Context Consistency Regularization for Label Sparsity in Time Series

Yooju Shin [1] **Susik Yoon** [2] **Hwanjun Song** [3] **Dongmin Park** [1] **Byunghyun Kim** [4] **Jae-Gil Lee** [4] **Byung Suk Lee** [5]

Abstract

Labels are typically sparse in real-world time series due to the high annotation cost. Recently, consistency regularization techniques have been used to generate artificial labels from unlabeled augmented instances. To fully exploit the sequential characteristic of time series in consistency regularization, we propose a novel method of data augmentation called *context-attached augmentation*, which adds preceding and succeeding instances to a target instance to form its augmented instance. Unlike the existing augmentation techniques that modify a target instance by directly perturbing its attributes, the context-attached augmentation generates instances augmented with varying contexts while maintaining the target instance. Based on our augmentation method, we propose a *context consistency regularization* framework, which first adds different contexts to a target instance sampled from a given time series and then shares unitary reliability-based cross-window labels across the augmented instances to maintain consistency. We demonstrate that the proposed framework outperforms the existing state-of-the-art consistency regularization frameworks through comprehensive experiments on real-world time-series datasets.

1 Introduction

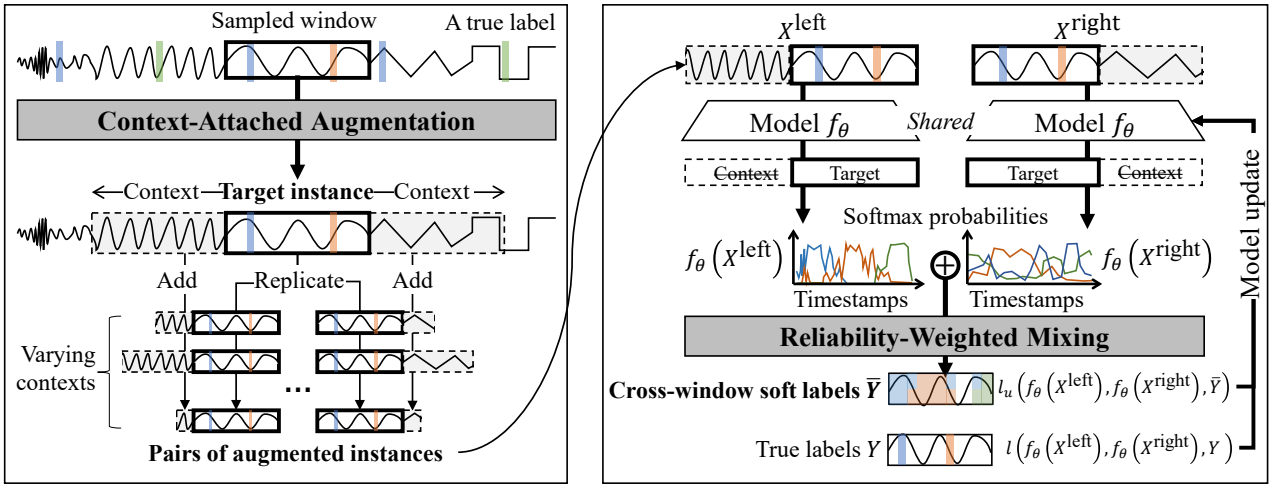
A time series is a collection of consecutive data points, often annotated with temporally coherent timestamp labels. As the boundaries of coherent labels are unknown in practice, a classifier is trained to predict the label of each timestamp (Farha & Gall, 2019). However, due to the length of and complexity in a time series, labeling every timestamp in the time series requires prohibitively high cost, and therefore, in reality, a lot of time series are only sparsely labeled (Moltisanti et al., 2019; Ma et al., 2020; Yoon et al., 2021; Shin et al., 2022). In this regard, consistency regularization is used as a promising way to train a model from sparse labels, by leveraging the model's output to infer new labels for unlabeled data points (Laine & Aila, 2017; Rizve et al., 2021). Consistency regularization enforces the consistency among the outputs of a model for a pair of augmented data instances, and thus data augmentation plays a key role in consistency regularization. Recent state-of-the-art consistency regularization methods, mostly developed for image data, necessitate domain-specific data augmentation such as color transformation and image rotation (Cubuk et al., 2020; Sohn et al., 2020; Zhang et al., 2021; Kim & Lee, 2022).

