MEDS-Torch: An ML Pipeline for Inductive Experiments for EHR Medical Foundation Models

Nassim Oufattole[1] nassim@mit.edu
Teya Bergamaschi[1] teya@mit.edu
Pawel Renc[2] prenc@mgh.harvard.edu
Aleksia Kolo[1] aleksiak@mit.edu
Matthew B.A. McDermott[3*] matthew_mcdermott@hms.harvard.edu
Collin Stultz[1,2,3*] cmstultz@csail.mit.edu

[1] Massachusetts Institute of Technology, Cambridge, MA, USA
[2] Massachusetts General Hospital, Boston, MA, USA
[3] Harvard Medical School, Boston, MA, USA

Abstract

We introduce MEDS-Torch, a scalable and extensible pipeline for inductive experiments with sequence models on medical datasets adhering to the MEDS format—a universal schema for medical time series data. Using this pipeline, we systematically compare three tokenization methods (Everything In Code, Triplet, and Text Code) and evaluate five transfer learning techniques, including autoregressive generative modeling and contrastive learning variations, across multiple predictive tasks on the MIMIC-IV EHR dataset. Our empirical analysis provides actionable insights into the effectiveness of each method, demonstrating significant performance differences among tokenization and pretraining combinations. By benchmarking these approaches against fully supervised learning models, we offer practical recommendations for selecting appropriate modeling strategies in diverse healthcare settings. MEDS-Torch streamlines the process of running controlled experiments on medical datasets and promotes reproducibility and standardization in EHR research through its exclusive dependence on the MEDS schema, facilitating more effective machine learning experiments in healthcare without reliance on dataset-specific nuances.

1 Intro

Processing and modeling electronic health record (EHR) data present significant challenges due to its complexity, high dimensionality, and heterogeneity [17, 21, 22]. Developing efficient pipelines that can handle diverse medical datasets is crucial for advancing predictive analytics in healthcare. However, the challenges of developing methods that generalize across different datasets have slowed down the creation of best ML practices on EHR datasets in the context of *transfer learning methods* and *input encoding* (i.e. how raw continuous and categorical time-series EHR data is converted into a sequence of inputs passed to a sequence model such as a transformer encoder or LSTM).

*Corresponding author

NeurIPS 2024 Workshop on Time Series in the Age of Large Models.

Unlike image or text data, tabular EHR data is often high-dimensional, sparse, irregularly sampled, and contains a mix of numerical and categorical variables. Additionally, the data can be noisy, incomplete, and subject to strict privacy constraints. Methods such as forecasting have yielded mixed results—underperforming compared to contrastive and multitask pretraining methods in some studies [14, 7], yet outperforming them in other studies [12]. These experiments are often conducted on different tasks and datasets, with varying standards for input encoding and different transfer learning techniques. This inconsistency creates a gap in the literature, leaving data scientists in healthcare institutions uncertain about which methods would work best for their specific, niche datasets.

Previous attempts to apply transfer learning and tokenization methods to EHR data have been limited by the lack of standardized data formats and pipelines that generalize across datasets. Many existing methods are tailored to specific datasets or require extensive preprocessing, making them unsuitable for broader applications. Furthermore, the complexity of EHR data has hindered the development of universally applicable tokenization and modeling techniques.

In this paper, we introduce MEDS-Torch[2], a scalable pipeline designed to generalize to any medical dataset adhering to the MEDS[2] format. We systematically compare three tokenization methods for tabular EHR data and evaluate five transfer learning techniques, including variations of autoregressive generative modeling and two variations of contrastive learning. Our evaluation spans several tasks across two distinct datasets, providing comprehensive insights into the performance of each method. Additionally, we benchmark these approaches against fully supervised learning models.

Our Contributions are:

1. **Standardized EHR analysis framework**: A scalable, extensible pipeline for processing and analyzing any MEDS-formatted medical dataset, featuring efficient preprocessing and promoting standardized, reproducible research in EHR modeling (Section 2).
2. **Comprehensive tokenization and transfer learning framework**: Implementation of three tokenization methods and five transfer learning techniques, enabling systematic comparisons of different approaches for EHR modeling (Section 3).
3. **Empirical analysis and insights**: Demonstration of MEDS-Torch's capabilities through extensive experiments across multiple tasks on MIMIC-IV, providing actionable insights into the effectiveness of various tokenization and transfer learning combinations for EHR data (Section 3).

