_{t-1} + \delta_{t-1}^d & t > 1, m_{t-1}^d = 0 \\ s_t - s_{t-1} & t > 1, m_{t-1}^d = 1 \\ 0 & t = 0 \end{cases}$$

where s_t represents the time associated with time step t.

GRU-D GRU-D or GRU-Decay (Che et al., 2018) contains modifications to the GRU RNN cell, allowing it to decay past observations to the mean imputation of a modality using a learnable decay rate. By additionally providing the measurement masks as an input the recurrent neural network the last feed in value. Learns how fast to decay back to a mean imputation of the missing data modality.

Phased-LSTM The PHASED-LSTM (Neil et al., 2016) introduced a biologically inspired time dependent gating mechanism of a Long short-term RNN cell (Hochreiter & Schmidhuber, 1997). This allows the network to handle event-based sequences with irregularly spaced observations, but not unaligned measurements. We thus additionally augment the input in a similar fashion as described for the GRU-SIMPLE approach.

Interpolation Prediction Networks IP-NETWORKS (Shukla & Marlin, 2019) apply multiple semi-parametric interpolation schemes to irregularly-sampled time series to obtain regularly-sampled representations that cover long-term trends, transients, and also sampling information. The parameters of the interpolation network are trained with the classifier in an end-to-end fashion.

Transformer In the TRANSFORMER architecture (Vaswani et al., 2017), the elements of a sequence are encoded simultaneously and information between sequence elements is captured using Multi-Head-Attention blocks. Transformers are typically used for sequence-to-sequence modelling tasks. In our setup, we adapted them to classification tasks by mean-aggregating the final representation. This representation is then fed into a one-layer MLP to predict logits for the individual classes.

A.3. Implementation details

All experiments were run using `tensorflow 1.15.2` and training was performed on `NVIDIA Geforce GTX 1080Ti` GPUs. In order to allow a fair comparison between methods, the input processing pipeline employed caching of model-specific representations and transformations of the data.

In contrast, due to the high complexity of the LATENT-ODE model, we relied on the implementation provided by the authors and introduced our datasets into their code. This introduces the following differences between the evaluation of LATENT-ODE compared to the other methods: 1. input processing pipeline is not cached 2. model code is written in PyTorch 3. due to an order of magnitude higher runtime, a thorough hyperparameter search was not feasible . This can introduce biases both in terms of runtime and performance compared to the other methods.

A.4. Training, Model Architectures, and Hyperparameter Search

General All models were trained using the Adam optimizer (Kingma & Ba, 2015), while log-uniformly sampling the learning rate between 0.01 and 0.0001. Further, the batch size of all methods was sampled from the values $(32, 64, 128, 256, 512)$.

Recurrent neural networks For the RNN based methods (GRU-SIMPLE, PHASED-LSTM, GRU-D and IP-NETS), the number of units was sampled from the values $(32, 64, 128, 256, 512, 1024)$. Further, recurrent dropout and input dropout were sampled from the values $(0.0, 0.1, 0.2, 0.3, 0.4)$. For the PHASED-LSTM method, however, we did not apply dropout to the recurrent state and the inputs, as the learnt frequencies were hypothesized to fulfil a similar function as dropout (Neil et al., 2016). We additionally sample parameters that are specific to PHASED-LSTM: if peephole connections should be used, the leak rate from $(0.001, 0.005, 0.01)$ and the maximal wavelength for initializing the hidden state phases from the range $(10, 100, 1000)$. For IP-NETS, we additionally sample the imputation stepsize uniformly from the range $(0.5, 1., 2.5, 5.)$ and the fraction of reconstructed data points from $(0.05, 0.1, 0.2, 0.5, 0.75)$.

Static variables were handled by computing the initial hidden state of the RNNs conditional on the static variables. For all methods, the computation was performed using a one-hidden-layer neural network with the number of hidden units set to the number of hidden units in the RNN.

SEFT-Attn We vary the number of layers, dropout in between the layers and the number of nodes per layer for both the encoding network h_θ and the aggregation network g_ψ from the same ranges. The number of layers is randomly sampled between 1 and 5, the number of nodes in a layer are uniformly sampled from the range $(16, 32, 64, 128, 256, 512)$ and the dropout fraction is sampled from the values $(0.0, 0.1, 0.2, 0.3)$. The width of the embedding space prior to aggregation is sampled from the values $(32, 64, 128, 256, 512, 1024, 2048)$. The aggregation function was set to be `sum` as described in the text. The number of dimensions used for the positional embedding τ is selected uniformly from $(4, 8, 16)$ and t, i.e. the maximum time scale, was selected from the values $(10, 100, 1000)$. The attention network f' was set to always use *mean* aggregation. Furthermore, we use a constant architecture for the attention network f' with 2 layers, 64 nodes per layer, 4 heads and a dimensionality of the dot product space d of 128. We sample the amount of attention dropout uniformly from the values $(0.0, 0.1, 0.25, 0.5)$.

Transformer We utilize the same model architecture as defined in Vaswani et al. (2017), where we use an MLP with a single hidden layer as a feed-forward network, with dimensionality of the hidden layer selected to be twice the model dimensionality. The Transformer architecture was applied to the time series by concatenating the vectors of each time point with a measurement indicator. If no value was measured, input was set to zero for this modality. The parameters for the Transformer network were sampled according to the following criteria: the dimensionality of the model was sampled uniformly from the values $(64, 128, 256, 512, 1024)$, the number of attention heads per layer from the values $(2, 4, 8)$, and the number of layers from the range $[1, 6] \in \mathbb{N}$. Moreover, we sampled the amount of dropout of the residual connections and the amount of attention dropout uniformly from the values $(0.0, 0.1, 0.2, 0.3, 0.5)$, and the maximal timescale for the time embedding from the values $(10, 100, 1000)$ (similar to the SeFT approach). Further, 1000 steps of warmup were applied, where the learning rate was linearly scaled from $lr_{\min} = 0$ to the learning rate $lr_{\max}$ sampled by the hyperparameter search.

