Appendix

A Proof

In this section, we present complete proofs of our theoretical study, starting with notations.

A.1 Notations

Fourier transform of a real-valued sample with a finite duration is obtained as in Equation 7.

$$\mathrm{X}_k = \mathcal{F}(\mathbf{x}) = \sum_{n=-\infty}^{\infty} \mathbf{x}_n e^{-j2\pi kn} \tag{7}$$

The amplitude and phase for each frequency are calculated from the Fourier transform as follows.

$$\begin{aligned} A(\mathbf{x}) &= \sqrt{\mathrm{Re}(\mathrm{X}_k)^2 + \mathrm{Im}(\mathrm{X}_k)^2} \\ P(\mathbf{x}) &= \mathrm{arctan2}(\mathrm{Im}(\mathrm{X}_k), \mathrm{Re}(\mathrm{X}_k)), \end{aligned} \tag{8}$$

where arctan is a 2-argument arctangent which is the angle measure in radians. The phasor, as in Figure 1, of a sample is represented as in Equation 9.

$$\mathrm{X}_k = \mathcal{F}(\mathbf{x}) = A(\mathbf{x}) e^{jP(\mathbf{x})} \tag{9}$$

A.2 Proof for Proposition 2.3

Proposition A.1 (Destructive Mixup)**.** *If Assumptions 2.1 and 2.2 hold, there exist $\lambda \sim Beta(\alpha, \alpha)$ or $\lambda \sim U(\beta, 1.0)$ with high values of β such that when linear mixup techniques are utilized, the lower bound of the mutual information for the augmented sample distribution decreases to zero.*

$$\begin{gathered} 0 \leq \mathcal{I}(\mathbf{y}; \mathbf{x}^+) < \mathcal{I}(\mathbf{y}; \mathbf{x}^*) \quad \text{where} \\ \mathbf{x}^+ = \lambda \mathbf{x} + (1-\lambda)\tilde{\mathbf{x}} \;\text{ and }\; \int_{f^*} S_{x^*}(f) = \int_{-\infty}^{\infty} S_{x^*}(f) \end{gathered} \tag{10}$$

Proof.

$$\mathbf{x}^+ = \lambda \mathbf{x} + (1-\lambda)\tilde{\mathbf{x}} \tag{11}$$

From the linearity of Fourier transformation and ignoring k in $\mathrm{X_k}$ for the sake of easiness.

$$\mathrm{X}^+ = \lambda \mathbf{X} + (1-\lambda)\tilde{\mathbf{X}} \tag{12}$$

$$\mathrm{X}^+ = \tilde{\mathbf{X}} + \lambda(\mathbf{X} - \tilde{\mathbf{X}}) \tag{13}$$

Let $\tilde{\mathbf{X}} = e^{-j\omega\boldsymbol{\phi}_k}\mathbf{X}\boldsymbol{\omega}_k$, where $\boldsymbol{\phi}_k$ and $\boldsymbol{\omega}_k$ are random phase and frequency modulators for each frequency, sampled from distributions $\boldsymbol{\phi}_k \sim \Phi$, $\boldsymbol{\omega}_k \sim \Omega$.

$$\mathrm{X}^+ = e^{-j\omega\boldsymbol{\phi}_k}\mathbf{X}\boldsymbol{\omega}_k + \lambda(\mathbf{X} - e^{-j\omega\boldsymbol{\phi}_k}\mathbf{X}\boldsymbol{\omega}_k) \tag{14}$$

$$\mathrm{X}^+ = \mathbf{X}\left[\lambda + e^{-j\omega\boldsymbol{\phi}_k}\boldsymbol{\omega}_k - \lambda e^{-j\omega\boldsymbol{\phi}_k}\boldsymbol{\omega}_k\right] \tag{15}$$

$$\mathrm{X}^+ = \mathbf{X}\left[\lambda + (1-\lambda)e^{-j\omega\boldsymbol{\phi}_k}\boldsymbol{\omega}_k\right] \tag{16}$$

$$\mathrm{X}^+ = \mathbf{X}\left[\lambda + (1-\lambda)\boldsymbol{\omega}_k(\cos(\omega\boldsymbol{\phi}_k) - j\sin(\omega\boldsymbol{\phi}_k))\right] \tag{17}$$

From the quasi-periodicity, assume that the frequency ranges of interest (f^*, i.e., k^*) are overlapped for both samples while the sampled random modulators have the following relationship.

$$\boldsymbol{\omega}_{k^*} \approx \frac{\lambda}{1-\lambda} \text{ and } \theta \equiv [\omega \boldsymbol{\phi}_{k^*}] \,(\mathrm{mod}\, 2\pi), \tag{18}$$

where θ is an odd multiple of π. Equation 17 can be simplified as follows.

$$\mathrm{X}^+_{k^*} = \mathbf{X}_{k^*} \left[\lambda + \lambda \cos\left(\omega \boldsymbol{\phi}_{k^*}\right)\right] \tag{19}$$

$$\mathrm{X}^+_{k^*} \approx 0 \tag{20}$$

$$\mathrm{X}^+_{k^*} = \sum_{n=-\infty}^{\infty} \mathbf{x}_n e^{-j2\pi k^* n} \longrightarrow \sum_{n=-\infty}^{\infty} \mathbf{x}_n e^{-j2\pi k^* n} \approx 0 \tag{21}$$

$$S_{x^+}(f^*) = \lim_{N\to\infty} \frac{1}{2N} \left| \sum_{n=-N}^{N} x_n e^{-j2\pi f^* n} \right|^2 \tag{22}$$

From Assumption 2.1,

$$\mathcal{I}(\mathbf{y}; \mathbf{x}^+) \propto \int_{f^*} S_{x^+}(f) \,/\, \int_{-\infty}^{\infty} S_{x^+}(f) \tag{23}$$

$$0 \leq \mathcal{I}(\mathbf{y}; \mathbf{x}^+) < \mathcal{I}(\mathbf{y}; \mathbf{x}^*) \tag{24}$$

We use Euler's formula to expand Equation 16 to 17. While we use the frequency bins (k) and frequency values in Hz (f) interchangeably for Equations 21 and 22. □

Although the above proof is to show the resulting instances may not contain any task-relevant information, it can also be demonstrated that the augmentation process can potentially discard the partial task-specific information (not whole) if $\boldsymbol{\phi}_k$ and $\boldsymbol{\omega}_k$ are close to indicated relationships.

A.3 Proof for Theorem 3.1

Theorem A.2 (Guarantees for Mixing)**.** *Under assumptions 2.1 and 2.2, given any $\lambda \in (0, 1]$, the mutual information for the augmented instance lower bounded by the sampled λ and anchor $\mathbf{x}$.*

$$\lambda \mathcal{I}(\mathbf{y}; \mathbf{x}) \leq \mathcal{I}(\mathbf{y}; \mathbf{x}^+) < \mathcal{I}(\mathbf{y}; \mathbf{x}^*) \;\text{ where }\; \mathbf{x}^+ = \mathcal{F}^{-1}(A(\mathbf{x}^+) \angle P(\mathbf{x}^+)) \tag{25}$$

Proof.

$$\begin{aligned} \mathbf{x}^+ &= \mathcal{F}^{-1}(A(\mathbf{x}^+) \angle P(\mathbf{x}^+)) \quad \text{where} \\ A(\mathbf{x}^+) &= \lambda_A A(\mathbf{x}) + (1 - \lambda_A) A(\tilde{\mathbf{x}}) \;\text{ and} \\ P(\mathbf{x}^+) &= \begin{cases} P(\mathbf{x}) - |\Delta\Theta| * (1 - \lambda_P), & \text{if } \Delta\Theta > 0 \\ P(\mathbf{x}) + |\Delta\Theta| * (1 - \lambda_P), & \text{otherwise} \end{cases} \end{aligned} \tag{26}$$

$$\mathrm{X}^+ = A(\mathbf{x}^+) e^{jP(\mathbf{x}^+)} \tag{27}$$

$$\left|\mathrm{X}^+\right| = \left| A(\mathbf{x}^+) e^{jP(\mathbf{x}^+)} \right| \tag{28}$$

$$\left|\mathrm{X}^+\right| = A(\mathbf{x}^+) \quad \text{where} \quad \left|\mathrm{X}^+_k\right| = \left| \sum_{n=-\infty}^{\infty} \mathbf{x}^+_n e^{-j2\pi kn} \right| \tag{29}$$

$$\left|\mathrm{X}^+\right| = \lambda A(\mathbf{x}) + (1 - \lambda) A(\tilde{\mathbf{x}}) \tag{30}$$

$$\left|\mathrm{X}^+\right| = \lambda \left| \sum_{n=-\infty}^{\infty} \mathbf{x}_n e^{-j2\pi kn} \right| + (1 - \lambda) \left| \sum_{n=-\infty}^{\infty} \tilde{\mathbf{x}}_n e^{-j2\pi kn} \right| \tag{31}$$

$$\lambda \left| \sum_{n=-\infty}^{\infty} \mathbf{x}_n e^{-j2\pi kn} \right| + (1 - \lambda) \left| \sum_{n=-\infty}^{\infty} \tilde{\mathbf{x}}_n e^{-j2\pi kn} \right| \geq \lambda \left| \sum_{n=-\infty}^{\infty} \mathbf{x}_n e^{-j2\pi kn} \right| \tag{32}$$

$$\int_{f^*} S_{x^+}(f) \geq \lambda \int_{f^*} S_x(f) \tag{33}$$

Using the $\int_{-\infty}^{\infty} S_{x^+}(f) = \int_{\infty}^{\infty} S_{\tilde{x}}(f)$ (i.e., both samples are normalized to have the same power) and Assumption 2.1,

$$\mathcal{I}(\mathbf{y}; \mathbf{x}^+) \propto \int_{f^*} S_{x^+}(f) \,/\, \int_{-\infty}^{\infty} S_{x^+}(f) \tag{34}$$

$$\lambda \mathcal{I}(\mathbf{y}; \mathbf{x}) \leq \mathcal{I}(\mathbf{y}; \mathbf{x}^+) < \mathcal{I}(\mathbf{y}; \mathbf{x}^*) \tag{35}$$

Proof is completed with Equation 35 by combining equations 33 and 34. □

Although this proof ignores the effect of phase mixing on the mutual information with the assumption 2.1, it is known that phase components carry semantically important features [45]. Therefore, it is necessary to note that the objective of this proof is to demonstrate that by applying mixup separately to the phase and amplitude components, we can avoid destructive interference.