2 MEDS-Torch

MEDS-Torch is a flexible and efficient pipeline designed for advanced machine learning on Electronic Health Records (EHR) data. It provides a comprehensive suite of tools for processing, modeling, and analyzing medical time-series data in the MEDS (Medical Event Data Standard) format.

MEDS-Torch implements a variety of tokenization strategies and transfer learning methods, enabling researchers to experiment with different approaches. The pipeline supports three main tokenization methods: Everything In Code (EIC), Triplet, and Text Code. These methods transform raw MEDS data into sequences suitable for input into deep learning models (See Figure 2 in the appendix for more information). For transfer learning, MEDS-Torch offers five distinct pretraining techniques, including event-based contrastive learning [7] (around random events), order contrastive pretraining [1], value forecasting [19], and two variations of autoregressive forecasting (see appendix section A.2 for more information on these pretraining methods). This diversity allows researchers to explore and compare different approaches for their specific tasks and datasets.

A key feature of MEDS-Torch is its ability to generalize to any medical dataset adhering to the MEDS format. The pipeline's dependence solely on the MEDS schema for input data and the MEDS label schema for supervised task labels ensures its applicability across various EHR datasets. This standardization allows researchers to easily apply the same models and techniques to different datasets without extensive modifications, facilitating reproducibility and comparability of results across studies. The MEDS format converts any subject time series data into a standardized table structure, enabling MEDS-Torch to process and model the data consistently regardless of its original format or source.

[2]Please view the MEDS-Torch codebase at https://github.com/Oufattole/meds-torch.

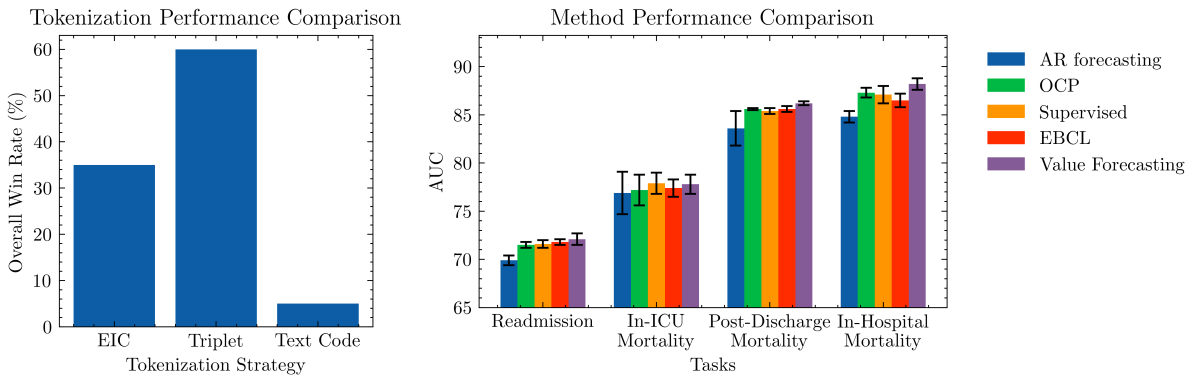
MEDS-Torch offers a user-friendly API that streamlines experiment execution. Leveraging Hydra [24] for configuration management, the pipeline enables easy customization and overriding of experiment parameters. We provide examples of common workflows, including supervised training, autoregressive model pretraining and finetuning, distributed hyperparameter tuning, and multiseed training jobs. These can be executed via a simple command-line interface (see Appendix B).[3]

3 Experiments and Results

The field of machine learning for Electronic Health Records (EHR) is characterized by fragmented model development and evaluation approaches. Researchers often implement models on a limited number of datasets, using custom implementations that are challenging to reproduce. This reproducibility challenge stems from difficulties in replicating both cohorts [10, 8] and model training recipes [15, 3]. While recent work has addressed cohort reproducibility [23], our work focuses on providing reproducible model training recipes. This approach enables practitioners to effectively compare methods and accelerates the identification of best practices for EHR modeling.

