A DATASETS

A.1 DATA PREPROCESSING

For all of the datasets, frames consisted of 2500 samples and consecutive frames had no overlap with one another. Data splits were always performed at the patient-level.

Chapman (Zheng et al., 2020). Each ECG recording was originally 10 seconds with a sampling rate of 500Hz. We downsample the recording to 250Hz and therefore each ECG frame in our setup consisted of 2500 samples. We follow the labelling setup suggested by Zheng et al. (2020) which resulted in four classes: Atrial Fibrillation, GSVT, Sudden Bradychardia, Sinus Rhythm. The ECG frames were normalized in amplitude between the values of 0 and 1.

PTB-XL (Wagner et al., 2020). Each ECG recording was originally 10 seconds with a sampling rate of 500Hz. We extract 5-second non-overlapping segments of each recording generating frames of length 2500 samples. We follow the diagnostic class labelling setup suggested by Strodthoff et al. (2020) which resulted in five classes: Conduction Disturbance (CD), Hypertrophy (HYP), Myocardial Infarction (MI), Normal (NORM), and Ischemic ST-T Changes (STTC). Furthermore, we only consider ECG segments with one label assigned to them. The ECG frames were standardized to follow a standard Gaussian distribution.

A.2 DATA SAMPLES

In this section, we outline the number of instances used during training.

Table 3: Number of instances (number of patients) used during training. These represent sample sizes for all 12 leads.

Dataset	Train	Validation	Test
Chapman	76,614 (6,387)	25,524 (2,129)	25,558 (2,130)
PTB-XL	22,670 (11,335)	3,284 (1,642)	3,304 (1,152)

B Implementation Details

B.1 Network Architectures

In this section, we outline the neural network architectures used for our experiments. More specifically, we use the architecture shown in Table 4 for all experiments pertaining to the Chapman dataset. Given the size of the PTB-XL dataset, we opted for a more complex network. We modified the ResNet18 architecture whereby the number of blocks per layer was reduced from two to one, effectively reducing the number of parameters by a factor of two. We chose this architecture after experimenting with several variants.

Table 4: Network architecture used for experiments conducted on the Chapman dataset. K, C_{in}, and C_{out} represent the kernel size, number of input channels, and number of output channels, respectively. A stride of 3 was used for all convolutional layers. E represents the dimension of the final representation.

Layer Number	Layer Components	Kernel Dimension
1	Conv 1D BatchNorm ReLU MaxPool(2) Dropout(0.1)	7 x 1 x 4 (K x C_{in} x C_{out})
2	Conv 1D BatchNorm ReLU MaxPool(2) Dropout(0.1)	7 x 4 x 16
3	Conv 1D BatchNorm ReLU MaxPool(2) Dropout(0.1)	7 x 16 x 32
4	Linear ReLU	320 x E

B.2 Experiment Details

Table 5: Batchsize and learning rates used for training with different datasets. The Adam optimizer was used for all experiments.

Dataset	Batchsize	Learning Rate
Chapman	256	10^{-4}
PTB-XL	128	10^{-5}

B.3 Baseline Implementations

B.3.1 DeepCluster

In the implementation by Caron et al. (2018), a forward pass of each instance in the training set is performed. This generates a set of representation which are then clustered, in an unsupervised manner, using K-means. This involves a decision regarding the value of K, i.e., the number of clusters. In our supervised setting, we have this information available and therefore set the value of K to be equal to the number of distinct cardiac arrhythmia classes. Once the clustering is complete, each instance is assigned a pseudo-label according to the cluster to which it belongs. Such pseudo-labels are used as the ground-truth for supervised training during the next epoch. We repeat this process after each epoch for a total of 30 epochs after realizing that the validation loss plateaus at that point.

B.3.2 IIC

In this implementation, the network is tasked with maximizing the mutual information between the representation of an instance and that of its perturbed counterpart. Such perturbations must be class-preserving and, in computer vision, consist of random crops, rotations, and modifications to the brightness of the images. In our setup involving time-series data, we perturb instances by using additive Gaussian noise in order to avoid erroneously flipping the class of a particular instance. In addition to the aforementioned, we implement the auxiliary over-clustering method suggested by the authors. This approach allows one to model additional 'distractor' classes that may be present in the dataset, and was shown by Ji et al. (2019) to improve generalization performance. In our setup, we set the number of total clusters to the number of attribute combinations, M.

B.3.3 SeLA

In this implementation, each instance is *assigned* a posterior probability distribution. For all instances, this results in an assigned matrix of posterior probability distributions. Each instance's label is obtained by identifying the index associated with the largest posterior probability distribution. Deriving the aforementioned matrix is the crux of SeLA. It does by solving the Sinkhorn-Knopp algorithm under the assumption that the dataset can be evenly split into K clusters. Our setup does not deviate from the original implementation found in (Asano et al., 2020).

