A DETAILS FOR NOTATIONS, IMPLEMENTATIONS AND DATASETS

A.1 NOTATION TABLE

For clarity, we have attached a notation table here, summarizing all symbols used in the main paper. This paper uses the generic notation $\mathbf{x}$ to denote the input features. We use plain letters for scalars, such as d, p, y; boldface lowercase letters for vectors or vector-valued functions, e.g., $\mathbf{v}$, $\mathbf{z}$, $\mathbf{h}_\theta(\cdot)$, $\mathbf{q}_\phi(\cdot)$, $\mathbf{g}_\xi(\cdot)$, $\mathbf{p}_\psi(\cdot)$, and they all have the same dimensionality; Euler script letters for sets, e.g., $\mathcal{D}$, $\mathcal{S}$.

Table 4: Notations used in `ManyDG`

Symbols	Descriptions
$d \in \mathcal{D}$, $d' \in \mathcal{D}'$	patient/domain set in training and test
$i \in \mathcal{S}^d$	data sample set of patient/domain d
$\mathbf{x}_i^d$, y_i^d	the i-th (sample, class label) for patient/domain d
$p(\cdot)$	meta domain distribution
$\mathbf{z}$	domain latent factor (representation)
$p(\cdot\|\mathbf{z})$	sample distribution conditioned on patient domain $\mathbf{z}$ and class label y
$p(y\|\mathbf{x})$	a generic notation for posterior of label probability given sample $\mathbf{x}$
$p(\mathbf{z}\|\mathbf{x})$	a generic notation for posterior of domain factor given sample $\mathbf{x}$
$p(y\|\mathbf{x},\mathbf{z})$	a generic notation for posterior of label probability given sample $\mathbf{x}$ and domain $\mathbf{z}$
$\mathbf{v}$	the feature representation of $\mathbf{x}$
$\hat{\mathbf{v}}$	reconstructed feature representation of $\mathbf{x}$
$\mathbf{v}_{\|\|\mathbf{z}}$, $\mathbf{v}_{\perp\mathbf{z}}$	the component of $\mathbf{v}$ that is parallel to or orthogonal to $\mathbf{z}$ space
$\mathbf{h}_\theta(\cdot) : \mathbf{x} \mapsto \mathbf{v}$	learnable feature extractor with parameter set θ
$\mathbf{q}_\phi(\cdot) : \mathbf{v} \mapsto \mathbf{z}$	learnable encoder network for $p(\mathbf{z}\|\mathbf{x})$ (on top of $\mathbf{h}_\theta(\cdot)$) with parameter set ϕ
$\mathbf{g}_\xi(\cdot) : \mathbf{v},\mathbf{z} \mapsto y$	learnable predictor network for $p(y\|\mathbf{x},\mathbf{z})$ (on top of $\mathbf{h}_\theta(\cdot)$) with parameter set ξ
$\mathbf{p}_\psi(\cdot) : \mathbf{z}, y \mapsto \mathbf{v}$	learnable decoder network for $p(\cdot\|\mathbf{z}, y)$ with parameter set ψ
$\langle\cdot\rangle$	vector inner product
$\|\cdot\|_{\mathcal{F}}$; $\|\cdot\|$	vector norm induced by measure $\mathcal{F}$; L_2-norm
`Weight`$(\cdot)$	the learned weight of a logistic regression task

A.2 IMPLEMENTING OUR DOUBLE DATA LOADER FOR TRAINING

Our method is trained end-to-end in a mini-batch fashion. First, during each training epoch, we will randomly shuffle the data samples within each patient domain and build a new data loader for our method. Second, for building the data loader, we will fold each patient data list into two half-lists and append them to two global data lists patient-by-patient. These two global lists have the same length and will be used to build the data loader. Upon using it, the data loader will output two data batches simultaneously (for the Siamese-type architecture), where the same indices correspond to the samples from the same patient. *At every epoch, only one pairing combination is used for the data from each patient, and the shuffle step between epochs ensures that every data pair within the same patient can have the chance to be trained together with equal probability.* Our double data loader building procedure does not incur extra time consumption compared to the model training time. Our model's space and time complexity are asymptotically the same as training the Base model, and empirically we have similar running time complexity as the Base model, shown in Appendix B.1.

A.3 DETAILS IN IMPLEMENTING THE LABEL PREDICTOR AND DATA DECODER

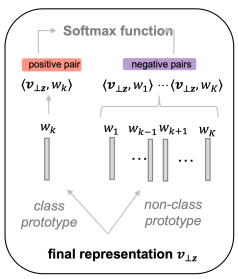

Figure 5: Predictor

Prototype-base Predictor As mentioned in the main text, we use a fully connected (FC) layer without the bias term to be the predictor, which is essentially a parameter matrix $\mathbf{W}$. Each row in the parameter matrices $\{\mathbf{w}_k\}_{k=1}^K$ has the same dimensionality as $\mathbf{v}$ and can be viewed as the prototype for the class $k \in \{1, 2, \ldots, K\}$. We can construct one positive pair between the final representation and the class prototype and $(K-1)$ negative pairs with the prototypes from other classes (in Figure 5). The final label probability is given by the softmax activation