We demonstrate the utility of MEDS-Torch in facilitating methodologically reproducible controlled experiments on EHR data. Our pipeline enables researchers to achieve actionable, dataset-specific insights from a suite of methods applicable to any medical dataset in the MEDS format. For this study, we focus on the MIMIC-IV [9] dataset, with ongoing collaborations to expand to other hospital datasets. We chose MIMIC-IV as our initial dataset due to its public availability and widespread use in the medical ML community. This choice allows for easier validation and comparison of our results with existing literature.

We experiment with three tokenization methods commonly found in the EHR sequence model literature: Everything In Code (EIC) [16], Triplet [7, 19], Text Code [5]. For each tokenization method, we implement two categories of transfer learning techniques: Contrastive Learning Methods include Order Contrastive Pretraining (OCP) [1] and Event-Based Contrastive Learning (EBCL) [7]. Forecasting Transfer Learning Methods include Autoregressive Forecasting [16, 4] and Value Forecasting [19]. We focus on contrastive learning methods due to their demonstrated effectiveness in healthcare settings [7, 1, 6, 11]. Forecasting methods were included as they have shown promise in capturing temporal dependencies in EHR data [16, 18].



(a) Overall win rate for each tokenization strategy, where a win is counted when a strategy achieves the highest average AUC (over 5 seeds) for a given task and method combination.

(b) Performance comparison of different pretraining methods across four clinical prediction tasks, showing AUC scores with standard deviation error bars over 5 seeds.

Figure 1: Performance analysis of different tokenization strategies and learning methods on clinical prediction tasks. Tasks include in-hospital mortality, in-ICU mortality, post-discharge mortality, and 30-day readmission prediction.

[3]For comprehensive guidance, users can refer to the documentation and tutorials in our codebase here: https://meds-torch.readthedocs.io/en/latest/inductive_experiments/.

Our results, as shown in Figure 1 (see all results in Appendix Table 1), indicate that the choice of tokenization method significantly impacts model performance across different tasks and learning methods. Key findings include: (1) Triplet tokenization generally outperforms other methods for contrastive learning and supervised learning tasks (visualized in Figure 1a). (2) Value Forecasting is the most performant pretraining method across most downstream tasks (visualized in Figure 1b).

The variability in performance across tokenization methods and learning approaches underscores the need for flexible, extensible frameworks like MEDS-Torch that allow researchers to easily experiment with different combinations. As we continue to expand our experiments to include more datasets and refine our methods, we anticipate that MEDS-Torch will play a crucial role in advancing the state-of-the-art in EHR-based predictive modeling. By providing a standardized platform for method comparison, we aim to accelerate the development of more accurate and reliable predictive models on any given healthcare dataset.

Our experiments were conducted on a machine with 8 V100 GPUs, 76 cores, and 768 GB RAM. We used a supervised transformer encoder [20] with 4 attention heads, 2 layers, a maximum sequence length of 128 tokens, and a token dimension of 128. Labels for each task were extracted from our MEDS dataset using the ACES tool [23], which enables the reproducible retrieval of identically defined cohorts across any medical dataset. We employed Ray Tune [13] for hyperparameter optimization and conducted multiple runs with different random seeds to ensure robust results. The Hydra framework [24] was used to manage our experimental configurations.

In our experiments, we distributed hyperparameter sweeps and multiseed jobs such that each seed or hyperparameter combination had exclusive access to one GPU, running in parallel with other configurations using Ray Tune [13]. This implementation allows for efficient utilization of computational resources and enables rapid experimentation across various model architectures and hyperparameters.