Latent-ODE We utilize the implementation from Rubanova et al. (2019) and extended the evaluation metrics and datasets to fit our scenario. Due to the long training time almost an order of magnitude longer than any other method considered a thorough hyperparameter search as executed for the other methods was not possible. We thus rely on the hyperparameters selected by the authors. In particular, we use their physionet 2012 dataset settings for all datasets. For further details see Table A.5.

Selected hyperparameters In order to ensure reproducibility, the parameters selected by the hyperparameter search are shown in Table A.5 for all model dataset combinations.

Table A.5. Best hyperparameters of all models on all datasets.

Model	P-Mortality	M-Mortality	P-Sepsis
GRU-D	lr: 0.00138, bs: 512, n_units: 128, dropout: 0.1, recurrent_dropout: 0.1	lr: 0.00016, bs: 32, n_units: 256, dropout: 0.0, recurrent_dropout: 0.2	lr: 0.0069, bs: 128, n_units: 512, dropout: 0.3, recurrent_dropout: 0.3
GRU-SIMPLE	lr: 0.00022, bs: 256, n_units: 256, dropout: 0.0, recurrent_dropout: 0.0	lr: 0.00011, bs: 32, n_units: 512, dropout: 0.3, recurrent_dropout: 0.4	lr: 0.00024, bs: 64, n_units: 1024, dropout: 0.3, recurrent_dropout: 0.3
IP-NETS	lr: 0.00035, bs: 32, n_units: 32, dropout: 0.4, recurrent_dropout: 0.3, imputation_stepsize: 1.0, reconst_fraction: 0.75	lr: 0.00062, bs: 16, n_units: 256, dropout: 0.2, recurrent_dropout: 0.1, imputation_stepsize: 1.0, reconst_fraction: 0.2	lr: 0.0008, bs: 16, n_units: 32, dropout: 0.3, recurrent_dropout: 0.4, imputation_stepsize: 1.0, reconst_fraction: 0.5
TRANSFORMER	lr: 0.00567, bs: 256, warmup_steps: 1000, n_dims: 512, n_heads: 2, n_layers: 1, dropout: 0.3, attn_dropout: 0.3, aggregation_fn: max, max_timescale: 1000.0	lr: 0.00204, bs: 256, warmup_steps: 1000, n_dims: 512, n_heads: 8, n_layers: 2, dropout: 0.4, attn_dropout: 0.0, aggregation_fn: mean, max_timescale: 100.0	lr: 0.00027, bs: 128, warmup_steps: 1000, n_dims: 128, n_heads: 2, n_layers: 4, dropout: 0.1, attn_dropout: 0.4, aggregation_fn: mean, max_timescale: 100.0
PHASED-LSTM	lr: 0.00262, bs: 256, n_units: 128, use_peepholes: True, leak: 0.01, period_init_max: 1000.0	lr: 0.00576, bs: 32, n_units: 1024, use_peepholes: False, leak: 0.01, period_init_max: 1000.0	lr: 0.00069, bs: 32, n_units: 512, use_peepholes: False, leak: 0.001, period_init_max: 100.0
LATENT-ODE	optimizer: Adamax, lr_schedule: exponential decay, lr: 0.01, bs: 50, rec-dims: 40, rec-layers: 3 gen-layers: 3, units: 50, gru-units: 50, quantization: 0.016, classification: True, reconstruction: True	optimizer: Adamax, lr_schedule: exponential decay, lr: 0.01, bs: 50, rec-dims: 40, rec-layers: 3 gen-layers: 3, units: 50, gru-units: 50, quantization: 0.016, classification: True, reconstruction: True	optimizer: Adamax, lr_schedule: exponential decay, lr: 0.01, bs: 50, rec-dims: 40, rec-layers: 3, gen-layers: 3, units: 50, gru-units: 50, quantization: 1, classification: True, reconstruction: True
SEFT-ATTN	lr: 0.00081, bs: 512, n_phi_layers: 4, phi_width: 128, phi_dropout: 0.2, n_psi_layers: 2, psi_width: 64, psi_latent_width: 128, dot_prod_dim: 128, n_heads: 4, attn_dropout: 0.5, latent_width: 32, n_rho_layers: 2, rho_width: 512, rho_dropout: 0.0, max_timescale: 100.0, n_positional_dims: 4	lr: 0.00245, bs: 512, n_phi_layers: 3, phi_width: 64, phi_dropout: 0.1, n_psi_layers: 2, psi_width: 64, psi_latent_width: 128, dot_prod_dim: 128, n_heads: 4, attn_dropout: 0.1, latent_width: 256, n_rho_layers: 2, rho_width: 512, rho_dropout: 0.1, max_timescale: 1000.0, n_positional_dims: 8	lr: 0.00011, bs: 64, n_phi_layers: 4, phi_width: 32, phi_dropout: 0.0, n_psi_layers: 2, psi_width: 64, psi_latent_width: 128, dot_prod_dim: 128, n_heads: 4, attn_dropout: 0.1, latent_width: 512, n_rho_layers: 3, rho_width: 128, rho_dropout: 0.0, max_timescale: 10.0, n_positional_dims: 16