B Datasets

In this section, we give details about the datasets that are used during our experiments.

B.1 Human Activity Recognition

UCIHAR Human activity recognition using smartphones dataset (UCIHAR) [34] is collected by 30 subjects within an age range of 16 to 48 performing six daily living activities with a waist-mounted smartphone. Six activities include walking, sitting, lying, standing, walking upstairs, and walking downstairs. Data is captured by 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50 Hz. We used the pre-processing technique the same as in [37, 19] such that the input contains nine channels with 128 features (it is sampled in sliding window of 2.56 seconds and 50% overlap, resulting in 128 features for each window). Windows are normalized to zero mean and unit standard deviation before feeding to models. Also, we follow the same experimental setup with prior works as follows. The experiments are conducted with a leave-one-domain-out strategy, where one of the domains is chosen to be the unseen target [19]. The contrastive pre-training is conducted with all subjects without any label information except the target one. Training of the linear layer, which is added to the frozen trained encoder, is only performed with the first five subjects of UCIHAR after excluding the target subject. In other words, if the target subject is 0, the subjects from 1 to 29 are used to train the encoder without any label information. Then, subjects from 1 to 4 are used to train the linear layer. And, evaluation is performed for subject 0. This is performed for the first five subjects with three random seeds and the mean value is reported.

HHAR Heterogeneity Dataset for Human Activity Recognition (HHAR) is collected by nine subjects within an age range of 25 to 30 performing six daily living activities with eight different smartphones—Although HHAR includes data from smartwatches as well, we use data from smartphones—that were kept in a tight pouch and carried by the users around their waists [35]. Subjects then perform 6 activities: 'bike', 'sit', 'stairs down', 'stairs up', 'stand', and 'walk'. Due to variant sampling frequencies of smart devices used in HHAR dataset, we downsample the readings to 50 Hz and apply 100 (two seconds) and 50 as sliding window length with step size, the windows are normalized to zero mean with unit standard deviation. We used the first four subjects (i.e., a, b, c, d) as source domains.

USC USC human activity dataset (USC-HAD) is composed of 14 subjects (7 male, 7 female, aged 21 to 49 with a mean of 30.1) executing 12 activities with a sensor on the front right hip. The data dimension is six (3-axis accelerometer, 3-axis gyroscope) and the sample rate is 100 Hz. 12 activities include walking forward, walking left, walking right, walking upstairs, walking downstairs, running forward, jumping up, sitting, standing, sleeping, elevator up, and elevator down. We used the pre-processing technique with a smaller window size such that the input contains six channels with 100 features (it is sampled in a sliding window of 1 second and 50% overlap, resulting in 100 features for each window). The same normalization is also applied to windows before feeding to models. We used the same setup with UCIHAR while source subjects are chosen as the last four this time.

B.2 Heart Rate Prediction

IEEE SPC This competition provided a training dataset of 12 subjects (SPC12) and a test dataset of 10 subjects [39]. The IEEE SPC dataset overall has 22 recordings of 22 subjects, ages ranging from 18 to 58 performing three different activities [83]. Each recording has sampled data from three accelerometer signals and two PPG signals along with the sampled ECG data and the sampling frequency is 125 Hz. All these recordings were recorded from the wearable device placed on the wrist of each individual. All recordings were captured with a 2-channel pulse oximeter with green LEDs, a tri-axial accelerometer, and a chest ECG for the ground-truth HR estimation. During our experiments, we used PPG channels. We choose the first five subjects of SPC12 as source domains similar to *activity recognition* setup while the last six subjects of SPC22 are used for source domains to prevent overlapping subjects with SPC12.

Dalia PPG dataset for motion compensation and heart rate estimation in Daily Life Activities (DaLia) was recorded from 15 subjects (8 females, 7 males, mean age of 30.6), where each recording was approximately two hours long. PPG signals were recorded while subjects went through different

daily life activities, for instance sitting, walking, driving, cycling, working, and so on. PPG signals were recorded at a sampling rate of 64 Hz. The first five subjects are used as source domains.

All PPG datasets are standardized as follows. Initially, a fourth-order Butterworth bandpass filter with a frequency range of 0.5–4 Hz is applied to PPG signals. Subsequently, a sliding window of 8 seconds with 2-second shifts is employed for segmentation, followed by z-score normalization of each segment. Lastly, the signal is resampled to a frequency of 25 Hz for each segment.

B.3 Cardiovascular disease (CVD) classification

CPSC China Physiological Signal Challenge 2018 (CPSC2018), held during the 7th International Conference on Biomedical Engineering and Biotechnology in Nanjing, China. This dataset consists of 6,877 (male: 3,699; female: 3,178) and 12 lead ECG recordings lasting from 6 seconds to 60 seconds with 500 Hz. We use the original labelling [40] with one normal and eight abnormal types as follows: atrial fibrillation, first-degree atrioventricular block, left bundle branch block, right bundle branch block, premature atrial contraction, premature ventricular contraction, ST-segment depression, ST-segment elevated. We resampled recordings to 100 Hz and exclude recordings of less than 10 seconds.

Chapman Chapman University, Shaoxing People's Hospital (Chapman) ECG dataset which provides 12-lead ECG with 10 seconds of a sampling rate of 500 Hz. The recordings are downsampled to 100 Hz, resulting in each ECG frame consisting of 1000 samples. The labeling setup follows the same approach as in [41] with four classes: atrial fibrillation, GSVT, sudden bradycardia, and sinus rhythm. The ECG frames are normalized to have a mean of 0 and scaled to have a standard deviation of 1. We split the dataset to 80–20% for training and testing as suggested in [41].

We choose leads I, II, III, and V2 during our experiments for both ECG datasets. We followed a similar setup with prior works [57] and considered each dataset as a single domain different from previous tasks. The fine-tuning of the linear layer, which is added to the frozen pre-trained encoder, is performed with 80% of the same domain.

B.4 Metrics

We used the common evaluation metrics in the literature for each task. Specifically, we used accuracy (Acc) and F1 score for *activity recognition* [19], mean absolute error (MAE), and root mean square error (RMSE) for *heart rate prediction* [39, 84], and the area under the ROC curve (AUC) for *cardiovascular disease classification* [57].

In this section, we explain how to calculate each metric for different time-series tasks. For activity recognition, the accuracy metric is computed by dividing the sum of true positives and true negatives by the total number of samples where a window has a single label. The MF1 score is calculated as a harmonic mean of the precision and recall where metrics are obtained globally by counting the total true positives, false negatives, and false positives similar to [19].

For heart rate prediction, the Mean Absolute Error (MAE) and Root-Mean-Square Error (RMSE) are calculated using the following equation:

$$\text{MAE} = \frac{1}{K}\sum_{k=1}^{K}|\text{HR}model(k) - \text{HR}ref(k)| \tag{36}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{k=1}^{K}(\text{HR}model(k) - \text{HR}ref(k))^2}{K}}, \tag{37}$$

where K represents the total number of segments. The variables HR$model(k)$ and HR$ref(k)$ denote the output of the model and reference heart rate values in beats-per-minute for the k^{th} segment, respectively. This performance metric is commonly used in PPG-based heart rate estimation studies [39]. The estimated heart rate values (HR$model(k)$) are obtained using our model, while the reference heart rate values (HR$ref(k)$) are directly taken from datasets.

The AUC score for CVD classification is calculated using the one-vs-one scheme where the average AUC is computed for all possible pairwise combinations of classes for both datasets.

C Baselines

C.1 Prior Mixup Techniques

In this section, we give a detailed explanation of each mixup technique we compare our proposed method.

LinearMix We apply linear mixup as in Equation 38 to generate positive samples, if $\mathbf{x}$ has more than one channel, mixup is applied independently for each of them.

$$\mathbf{x}^+ = \lambda \mathbf{x} + (1-\lambda)\tilde{\mathbf{x}} \tag{38}$$

BinaryMix We implement the binary mixup [43] by swapping the elements of $\mathbf{x}$ with the elements of another randomly chosen sample $\tilde{\mathbf{x}}$ as shown below.

$$\mathbf{x}^+ = \mathbf{m} \odot \mathbf{x} + (1-\mathbf{m}) \odot \tilde{\mathbf{x}}, \tag{39}$$

where $\mathbf{m}$ is a binary mask sampled from a Bernouilli(ρ) with high values, and $\odot$ stands for Hadamard product.

