Appendix A: HiRID Dataset Details

Dataset description

The HiRID ICU data is provided by the University Hospital Bern, Bern, Switzerland. The dataset consists of 33,905 patients whose length of stays in the ICU range from 0 to 28 days. A more detailed summary of the HiRID cohort statistics can be found in Table 1.

For all patients, a variety of clinical measurements are collected. It represents a set of 710 variables that can be categorized into the following types:

- Demographics (e.g.: Sex, Age, Height)
- Bedside vital signs (e.g.: Heart rate)
- Settings of medical devices (e.g.: Mechanical ventilation)
- Manual observations (e.g.: Urine output)
- Lab measurements (e.g.: Lactate)
- Treatments (e.g.: Vasopressor agents)

One notable difference between these types of measurements is the resolution at which they are provided. Bedside vital signs are provided at regular intervals of 2 min, whereas lab measurements are only available every couple of hours at best. The frequency of recording discrepancy existing between different variables yields unique challenges for machine learning models. Further details about the available clinical measurements can be found in the official documentation of HiRID[1] as well as in our software repository[2]

Data preprocessing

Artifact removal The HiRID data when directly exported from the data management system contains various types of artifacts. The most common one concerns measurement values that are out of the normal range defined by clinicians. Another artifact that we observed is that the time of the first measurement record of cumulative variables is at noon by default, which could be much earlier than the admission time of the patient. This, if not taken into account correctly, will affect the rate calculation for those cumulative variables. To solve these timestamp issues, we clip the time of the first record of the cumulative variables to the ICU admission time of the patient.

Variable merging There are often various variables in the ICU EHRs that have the same or similar clinical meaning. As an example, there exist three variables for temperature, namely core body temperature, auxiliary temperature, and rectal temperature. Usually, only one or few variables from one medical concept are measured for a patient, therefore, merging variables with the same clinical concept into one meta-variable reduces the sparsity of the feature matrices. Another advantage is that machine learning models trained on medical concept features are more transferable to other hospitals than those trained on the original clinical variables because hospitals usually have their own

[1] URL for the official HiRID documentation `https://hirid.intensivecare.ai/`

[2] Table containing all information for each variable: `https://github.com/ratschlab/HIRID-ICU-Benchmark/blob/master/preprocessing/resources/varref.tsv`

35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks.

Table 1: Cohort statistics of the public HiRID dataset v1.1.1, as released on Physionet, which was used for the HiRID-ICU benchmark

Sex	Female		12,138	patients
	Male		21,767	patients
Age group	[20,30)		1,042	patients
	[30,40)		1,278	patients
	[40,50)		2,649	patients
	[50,60)		5,194	patients
	[60,70)		8,241	patients
	[70,80)		9,445	patients
	[80,90)		5,534	patients
	[90,100)		522	patients
Discharge status	Alive		31,604	patients
	Dead		2,062	patients
	Unknown		239	patients
APACHE group (Patient phenotype)	Cardiovascular	Surgical	8,125	patients
		Non-surgical	4,356	patients
	Neurologic	Surgical	3,938	patients
		Non-surgical	6,014	patients
	Gastrointestinal	Surgical	1,768	patients
		Non-surgical	1,918	patients
	Respiratory	Surgical	631	patients
		Non-surgical	2,399	patients
	Trauma	Surgical	239	patients
		Non-surgical	1,522	patients
	Other medical diseases		1,428	patients
	Other surgical		568	patients
	Metabolic/Endocrinology		630	patients
	Hematologic		99	patients
	Renal surgical		81	patients
	Unknown		189	patients
Length of stay	Median		0.95	days
	Range		(0,28]	days

variable coding scheme. We hence incorporated variable-merging in our first pre-processing step. A challenge that arises from merging pharmaceutical variables is the dosage normalization across drugs with similar active ingredients. Therefore, instead of using the drug dosage as features, which likely contain invalid information, we use "presence/absence" binary indicators of drugs as features for the machine learning models.

Definition of pharmaceutical acting periods When a patient has been administrated a drug at a time point, only this time point contains a measurement. However, each drug is active for a specific duration after this administration time. For this reason, we ensure to propagate the measurement for a duration corresponding to the drug acting period. We set the value back to zero after this period ensuring compatibility with forward filling imputation.

Conversion of cumulative values to rates There are a few fluid output variables whose measurement values are recorded as cumulative every 12 hours starting from noon on the ICU admission day. Since the cumulative signals are periodic, we converted them to hourly fluid rates which are more interpretable and amenable as machine learning features and for the endpoint definition.

Endpoint definition and statistics

Endpoint extraction algorithms

Our benchmark suite contains a range of clinically relevant tasks chosen to range over a spectrum of different properties of tasks, e.g. task type from the machine learning point-of-view (regression, binary classification, multi-class classification), point of the ICU stay at which the prediction is made (fixed time-point vs. predictions made throughout the stay), as well as the degree of class imbalance (highly imbalanced tasks vs. more balanced tasks). While in the main paper we describe the main idea of each task, the full definition and implementation details of the extraction algorithm are described below.