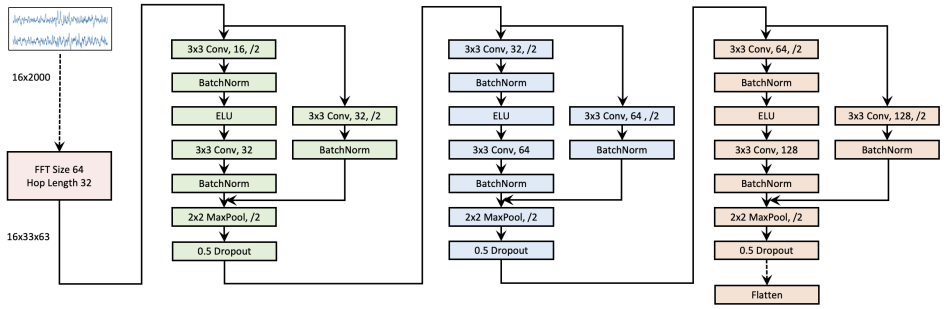

Figure 6: Shared backbone for seizure detection task

form,

$$p(y=k|\mathbf{x},\mathbf{z}) = \mathbf{g}_{\xi,k}(\mathbf{v}_{\perp\mathbf{z}}) = \frac{\exp(\langle \mathbf{w}_k, \mathbf{v}_{\perp\mathbf{z}} \rangle/\tau)}{\sum_i \exp(\langle \mathbf{w}_i, \mathbf{v}_{\perp\mathbf{z}} \rangle/\tau)}. \tag{16}$$

Here, $\langle \cdot, \cdot \rangle$ is the notation for vector inner product, and τ is the temperature (He et al., 2020). With the prototype-based design, our predictor is also amenable for use with recent supervised contrastive loss (Yao et al., 2022; Khosla et al., 2020); however, we leave this as future work.

Prototype-base Data Decoder In the main text, we use a decoder network $\mathbf{p}_\psi(\cdot|\mathbf{z}, y)$ to reconstruct another sample from the same domain based on the domain factor $\mathbf{z}$ and the class label y. In the implementation, we do not directly input a combination of $\mathbf{z}$ and the categorical label y. For parameterizing the class label, we also use the prototype $\mathbf{w}_y$ as the class label information and concatenate it with the domain factor $\mathbf{z}$ for reconstruction.

A.4 More information for datasets and implementation

IIIC Seizure requested from (Jing et al., 2018; Ge et al., 2021). We could not find the license, as stated in Jing et al. (2018) "the local IRB waived the requirement for informed consent for this retrospective analysis of EEG data".

We use 16-channel 10-second signals sampled at 200Hz. Thus, the raw data inputs are a matrix 16×2000, and the values are measurement amplitudes. We use *torch.stft* to implement the short-time Fourier transform (STFT) for obtaining the spectrogram. The FFT window size is 64, the hop length (overlapping window) is 32, and the imaginary and real parts are square-summed to be the energy density in the frequency domain. After STFT, the sample size is $16 \times 33 \times 63$. We further normalize the values and take the logarithm on one side of the spectrogram as the input to the backbone model. The data processing steps largely followed Jing et al. (2018). Seizure detection is a six-class classification problem, and we use some common combinations of hyperparameters: 128 as the batch size, 128 as the hidden representation size, 50 as the training epochs ($50 \times \frac{\text{size of an epoch}}{\text{size of an episode}}$ as the number of episodes for MLDG baseline, which follows MAML, the same setting for other datasets), 5×10^{-4} as the learning rate with Adam optimizer and 1×10^{-5} as the weight decay for all models. Our model uses $\tau = 0.5$ as the temperature (the same for other datasets). Other hyperparameters in baselines are mostly following their original default values. Some other variants of hyperparameter combinations have also been tested with the validation set (on the following three datasets, we also do the same procedures), which do not show significant differences. We report our backbone model in Figure 6.

The original dataset provides a vote count distribution over six labels as the targets in Seizure. We filter out the data that have fewer than five expert votes. We use the vote counts for calculating the following soft cross entropy loss (CEL) and use the majority label (that has the most significant vote) for calculating the evaluation metrics.

For calculating the supervised loss in Seizure, we use a customized version of cross entropy loss. Our customized CEL is a weighted average form of the standard CEL for the classification task.

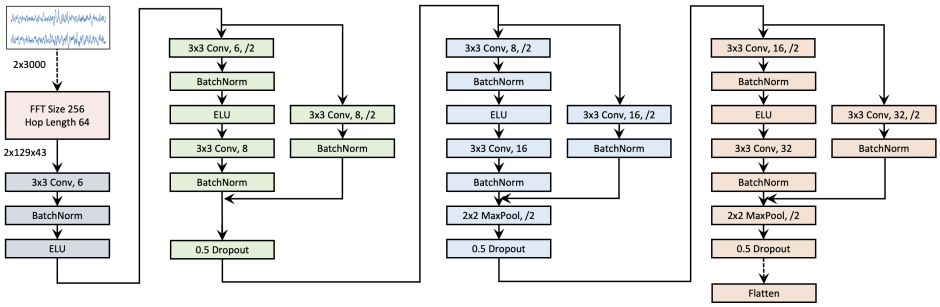

Figure 7: Shared backbone for sleep staging task

We employ the customized CEL form here, considering that for Seizure (and some other expert-annotated datasets), the labeling experts might have disagreements on the true label of ambiguous samples and thus usually resulting in a vote count distribution instead of a single label. The customized CEL can capture more information (e.g., how hard the sample is) than the standard CEL in such cases. The customized CEL will reduce to the standard CEL if without labeling disagreements. We give the definition of our customized CEL below,

$$\mathcal{L}_4 = -\sum_{k\in\mathcal{S}} \frac{1}{|\mathcal{S}|} \log p(y=k|\mathbf{x},\mathbf{z}). \tag{17}$$

where $\mathcal{S} \subset \{1,\cdots,K\}$ includes all class indices that have more votes than half of the maximum class votes. For example, if the votes for a sample are $[8,0,5,3,2,1]$, then $\mathcal{S}=\{1,3\}$ where 8 and 5 are viewed as valid votes, and other classes are considered minor classes, which will not be used for calculating the loss. For common classification task with single labels, $|\mathcal{S}|=1$.