GeometricMix In Geometric Mixup, we create a positive sample corresponding to a sample $\mathbf{x}$ by taking its weighted-geometric mean with another randomly chosen sample $\tilde{\mathbf{x}}$ same as [22] as shown below.

$$\mathbf{x}^+ = \mathbf{x}^{\lambda} + \tilde{\mathbf{x}}^{(1-\lambda)} \tag{40}$$

CutMix Cutmix is implemented similarly to Binarymix. However, instead of changing each sample point with a probability, we cut a continuous portion using a rectangle mask $\mathbf{M}$ from a signal $\mathbf{x}$ and replace it with the same portion of another randomly chosen one $\tilde{\mathbf{x}}$. The starting point of the mask is uniformly sampled while its length is sampled from lower values such that the augmented sample is more similar to the anchor. If the signal has multiple channels, this process is applied to all channels in the same section.

$$\mathbf{x}^+ = \mathbf{M} \odot \mathbf{x} + (1-\mathbf{M}) \odot \tilde{\mathbf{x}}, \quad \text{and} \quad \mathbf{M} = \text{rect}\left(\frac{b}{a}\right), \tag{41}$$

where b and a are the starting point and length of the rectangle wave, respectively.

AmplitudeMix AmplitudeMix is introduced for domain adaptation problems by mixing the amplitude information of images without mixing the phase of two samples [45]. In our setup, we perform amplitude mixing on the time series data across all channels while keeping the phase component unchanged. In other words, we perform the following operations.

$$\begin{gathered} \mathbf{x}^+ = \mathcal{F}^{-1}(A(\mathbf{x}^+) \angle P(\mathbf{x}^+)) \quad \text{where} \\ A(\mathbf{x}^+) = \lambda_A A(\mathbf{x}) + (1-\lambda_A) A(\tilde{\mathbf{x}}) \quad \text{and} \quad P(\mathbf{x}^+) = P(\mathbf{x}) \end{gathered} \tag{42}$$

SpecMix We implement the SpecMix by applying CutMix to the spectrogram of time-series where the spectrogram is calculated using the short-time Fourier transform as follows.

$$\mathrm{X}_k^+ = \sum_{n=-\infty}^{\infty} \mathbf{x}_n g[n - mR] e^{-j2\pi kn}, \tag{43}$$

where $g[n - mR]$ is an analysis window of length M with hop length of R over the signal and calculating the discrete Fourier transform (DFT) of each segment of windowed data. The length of the Fourier transform is set to the sample size of the input time series while the hop and window parameters are set to the quarter of the length.

C.2 Prior Methods for Sample Generation

In this section, we give a detailed explanation of prior methods for data generation methods.

Traditional Augmentations We apply two separate data augmentation to the anchor for creating two instances, and the encoders are trained to maximize agreement using the contrastive loss in [15]. We search mainly for augmentations that are known in state-of-the-art works [19]. The detailed augmentations are given in Table 22.

InfoMin We train a model $g_\theta(.)$, which is restricted to sample-wise 1×1 convolutions and ReLU activations same as in [48], to decrease the mutual information between two instances. In the original paper, the input sample is split into two instances (X_1 and $X_{2:3}$) and then adversarial training is performed. As we do not have RGB channels for time-series data, we added Gaussian noise to the signal for creating other instances and then perform adversarial training.

NNCLR We follow a similar setup to SimCLR by applying two separate data augmentations, then we use nearest neighbors in the learned representation space as the positive in contrastive losses [49].

PosET We perform the dimension level mixing with extrapolation of positive features as follows:

$$\mathbf{z}^+ = \lambda \odot \mathbf{z} + (1 - \lambda) \odot \tilde{\mathbf{z}}, \tag{44}$$

where $\odot$ is Hadamard product, and $\lambda \sim Beta(\alpha, \alpha)$. We add 1 to sampled λ for extrapolation as in [50].

GenRep In the original implementation of GenRep, the authors use implicit generative models (IGMs) such as BigBiGAN [85] that are trained with millions of images to create the anchor and positive instance by sampling nearby latent vectors. However, as the number of samples for training is limited in time series and there is a well-trained generator for different time-series tasks, we use our trained VAE for sampling nearby latent vectors as positives. Mainly, we sample an anchor from real data, feed it to the encoder, add a Gaussian noise sampled from a truncated normal distribution, and use the output of the decoder for the positive sample with the anchor.

STAug The Spectral and Time Augmentation (STAug) method is specifically proposed for the time-series forecasting task, where the authors apply the empirical mode decomposition to decompose time series into multiple subcomponents, then reassemble these subcomponents with random weights to generate new synthetic series. Finally, in the time domain, the method uses the linear mixup to generate samples from the reassembled components. Although, the mixing coefficient sampled from a beta distribution in the original implementation, we observe significant performance decreases when the same distribution with parameters is used in our experiments, possibly due to the generated samples being far away from the anchor. We, therefore, investigate the case when the mixing coefficient is sampled from uniform distribution with high values, e.g., same as our method. Since there is no

Augmentation Bank The augmentation bank that perturbs frequency components of a time-series signal is proposed in [21] where the authors use it for unsupervised domain adaptation with a different framework than SimCLR, namely time-frequency consistency (TF-C). As it is a novel data augmentation technique, we have implemented the frequency augmentation bank as a baseline while using the SimCLR framework for a fair comparison with other methods. The authors also employed a collection of time-based augmentations for the time-domain contrastive encoder. Nonetheless, since these augmentations have already been studied in previous CL setups, we chose to exclusively utilize the frequency augmentation bank. In the paper, the authors mentioned using a small budget with low-frequency perturbations results in a performance increase, thus we chose the budget with a single frequency while choosing the $\alpha = 0.5$ with the same settings in the paper.

DACL We perform the mixup for hidden representations, i.e., before applying projection-head, as follows.

$$\mathbf{v}^{+} = \lambda \mathbf{v} + (1 - \lambda)\tilde{\mathbf{v}}, \tag{45}$$

where $\mathbf{v}$ is the fixed-length hidden representations of samples while λ is sampled from uniform distribution with high values.

IDAA We follow the original implementation of authors with their proposed VAE architecture while optimizing the adversarial strength for each time-series task. We apply the FGSM adversarial attack the same as in the original implementation [53] by perturbing the encoded representation of a sample while adding noises along the gradient sign's direction of the loss.

One setup difference between this section and the previous mixup methods is that when we compare our work with PosET, GenRep, DACL, and IDAA, we apply the best traditional data augmentation techniques, which are used for SimCLR implementation, to the specific positive data generation mechanisms. The reason for this approach is that the original implementations of certain works indicate that the proposed methods achieve optimal results when used in conjunction with known augmentations, where our observations align with these findings.

The detailed hyperparameters for each baseline with the corresponding time series tasks are given in the following section.

D Implementation Details

D.1 Parameters for mixing

In this section, we provide the parameters that are used during our experiments. To determine the optimal parameters of the baselines for each task, we conduct a grid search. This search is performed on a small validation set taken from the largest dataset of the respective tasks, which are USC, Dalia and Chapman. We believe that this approach ensures fairness and produces more realistic results, as dataset-specific optimizations can lead to overfitting of parameters, particularly in smaller and less diverse datasets.

Table 7: Parameters for baselines

Method	*Activity Recognition*	*Heart rate Prediction*	*CVD Classification*
Linear Mixup	$\lambda \sim U(0.9, 1)$	$\lambda \sim U(0.9, 1)$	$\lambda \sim U(0.85, 1)$
Binary Mixup	$m \sim U(0.8, 1)$	$m \sim U(0.9, 1)$	$m \sim U(0.9, 1)$
Geometric Mixup	$\lambda \sim U(0.9, 1)$	$\lambda \sim U(0.9, 1)$	$\lambda \sim U(0.9, 1)$
CutMix	$b \sim U(0, 1)$	$b \sim U(0, 1)$	$b \sim U(0, 1)$
	$a \sim U(0.1, 0.4)$	$a \sim U(0.1, 0.3)$	$a \sim U(0.1, 0.3)$
AmplitudeMix	$\lambda_A \sim U(0.9, 1)$	$\lambda_A \sim U(0.9, 1)$	$\lambda_A \sim U(0.8, 1)$
SpecMix	$b \sim U(0, 1)$	$b \sim U(0, 1)$	$b \sim U(0, 1)$
	$a \sim U(0.1, 0.4)$	$a \sim U(0.1, 0.3)$	$a \sim U(0.1, 0.3)$
PosET	$\lambda \sim Beta(2, 2)$	$\lambda \sim Beta(2, 2)$	$\lambda \sim Beta(2, 2)$
GenRep	$\lambda \sim \mathcal{N}^t(0, 0.2, 1.0)$	$\lambda \sim \mathcal{N}^t(0, 0.25, 1.0)$	$\lambda \sim \mathcal{N}^t(0, 0.2, 1.0)$
DACL	$\lambda \sim U(0.9, 1)$	$\lambda \sim U(0.9, 1)$	$\lambda \sim U(0.85, 1)$
IDAA	$\delta = 0.1$	$\delta = 0.15$	$\delta = 0.2$
Ours	$\lambda_A \sim U(0.7, 1)$, $\lambda_P \sim U(0.9, 1)$	$\lambda_A \sim U(0.7, 1)$, $\lambda_P \sim U(0.9, 1)$	$\lambda_A \sim U(0.7, 1)$, $\lambda_P \sim U(0.9, 1)$
	$\epsilon = 0.7$, $\lambda_A, \lambda_P \sim \mathcal{N}^t(0.9, 0.1, 0.9)$	$\epsilon = 0.8$, $\lambda_A, \lambda_P \sim \mathcal{N}^t(1, 0.1, 0.9)$	$\epsilon = 0.7$, $\lambda_A, \lambda_P \sim \mathcal{N}^t(1, 0.1, 0.9)$

D.2 Baseline Encoder Architecture

For the baseline encoder model, we adopt the DeepConvLSTM as in [19] where the architecture has 4 convolutional layers with 5×1 size of 64 kernels while ReLU is followed each convolution. After the convolutions, the tensor is passed through a dropout layer with a dropout rate of 0.5 to prevent overfitting. Then, the output of dropout is fed into the 2-layer LSTM with 128 units. After training

the baseline encoder, we attach a linear layer and freeze the previous layers for fine-tuning. This architecture is widely used for the datasets we used during our experiments [37, 83, 19], we therefore adopt the same network across tasks.