4 Conclusion

In this paper, we introduced MEDS-Torch, a versatile and efficient pipeline for training sequence models on EHR data in the standardized MEDS format. Our key contributions include: (1) the development of a scalable and generalizable tool for processing medical datasets in the MEDS format, (2) the implementation of computationally efficient preprocessing and caching mechanisms, (3) support for three different tokenization methods, (4) implementation of five transfer learning techniques, and (5) empirical analysis across multiple tasks demonstrating the ease of performance comparisons on MEDS-formatted datasets. Through our systematic evaluation of tokenization methods (EIC, Triplet, and Text Code) across multiple tasks on the MIMIC-IV dataset, we have demonstrated the significant impact of tokenization choices on model performance. The MEDS-Torch pipeline not only streamlines the process of running controlled experiments on medical datasets but also promotes reproducibility and standardization in EHR research. By providing a flexible, extensible framework for method comparison, MEDS-Torch enables researchers and healthcare institutions to systematically evaluate modeling strategies tailored to their specific datasets and tasks, potentially accelerating the development and deployment of more accurate and reliable predictive models in healthcare settings.

While our study provides valuable insights, it also has limitations that point to exciting directions for future work. We plan to expand our evaluation to a broader range of EHR data sources beyond MIMIC-IV, further validating our pipeline's generalizability. Leveraging our existing support for architecture searching via Hydra and Ray Tune, we aim to provide comprehensive documentation, scripts, and results from architecture searches across more tasks, datasets, and learning methods, accompanied by compute time/memory benchmarks. Additionally, we intend to extend our framework to support generative model evaluations and incorporate new pretraining strategies, including masked imputation and additional contrastive learning methods. These enhancements will further establish MEDS-Torch as a comprehensive tool for advancing EHR-based machine learning research and applications.

References

[1] Monica N Agrawal, Hunter Lang, Michael Offin, Lior Gazit, and David Sontag. Leveraging time irreversibility with order-contrastive pre-training. In *International Conference on Artificial*

Intelligence and Statistics, pages 2330–2353. PMLR, 2022.

[2] Bert Arnrich, Edward Choi, Jason Alan Fries, Matthew B.A. McDermott, Jungwoo Oh, Tom Pollard, Nigam Shah, Ethan Steinberg, Michael Wornow, and Robin van de Water. Medical event data standard (MEDS): Facilitating machine learning for health. In *ICLR 2024 Workshop on Learning from Time Series For Health*, 2024.

[3] Andrew L Beam, Arjun K Manrai, and Marzyeh Ghassemi. Challenges to the reproducibility of machine learning models in health care. *Jama*, 323(4):305–306, 2020.

[4] Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[5] Kyunghoon Hur, Jungwoo Oh, Junu Kim, Jiyoun Kim, Min Jae Lee, Eunbyeol Cho, Seong-Eun Moon, Young-Hak Kim, Louis Atallah, and Edward Choi. Genhpf: General healthcare predictive framework for multi-task multi-source learning. *IEEE Journal of Biomedical and Health Informatics*, 28(1):502–513, 2024.

[6] Aapo Hyvarinen and Hiroshi Morioka. Nonlinear ICA of Temporally Dependent Stationary Sources. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 460–469. PMLR, 20–22 Apr 2017.

[7] Hyewon Jeong, Nassim Oufattole, Matthew Mcdermott, Aparna Balagopalan, Bryan Jangeesingh, Marzyeh Ghassemi, and Collin Stultz. Event-based contrastive learning for medical time series, 2024.

[8] Alistair E. W. Johnson, Tom J. Pollard, and Roger G. Mark. Reproducibility in critical care: a mortality prediction case study. In Finale Doshi-Velez, Jim Fackler, David Kale, Rajesh Ranganath, Byron Wallace, and Jenna Wiens, editors, *Proceedings of the 2nd Machine Learning for Healthcare Conference*, volume 68 of *Proceedings of Machine Learning Research*, pages 361–376. PMLR, 18–19 Aug 2017.

[9] Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, et al. Mimic-iv, a freely accessible electronic health record dataset. *Scientific data*, 10(1):1, 2023.

[10] Alistair EW Johnson, Tom J Pollard, and Roger G Mark. Reproducibility in critical care: a mortality prediction case study. In *Machine learning for healthcare conference*, pages 361–376. PMLR, 2017.

[11] Ryan King, Tianbao Yang, and Bobak J. Mortazavi. Multimodal pretraining of medical time series and notes. In Stefan Hegselmann, Antonio Parziale, Divya Shanmugam, Shengpu Tang, Mercy Nyamewaa Asiedu, Serina Chang, Tom Hartvigsen, and Harvineet Singh, editors, *Proceedings of the 3rd Machine Learning for Health Symposium*, volume 225 of *Proceedings of Machine Learning Research*, pages 244–255. PMLR, 10 Dec 2023.