Sleep-EDF [1] (Kemp et al., 2000) This dataset is under Open BSD 3.0 License[2]. It contains other Polysomnography (PSG) signals such as (horizontal) EOG, and submental chin EMG. We only use two EEG channels, Fpz-Cz and Pz-Oz, and a raw 30-second sample has size 2×3000. We also do STFT with an FFT window size of 256, hop length of 64 and normalize and logarithm operations on one side of the spectrogram. The data processing step[3] largely follows Yang et al. (2021b). The final size of a sample is $2\times 129\times 43$. We select combinations of hyperparameters: 256 as the batch size, 128 as the hidden representation size, 50 as the training epochs, 5×10^{-4} as the learning rate with Adam optimizer, and 1×10^{-5} as the weight decay for all models. The backbone model is reported in Figure 7.

MIMIC-III [4] (Johnson et al., 2016) This dataset is under PhysioNet Credentialed Health Data License 1.5.0[5]. The dataset is publicly available with patients who stayed in intensive care unit (ICU) in the Beth Israel Deaconess Medical Center for over 11 years. It consists of 50,206 medical encounter records. Following the preprocessing step from Yang et al. (2021a); Shang et al. (2019). We filter out the patients with only one visit. Diagnosis, procedure and medication information is extracted in "DIAGNOSES_ICD.csv", "PROCEDURES_ICD.csv" and "PRESCRIPTIONS.csv" from original MIMIC database. We merge these three sources by patient id and visit id. After the merging, diagnosis and procedure are ICD-9 coded, and they will be transformed into multi-hot vectors before training. There are in total 6,350 patients, 15,032 visits, 1,958 types of diagnoses, 1,430 types of procedures, and 112 ATC-4 coded medications. For each visit, we use the diagnosis and procedure information (two ICD code sets) and previous visit information (two ICD code sets and one ATC code set per visit) to predict the current medication set, which is a sequential multi-label prediction task. In this task, we use binary cross entropy loss for each medication, following Yang et al. (2021a); Shang et al. (2019) and then aggregate the loss. The backbone model is borrowed

[1] https://www.physionet.org/content/sleep-edfx/1.0.0/
[2] https://www.physionet.org/content/sleep-edfx/view-license/1.0.0/
[3] https://github.com/ycq091044/ContraWR
[4] https://physionet.org/content/mimiciii/1.4/
[5] https://physionet.org/content/mimiciii/view-license/1.4/

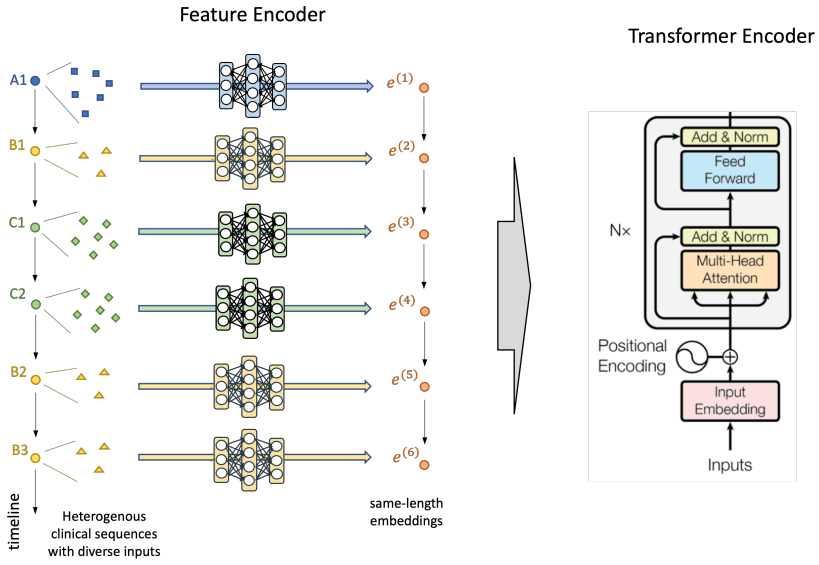

Figure 8: Shared backbone for hospitalization prediction task. Different colored circles mean different event types and different feature inputs.

from a recent paper Shang et al. (2019) with the same model hyperparameter set. For other hyperparameters, we use 64 as the batch size, 64 as hidden representation, 100 as the training epochs, 1×10^{-3} as the learning rate with Adam optimizer and 1×10^{-5} as the weight decay for all models. We adjust the conditional mutual reconstruction part of our model by using the average of class prototypes as the label information for this setting.