D.3 VAE Models

We use the total correlation variational autoencoder (β-TCVAE) [86] to calculate the distance between two encoded samples in the latent space. We train the model for 100 epochs with a learning rate of $1e-3$ while setting the batch size to 2048. The latent dimensions and the β values are set to 10 and 5, respectively. Below, we present the tables providing detailed information about the architectures of the encoder and decoder for datasets. The output of convolutional layers is fed to the batch normalization before the activation layer is applied. For tasks *Heart rate Prediction* and *CVD Classification*, we use task-specific encoder and decoder as the number of channels and input size for datasets in each task are the same. However, two different networks, one for UCIHAR and one for others, are designed for the *Activity Recognition* due to different number of input channels.

Table 8: Encoder Network for UCIHAR in *Activity Recognition*

Encoder					
Layer Name	Output size	# of kernels	Kernel size	Stride	Activation
Input	Nx1x128x9				
Convolution	Nx32x60x7	32	9x3	2x1	ReLU
Convolution	Nx32x27x5	32	7x3	2x1	ReLU
Convolution	Nx64x8x3	64	5x3	3x1	ReLU
Convolution	Nx128x2x1	128	5x3	2x1	ReLU
Convolution	Nx512x1x1	512	2x1	1x1	ReLU
Convolution	Nx20x1x1	10	1x1	1x1	

Table 9: Decoder Network for UCIHAR in *Activity Recognition*

Decoder					
Layer Name	Output size	# of kernels	Kernel size	Stride	Activation
Input	Nx1x10x1				
Transposed Convolution	Nx512x2x9	512	2x9	1x1	ReLU
Transposed Convolution	Nx128x8x9	128	4x1	6x1	ReLU
Transposed Convolution	Nx64x16x9	64	4x1	2x1	ReLU
Transposed Convolution	Nx32x32x9	32	4x1	2x1	ReLU
Transposed Convolution	Nx32x64x9	32	4x1	2x1	ReLU
Transposed Convolution	Nx1x128x9	1	4x1	2x1	

Table 10: Encoder Network for USC and HHAR in *Activity Recognition*

Encoder					
Layer Name	Output size	# of kernels	Kernel size	Stride	Activation
Input	Nx1x100x6				
Convolution	Nx32x46x5	32	9x2	2x1	ReLU
Convolution	Nx32x20x4	32	9x2	2x1	ReLU
Convolution	Nx64x8x3	64	5x2	2x1	ReLU
Convolution	Nx128x2x2	128	5x2	2x1	ReLU
Convolution	Nx512x1x1	512	2x2	1x1	ReLU
Convolution	Nx20x1x1	10	1x1	1x1	

Table 11: Decoder Network for USC and HHAR in *Activity Recognition*

Decoder					
Layer Name	Output size	# of kernels	Kernel size	Stride	Activation
Input	Nx1x10x1				
Transposed Convolution	Nx512x2x6	512	2x6	1x1	ReLU
Transposed Convolution	Nx128x6x6	128	6x1	2x1	ReLU
Transposed Convolution	Nx64x12x6	64	4x1	2x1	ReLU
Transposed Convolution	Nx32x25x6	32	5x1	2x1	ReLU
Transposed Convolution	Nx32x50x6	32	4x1	2x1	ReLU
Transposed Convolution	Nx1x100x6	1	4x1	2x1	

Table 12: Encoder Network for *Heart rate Prediction*

Encoder					
Layer Name	Output size	# of kernels	Kernel size	Stride	Activation
Input	Nx1x200x1				
Convolution	Nx32x94x1	32	13x1	2x1	ReLU
Convolution	Nx32x43x1	32	9x1	2x1	ReLU
Convolution	Nx64x18x1	64	9x1	2x1	ReLU
Convolution	Nx128x6x1	128	7x1	2x1	ReLU
Convolution	Nx512x1x1	512	5x1	2x1	ReLU
Convolution	Nx20x1x1	20	2x1	1x1	

Table 13: Decoder Network for *Heart rate Prediction*

Decoder					
Layer Name	Output size	# of kernels	Kernel size	Stride	Activation
Input	Nx1x10x1				
Transposed Convolution	Nx512x6x1	512	6x1	1x1	ReLU
Transposed Convolution	Nx128x12x1	128	4x1	2x1	ReLU
Transposed Convolution	Nx64x25x1	64	5x1	2x1	ReLU
Transposed Convolution	Nx32x50x1	32	4x1	2x1	ReLU
Transposed Convolution	Nx32x100x1	32	4x1	2x1	ReLU
Transposed Convolution	Nx1x200x1	1	4x1	2x1	

Table 14: Encoder Network for *CVD Classification*

Encoder					
Layer Name	Output size	# of kernels	Kernel size	Stride	Activation
Input	Nx1x1000x4				
Convolution	Nx32x330x3	32	12x2	3x1	ReLU
Convolution	Nx32x107x2	32	10x2	3x1	ReLU
Convolution	Nx64x34x1	64	8x2	3x1	ReLU
Convolution	Nx128x9x1	128	8x1	3x1	ReLU
Convolution	Nx512x1x1	512	7x1	3x1	ReLU
Convolution	Nx20x1x1	20	1x1	1x1	

Table 15: Decoder Network for *CVD Classification*

Decoder					
Layer Name	Output size	# of kernels	Kernel size	Stride	Activation
Input	Nx1x10x1				
Transposed Convolution	Nx512x4x4	512	6x1	1x1	ReLU
Transposed Convolution	Nx128x12x4	128	4x1	2x1	ReLU
Transposed Convolution	Nx64x36x4	64	5x1	2x1	ReLU
Transposed Convolution	Nx32x109x4	32	4x1	2x1	ReLU
Transposed Convolution	Nx32x331x4	32	4x1	2x1	ReLU
Transposed Convolution	Nx1x1000x4	1	4x1	2x1	

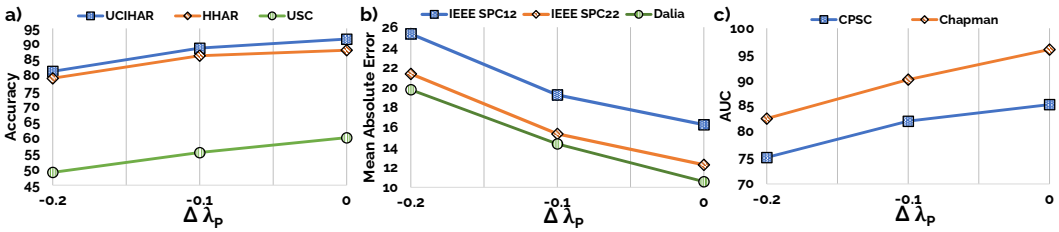

Figure 3: The experiment regarding the effect of phase mixup coefficients in eight datasets. **a)** shows the performance in *activity recognition*, **b)** is for *heart rate prediction* using PPG, and finally **c)** shows the *cardiovascular disease classification*

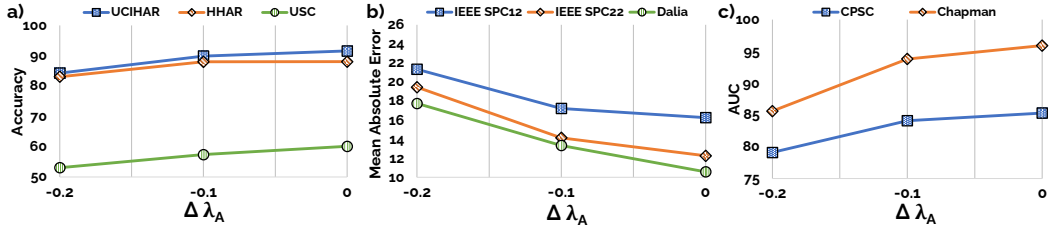

Figure 4: The experiment regarding the effect of amplitude mixup coefficients in eight datasets. **a)** shows the performance in *activity recognition*, **b)** is for *heart rate prediction* using PPG, and finally **c)** shows the *cardiovascular disease classification*

E Additonal Results

E.1 The effect and robustness of mixing coefficients

In this section, our experiments focus on observing the impact of a diverse range of mixing coefficients for both phase and amplitude components. We decrease the lower threshold of distributions for sampling the mixing coefficient by 0.1 and 0.2. For example, normally the phase mixup coefficient for *Activity Recognition* is sampled from truncated normal $\lambda_P \sim \mathcal{N}^t(1, 0.1, 0.9)$ and uniform $\lambda_P \sim U(0.9, 1)$. We decrease the low threshold value from 0.9 to 0.8 and 0.7 and report the results for both phase and amplitude. The results are reported in Figures 3 and 4 for eight datasets.