Dependencies on previous pipeline stages. Static information about the patient is extracted from the `static.parquet` file, in which the mortality status and the APACHE-II/IV codes for the admission are stored. This file is derived from the `general_table.parquet` and `observations` tables, where APACHE II/IV codes are stored, during the data preprocessing pipeline stage. Respiratory/kidney/circulatory failure labels are generated from the merged stage of the data, via an intermediate imputation step, in which measurements are forward filled to statistics about the number of real measurements in time grid intervals were pre-computed. These simplified the implementation of the endpoints, whose extraction algorithms use a combination of imputed values and real measurement detection strategies.

Label masking on presence of vital sign monitoring. All tasks described below were additionally masked with a condition on a current connection to vital sign monitors. This implied setting a valid label (according to the below task definitions) to invalid (`NaN`) if the patient had no heart rate measurement (`vm1`) in the 15 minutes surrounding the time grid point. Since the expected frequency of heart rate measurements is 1 sample / 2 minutes, it is likely the patient is disconnected when this condition is satisfied, and no reliable predictions on their state can be made.

Patient phenotyping. To derive the APACHE diagnostic group of the patient, the two fields `APACHE-II group` and `APACHE-IV group` of the static HiRID data are used. To map these two fields to a standardized encoding, the conversion Table 2 is used. If the patient had a valid `APACHE-II group` entry and its code was mapped in the table, the target `APACHE group` code was used. Otherwise, we tried to fall back to the `APACHE-IV group` entry, where again if it was present and mapped, the corresponding target `APACHE group` code was used. If still no code could be extracted, the prediction task was defined as invalid for the patient. The label of the time-point 24h after ICU admission was set to the class category of the `APACHE group`, defining a multi-class classification problem to be solved once in the stay. If the admission was shorter than 24h, no label for the patient was assigned.

Table 2: Mapping between APACHE II and IV group codes to one unique APACHE group diagnostic group encoding, which is used for the patient phenotyping task.

APACHE II Code	APACHE IV Code	Original group name	Target group	Target group name
98	190	Cardiovascular	1	Cardiovascular
99	191	Respiratory	2	Respiratory
100	192	Gastrointestinal	3	Gastrointestinal
101	193	Neurologic	4	Neurologic
	197	Urogenital	6	Other medical diseases
102		Sepsis	6	Other medical diseases
106	198	Other	6	Other medical diseases
	206	Intoxication	6	Other medical diseases
103	194	Trauma not surgical	7	Trauma not surgical
104	195	Metabolic/Endocrinology	8	Metabolic/Endocrinology
105	196	Hematologic	9	Hematologic
107	199	(Cardio)vascular surgical	11	(Cardio)vascular surgical
108	201	Respiratory surgical	12	Respiratory surgical
109	200	Gastrointestinal surgical	13	Gastrointestinal surgical
110	202	Neurologic surgical	14	Neurologic surgical
111	203	Trauma surgical	15	Trauma surgical
112	204	Renal surgical	16	Renal surgical
113		Gynacology surgical	17	Other surgical
114		Orthopedics surgical	17	Other surgical
	205	Other surgical	17	Other surgical
		Unknown	18	Unknown

Mortality. First, the ICU mortality label was extracted from the general data table of the HiRID database. The label of the time-point 24 hours after ICU admission was set to 1 (positive) if the patient died at the end of the stay according to this field, and 0 (negative) otherwise, defining a binary classification problem to be solved once per stay. If the admission was shorter than 24 hours, no label was assigned to the patient.

Remaining length of stay. The continuous regression label of a time-point in the ICU stay was defined as the number of hours that remain until the end-of-the-stay, i.e. the first label of the ICU stay is equal to the ICU stay length (in hours) and the last label just before dispatch is defined as 0. To determine the beginning and end of the ICU stay, the first and last observed heart rate measurements (`vm1`) were used.

Kidney function. As a basis for extracting the kidney function label, the urine output rate (`vm24`) variable and the weight variable (`vm131`) were used. The valid labels were anchored to real measurements, i.e. updates, of the urine output rate. Only time-points exactly 2h before an update of the urine output rate were assigned a prediction label. The overall urine output in the 2h after this time point was found by computing the integral of the urine output rate variable (`vm24`) over this 2h period. The overall urine output in the time interval was then normalized to the weight of the patient (unit `kg`) at the time of the prediction, and then standardized to a rate/hour. Hence, the unit of the regression output is `ml/kg/h`.

Circulatory failure. As a basis for computing the circulatory failure status of a patient at every time-point of the ICU stay, the following variables were used: Mean arterial pressure (`vm5`), arterial lactate (`vm136`) and the vasopressive agents Milrinone (`pm42`), Dobutamine (`pm41`), Levosimendan (`pm43`), Theophylline (`pm44`), Norepinephrine (`pm39`), Epinephrine (`pm40`) and Vasopressin (`pm45`). For defining the label at time t a centralized window of size 2h was anchored at t, and the lactate/MAP conditions in this window were separately analyzed. The time-point was assigned a (tentative) positive label if, for at least 2/3 of the time-points inside the 2h, the lactate condition was satisfied, and for at least 2/3 of the time-points inside the 2h, the MAP condition was satisfied. The time-points at which they have to be satisfied do not necessarily have to coincide. The conditions defining the circulatory failure state are listed below:

- **Lactate condition**. Elevated arterial lactate (> 2 mmol/l)
- **MAP/vasopressor condition**. Low MAP (< 65 mmHg) or any dose of any of the considered vasopressors (`pm39, pm40, pm41, pm42, pm43, pm44, pm45`) given to the patient, which could potentially mask a low MAP.