eICU [6] (Pollard et al., 2018) This dataset is under PhysioNet Credentialed Health Data License 1.5.0[7]. This data covers patients admitted to critical care units from 2014 to 2015. We define each encounter as identified by *patientunitstayid* in the data tables. Data processing step largely follows Choi et al. (2020). We consider the following five event categories and the features: (i) diagnosis: diagnosis string features (in total 3,845 types) and the ICD-9 codes (in total 1,195 types); (ii) lab test: lab test types (in total 158 types) and the corresponding numeric results; (iii) medication: medication names (in total 291 types), prescription and the pick list frequency with which the drug is taken; (iv) physical exam: the system root path of the physical exam (in total 462 types), which indicates the types; (v) treatment: the path of the treatment (in total 2,695 types). The "offset" time in raw data tables is used as the timestamp for each event since they are offset with respect to the admission time. For each encounter, we can collect a heterogeneous event sequence following time order for predicting the risk of re-admitted into ICU (re-hospitalized) within next 15 days (Choi et al., 2020). In this task, we learn two class prototypes and finally feed the first column of the output softmax logits for the binary cross entropy loss (which only takes *batch_size* $\times$ *1* format as input in *torch.nn.functional.binary_cross_entropy_loss*). For hyperparameters, we use 256 as the batch size, 128 as hidden representation, 50 as the training epochs, 5×10^{-4} as the learning rate with Adam optimizer, and 1×10^{-5} as the weight decay for all models. The backbone model is based on 3-layer Transformer encoder blocks, before which we first use separate encoders for encoding different types of events into the same-length embeddings. We show an illustration in Figure 8.

We report the label distribution information in Table 5.

B Additional Experimental Results

We present additional experiments in this appendix section due to space limitations.

[6] https://physionet.org/content/eicu-crd/2.0/
[7] https://physionet.org/content/eicu-crd/view-license/2.0/

Table 5: Label information of four datasets

Dataset	Task	Label Distribution
Seizure	multi-class	OTHER: 26.4%, ESZ: 3.7%, LPD: 19.7%, GPD: 24.2%, LRDA: 14.4%, GRDA: 11.7%
Sleep-EDF	multi-class	Wake: 68.8%, N1: 5.2%, N2: 16.6%, N3: 3.2%, REM: 6.2%
MIMIC-III	multi-label	max # of med. is 64, avg # of med. is 11.65
eICU	binary	hospitalized: 83.78%, non-hospitalized: 16.22%

B.1 Running time comparison

This section compares the running time of all models in four healthcare tasks. As mentioned before, all models use the same backbone and batch size. When recording the running time, we duplicated the environment mentioned in Section 4.1, stopped other programs, and ran all the models one by one on one GPU. We record the first 23 epochs of all models and drop the first three epochs (since they might be unstable). We report the mean time cost per epoch in Table. MLDG uses an episodic training strategy, different from epoch-based training. Thus, we calculate the *equivalent running time* for MLDG, which is that we first average the episode time by the sample size and then multiply the per-sample time by the overall training data size.

We report the results in Table 6, which shows our method is as efficient as the Base model in each task. CondAdv and DANN are two adversarial training models which rely on an additional domain discriminator. In our cases with patient-induced domains, we have more domains to deal with (than previous domain generalization settings). Thus, their discriminator can be less effective and will cost more time.

Table 6: Running time comparison (seconds per epoch)

Model	Seizure Detection	Sleep Staging	Drug Recommendation	Hospitalized Prediction
Base	7.128 ± 0.4000	17.91 ± 0.1885	3.306 ± 0.0219	541.4 ± 12.36
CondAdv	19.95 ± 0.1673	20.60 ± 0.1498	3.472 ± 0.0248	580.1 ± 16.87
DANN	11.71 ± 0.3037	18.29 ± 1.0210	3.518 ± 0.0256	585.3 ± 8.282
IRM	7.598 ± 0.3639	19.23 ± 0.5725	3.416 ± 0.0022	558.2 ± 4.022
SagNet	12.75 ± 0.4110	35.57 ± 0.0186	/	/
PCL	8.743 ± 0.4869	21.70 ± 0.3514	/	577.9 ± 10.24
MLDG (equivalent)	13.87 ± 0.4965	31.89 ± 0.2578	4.210 ± 0.0448	663.6 ± 14.05
ManyDG	7.996 ± 0.2862	19.23 ± 0.2744	3.462 ± 0.0648	558.1 ± 14.27

B.2 Statistical testing with p-values

We also conduct T-tests on the results in Section 4.2 and Section 4.3 and calculated the p-values in the following Table 7 and Table 8. Commonly, a p-value smaller than 0.05 would be considered significant. We use **bold** font for p-values that is larger than 0.05. We can conclude that our performance gain is significant over the baselines in most of the cases.

Table 7: p-values under T-test on biosignal classification tasks

Comparison	Seizure Detection			Sleep Staging		
	Accuracy	Kappa	Avg. F1	Accuracy	Kappa	Avg. F1
Base vs ManyDG	1.8277e-04	2.3857e-03	3.1177e-03	3.9742e-02	3.1450e-02	**2.7878e-01**
CondAdv vs ManyDG	1.6107e-03	1.1745e-02	1.2661e-04	6.3333e-03	**1.5790e-01**	1.4948e-02
DANN vs ManyDG	2.3134e-04	2.1345e-03	6.9548e-05	3.2046e-02	1.0646e-02	**2.7459e-01**
IRM vs ManyDG	1.9953e-03	1.4039e-03	1.4164e-03	6.9382e-04	4.2033e-02	**4.4536e-01**
SagNet vs ManyDG	6.4119e-04	7.1616e-02	**5.2789e-02**	3.0083e-01	**4.5421e-01**	**7.3663e-01**
PCL vs ManyDG	1.8472e-02	9.1054e-02	1.5726e-02	**1.3048e-01**	**9.8423e-02**	**5.4170e-01**
MLDG vs ManyDG	9.1763e-03	4.9798e-03	3.0785e-03	7.4243e-03	2.0582e-03	4.1056e-02