From Figures 3 and 4, it can be inferred that the phase component is more sensitive to the changes. In other words, a significant decrease in performance is observed when the mixing coefficients for the phase are sampled from lower values whereas this effect is not as much as severe for the amplitude coefficient, indicating that the amplitude of frequencies is more robust to changes compared to phase.

E.2 The performance in other frameworks

In this section, we investigate the effect of data augmentations in three different unsupervised learning frameworks which are SimCLR [15], BYOL [59] and TS-TCC [16]. For BYOL, the hidden size of the projector is set to 128, the exponential moving average parameter is set to 0.996. For TS-TCC, the λ_1 and λ_2 coefficients of temporal and contextual contrasting losses are set to 1, the same as in the original implementation. In TS-TCC, the authors proposed to use a weak (jitter and scale) and strong (permutation and jitter) augmentation together, which is shown as TS-TCC + Traditional Augs in the tables. During our experiments, we followed the original implementation of TS-TCC and applied additional augmentations after the strong one without changing the original contrastive learning framework. We set the scaling ratio to 2 and 10 for permutation (splitting the signal into a random number of segments with a maximum of 10 and randomly shuffling them). These parameters for augmentation strengths are set to the same values as in the original implementation.

Table 16: Performance comparison of our method in different CL frameworks for *Activity Recognition*

Method	UCIHAR		HHAR		USC	
	ACC↑	MF1↑	ACC↑	MF1↑	ACC↑	MF1↑
SimCLR + Traditional Augs.	87.05 ± 1.07	86.13 ± 0.96	85.48 ± 1.16	84.31 ± 1.31	53.47 ± 1.10	52.09 ± 0.95
SimCLR + Aug. Bank	65.27 ± 1.12	71.16 ± 1.24	67.95 ± 1.45	75.13 ± 1.32	43.28 ± 4.37	47.31 ± 4.68
SimCLR + DACL	73.12 ± 1.23	66.28 ± 1.11	80.89 ± 0.91	81.31 ± 0.78	53.61 ± 2.60	51.76 ± 2.21
SimCLR + Ours	91.60 ± 0.65	90.46 ± 0.53	88.05 ± 1.05	87.95 ± 1.10	**60.13** ± 0.75	**59.13** ± 0.69
BYOL + Traditional Augs.	83.41 ± 0.95	82.13 ± 1.12	86.41 ± 0.97	86.31 ± 1.10	58.34 ± 1.15	55.04 ± 1.15
BYOL + Aug. Bank	73.71 ± 0.74	69.80 ± 1.10	84.60 ± 0.93	84.65 ± 1.03	52.00 ± 1.21	49.14 ± 1.18
BYOL + DACL	73.86 ± 1.12	70.46 ± 1.24	82.76 ± 1.04	84.89 ± 0.93	47.14 ± 2.08	45.34 ± 2.98
BYOL + Ours	87.01 ± 1.10	84.92 ± 1.13	**90.31** ± 1.16	**90.45** ± 1.31	56.87 ± 0.91	55.01 ± 0.95
TS-TCC + Traditional Augs.	90.95 ± 0.87	90.30 ± 0.64	35.57 ± 1.43	40.13 ± 1.67	39.76 ± 1.61	43.12 ± 1.10
TS-TCC + Aug. Bank	76.78 ± 0.95	76.52 ± 0.97	20.25 ± 1.54	19.25 ± 1.32	21.37 ± 1.78	20.15 ± 1.48
TS-TCC + DACL	73.86 ± 1.12	70.46 ± 1.24	33.89 ± 1.87	37.41 ± 1.39	36.74 ± 1.36	40.18 ± 1.45
TS-TCC + Ours	**91.86** ± 0.97	**91.92** ± 1.02	38.45 ± 1.12	43.52 ± 1.33	42.61 ± 1.92	45.06 ± 1.11

Table 17: Performance comparison of our method in different CL frameworks for *Heart Rate Prediction*

Method	IEEE SPC12		IEEE SPC22		DaLia	
	MAE↓	RMSE↓	MAE↓	RMSE↓	MAE↓	RMSE↓
SimCLR + Traditional Augs.	20.67 ± 1.13	26.35 ± 0.98	16.84 ± 1.10	22.23 ± 0.72	12.01 ± 0.65	21.09 ± 0.86
SimCLR + Aug. Bank	27.31 ± 2.17	37.93 ± 2.96	27.84 ± 2.03	36.41 ± 3.98	35.87 ± 4.18	40.61 ± 3.74
SimCLR + DACL	21.85 ± 1.63	28.17 ± 1.75	14.67 ± 1.10	20.06 ± 1.21	18.44 ± 1.32	25.61 ± 1.45
SimCLR + Ours	16.26 ± 0.72	22.48 ± 0.95	**12.25** ± 0.47	**18.20** ± 0.61	**10.57** ± 0.55	**20.37** ± 0.73
BYOL + Traditional Augs.	20.68 ± 0.98	27.11 ± 0.85	21.16 ± 1.10	26.83 ± 1.05	12.03 ± 0.75	20.77 ± 0.83
BYOL + Aug. Bank	26.08 ± 1.05	32.62 ± 0.93	21.87 ± 1.03	29.13 ± 1.03	18.63 ± 0.91	28.30 ± 0.87
BYOL + DACL	26.45 ± 1.23	33.50 ± 1.32	21.29 ± 1.13	27.34 ± 1.33	15.11 ± 0.93	23.21 ± 0.83
BYOL + Ours	19.85 ± 0.88	26.10 ± 0.94	22.08 ± 1.24	28.20 ± 1.13	11.45 ± 0.63	20.38 ± 0.80
TS-TCC + Traditional Augs.	11.08 ± 1.03	16.97 ± 0.92	16.10 ± 1.23	26.11 ± 1.11	16.18 ± 1.03	24.27 ± 0.95
TS-TCC + Aug. Bank	11.44 ± 1.01	17.06 ± 0.94	13.79 ± 1.21	22.41 ± 1.08	17.28 ± 1.12	25.41 ± 0.98
TS-TCC + DACL	11.60 ± 1.16	18.26 ± 1.20	15.25 ± 1.26	24.40 ± 1.10	16.27 ± 1.16	24.28 ± 0.97
TS-TCC + Ours	**10.82** ± 0.65	**16.93** ± 0.73	13.63 ± 1.02	21.80 ± 1.11	15.90 ± 0.57	23.81 ± 0.89

Tables 16 17 and 18 compares the performance of three data augmentation techniques, traditional time-series augmentations, DACL and our proposed method, in contrastive learning frameworks of BYOL, SimCLR, and TS-TCC.

Table 18: Performance comparison of our method in different CL frameworks for *CVD classification*

Method	CPSC 2018	Chapman
	AUC↑	AUC↑
SimCLR + Traditional Augs.	67.86 ± 3.41	74.69 ± 2.04
SimCLR + Aug. Bank	81.78 ± 1.24	94.75 ± 0.90
SimCLR + DACL	82.38 ± 0.84	92.28 ± 0.97
SimCLR + Ours	85.30 ± 0.45	**95.90** ± 0.82
BYOL + Traditional Augs	75.41 ± 1.34	85.63 ± 1.43
BYOL + Aug. Bank	83.51 ± 1.12	91.03 ± 1.18
BYOL + DACL	77.61 ± 1.16	81.62 ± 1.24
BYOL + Ours	83.25 ± 1.03	91.23 ± 1.15
TS-TCC + Traditional Augs	87.07 ± 1.10	92.03 ± 1.17
TS-TCC + Aug. Bank	86.67 ± 1.04	92.15 ± 1.02
TS-TCC + DACL	87.63 ± 0.83	92.21 ± 0.86
TS-TCC + Ours	**88.05** ± 0.37	92.11 ± 0.75

The results show that the BYOL is more robust to the choice of augmentations than SimCLR, which is also indicated in the original paper [59]. Also, another important outcome of this ablation experiment is that when the TS-TCC framework is used for datasets HHAR and USC, the performance decreases compared to other datasets. A possible explanation for this decrease in the TS-TCC might be the hyper-parameters of the augmentations that are used in the paper. The authors change the strength of the permutation window from dataset to dataset. In our experiments, we used the same hyperparameter for all activity recognition datasets, which can explain the outcome. This ablation experiment also shows that the degree of traditional augmentations is important for contrastive learning to learn class invariant representations.

E.3 Do we still need data augmentations?

In this section, we conduct experiments to observe the performance of methods without additional augmentations. During our experiments, we searched for the best traditional augmentation technique for each method in a given task. We searched over common time series augmentation methods in literature (Table 22), and applied them with baselines. Specifically, we apply *Resample* for *Activity Recognition*, *Permutation with Noise* for *Heart rate Prediction* and *Noise with Scaling* for *CVD Classification*. We have observed that these augmentations yield the best results for all baselines when applied prior to the proposed techniques. However, for GenRep, we found that applying the augmentations after generating instances results in better performance, similar to the original work [51]. We, therefore, apply these specified augmentations for each baseline and report the corresponding results.