B.3.4 DTCluster

In this implementation, the distance between each representation and each cluster prototype is calculated to generate a probability distribution over classes, p. The distribution, p, is encouraged to be similar to a target distribution, q, by minimizing the KL divergence of these two distributions. In the original unsupervised implementation, the target distribution is a sharper version of the empirical distribution (Han et al., 2019). In our supervised implementation, we initialize the prototypes as is done with DROPS and modify the target distribution to incorporate labels. As with the soft-assignment used in DROPS, we aim for a target distribution that reflects discrepancies, d, between the representation attributes, A_i, and the prototype attributes, A_k. Mathematically, our target distribution, q, is as follows:

$$q = \frac{e^{\omega_{ik}}}{\sum_j^{|L|} e^{\omega_{ij}}} \tag{11}$$

$$\omega_{ik} = \begin{cases} \frac{e^{d(A_i, A_k)}}{\sum_j^{|L|} e^{d(A_i, A_j)}} & \text{if disease class}_i = \text{disease class}_k \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

$$d(A_i, A_k) = [\delta(\text{disease class}_i = \text{disease class}_k) + \delta(\text{sex}_i = \text{sex}_k) + \delta(\text{age}_i = \text{age}_k)] \cdot \frac{1}{\tau_\omega} \tag{13}$$

C Discovering Attribute-Specific Features Within Clinical Prototypes

We have shown that clinical prototypes can be deployed successfully for retrieval and clustering purposes while managing to capture relationships between attributes. In this section, we aim to quantify the relationship between clinical prototypes and explore their features further. In Fig. 5, we illustrate a matrix of the clinical prototypes ($M = 32$) along the rows and their corresponding features ($E = 128$) along the columns. By implementing the hierarchical agglomerative clustering (HAC) algorithm, we cluster these clinical prototypes and arrive at the dendrogram presented along the rows of Fig. 5. In addition to being correctly clustered according to class labels, they are also more similar to one another based on their attributes. This can be seen by the attribute combination descriptions in the right column. This finding supports our earlier claim that clinical prototypes do indeed capture relationships between attributes.

Motivated by recent work on disentangled representations, whereby representations can be factorized into multiple sub-groups each of which correspond to a particular abstraction, we chose to cluster the *features* of the clinical prototypes, resulting in the dendrogram presented along the columns of Fig. 5. The intuition is that by clustering we may discover attribute-specific feature subsets. We show that these features can indeed be clustered into three main groups, potentially coinciding with our pre-defined attributes. Such a process can improve the interpretability of clinical prototypes and lead to insights about how they can be further manipulated for retrieval purposes, for instance, by altering a subset of features.

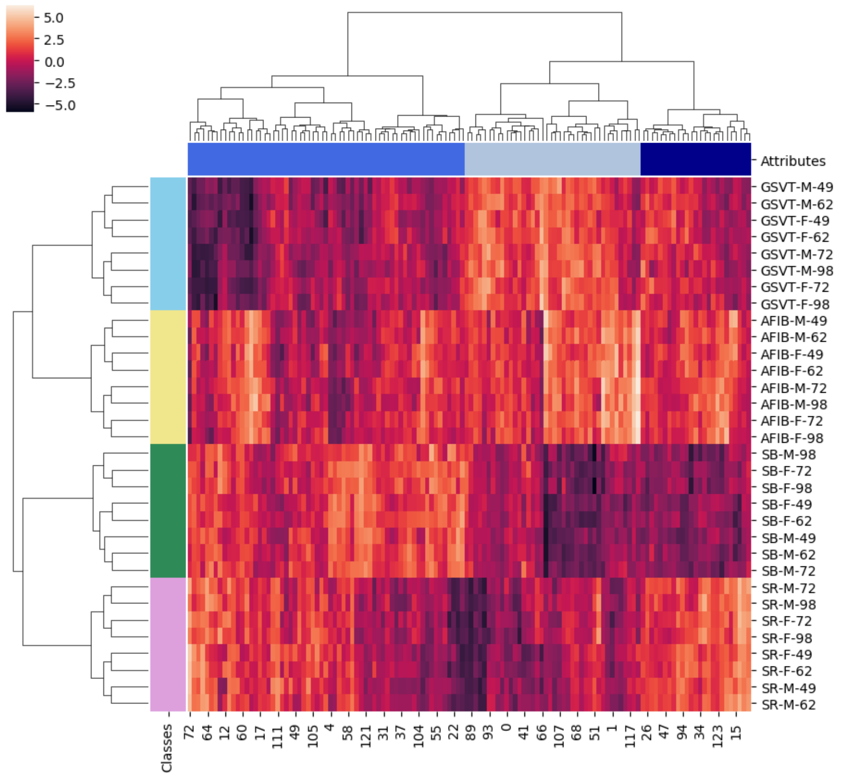

Figure 5: Hierarchical agglomerative clustering of the learned clinical prototypes in M. **(Rows)** Clustering is performed using the 128-dimensional features resulting in 4 major clusters corresponding to the 4 classes. Clinical prototypes with similar attributes are also clustered together. The rows are labelled according to the attribute combination, m. **(Columns)** Clustering is performed whereby each of the 128 *features* is treated as an instance, resulting in 3 major clusters which are hypothesized to correspond to the 3 attributes: class, sex, and age. This suggests that disentangled, attribute-specific features may have been learned.