B.3 Cosine similarity on domain representations

In this section, we load the pre-trained embedding $\mathbf{z}$ from four healthcare tasks. To give more insights into the learned domain factor $\mathbf{z}$, we compute the cosine similarity of the estimated $\mathbf{z}$ within

Table 8: p-values under T-test on EHR classification tasks

Comparison	Drug Recommendation			Hospitalized Prediction		
	Jaccard	Avg. AUPRC	Avg. F1	AUPRC	F1	Kappa
Base vs ManyDG	1.5738e-03	1.0867e-04	8.9086e-04	7.8500e-03	3.4105e-04	7.0435e-04
CondAdv vs ManyDG	2.2134e-02	5.5230e-05	2.9717e-02	2.6847e-03	1.8490e-02	2.4544e-02
DANN vs ManyDG	2.0507e-02	1.9382e-03	**9.6398e-02**	**6.0040e-02**	3.6720e-03	3.2901e-03
IRM vs ManyDG	2.2013e-03	4.5726e-04	3.5304e-03	**1.5173e-01**	8.1280e-03	2.5219e-02
PCL vs ManyDG	/	/	/	4.3115e-03	4.3811e-03	3.8113e-02
MLDG vs ManyDG	1.4382e-05	2.3040e-05	1.4652e-03	2.4545e-04	4.0163e-04	1.7682e-02

the same domains and the cosine similarity of $\mathbf{z}$ cross domains. We average the computed cosine similarity values over each embedding pair and show the results in Table 9.

Table 9: Cosine similarity of within-domain and cross-domain factors $\mathbf{z}$

Cosine Similarity	Seizure Detection	Sleep Staging	Drug Recommendation	Hospitalization Prediction	Average
Avg. of within-domain $\mathbf{z}$	0.9789	0.9672	0.9993	0.9619	0.9768 ± 0.0144
Avg. of cross-domain $\mathbf{z}$	0.9685	0.9584	0.9965	0.9523	0.9689 ± 0.0169

It is interesting that both the within-domain and cross-domain similarity are pretty high, e.g., larger than 0.95. The second finding is that within-domain similarity is larger than cross-domain similarity (though it seems insignificant, we will explain why this phenomenon is normal and our design is however useful). The reason is that we only enforce the similarity of $\mathbf{z}$ from the same domain as one objective $\mathcal{L}_{\text{sim}}$ but never enforce the dissimilarity of $\mathbf{z}$ from different domains (if we did, then the values in second row of the table are expected to decrease).

A very similar phenomenon has been observed in Grill et al. (2020) from the self-supervised contrastive learning area. Before the proposal of Grill et al. (2020), researchers will simultaneously enforce the similarity of positive samples and the dissimilarity of negative samples He et al. (2020); Chen et al. (2020) for learning unsupervised features invariant to data augmentations. However, Grill et al. (2020) claims that enforcing the similarity of positive pairs along is empirically good enough for learning unsupervised features. In the implementation, the cosine similarity of positive pairs is learned to approach 1.0, and the negative pairs will chase after to achieve a slightly lower score (often times higher than 0.95). They show better results on ImageNet against previous approaches He et al. (2020); Chen et al. (2020) (that enforces both similarity and dissimilarity).

Our method empirically works well on four datasets. Further discussion on this topic is beyond the main scope of the paper. We will explore this direction for future work.

B.4 Ablation Study on Objectives and Their Combinations

Ablation Study on Combinations of Objectives First, we design ablation studies to test the effectiveness of each objective. Recall that, we have used the following final loss in the main text:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{MMD}} + \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{sim}}. \tag{18}$$

Here, $\mathcal{L}_{\text{sup}}$ is the supervised cross entropy loss, $\mathcal{L}_{\text{MMD}}$ is the loss to enforce $\mathbf{v}$ and $\mathbf{z}$ to be in the same space, $\mathcal{L}_{\text{rec}}$ is the mutual reconstruction loss between two data samples, and $\mathcal{L}_{\text{sim}}$ enforces the similarity of two latent factors from the same patient.

In this section, we test the following loss combinations on the seizure detection task: **(i) Case 1**: Only $\mathcal{L}_{\text{sup}}$; **(ii) Case 2**: $\mathcal{L}_{\text{sup}}$ and $\mathcal{L}_{\text{MMD}}$; **(iii) Case 3**: $\mathcal{L}_{\text{sup}}$ and $\mathcal{L}_{\text{rec}}$; **(iv) Case 4**: $\mathcal{L}_{\text{sup}}$, $\mathcal{L}_{\text{MMD}}$ and $\mathcal{L}_{\text{rec}}$. We show the final performance results and the learning curve of each objective in Figure 9. The results prove the necessity of all our objectives. We provide discussions and insights on the effectiveness of individual objectives:

- With different objectives, the overall losses all decrease, which means the model trains successfully in each case.