Different from other baselines, we observed performance increases for a few datasets when GenRep is applied without any augmentations. This phenomenon can be attributed to the generation of low-quality and less realistic positive samples, where additional augmentations lead to alterations in semantic information, due to less number of samples during training VAE models. However, in the end, we observe that applying additional augmentations always increases the performance on average for all baselines in each task.

Table 19: Performance comparison of methods without Augs. in *Activity Recognition* datasets

Method	UCIHAR		HHAR		USC	
	ACC↑	MF1↑	ACC↑	MF1↑	ACC↑	MF1↑
IDAA [53]	82.23 ± 0.69	79.84 ± 0.89	**88.98** ± 0.62	**89.01** ± 0.55	59.23 ± 1.10	56.11 ± 1.54
w/o Aug.	64.42 (-17.81)	65.17 (-14.67)	86.44 (-2.54)	86.31 (-2.70)	35.22 (-24.01)	33.62 (-22.59)
GenRep [51]	87.22 ± 1.05	86.48 ± 0.95	87.05 ± 0.95	86.45 ± 0.90	50.13 ± 2.85	49.50 ± 2.73
w/o Aug.	88.01 (+0.79)	88.12 (+1.64)	86.51 (-0.54)	86.33 (-0.22)	48.31 (-1.82)	47.33 (-2.17)
DACL [22]	73.12 ± 1.23	66.28 ± 1.11	80.89 ± 0.91	81.31 ± 0.78	53.61 ± 2.60	51.76 ± 2.21
w/o Aug.	45.17 (-27.95)	44.84 (-21.44)	56.70 (-24.19)	56.55 (-25.76)	27.12 (-26.49)	26.99 (-24.77)
Ours	**91.60** ± 0.65	**90.46** ± 0.53	88.05 ± 1.05	87.95 ± 1.10	**60.13** ± 0.75	**59.13** ± 0.69
w/o Aug.	84.04 (-5.56)	83.34 (-7.12)	86.70 (-1.35)	86.72 (-1.23)	45.55 (-14.58)	44.94 (-14.19)

Table 20: Performance comparison of methods without Augs. in *Heart Rate Prediction* datasets

Method	IEEE SPC12		IEEE SPC22		DaLia	
	MAE↓	RMSE↓	MAE↓	RMSE↓	MAE↓	RMSE↓
IDAA [53]	19.02 ± 0.96	27.42 ± 1.11	15.37 ± 1.21	22.41 ± 1.42	11.12 ± 0.64	20.45 ± 0.69
w/o Aug.	20.19 (+1.17)	28.51 (+1.09)	16.34 (+0.97)	25.75 (+3.34)	16.01 (+4.89)	25.62 (+5.17)
GenRep [51]	21.02 ± 1.41	28.42 ± 1.65	15.67 ± 1.23	22.33 ± 1.43	25.41 ± 1.62	36.83 ± 1.87
w/o Aug.	20.51 (-0.51)	28.35 (-0.07)	23.07 (+7.40)	33.20 (+10.87)	20.03 (-5.38)	31.01 (-5.82)
DACL [22]	21.85 ± 1.63	28.17 ± 1.75	14.67 ± 1.10	20.06 ± 1.21	18.44 ± 1.32	25.61 ± 1.45
w/o Aug.	22.75 (+0.90)	29.90 (+1.73)	20.88 (+6.21)	29.51 (+2.70)	28.24 (+9.45)	37.33 (+11.72)
Ours	**16.26** ± 0.72	**22.48** ± 0.95	**12.25** ± 0.47	**18.20** ± 0.61	**10.57** ± 0.55	**20.37** ± 0.73
w/o Aug.	19.41 (+3.15)	26.23 (+3.75)	16.41 (+4.16)	25.71 (+7.51)	16.73 (+6.16)	27.43 (+7.06)

Table 21: Performance comparison of methods without Augs. in *CVD classification* datasets

Method	CPSC 2018	Chapman
	AUC↑	AUC↑
IDAA [53]	80.90 ± 0.73	93.63 ± 0.91
w/o Aug.	79.00 (-1.90)	92.37 (-1.26)
GenRep [51]	52.49 ± 3.43	86.72 ± 1.13
w/o Aug.	45.17 (-7.32)	84.51 (-2.21)
DACL [22]	82.38 ± 0.84	92.28 ± 0.97
w/o Aug.	73.00 (-9.38)	75.10 (-17.18)
Ours	**85.30** ± 0.45	**95.90** ± 0.82
w/o Aug.	79.67 (-5.63)	93.48 (-2.42)

Table 22: Common time series augmentations [19]

Domain	Augmentation	Details
Time	Noise	Add Gaussian noise sampled from normal distribution, $\mathcal{N}(0, 0.4)$
	Scale	Amplify channels by a random distortion sampled from normal distribution $\mathcal{N}(2, 1.1)$
	Shuffle	Randomly permute the channels of the sample. (Not available for *Heart rate Prediction*)
	Negate	Multiply the value of the signal by a factor of -1
	Permute	Split signals into no more than 5 segments, then permute the segments and combine them into the original shape
	Resample	Interpolate the time-series to 3 times its original sampling rate and randomly down-sample to its initial dimensions
	Rotation	Rotate the 3-axial (x, y, and z) readings of each IMU sensor by a random degree, which follows a uniform around a random axis in the 3D space. (Only applied for *Activity Recognition*)
	Time Flip	Flip the time series in time for all channels, i.e., $\mathbf{x}_{Aug}[n] = \mathbf{x}[-n]$
	Random Zero Out	Randomly chose a section to zero out
	Permutation + Noise	Combination of Permutation and Noise
	Noise + Scale	Combination of Noise and Scaling
Frequency	Highpass	Apply a highpass filter in the frequency domain to reserve high-frequency components
	Lowpass	Apply a lowpass filter in the frequency domain to reserve low-frequency components
	Phase shift	Shift the phase of time-series data with a randomly generalized number
	Noise in Frequency	Add Gaussian noise, sampled from normal distribution $\mathcal{N}(0, 0.5)$, to the frequency spectrum

E.4 The effect of interpolating phase components

Here, we investigate the effect of phase interpolation of two samples on the CL performance. In our proposed method, we bring the phase components of the two coherent signals together by adding a small value to the anchor's phase in the direction of the other sample. In this section, we apply the opposite case of our proposed method and increase the gap of phase difference between the anchor and randomly chosen sample. However, we mix their amplitudes according to our proposed method to only observe the phase effect. In other words, we perform the mixup as in Equation 46. Note that the phase mixing in Equation 46 differs from the proposed method only by the sign change.

$$\begin{aligned} \mathbf{x}^{+} &= \mathcal{F}^{-1}(A(\mathbf{x}^{+}) \angle P(\mathbf{x}^{+})) \quad \text{where} \\ A(\mathbf{x}^{+}) &= \lambda_A A(\mathbf{x}) + (1-\lambda_A) A(\tilde{\mathbf{x}}) \quad \text{and} \\ P(\mathbf{x}^{+}) &= P(\mathbf{x}) + \Delta\Theta * (1-\lambda_P) \end{aligned} \tag{46}$$

Also, It is important to note that we sample the mixing coefficients for both amplitude and phase from the same distributions in the proposed method to have a fair comparison. Tables 23 24 25.

Table 23: Performance comparison of our method and its ablation regarding the phase interpolation in SimCLR and BYOL frameworks for *Activity Recognition*

Method	UCIHAR		HHAR		USC	
	ACC↑	MF1↑	ACC↑	MF1↑	ACC↑	MF1↑
SimCLR + Traditional Augs.	87.05 ± 1.07	86.13 ± 0.96	85.48 ± 1.16	84.31 ± 1.31	53.47 ± 1.10	52.09 ± 0.95
SimCLR + Phase Gap	79.62 ± 1.10	80.57 ± 1.03	86.55 ± 0.83	86.68 ± 0.71	53.61 ± 2.60	51.76 ± 2.21
SimCLR + Ours	**91.60** ± 0.65	**90.46** ± 0.53	88.05 ± 1.05	87.95 ± 1.10	**60.13** ± 0.75	**59.13** ± 0.69
BYOL + Traditional Augs.	83.41 ± 0.95	82.13 ± 1.12	86.41 ± 0.97	86.31 ± 1.10	58.34 ± 1.15	55.04 ± 1.15
BYOL + Phase Gap	78.66 ± 0.63	75.45 ± 1.02	85.82 ± 0.91	85.16 ± 0.92	56.14 ± 0.67	56.20 ± 0.75
BYOL + Ours	87.01 ± 1.10	84.92 ± 1.13	**90.31** ± 1.16	**90.45** ± 1.31	56.87 ± 0.91	55.01 ± 0.95

Table 24: Performance comparison of our method and its ablation regarding the phase interpolation in SimCLR and BYOL frameworks for *Heart Rate Prediction*