Using these endpoint annotations of "circulatory failure" and "no circulatory failure", a label at time point t is either (1) *invalid* (no prediction made), if the patient was in circulatory failure at t, (2) *positive* if the patient was not circulatory failure at t, but there was a new onset of circulatory failure in the next 12h, and (3) *negative* if the patient was not in circulatory failure at t, and there was no onset of circulatory failure in the next 12h.

Respiratory failure. Estimating the respiratory failure status of a patient at a given time-point involves estimating (1) their instantaneous FiO_2 value, (2) their instantaneous PaO_2 value and computing the P/F ratio as P/F = PaO_2/FiO_2, and then labeling the time series according to the threshold P/F = 300 mmHg, to define (mild) failure and stability periods.

We distinguished between the following cases to estimate the FiO_2 value at a time-point t.

- FiO_2 from ventilator: If there was a FiO_2 measurement (`vm58`) in the last 30 min and the patient was on the ventilator or the ventilation mode NIV (`vm60`) was active, then the last FiO_2 measurement (`vm58`) was directly used as the estimate.
- FiO_2 from supplementary oxygen (oxygen mask): If there was a real measurement in supplementary oxygen (`vm23`) in the last 12h, then it was forward filled and used to estimate FiO_2 via the conversion Table 3.
- FiO_2 from ambient air: An ambient air FiO_2 assumption was made, and FiO_2 was estimated as 21 %.

Table 3: Supplemental Oxygen to FiO_2 conversion table used for determining the continuous FiO_2 estimate.

Supp. oxygen [l]	FiO_2 [%]	Supp. oxygen [l]	FiO_2 [%]
1	26	9	63
2	34	10	66
3	39	11	67
4	45	12	69
5	49	13	70
6	54	14	73
7	57	15	75
8	58	>15	75

We distinguished between the following cases tor estimating the PaO_2 value at a time-point. As source variables peripheral oxygen saturation (`vm20`) and PaO_2 from ABGA (`vm140`) were used. Hereby the SpO2 variable was pre-smoothed with a percentile (75 % percentile) moving window filter of size 30 min to remove spuriously low outlier measurements.

- Real PaO_2 measurement: If there was a real PaO_2 (`vm140`) from an arterial blood gas analysis available in the last 30 minutes, then the estimate was defined directly by the measurement.
- Ellis estimate from SpO_2: Otherwise the last measurement of peripheral oxygen saturation measured by pulse oximetry (`vm20`) was used to compute an estimate of PaO_2 according to Ellis et al. [1]. If there is no real SpO_2 measurement in the last 24h, a default value of 98 % was assumed for the estimation.

First, the resulting PaO_2 estimate was smoothed with a Nadaraya-Watson kernel smoother with a bandwidth of 20 min. Then, each so-derived PaO_2 estimate was weighted by a squared exponential kernel and converted to the closest plausible PaO_2 measurement. The scale of the kernel function was chosen such that a1h distance with the measurement time resulted in a weight of $\frac{1}{3}$.

The time series was labeled for respiratory failure by using forward-facing windows anchored at time-point t. A patient is considered as being in respiratory failure at t if, for at least 2/3 time-points of the next 2h, either of the following conditions hold:

- The estimated P/F ratio is < 300 mmHg and the patient is not on mechanical ventilation **OR**
- The estimated P/F ratio is < 300 mmHg and the patient is on mechanical ventilation, but the PEEP value is not densely measured (no real measurement in 30 min) **OR**
- The estimated P/F ratio is < 300 mmHg and the patient is on mechanical ventilation, and the PEEP is densely measured and the PEEP is > 4 mmHg.

After the definition of event periods, their edges were corrected according to estimates of the P/F ratio at the given time points. If before the event P/F< 300 mmHg, the event is further extended in the past. Likewise, if the P/F> 300 mmHg condition holds after the event, it is extended into the future.

Finally, small events shorter than 4h, preceded and succeeded by two stability periods, of which one is longer than 4h are deleted. This is because these likely represent intermittent/spurious instances of respiratory failure. Conversely, short stability periods shorter than 4h, preceded and followed by periods of respiratory failure, of which one is longer than 4h, are defined as respiratory failure. Likely, during these periods, the patient does not recover from the failure in a clinical sense.

Statistics on annotated failure events and labels

Below we display statistics on the data resulting from the endpoint stage, and from the label stage, which was used directly as inputs for the machine learning models.

Respiratory failure

Table 4: Summary statistics about annotated respiratory failure (oxygenation failure with a P/F ratio <300 mmHg) events in the pre-processed data-set. The label prevalence refers to the observed probability of developing respiratory failure in the next 12h, given that the patient is currently stable.

Respiratory failure		
Patients with events	83.08 %	28,157 admissions
Per-time prevalence of resp. failure	61.83 %	
Median # events per patient (if ≥ 1 event) [IQR]	1 [1-2]	
Median event duration [IQR]	9.5 [4.1-20.6] hours	
Median time to first event (if ≥ 1 event) [IQR]	1.58 [0-11] hours	
Median gap length between two events [IQR]	4.4 [2.1-11] hours	
Overall label prevalence (Stable->Failure 12h)	8.6 %	
Median label prevalence p/patient (if ≥ 1 event) [IQR]	0 [0-60.1] %	

Circulatory failure

Table 5: Summary statistics about annotated circulatory failure (elevated lactate and abnormally low MAP or vasopressive agents) events in the pre-processed data-set. The label prevalence refers to the observed probability of developing circulatory failure in the next 12h, given that the patient is currently stable.