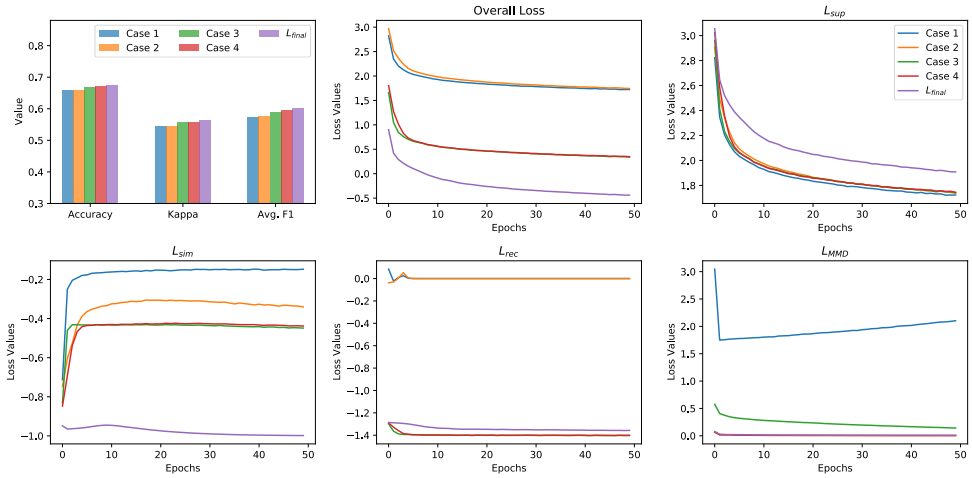

Figure 9: Results of ablation studies on different objective combinations

- **In case 1 (only $\mathcal{L}_{sup}$, no $\mathcal{L}_{MMD}$, $\mathcal{L}_{rec}$, $\mathcal{L}_{sim}$)**, there is no interaction between two sides of the Siamese architecture. **Our method performs similarly to the Base model.** Reflected on the loss curves: $\mathcal{L}_{sim} = -0.17$ (domain factors are not similar, ideal value is -1.0); $\mathcal{L}_{rec} = 0$ (no reconstruction effect, ideal value is -2.0); $\mathcal{L}_{MMD} = 2.0$ (embedding spaces are not aligned, ideal value is 0).
- **In case 2 (only $\mathcal{L}_{sup}$, $\mathcal{L}_{MMD}$, no $\mathcal{L}_{rec}$, $\mathcal{L}_{sim}$):** There is no interaction between two sides of the Siamese architecture. **Our method performs similarly to the Base model.** However, the MMD objective improves the orthogonal projection, which explains the slight improvements over **case 1**. Reflected on the loss curves: $\mathcal{L}_{sim} = -0.45$ (domain factors are somewhat similar, ideal value is -1.0); $\mathcal{L}_{rec} = 0$ (no reconstruction effect, ideal value is -2.0); $\mathcal{L}_{MMD} = 0$ (embedding spaces are aligned, ideal value is 0).
- **In case 3 (only $\mathcal{L}_{sup}$, $\mathcal{L}_{rec}$, no $\mathcal{L}_{MMD}$, $\mathcal{L}_{sim}$): The reconstruction objective ensures the interactions between two-sided pipeline, which is essential for learning the domain and domain-invariance representations.** We find that without $\mathcal{L}_{MMD}$, our model can learn to align the $\mathbf{v}$ and $\mathbf{z}$ embedding spaces and use orthogonal projection for prediction. According to the bar chart in Figure 9, **the $\mathcal{L}_{rec}$ brings the most performance gains** compared to other objectives (exclude $\mathcal{L}_{sup}$). Reflected on the loss curves: $\mathcal{L}_{sim} = -0.45$ (domain factors are somewhat similar, ideal value is -1.0); $\mathcal{L}_{rec} = -1.4$ (reconstruction effect, ideal value is -2.0); $\mathcal{L}_{MMD} = 0.15$ (embedding spaces are nearly aligned, ideal value is 0).
- **In case 4 ($\mathcal{L}_{sup}$, $\mathcal{L}_{rec}$, $\mathcal{L}_{MMD}$, no $\mathcal{L}_{sim}$):** Adding $\mathcal{L}_{MMD}$ can ensure the perfect alignment of the $\mathbf{v}$ and $\mathbf{z}$ embedding spaces and brings further improvements over **case 3**. Reflected on the loss curves: $\mathcal{L}_{sim} = -0.45$ (domain factors are somewhat similar, ideal value is -1.0); $\mathcal{L}_{rec} = -1.4$ (reconstruction effect, ideal value is -2.0); $\mathcal{L}_{MMD} = 0$ (embedding spaces are aligned, ideal value is 0).
- Adding $\mathcal{L}_{sim}$ for **case 4** leads to the final objective in our method. $\mathcal{L}_{sim}$ maximizes the similarity of the domain factors and brings further accuracy improvements.

In summary, $\mathcal{L}_{sup}$ brings the label supervision (the most important one), $\mathcal{L}_{rec}$ brings additional information (tells the model that two samples are from the same domain), $\mathcal{L}_{MMD}$ improves the orthogonal projection (for better disentanglement), and $\mathcal{L}_{sim}$ improves the learned domain factor (for better orthogonal projection).