[12] Alex Labach, Aslesha Pokhrel, Xiao Shi Huang, Saba Zuberi, Seung Eun Yi, Maksims Volkovs, Tomi Poutanen, and Rahul G. Krishnan. Duett: Dual event time transformer for electronic health records, 2023.

[13] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[14] Matthew McDermott, Bret Nestor, Evan Kim, Wancong Zhang, Anna Goldenberg, Peter Szolovits, and Marzyeh Ghassemi. A comprehensive ehr timeseries pre-training benchmark. In *Proceedings of the Conference on Health, Inference, and Learning*, CHIL '21, page 257–278, New York, NY, USA, 2021. Association for Computing Machinery.

[15] Matthew B. A. McDermott, Shirly Wang, Nikki Marinsek, Rajesh Ranganath, Luca Foschini, and Marzyeh Ghassemi. Reproducibility in machine learning for health research: Still a ways to go. *Science Translational Medicine*, 13(586):eabb1655, 2021.

[16] Pawel Renc, Yugang Jia, Anthony E. Samir, Jaroslaw Was, Quanzheng Li, David W. Bates, and Arkadiusz Sitek. Zero shot health trajectory prediction using transformer. March 2024.

[17] Benjamin Shickel, Patrick James Tighe, Azra Bihorac, and Parisa Rashidi. Deep ehr: a survey of recent advances in deep learning techniques for electronic health record (ehr) analysis. *IEEE journal of biomedical and health informatics*, 22(5):1589–1604, 2017.

[18] Ethan Steinberg, Jason Fries, Yizhe Xu, and Nigam Shah. Motor: A time-to-event foundation model for structured medical records. *arXiv preprint arXiv:2301.03150*, 2023.

[19] Sindhu Tipirneni and Chandan K Reddy. Self-supervised transformer for sparse and irregularly sampled multivariate clinical time-series. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(6):1–17, 2022.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[21] Cao Xiao, Edward Choi, and Jimeng Sun. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *Journal of the American Medical Informatics Association*, 25(10):1419–1428, 2018.

[22] Feng Xie, Han Yuan, Yilin Ning, Marcus Eng Hock Ong, Mengling Feng, Wynne Hsu, Bibhas Chakraborty, and Nan Liu. Deep learning for temporal data representation in electronic health records: A systematic review of challenges and methodologies. *Journal of biomedical informatics*, 126:103980, 2022.

[23] Justin Xu, Jack Gallifant, Alistair E. W. Johnson, and Matthew B. A. McDermott. Aces: Automatic cohort extraction system for event-stream datasets, 2024.

[24] Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019.

A Tokenization and Pretraining Method Details

A.1 Tokenization

In this paper, we explore three distinct tokenization methods (represented in Figure 2): Everything In Code (EIC), Triplet, and Text Code. For detailed explanations of each method, see Figure 2.

A.2 Pretraining Methods

We currently support five distinct pretraining techniques: (1) Event-Based contrastive pretraining, (2) Order contrastive pretraining, (3) Value forecasting, (4) Autoregressive EIC pretraining, and (5) Autoregressive triplet forecasting.

The first three methods rely on support for a random-window dataset class, where the patient's history is loaded and partitioned into two adjacent windows. Extra measures are taken to keep windows within the user-defined maximum sequence length for their architecture. **Event-Based contrastive pretraining** creates positive pairs from observation windows immediately preceding and following an event in a patient's trajectory. In contrast, windows from different patients are considered negative pairs. The objective of pretraining is to minimize the distance between positive pairs in the latent space while maximizing the separation from negative pairs. **Order contrastive pretraining** (based on [1]) involves drawing pairs of observation windows from a patient's trajectory, with the task of determining whether these windows are in their correct chronological order. By challenging the model to differentiate between correctly and incorrectly ordered sequences, this approach encourages the learned representations to effectively capture the temporal structure and sequential dependencies within the data. **Value forecasting** (based on [19]) takes two adjacent windows of time, the prior window is input into the encoder, and the second window is used to create labels, the model must predict the presence of codes (binary cross entropy loss) and the numeric value of all present codes with a present numeric value (MSE loss).