Method	IEEE SPC12		IEEE SPC22		DaLia	
	MAE↓	RMSE↓	MAE↓	RMSE↓	MAE↓	RMSE↓
SimCLR + Traditional Augs.	20.67 ± 1.13	26.35 ± 0.98	16.84 ± 1.10	22.23 ± 0.72	12.01 ± 0.65	21.09 ± 0.86
SimCLR + Phase Gap	18.90 ± 1.43	25.29 ± 1.56	14.60 ± 1.03	19.84 ± 1.15	17.57 ± 1.13	27.72 ± 1.35
SimCLR + Ours	**16.26** ± 0.72	**22.48** ± 0.95	**12.25** ± 0.47	**18.20** ± 0.61	**10.57** ± 0.55	**20.37** ± 0.73
BYOL + Traditional Augs.	20.68 ± 0.98	27.11 ± 0.85	21.16 ± 1.10	26.83 ± 1.05	12.03 ± 0.75	20.77 ± 0.83
BYOL + Phase Gap	25.93 ± 0.96	32.68 ± 0.90	21.87 ± 1.03	29.13 ± 1.03	17.46 ± 0.83	27.24 ± 0.83
BYOL + Ours	19.85 ± 0.88	26.10 ± 0.94	22.08 ± 1.24	28.20 ± 1.13	11.45 ± 0.63	20.38 ± 0.80

Table 25: Performance comparison of our method and its ablation regarding the phase interpolation in SimCLR and BYOL frameworks for *CVD classification*

Method	CPSC 2018	Chapman
	AUC↑	AUC↑
SimCLR + Traditional Augs.	67.86 ± 3.41	74.69 ± 2.04
SimCLR + Phase Gap	77.45 ± 1.10	91.95 ± 0.91
SimCLR + Ours	**85.30** ± 0.45	**95.90** ± 0.82
BYOL + Traditional Augs	75.41 ± 1.34	85.63 ± 1.43
BYOL + Phase Gap	83.11 ± 1.03	91.02 ± 1.11
BYOL + Ours	83.25 ± 1.03	91.23 ± 1.15

E.5 The comparison of Mixup methods

In this section, we give a detailed comparison of prior mixup methods with ours below tables, which are the explicit numbers for Figure 2. Our method demonstrates superior performance compared to previous mixup techniques in 11 out of 14 metrics, indicating its effectiveness. Additionally, the Amplitude Mixup technique, which yields comparable results in two datasets, further supports our claim regarding the destructive effect of simultaneously mixing phase and magnitude for time series. The relatively lower performance of Amplitude Mixup for some datasets can be explained by its limited diversity in generating positive samples since this technique has no solution for mixing the phase of samples in randomly chosen pairs. In other words, as the phase of the augmented instance is the same as the anchor in Amplitude Mix, the diversity of generated positive samples is less compared to other techniques.

Table 26: Performance comparison of ours with prior mixups in *Activity Recognition* datasets

Method	UCIHAR		HHAR		USC	
	ACC↑	MF1↑	ACC↑	MF1↑	ACC↑	MF1↑
Geo	36.31 ± 10.15	33.21 ± 12.25	33.16 ± 8.32	31.15 ± 9.25	24.85 ± 9.43	21.64 ± 8.94
Amp	81.76 ± 0.89	80.78 ± 0.78	**87.85** ± 0.83	**85.53** ± 1.10	41.29 ± 0.56	39.77 ± 1.03
Spec	40.14 ± 2.05	38.34 ± 1.95	56.73 ± 2.01	53.54 ± 1.98	23.45 ± 2.55	21.30 ± 2.41
Cut	50.21 ± 1.34	48.23 ± 1.23	57.71 ± 1.12	53.87 ± 1.09	25.63 ± 2.95	23.41 ± 3.11
Binary	74.13 ± 1.12	71.31 ± 1.10	77.12 ± 0.75	75.23 ± 0.95	42.21 ± 0.97	41.53 ± 1.10
Linear	82.23 ± 2.10	80.25 ± 1.93	80.11 ± 2.05	81.31 ± 1.73	40.15 ± 1.43	39.71 ± 1.14
Ours	**84.30** ± 0.73	**83.23** ± 0.58	84.51 ± 1.10	83.98 ± 1.03	**45.36** ± 0.97	**43.14** ± 0.81

Table 27: Performance comparison of ours with prior mixups in *Heart Rate Prediction* datasets

Method	IEEE SPC12		IEEE SPC 22		DaLia	
	MAE↓	RMSE↓	MAE↓	RMSE↓	MAE↓	RMSE↓
Geo	32.65 ± 7.25	48.90 ± 9.87	37.15 ± 6.74	36.32 ± 6.21	38.45 ± 7.31	41.32 ± 6.21
Amp	23.01 ± 0.95	30.10 ± 1.04	18.07 ± 1.13	23.13 ± 1.43	19.05 ± 1.63	30.41 ± 1.65
Spec	24.09 ± 4.10	38.41 ± 3.98	24.41 ± 4.10	29.93 ± 4.10	26.71 ± 4.34	35.31 ± 3.93
Cut	24.98 ± 3.93	35.67 ± 4.15	21.77 ± 4.45	28.43 ± 3.97	31.75 ± 4.10	43.56 ± 3.88
Binary	32.23 ± 1.67	40.21 ± 1.98	22.55 ± 1.87	28.78 ± 2.10	19.71 ± 2.15	28.83 ± 2.45
Linear	24.31 ± 1.54	31.29 ± 1.75	18.52 ± 1.43	22.54 ± 1.49	24.16 ± 1.89	32.46 ± 1.97
Ours	**21.13** ± 0.89	**28.21** ± 1.15	**16.17** ± 0.85	**21.13** ± 1.05	**16.64** ± 1.20	**28.43** ± 1.43

Table 28: Performance comparison of ours with prior mixups in *CVD classification* datasets

Method	CPSC 2018	Chapman
	AUC↑	AUC↑
Geo	45.65 ± 6.43	61.32 ± 5.79
Amp	**84.10** ± 1.05	89.83 ± 1.12
Spec	69.26 ± 3.10	70.48 ± 3.05
Cut	72.20 ± 2.98	79.23 ± 2.75
Binary	80.53 ± 1.62	82.56 ± 1.45
Linear	78.02 ± 1.43	90.21 ± 1.15
Ours	83.79 ± 1.10	**93.85** ± 1.05

F Illustrative Examples

In this section, we show examples of the destructive behavior of linear mixup and how our proposed mixup technique solves this problem. In Figure 5 **a)**, we show two PPG waveforms that are obtained from IEEE SPC15 with the same label i.e., the same heart rate value. Also, we give the corresponding frequency domain transformations of these two waveforms in Figure 5 **b)** where the frequency axis is converted to heart rate in beats-per-minute i.e., 1 Hz corresponds to 60 bpm.

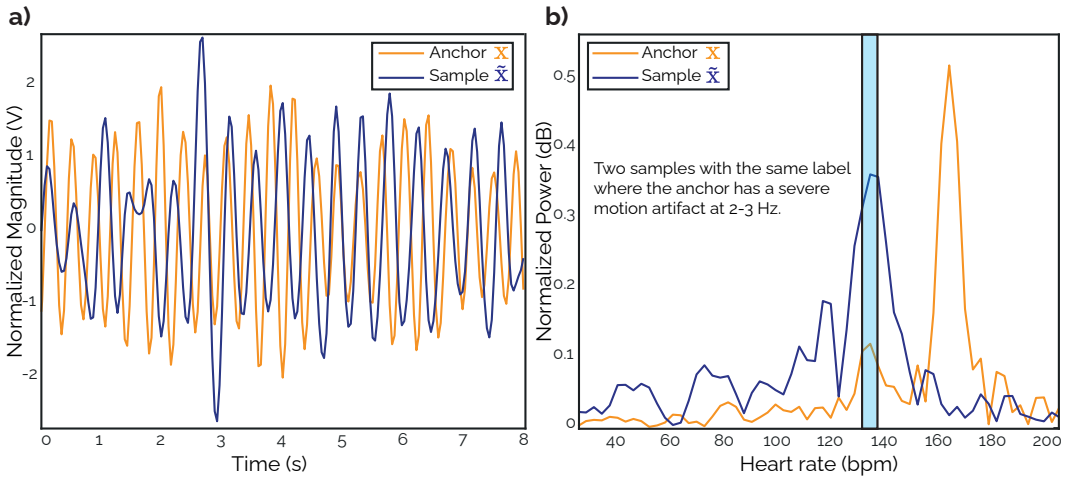

Figure 5: **a)** The waveforms of anchor and random sample, **b)** The frequency domain ($A(\mathbf{x})$) representation of two samples.

When the linear mixup is applied as in Equation 47 with a λ of 0.9, the resulting waveform is anticipated to contain heart rate information to an extent similar to both the anchor and the sample.

$$\mathbf{x}^{+} = \lambda \mathbf{x} + (1 - \lambda)\tilde{\mathbf{x}} \tag{47}$$

However, when there is a phase difference greater than $\pi/2$ between these two samples in the frequencies where the task-specific information is carried, the linear mixup destroys the information.