Ablation Study on Combinations of Weights Next, on the seizure detection task, we add weights as hyperparameters $\lambda_1, \lambda_2, \lambda_3 > 0$ to combine our proposed objectives,

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{sup}} + \lambda_1 \cdot \mathcal{L}_{\text{MMD}} + \lambda_2 \cdot \mathcal{L}_{\text{rec}} + \lambda_3 \cdot \mathcal{L}_{\text{sim}}. \tag{19}$$

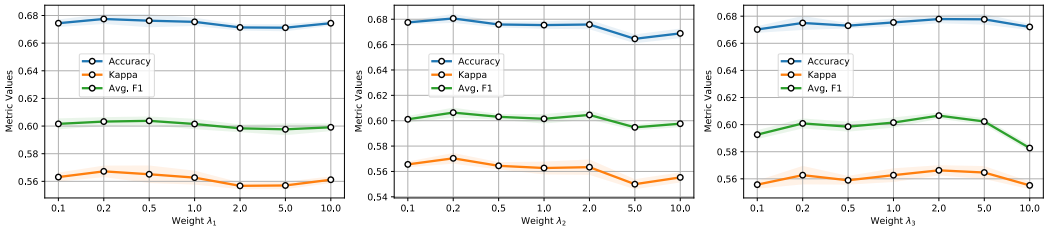

Figure 10: Results on individual weight variation

By default, we use $\lambda_1 = \lambda_2 = \lambda_3 = 1.0$ in the experiments of main text. This section first tests different variations of $\lambda_1, \lambda_2, \lambda_3$ individually. The results are shown in Figure 10. When increasing one of the weights from 0.1 to 10.0 and fix the others, we find that all three metrics will follow a similar pattern: first become better and then slightly decreases. From the figure, we can conclude that using $\lambda_1 = \lambda_2 = \lambda_3 = 1.0$ is indeed a decent choice without using exhaustive search. However, there are minor variations in performance, and apparently we can find a (slightly) better combination than default.

Based on the individual performance, we guess that $\lambda_1 = 0.2, \lambda_2 = 0.2, \lambda_3 = 2.0$ might be a better combination from a greedy perspective. Thus, we train the model again on this combination and find that it gives marginal improvements over our default choice, shown in Table 10.

Table 10: Comparison with greedy weight combination

Weights	Accuracy	Kappa	Avg. F1
$\lambda_1 = \lambda_2 = \lambda_3 = 1.0$	0.6754 ± 0.0079	0.5627 ± 0.0066	0.6015 ± 0.0120
$\lambda_1 = 0.2, \lambda_2 = 0.2, \lambda_3 = 2.0$	0.6785 ± 0.0045	0.5654 ± 0.0062	0.6031 ± 0.0089

B.5 Scatter plot of $\mathbf{v}_{\perp \mathbf{z}}, \mathbf{v}_{||\mathbf{z}}$

To provide more insights on the learned embeddings, we plot the L2-norm statistics of $\mathbf{v}_{\perp \mathbf{z}}, \mathbf{v}_{||\mathbf{z}}$ on four tasks for both training and test. Specifically, we draw a scatter plot using L2-norm of $\mathbf{v}_{\perp \mathbf{z}}$ as x-axis and L2-norm of $\mathbf{v}_{||\mathbf{z}}$ as y-axis.

For IIIC seizure dataset, we randomly select 25 epochs (3,200 samples) from the training set and select 25 epochs (3,200 samples) from the test set. For Sleep-EDF sleep staging dataset, we randomly select 25 epochs (3,200 samples) from the training set and 25 epochs (3,200 samples) from the test set. For MIMIC-III drug recommendation dataset, we randomly select 25 epochs (1,600 samples) from the training set and 25 epochs (1,600 samples) from the test set. For eICU hospitalization prediction dataset, we randomly select 25 epochs (1,600 samples) from the training set and 10 epochs (640 samples) from the test set. The results are shown in Figure 11.

Result Analysis From Figure 11, we can roughly come up with three findings: (i) The scatter plot distributions for training and test look similar on all datasets and tasks, which means our approach is well-trained and generaliable (at least on the norm space); (ii) The magnitude (norm) of $\mathbf{v}_{||\mathbf{z}}$ is generally larger than $\mathbf{v}_{\perp \mathbf{z}}$ on all datasets, which means the domain information $\mathbf{v}_{||\mathbf{z}}$ is indeed a large component on the extracted features $\mathbf{v}$. This finding again verifies the necessity of removing the domain information during the prediction; (iii) The correlations between norms of $\mathbf{v}_{||\mathbf{z}}$ and $\mathbf{v}_{\perp \mathbf{z}}$ are slightly different across different tasks. In the IIIC seizure prediction task and drug MIMIC recommendation task, their norms seem a bit positively correlated. In Sleep-EDF sleep staging task, their norms seem uncorrelated. In eICU hospitalization prediction task, their norms seem a bit negative correlated. We conclude that the norm correlations are highly dependent on the dataset characteristics.

B.6 Performance comparison with simple decomposition model

In this section, we compare our model performance with two simple decomposition models. They have the same architectural designs with different objective functions. We describe the architecture:

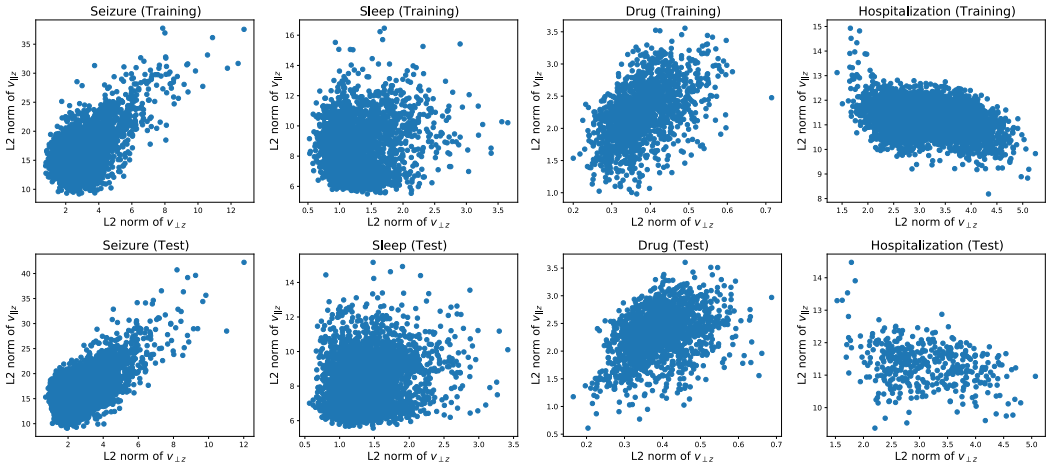

Figure 11: Scatter plot of $\mathbf{v}_{\perp \mathbf{z}}$ and $\mathbf{v}_{||\mathbf{z}}$