Circulatory failure		
Patients with events	25.58 %	8,669 admissions
Per-time prevalence of circ. failure	6.70 %	
Median # events per patient (if $\geq$ 1 event) [IQR]	1 [1-2]	
Median event duration [IQR]	2.9 [1.1-7.6] hours	
Median time to first event (if $\geq$ 1 event) [IQR]	1.9 [0.5-7.3] hours	
Median gap length between two events [IQR]	3.3 [1.3-9.6] hours	
Overall label prevalence (Stable->Failure 12h)	1.4 %	
Median label prevalence p/patient (if $\geq$ 1 event) [IQR]	3.8 [0-19.1] %	

Mortality prediction / Patient phenotyping

Table 6: Summary statistics about the mortality prediction and patient phenotyping tasks defined at 24h after ICU admission. The last column shows the number of admissions assigned this label.

Percentage of ICU stays with length >24h	44.1 %	14,962 admissions
Mortality prediction		
Overall label prevalence (Mortality at 24 hours)	8.38 %	
Patient phenotyping		
Prevalence 'Neurologic' (4)	23.6 %	3,511 admissions
Prevalence '(Cardio)vascular' (1)	15.9 %	2,367 admissions
Prevalence '(Cardio)vascular surgical (11)	12.4 %	1,848 admissions
Prevalence 'Respiratory' (2)	11.0 %	1,635 admissions
Prevalence 'Neurologic surgical' (14)	7.4 %	1,106 admissions
Prevalence 'Trauma not surgical' (7)	6.7 %	990 admissions
Prevalence 'Gastrointestinal' (3)	6.3 %	930 admissions
Prevalence 'Other medical disease' (6)	5.2 %	778 admissions
Prevalence 'Gastrointestinal surgical' (13)	4.8 %	713 admissions
Prevalence 'Metabolic/Endocrinology' (8)	2.3 %	344 admissions
Prevalence 'Respiratory surgical' (12)	1.7 %	246 admissions
Prevalence 'Other surgical' (17)	1.2 %	181 admissions
Prevalence 'Trauma surgical' (15)	1.0 %	148 admissions
Prevalence 'Hematologic' (9)	0.4 %	57 admissions
Prevalence 'Renal surgical' (16)	0.2 %	27 admissions

Kidney function / Remaining-length-of-stay regression

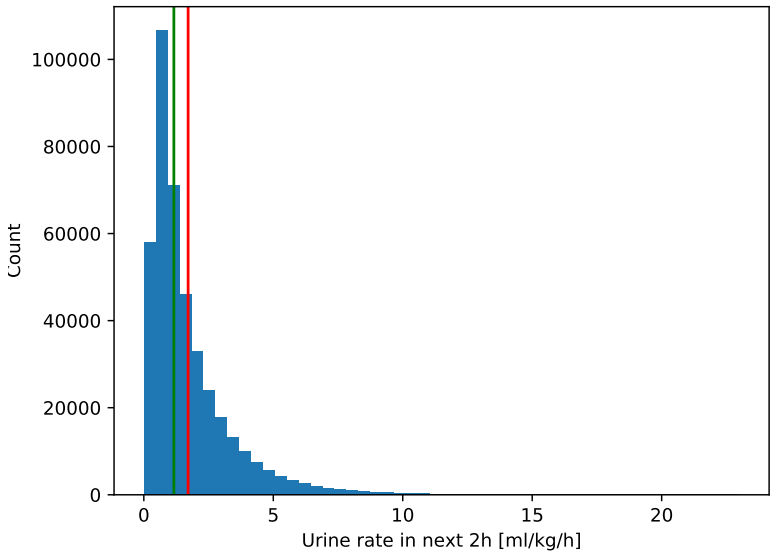

Figure 1: Marginal distribution of the urine regression labels. The vertical red line denotes the mean, and the vertical green line the median of the distribution.

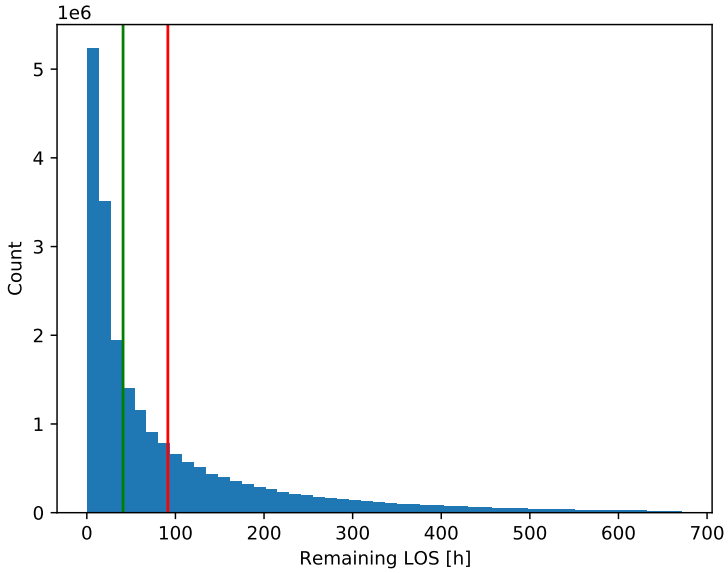

Figure 2: Marginal distribution of the remaining length-of-stay regression labels. The vertical red line denotes the mean, and the vertical green line the median of the distribution.