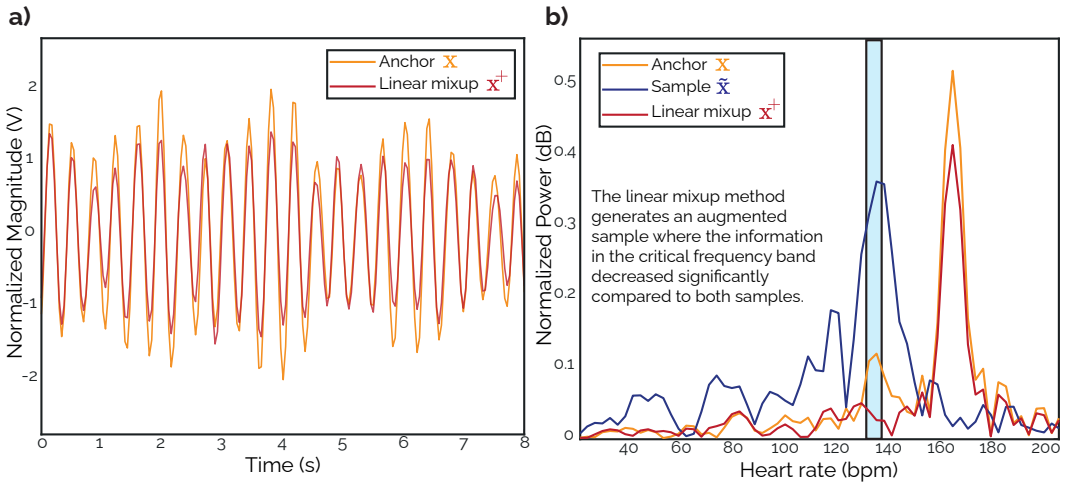

Figure 6: **a)** The waveform of anchor and augmented sample with linear mixup, **b)** The frequency domain ($A(\mathbf{x})$) representations of samples where the augmented waveform has lost all the information in the critical frequency band, i.e., the task-specific information is lost.

Figures 5 and 6 demonstrate the destructive behavior of linear mixup instead of feature interpolation. The linear mixup technique destroys the task-specific information even though the two samples have the same labels and the mixup ratio is relatively high.

As our proposed mixup prevents this problem and interpolates between features of two samples, the information is not lost but rather enhanced as both samples have the same label, shown in Figure 7.

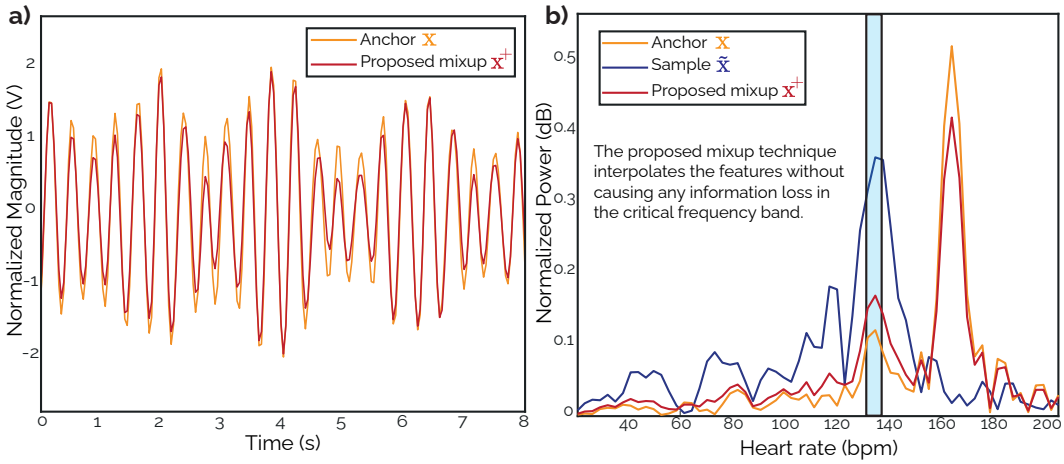

Figure 7: **a)** The waveform of anchor and augmented sample with proposed mixup technique, **b)** The frequency domain ($A(\mathbf{x})$) representations of samples where the augmented waveform carries the information in the critical frequency band as an interpolation of two samples.

As can be seen From figures 6 and 7, our proposed mixup technique not only prevents information loss due to linear mixup but also generates an interpolated sample.

G Performance in Supervised Learning Paradigm

We also conduct experiments in the supervised learning paradigm with our proposed mixup method to see its effectiveness in different learning paradigms. We compare the performance of our method with prior mixup techniques. During the experiments, we follow the original implementation where the mixup is applied to the same minibatch after random shuffling. In the seminal work of mixup [31], the authors stated that interpolating only between inputs with equal labels does not lead to performance gains. Therefore, we only perform the tailored mixup without implementing any VAEs to check the similarity of the randomly chosen samples. We implement the tailored mixup for the supervised learning paradigm as follows.

$$\mathbf{x}^+ = \mathcal{F}^{-1}(A(\mathbf{x}^+) \angle P(\mathbf{x}^+)) \quad \text{where} \quad A(\mathbf{x}^+) = \lambda_A A(\mathbf{x}) + (1-\lambda_A) A(\tilde{\mathbf{x}}) \quad \text{and}$$
$$P(\mathbf{x}^+) = \begin{cases} P(\mathbf{x}) - |\Delta\Theta| * (1-\lambda_P), & \text{if } \Delta\Theta > 0 \text{ and } \lambda_A \geq 0.5 \\ P(\mathbf{x}) + |\Delta\Theta| * (1-\lambda_P), & \text{if } \Delta\Theta \leq 0 \text{ and } \lambda_A \geq 0.5 \\ P(\tilde{\mathbf{x}}) - |\Delta\Theta| * (1-\lambda_P), & \text{if } \Delta\Theta > 0 \text{ and } \lambda_A < 0.5 \\ P(\tilde{\mathbf{x}}) + |\Delta\Theta| * (1-\lambda_P), & \text{if } \Delta\Theta \leq 0 \text{ and } \lambda_A < 0.5 \end{cases} \tag{48}$$
$$y^+ = \lambda_A y_{\mathbf{x}} + (1-\lambda_A) y_{\tilde{\mathbf{x}}},$$

where the coefficient for the λ_A is chosen from a beta distribution with $\alpha \in [0.1, 0.4]$ within the same range of the original implementation [31]. The mixing for the phase is constrained to our original implementation with a uniform $\lambda_P \sim U(0.9, 1)$. We searched for the best α value for each time-series task and augmentation method. Unlike linear mixup and our mixup approach, for cutmix, we followed the recommendation from the original paper and searched the α value close to 1.

Table 29: Performance comparison in *Activity Recognition* within supervised learning scheme

Method	UCIHAR		HHAR		USC	
	ACC↑	MF1↑	ACC↑	MF1↑	ACC↑	MF1↑
W/o Augs.	65.66 ± 0.23	61.21 ± 0.15	91.58 ± 0.07	91.64 ± 0.11	71.93 ± 0.54	68.43 ± 0.78
Linear Mix	77.06 ± 0.18	73.21 ± 0.17	93.64 ± 0.17	93.67 ± 0.08	74.45 ± 0.28	71.93 ± 0.43
Amp Mix	70.96 ± 0.19	67.14 ± 0.33	92.50 ± 0.15	92.54 ± 0.10	74.02 ± 0.19	71.90 ± 0.26
Binary Mix	69.01 ± 0.36	71.63 ± 0.11	92.36 ± 0.19	92.42 ± 0.10	72.81 ± 0.15	70.98 ± 0.35
CutMix	67.14 ± 0.54	63.31 ± 0.48	90.37 ± 0.43	90.36 ± 0.76	57.89 ± 0.34	61.45 ± 0.57
Ours	**81.60** ± 0.15	**79.35** ± 0.13	**94.02** ± 0.05	**94.00** ± 0.06	**74.85** ± 0.19	**72.45** ± 0.34

Table 30: Performance comparison in *Heart Rate Prediction* within supervised learning scheme

Method	IEEE SPC12		IEEE SPC 22		DaLia	
	MAE↓	RMSE↓	MAE↓	RMSE↓	MAE↓	RMSE↓
W/o Augs.	20.01 ± 0.03	27.16 ± 0.05	20.29 ± 0.87	26.60 ± 1.13	6.58 ± 0.10	11.30 ± 0.58
Linear Mix	20.07 ± 0.09	26.93 ± 0.10	19.98 ± 0.12	24.90 ± 0.51	6.97 ± 0.14	12.07 ± 0.51
Amp Mix	20.14 ± 0.07	26.98 ± 0.07	19.61 ± 0.07	**24.11** ± 0.21	11.20 ± 0.17	16.07 ± 0.43
Binary Mix	21.05 ± 0.13	27.02 ± 0.08	19.62 ± 0.10	25.23 ± 0.13	7.35 ± 0.16	12.17 ± 0.53
CutMix	20.12 ± 0.06	**26.89** ± 0.11	19.64 ± 0.13	24.18 ± 0.20	10.78 ± 1.23	14.40 ± 1.43
Ours	**19.97** ± 0.05	26.98 ± 0.10	**19.45** ± 0.12	24.35 ± 0.18	**6.49** ± 0.08	**11.69** ± 0.10

Table 31: Performance comparison in *CVD classification* within supervised learning scheme

Method	CPSC 2018	Chapman
	AUC↑	AUC↑
W/o Augs.	82.01 ± 0.51	92.27 ± 0.35
Linear Mix	80.29 ± 0.93	93.02 ± 0.33
Amp Mix	80.01 ± 0.36	89.11 ± 0.27
Binary Mix	78.10 ± 0.98	80.31 ± 0.36
CutMix	80.75 ± 0.78	89.17 ± 0.58
Ours	**83.75** ± 0.32	**95.26** ± 0.24