- We assume the model has a **feature encoder** from $\mathbf{x} \rightarrow \mathbf{v}$, and $\mathbf{v}$ contains both **domain information** and **domain-invariant label information**.
- We assume the model has a **domain information encoder** that maps $\mathbf{v}$ into a domain representation $\mathbf{z}$ and has a **label information encoder** that maps $\mathbf{v}$ into a label representation $\mathbf{u}$.
- There are four loss objectives, which (i) **(supervised loss)** minimizes the supervised prediction loss with a final prediction layer on $\mathbf{u}$, similarity on $\mathbf{u}$ as well; (ii) **(reconstruction loss)** minimizes the discrepancy between $\mathbf{v}$ and the reconstructed $\hat{\mathbf{v}}$ from $\mathbf{z}, \mathbf{u}$, similarity for $\mathbf{v}'$ and $\hat{\mathbf{v}}'$ from $\mathbf{z}', \mathbf{u}'$; (iii) **(domain similarity loss)** maximizes the similarity of $\mathbf{z}$ and $\mathbf{z}'$ between two samples of the same patient domain; (iv) **(orthogonality loss)** enforces the orthogonality of $\mathbf{z}, \mathbf{u}$ and the orthogonality of $\mathbf{z}', \mathbf{u}'$.

For the **first simple decomposition (SimD1) model**, we follow this domain adaptation work (Bousmalis et al., 2016), which also learns the domain-specific information and domain-shared information separately by two neural networks. It uses (i) cross-entropy loss as the **supervised loss**; (ii) mean square error (MSE) as the **reconstruction loss**; (iii) scale-invariant MSE (Eigen et al., 2014) as the **domain similarity loss**; (iv) Frobenius norm as the **orthogonality loss**.

For the **second simple decomposition (SimD2) model**, we use the loss design from our approach and use (i) cross-entropy loss as the **supervised loss**; (ii) normalized cosine similarity as the **reconstruction loss**; (iii) normalized cosine similarity as the **domain similarity loss**; (iv) Frobenius norm as the **orthogonality loss**.

Result Analysis We show the performance compared of these two simple decomposition models with our proposed `ManyDG` in Table 11 and Table 12. In summary, our `ManyDG` outperforms two simple decomposition models significantly, which shows the effectiveness of our mutual reconstruction and orthogonal projection steps. The SimD2 model also performs better than SimD1 model consistently. The only differences between them are the metrics used for each objective design. We find that SimD1 needs to carefully find the hyperparameter weights for balancing different objectives, while the objectives in SimD2 are all well-scaled and can be used together without weights. Though we have selected decent weight combinations for SimD1, it is still inferior to SimD2.

Table 11: Result comparison on health monitoring datasets

Models	Seizure Detection			Sleep Staging		
	Accuracy	Kappa	Avg. F1	Accuracy	Kappa	Avg. F1
SimD1	0.5954 ± 0.0127	0.4792 ± 0.0197	0.4998 ± 0.0242	0.8862 ± 0.0015	0.7855 ± 0.0149	0.6962 ± 0.0025
SimD2	0.6499 ± 0.0153	0.5485 ± 0.0065	0.5686 ± 0.0092	0.9017 ± 0.0073	0.7978 ± 0.0205	0.6989 ± 0.0141
`ManyDG`	**0.6754 ± 0.0079**	**0.5627 ± 0.0066**	**0.6015 ± 0.0120**	**0.9055 ± 0.0054**	**0.7998 ± 0.0124**	**0.7015 ± 0.0027**

Table 12: Result comparison on open EHR databases

Models	Drug Recommendation			Hospitalization Prediction		
	Jaccard	Avg. AUPRC	Avg. F1	AUPRC	F1	Kappa
SimD1	0.4894 ± 0.0087	0.7472 ± 0.0065	0.6482 ± 0.0192	0.6985 ± 0.0153	0.6368 ± 0.0026	0.5079 ± 0.0118
SimD2	0.4901 ± 0.0091	0.7536 ± 0.0115	0.6534 ± 0.0127	0.7047 ± 0.0105	0.6584 ± 0.0048	0.5260 ± 0.0103
ManyDG	**0.5175 ± 0.0130**	**0.7746 ± 0.0035**	**0.6737 ± 0.0141**	**0.7258 ± 0.0132**	**0.6752 ± 0.0025**	**0.5531 ± 0.0144**

B.7 More explanations on the learned linear weights

In this section, we discuss the relatively low cosine similarity in Table 3. Compared to the first row (predicting the labels, e.g., 6-class classification in seizure detection), the second row corresponds to predicting one of the many domains (e.g., 2,702-class classification in seizure detection). Using the linear prediction model, the performances of the second row are naturally not as good as the first row. As a result, the learned coefficients in the second row are less similar than those learned in the first row. This may also explain why sleep staging has the highest similarity in the second row (since this task only has 78 domains).