References

[1] RK Ellis. Determination of po2 from saturation. *Journal of applied physiology*, 67(2):902–902, 1989.

Appendix B: HiRID-ICU Pipeline Details

In this section we provide a description of the full pipeline of the HiRID-ICU Benchmark and go through the main steps in detail. We also address the ML Reproducibility checklist questions[1].

0. *Data loading*. Request access and then download the data from `https://physionet.org/content/hirid/1.1.1/` (see how to request access to the data in the README section of the Software Repository[2].) The data is available under the PhysioNet Credentialed Health Data License 1.5.0.
1. *Data preprocessing*. The preprocessing step is described in detail in the APPENDIX A: DATASET DETAILS. It is composed of several stages: merge stage, resample stage, feature extraction stage.
2. *Task implementation*. In this stage we construct the state annotations and machine learning labels for our predefined set of tasks.
3. *Model training and evaluation.* We select and train both ML and DL models on the training and validation sets. We evaluate model performance on the test set.

The script `run.py` in the `icu_benchmarks` folder unites three stages of the pipeline which can be used individually: Preprocessing (Section B.1), Task implementation (Section B.2), and Evaluation (Section B.3). Each stage has an associated help function. Below we describe each stage in detail.

Preprocessing

The method `preprocess` unites all necessary steps to generate the input for the Deep Learning (DL) and Machine Learning (ML) models:

1. `run_merge_step` — Converts ICU data from table format to a matrix format that can be input to the machine learning model (see details in APPENDIX A: DATASET DETAILS).
2. `run_resample_step` — Re-samples data to a regularly sampled 5min resolution.
3. `run_feature_extraction_step` — Hand-engineered feature extraction for classical machine learning models.
4. `run_build_dl` — Data splitting and preprocessing specific to ML.

Running these steps requires the following arguments:

1. `hirid-data-root` — Path to the unpacked parquet files containing the HiRID data, downloaded from Physionet.
2. `work-dir` — Path to the working directory of the user.
3. `var-ref-path` — Path to the file containing meta-data about variables necessary for our pre-processing pipeline.
4. `split-path` — Path to the exact split of the data for model training and evaluation.
5. `nr-workers` — Number of workers to use during training. It depends on the user hardware capacity.

[1] ML Reproducibility checklist

[2] `https://github.com/ratschlab/HIRID-ICU-Benchmark/`

To run the whole preprocessing, the command below is used:

```
icu-benchmarks preprocess --hirid-data-root [path to unpacked parquet files] \
                          --work-dir [output directory] \
                          --var-ref-path ./preprocessing/resources/varref.tsv \
                          --split-path ./preprocessing/resources/split.tsv \
                          --nr-workers 8
```

Task Implementation

The second stage of the pipeline, consisting of extracting task labels, is constructed around the following three steps:

1. `imputation_for_endpoints` — Imputes data for endpoint generation.
2. `generate_endpoints` — Generates the endpoints i.e. verify if conditions for events are matched at each timestep.
3. `generate_labels` — Extract final labels from endpoints.

As mentioned in the paper, we construct a set of 6 tasks, relevant for healthcare workers:

1. `Mortality_At24Hours` — Mortality prediction at 24h after admission in the ICU (further Mortality).
2. `Phenotyping_APACHEGroup` — Patient APACHE group classification 24h after admission in the ICU (further Patient Phenotyping).
3. `Dynamic_CircFailure_12Hours` — Continuous prediction of occurrence of circulatory failure in the next 12 hours (further Circulatory Failure).
4. `Dynamic_RespFailure_12Hour` — Continuous prediction of occurrence of respiratory failure in the next 12 hours (further Respiratory Failure).
5. `Remaining_LOS_Reg` — Continuous prediction of the remaining stay duration (further Remaining LOS).
6. `Dynamic_UrineOutput_2Hours_Reg` — Continuous prediction of patient urine production in the next 2h (further Kidney Function).

Model Training and Evaluation

In the final part of the pipeline, we provide the user with a choice of model to train:

1. from the list of DL models: Transformer, LSTM, GRU and Temporal CNN. All these models use the class `DLWrapper` defining methods mechanics for deep learning models' training and evaluation.
2. from the list of the ML Models: Gradient Boosting method and Logistic Regression. In the same manner as DL models, these models have a `MLWrapper` class.

All models are trained on the training set. The validation set is used for early stopping and model selection through a random search. After the training procedure, the model performance is evaluated on the test set.

To run the training, the following command is used (as an example, we provide here the command for GRU model training for the `Dynamic_CircFailure_12Hours` task):

```
icu-benchmarks train -c configs/hirid/Classification/GRU.gin \
                     -l [path to logdir] \
                     -t Dynamic_CircFailure_12Hours\
                     -sd 1111
```

- Argument `-l` specifies the path to the directory where the trained model and meta-data will be stored.
- Argument `-t` specifies the selected task.
- Argument `-sd` specifies the random seed.
- Argument `-c` specifies path to `gin-config` configuration file. This file should include the path to the data after preprocessing, the task, the model with all hyper-parameters.