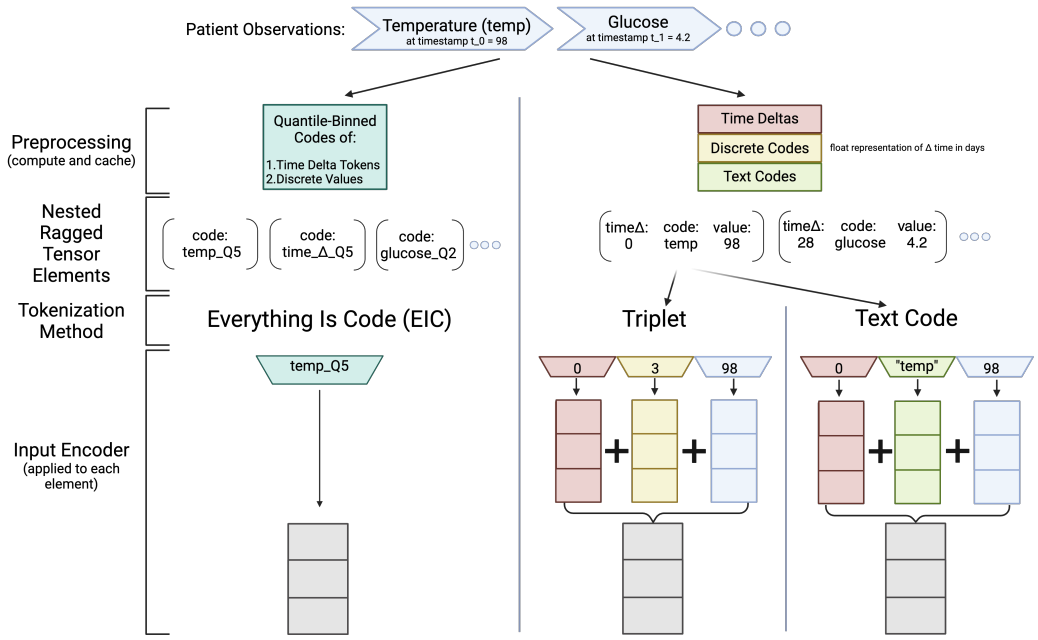


Figure 2: Comparison of three tokenization methods for medical time series data. Everything In Code (EIC) converts all observations into categorical tokens by quantile binning values and discretizing time deltas. Triplet represents each observation as three separate embeddings (time delta, code, numeric value) that are summed together. The continuous components (time delta and numeric value) are embedded using learned feed-forward networks, while codes use learned codebook embeddings (i.e. each unique code is stored as a unique integer which we learn an embedding for). Text Code is similar to Triplet but maintains codes as text strings rather than numeric IDs, allowing semantic embedding. The figure shows the transformation process from raw patient observations through preprocessing to nested tensor representations, and finally to the embedded form for each method.

The other two methods use a single window of input data and the base PyTorch dataset class: *Autoregressive EIC pretraining* (based on [16]) is the task of predicting the next code, just as GPT-style language models [4] are pretrained, as the patient's history is assumed to be preprocessed into a categorical sequence. *Autoregressive triplet forecasting* is an autoregressive forecasting task where the model has to predict the next observation by simultaneously predicting the triplet of time, code, and value, it is a natural extension of Autoregressive EIC forecasting to the triplet and text code input encoded data. MSE loss is used over the time and numeric values while cross entropy is applied for the code.