An example of the configuration file for the GRU model is provided below[3].

```
import gin.torch.external_configurables
import icu_benchmarks.models.wrappers
import icu_benchmarks.models.encoders
import icu_benchmarks.models.utils
import icu_benchmarks.data.loader


EMB = 231
LR = 3e-4
HIDDEN = 64
NUM_CLASSES = 2
DEPTH = 1
BS = 64
EPOCHS = 1000
TASK = 'Dynamic_CircFailure_12Hours'
RES = 1
RES_LAB = 1
MAXLEN = 2016
LOSS_WEIGHT = None

# Train params
train_common.model = @DLWrapper()
train_common.dataset_fn = @ICUVariableLengthDataset
train_common.data_path = [path to data] # TODO add by user
train_common.weight = %LOSS_WEIGHT
train_common.do_test = True

DLWrapper.encoder = @GRU()
DLWrapper.loss = @cross_entropy
DLWrapper.optimizer_fn = @Adam
DLWrapper.train.epochs = %EPOCHS
DLWrapper.train.batch_size = %BS
DLWrapper.train.patience = 10
DLWrapper.train.min_delta = 1e-4


ICUVariableLengthLoaderTables.splits = ['train','test','val']
ICUVariableLengthLoaderTables.task = %TASK
ICUVariableLengthLoaderTables.data_resampling = %RES
ICUVariableLengthLoaderTables.label_resampling = %RES_LAB
ICUVariableLengthDataset.maxlen = %MAXLEN

# Optimizer params
Adam.lr = %LR
Adam.weight_decay = 1e-6

# Encoder params
GRU.input_dim = %EMB
GRU.hidden_dim = %HIDDEN
GRU.layer_dim = %DEPTH
GRU.num_classes = %NUM_CLASSES
```

[3]See examples of configuration files for all the tasks and models in the `config` folder of the software repository

We define all macros at the top of the file and use them to configure some classes and functions. Among them, we have two custom function for data loading:

- `ICUVariableLengthLoaderTables` — Main Loader class allowing to sample patient from the data.
- `ICUVariableLengthDataset` — `pytorch` dataset wrapper to ship data to the GPU.

Technical Specifics for Reproducibility

Libraries A full list of libraries and the version we used is provided in the `environment.yml` file. The most important ones are the following: pytorch 1.8.1, scikit-learn 0.24.1, ignite 0.4.4, CUDA 10.2.89, cudNN 7.6.5.

Infrastructure We follow all guidelines provided by `pytorch` documentation to ensure reproducibility of our results. However, reproducibility across devices is not ensured. Thus we provide here the characteristics of our infrastructure. We trained deep learning methods on a single `NVIDIA RTX2080Ti` with a `Xeon E5-2630v4` core. For other methods we trained models on either `Xeon E5-2697v4` cores or `Xeon Gold 6140` cores.

Method For the main experiment, 10 different random initializations were used for each model; For the ablation study, 5 were used. For all models, we tuned specific hyper-parameters using random search with 100 iterations for stay-level tasks and 50 iterations for the dynamic ones. Each random set of parameters was run with 3 different random initializations. Early stopping with 10 step patience on the loss was used as a stopping criterion. We then chose hyper-parameters on AUPRC, balanced accuracy, and MAE for respectively, binary, multi-class, and regression tasks.

Complexity of training Among the deep learning methods, transformer memory complexity, with regard to the sequence length, is quadratic. This has forced us to reduce both batch size and the number of parameters of this model for the dynamic tasks. Also, to ensure reproducibility, using deterministic algorithms particularly slows down TCN training. With our hardware, training deep learning methods takes less than 1h for stay-level tasks and less than 6h for dynamic ones on a single GPU. To reduce RAM consumption we provide the possibility to load data only at inference time with parameter `on_RAM` in our loader. In that case, training is slightly slower but requires only around 8GB of RAM.

On the other hand, ML methods are faster to train but more RAM-consuming. Training any model takes less than 4h on 4 CPUs. However, peak memory can exceed 100GB when using hand-engineered features.

Hyperparameters Search

In this section we detail the range of hyperparameters we searched over and the one we used for our experiments.

Common Hyperparameters

For the training of DL methods, we used certain parameters across multiple tasks and architecture as reported in Table 1. For the ML methods, we fixed certain parameters as in the original HiRID paper. For LGBM, we set the bagging frequency to 1, the number of leaves to $2^{\textbf{depth}}$, and the minimum number of children per leaf to 1000. In the rest of the section, we report the hyperparameters we searched over in our experiments.

Models	Optimizer	Weight Decay	Batch Size Online	Batch Size Stay Level
LSTM	Adam	1e-6	64	64
GRU	Adam	1e-6	64	64
TCN	Adam	1e-6	64	64
Transformer	Adam	1e-6	8	16

Table 1: Fixed Hyperparameters for DL Methods

Deep Learning model Hyperparameters

In this section, we detail the range of hyperparameters considered for LSTM, GRU, TCN and transformer models.

LSTM The range of hyperparameters considered for the LSTM Model can be found in Table 2.

Task	Learning Rate	Drop-out	Depth	Hidden Dimension	Loss Weighting
Mortality	(1e-5, 3e-5, **1e-4**, 3e-4)	(0.0, **0.1**, 0.2, 0.3, 0.4)	(**1**, 2, 3)	(32, 64, **128**, 256)	(**None**, Balanced)
Phenotyping	(1e-5, 3e-5, 1e-4, **3e-4**)	(**0.0**, 0.1, 0.2, 0.3, 0.4)	(**1**, 2, 3)	(32, 64, 128, **256**)	(None, **Balanced**)
Circ. Failure	(1e-5, 3e-5, 1e-4, **3e-4**)	(0.0, 0.1, 0.2, 0.3, **0.4**)	(1, **2**, 3)	(32, 64, 128, **256**)	(**None**, Balanced)
Resp. Failure	(1e-5, 3e-5, **1e-4**, 3e-4)	(0.0, 0.1, 0.2, 0.3, **0.4**)	(1, **2**, 3)	(32, 64, 128, **256**)	(**None**, Balanced)
Urine Output	(1e-5, 3e-5, 1e-4, **3e-4**)	(0.0, 0.1, 0.2, **0.3**, 0.4)	(1, 2, **3**)	(32, 64, **128**, 256)	N.A
Rem. LOS	(1e-5, 3e-5, 1e-4, **3e-4**)	(0.0, 0.1, **0.2**, 0.3, 0.4)	(1, 2, **3**)	(32, 64, 128, **256**)	N.A

Table 2: Hyperparameter search range for LSTM. In **bold** are the parameters we selected using random search.

GRU The range of hyperparameters considered for the GRU Model can be found in Table 3.

Task	Learning Rate	Drop-out	Depth	Hidden Dimension	Loss Weighting
Mortality	(1e-5, 3e-5, 1e-4, **3e-4**)	(**0.0**, 0.1, 0.2, 0.3, 0.4)	(1, **2**, 3)	(32, **64**, 128, 256)	(**None**, Balanced)
Phenotyping	(**1e-5**, 3e-5, 1e-4, 3e-4)	(0.0, 0.1, 0.2, 0.3, **0.4**)	(**1**, 2, 3)	(32, 64, 128, **256**)	(None, **Balanced**)
Circ. Failure	(1e-5, 3e-5, 1e-4, **3e-4**)	(0.0, 0.1, **0.2**, 0.3, 0.4)	(1, **2**, 3)	(32, 64, 128, **256**)	(**None**, Balanced)
Resp.Failure	(1e-5, 3e-5, **1e-4**, 3e-4)	(0.0, 0.1, 0.2, 0.3, **0.4**)	(1, 2, **3**)	(32, 64, 128, **256**)	(**None**, Balanced)
Urine Output	(1e-5, 3e-5, 1e-4, **3e-4**)	(0.0, 0.1, 0.2, **0.3**, 0.4)	(1, 2, **3**)	(32, 64, 128, **256**)	N.A
Rem. LOS	(1e-5, 3e-5, 1e-4, **3e-4**)	(0.0, 0.1, 0.2, **0.3**, 0.4)	(1, **2**, 3)	(32, 64, **128**, 256)	N.A

Table 3: Hyperparameter search range for GRU. In **bold** are the parameters we selected using random search.

TCN The range of hyperparameters considered for the TCN Model can be found in Table 4. Note that we do not consider any depth factor as it is fully determined by the kernel size and the sequence length.

Task	Learning Rate	Drop-out	Kernel	Hidden Dimension	Loss Weighting
Mortality	(1e-5, 3e-5, **1e-4**, 3e-4)	(**0.0**, 0.1, 0.2, 0.3, 0.4)	(2, **4**, 8, 16, 32)	(32, 64, 128, **256**)	(**None**, Balanced)
Phenotyping	(1e-5, 3e-5, **1e-4**, 3e-4)	(0.0, 0.1, **0.2**, 0.3, 0.4)	(2, 4, 8, 16, **32**)	(32, 64, **128**, 256)	(None,**Balanced**)
Circ. Failure	(1e-5, 3e-5, 1e-4, **3e-4**)	(0.0, **0.1**, 0.2, 0.3, 0.4)	(2, **4**, 8, 16, 32)	(32, 64, 128, **256**)	(**None**, Balanced)
Resp. Failure	(1e-5, 3e-5, 1e-4, **3e-4**)	(0.0, 0.1, 0.2, 0.3, **0.4**)	(2, 4, **8**, 16, 32)	(32, **64**, 128, 256)	(**None**, Balanced)
Urine Output	(1e-5, 3e-5, 1e-4, **3e-4**)	(0.0, 0.1, **0.2**, 0.3, 0.4)	(2, 4, **8**, 16, 32)	(32, 64, 128, **256**)	N.A
Rem. LOS	(1e-5, 3e-5, 1e-4, **3e-4**)	(0.0, 0.1, 0.2, **0.3**, 0.4)	(2, 4, 8, 16, **32**)	(32, 64, **128**, 256)	N.A

Table 4: Hyperparameter search range for TCN. In **bold** are the parameters we selected using random search.

Transformer The range of hyperparameters considered for Transformer Model can be found in Table 5 and Table 6. We considered smaller parameters for online tasks due to GPU memory limitations.