B Code Examples

B.0.1 Environment Variables

The following environment variables are used in the code examples:

- `PATHS_KWARGS`: Defines data paths, e.g., `paths.data_dir=/path/to/data paths.meds_cohort_dir=/path/to/meds paths.output_dir=/path/to/output`
- `TASK_KWARGS`: Defines task parameters for supervised learning, e.g., `data.task_name=mortality data.task_root_dir=/path/to/task/labels`

C Command-line Examples

Listing 1: Training a supervised model on GPU

```
MEDS-Torch-train trainer=gpu $PATHS_KWARGS $TASK_KWARGS
```

Listing 2: Pretraining an autoregressive forecasting model

```
MEDS-Torch-train trainer=gpu $PATHS_KWARGS model=eic_forecasting
```

Listing 3: Parameter overriding and hyperparameter tuning

```
MEDS-Torch-train trainer.max_epochs=20 data.batch_size=64 $PATHS_KWARGS $TASK_KWARGS
MEDS-Torch-tune experiment=experiment.yaml callbacks=tune_default $PATHS_KWARGS $TASK_KWARGS hparams_search=ray_tune
```

C.0.1 Example Configuration File

Listing 4: Sample `experiment.yaml` file

```yaml
# @package _global_

defaults:
  - override /data: pytorch_dataset
  - override /logger: wandb
  - override /model/backbone: triplet_transformer_encoder
  - override /model/input_encoder: triplet_encoder
  - override /model: supervised
  - override /trainer: gpu

tags: [mimiciv, triplet, transformer_encoder]

seed: 0

trainer:
  min_epochs: 1
  max_epochs: 10
  gradient_clip_val: 1.0

data:
  dataloader:
    batch_size: 64
    num_workers: 6
  max_seq_len: 128
  collate_type: triplet
  subsequence_sampling_strategy: to_end

model:
  token_dim: 128
  optimizer:
    lr: 0.001
  backbone:
    n_layers: 2
    nheads: 4
    dropout: 0

logger:
  wandb:
    tags: ${tags}
    group: mimiciv_tokenization
```

This `experiment.yaml` file configures a supervised learning experiment using a triplet transformer encoder on MIMIC-IV data. It sets up various parameters for the trainer, data loading, model architecture, and logging.

D All Results

Table 1: **MIMIC-IV**: Mean ± STD AUC (over five trials) of tokenization Methods across tasks. hosp-Mortality refers to in-hospital mortality after 24 hours. ICU-Mortality refers to in-ICU mortality. Discharge-Mortality predicts whether a subject will survive 1 year after discharge. Readmission is the task of predicting if a subject is readmitted within 30 days.

Method	Type	Tokenization	hosp-Mortality	ICU-Mortality	Discharge-Mortality	Readmission
Contrastive	OCP	EIC	86.1 ± 0.3	77.2 ± 0.4	84.1 ± 0.1	70.3 ± 0.3
		Text Code	80.0 ± 0.5	69.3 ± 1.6	81.6 ± 0.1	68.3 ± 0.3
		Triplet	87.3 ± 0.4	75.8 ± 0.4	85.6 ± 0.1	71.5 ± 0.2
	EBCL	EIC	86.1 ± 0.7	76.6 ± 0.6	84.0 ± 0.0	70.6 ± 0.2
		Text Code	85.2 ± 0.3	74.9 ± 0.9	83.9 ± 0.3	70.1 ± 0.3
		Triplet	86.5 ± 0.2	77.4 ± 0.7	85.6 ± 0.1	71.8 ± 0.1
Forecasting	AR	EIC	84.8 ± 0.1	76.9 ± 0.5	83.6 ± 0.3	69.9 ± 0.2
		Text Code	81.9 ± 0.4	71.6 ± 1.3	81.9 ± 0.4	69.1 ± 0.3
		Triplet	74.0 ± 0.6	65.1 ± 2.2	76.0 ± 1.8	64.4 ± 0.5
	Value	EIC	86.1 ± 0.6	77.8 ± 1.0	84.0 ± 0.2	70.2 ± 0.6
		Text Code	85.2 ± 0.2	75.8 ± 0.6	85.8 ± 0.1	72.1 ± 0.1
		Triplet	88.2 ± 0.1	77.3 ± 0.8	86.2 ± 0.1	72.1 ± 0.1
Supervised	-	EIC	86.4 ± 0.2	77.9 ± 0.4	84.0 ± 0.1	70.0 ± 0.1
		Text Code	85.3 ± 0.3	75.5 ± 0.7	85.0 ± 0.3	70.8 ± 0.3
		Triplet	87.1 ± 0.9	77.4 ± 1.1	85.4 ± 0.2	71.5 ± 0.2