Task	Learning Rate	Attention Drop-out	Nb. Heads	Depth
Mortality	(**1e-5**, 3e-5, 1e-4, 3e-4)	(0.0, **0.1**, 0.2, 0.3, 0.4)	(1, 2, **4**, 8)	(1, **2**, 3)
Phenotyping	(1e-5, 3e-5, 1e-4, **3e-4**)	(0.0, 0.1, **0.2**, 0.3, 0.4)	(1, 2, 4, **8**)	(1, **2**, 3)
Circ. Failure	(1e-5, **3e-5**, 1e-4, 3e-4)	(**0.0**, 0.1, 0.2, 0.3, 0.4)	(**1**, 2, 4)	(1, 2, **3**)
Resp. Failure	(1e-5, 3e-5, 1e-4, **3e-4**)	(**0.0**, 0.1, 0.2, 0.3, 0.4)	(**1**, 2, 4)	(1, **2**, 3)
Urine Output	(1e-5, **3e-5**, 1e-4, 3e-4)	(0.0, **0.1**, 0.2, 0.3, 0.4)	(**1**, 2, 4, 8)	**1**
Rem. LOS	(1e-5, 3e-5, 1e-4, **3e-4**)	(**0.0**, 0.1, 0.2, 0.3, 0.4)	(**1**, 2, 4, 8)	**1**

Table 5: Hyperparameter search range for the Transformer. In **bold** are the parameters we selected using random search.

Task	Attention Drop-out	Hidden Dimension	Loss Weighting
Mortality	(**0.0**, 0.1, 0.2, 0.3, 0.4)	(32, 64, 128, **256**)	(**None**, Balanced)
Phenotyping	(**0.0**, 0.1, 0.2, 0.3, 0.4)	(**32**, 64, 128, 256)	(None, **Balanced**)
Circ. Failure	(0.0, 0.1, 0.2, 0.3, **0.4**)	(32, 64, **128**)	(**None**, Balanced)
Resp. Failure	(0.0, 0.1, 0.2, **0.3**, 0.4)	(32, **64**, 128)	(**None**, Balanced)
Urine Output	(0.0, **0.1**, 0.2, 0.3, 0.4)	(32, **64**, 128)	N.A
Rem. LOS	(**0.0**, 0.1, 0.2, 0.3, 0.4)	(32, 64, **128**)	N.A

Table 6: Hyperparameter search range for Transformer. In **bold** are the parameters we selected using random search.

Machine Learning Models Hyperparameters

Gradient Boosting The range of hyperparameters considered for the gradient boosting method, LightGBM framework[4] can be found in Table 7 and 8 :

Task	Depth	Colsample_bytree[5]	Subsample[6]
Mortality	(3, 4, 5, 6, **7**)	(0.33, 0.66, **1.00**)	(**0.33**, 0.66, 1.00)
Phenotyping	(**3**, 4, 5, 6, 7)	(0.33, **0.66**, 1.00)	(0.33, **0.66**, 1.00)
Circulatory Failure	(3, **4**, 5, 6, 7)	(**0.33**, 0.66, 1.00)	(**0.33**, 0.66, 1.00)
Respiratory Failure	(3, 4, 5, 6, **7**)	(**0.33**, 0.66, 1.00)	(**0.33**, 0.66, 1.00)
Urine Output	(3,**4**, 5, 6, 7)	(0.33, 0.66, **1.00**)	(0.33, 0.66, **1.00**)
Remaining Length-of-Stay	(3, 4, 5, 6, **7**)	(**0.33**, 0.66, 1.00)	(0.33, 0.66, **1.00**)

Table 7: Hyperparameter search range for LGBM. In **bold** are the parameters we selected using random search.

Task	Depth	Colsample_bytree[7]	Subsample[8]
Mortality	(3, 4, 5, 6, **7**)	(0.33, 0.66, **1.00**)	(0.33, 0.66, **1.00**)
Phenotyping	(3, 4, **5**, 6, 7)	(**0.33**, 0.66, 1.00)	(**0.33**, 0.66, 1.00)
Circulatory Failure	(3, **4**, 5, 6, 7)	(**0.33**, 0.66, 1.00)	(0.33, **0.66**, 1.00)
Respiratory Failure	(3, 4, 5, **6**, 7)	(**0.33**, 0.66, 1.00)	(0.33, **0.66**, 1.00)
Urine Output	(3, 4, 5, **6**, 7)	(0.33, 0.66, **1.00**)	(0.33, 0.66, **1.00**)
Remaining Length-of-Stay	(3, 4, 5, 6, **7**)	(0.33, **0.66**, 1.00)	(**0.33**, 0.66, 1.00)

Table 8: Hyper-parameters range for LGBM w. features. In **bold** are the parameters we selected using random search.

Logistic Regression The search range of hyperparameters considered for Logistic Regression[9] can be found in Table 9:

Task	C	Penalty
Mortality	(0.001, 0.01, 0.1, **1**, 10)	('l1', **'l2'**)
Phenotyping	(0.001, 0.01, **0.1**, 1, 10)	('l1', **'l2'**)
Circulatory Failure	(0.001, **0.01**, 0.1, 1, 10)	('l1', **'l2'**)
Respiratory Failure	(**0.001**, 0.01, 0.1, 1, 10)	('l1', **'l2'**)

Table 9: Hyperparameter search range for Logistic Regression. In **bold** are the parameters we selected using random search.

[4] `https://lightgbm.readthedocs.io/en/latest/`

[5] Subsample ratio of columns when constructing each tree.

[6] Subsample ratio of the training instance

[7] Subsample ratio of columns when constructing each tree.

[8] Subsample ratio of the training instance

[9] scikit-learn framework