Such conventional data augmentation generates multiple different instances from a target instance (i.e., an instance for pseudo-labeling) by way of data perturbation. If data instances are independent of one another as in image data, there is no other way than to perturb the target instance itself. In contrast, using the sequential nature of time series, where data instances (segments or data points) are temporally correlated, it is feasible to generate multiple different instances from a target instance without perturbing it but by attaching its surrounding sequence (i.e., *context*). As shown in Figure 1(a), given a target instance in a time series, contexts of varying lengths are attached to the target instance to generate different pairs of "augmented" instances. We call this type of data augmentation the *context-attached augmentation*.

The key property of context-attached augmentation is to achieve the effect of data augmentation *without* perturbing a target instance. Being free of data perturbation brings several benefits. First, consistency between augmented instances can be enforced more reliably because a target instance itself is exactly the same among its augmented instances. Second, a sufficient number of augmented instances can be easily obtained by varying contexts. Third, it is computationally inexpensive, only requiring the retrieval of a sub-sequence from a time series. Moreover, context-attached augmentation can be used together with conventional data augmentation such as jittering and scaling. Thus, the novel concept of context-attached augmentation opens a new direction of context consistency regularization.

[1]Graduate School of Data Science, KAIST, Korea [2]Department of Computer Science, University of Illinois at Urbana-Champaign, USA [3]AWS AI Labs, USA [4]School of Computing, KAIST, Korea [5]Department of Computer Science, University of Vermont, USA. Correspondence to: Jae-Gil Lee <jaegil@kaist.ac.kr>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).



(a) Context-attached augmentation.

(b) Model update with cross-window soft labels.

Figure 1. Key technique and overall procedure of CrossMatch.

Despite its time-series-savvy concept and big benefits, applying context-attached augmentation for consistency regularization is challenging. First, it requires determining the proper number and length of contexts based on the trade-off between the expected performance improvement and training cost. Note that varying the context length in augmented instances incurs different complexity for a downstream task; intuitively, compared with the conventional data augmentation, a short context gives weak augmentation and a long context gives strong augmentation considering the intensity of perturbation. Second, a new consistency regularization method is needed to fully benefit from context-attached augmentation which does not perturb a target instance unlike the conventional data augmentation.

In this work, we propose a novel *context consistency regularization* framework with context-attached augmentation for time series, called *CrossMatch*. Given a target instance, CrossMatch generates a *pair* of augmented instances by attaching contexts to the preceding and succeeding positions of the target instance, respectively (Figure 1(a)). We theoretically and empirically show that such a minimal context-attached augmentation is sufficient to take a full advantage of the context consistency regularization. Then, CrossMatch conducts reliability-weighted mixing to generate cross-window soft labels from the augmented instances for regularizing the inference of the target instance (Figure 1(b)). In existing consistency regularization methods such as MixMatch (Berthelot et al., 2019) and FixMatch (Sohn et al., 2020), an artificial label is created as a form of a *hard* label; because the model's outputs for augmented instances could be biased by the perturbation of the target instance, the most confident label is chosen by averaging and sharpening in MixMatch and weak augmentation and thresholding in FixMatch. In CrossMatch, on the other hand, the model's outputs from the two augmented instances are *crossed* in the form of a *soft* label; due to its *target-preserving* property, the outputs with different contexts can be considered equally meaningful. As shown in Figure 1(b), a pair of augmented instances generated from a target instance is fed to a model to get the two softmax outputs of the target instance. Then, a single set of *cross-window* soft labels is derived and enforced to each output for consistency regularization. These augmentation and regularization procedures are repeated in each training iteration.

Overall, the main contributions are summarized as follows:

- Proposing a novel context consistency regularization framework equipped with context-attached augmentation;
- Analyzing the impact of varying contexts theoretically and empirically to explain why CrossMatch works;
- Achieving higher classification accuracy than existing state-of-the-art methods (the FixMatch-style methods with jittering and scaling) by up to 23 percentage points.

2 Related Work

2.1 Data Augmentation

Data augmentation perturbs given data instances to generate diverse and sufficient data instances to prevent overfitting (Shorten & Khoshgoftaar, 2019). The techniques used in data augmentation usually manipulate features and can be classified into two categories: (1) inner-augmentation that changes the features within a data instance and (2) inter-augmentation that exploits features across multiple data instances. Rotation, flipping, scaling, cutout, and random erasing are the examples of inner-augmentation (DeVries & Taylor, 2017; Cubuk et al., 2019). Mix-up, cut-mix, and copy-paste are the representatives of inter-augmentation that mixes two full or partial images (Zhang et al., 2018; Yun

et al., 2019; Ghiasi et al., 2021). However, these studies have not considered context-additive augmentation because they deal with independent data instances such as images.

Time-series augmentation techniques have been devised in recent literature (Wen et al., 2021). They include jittering, scaling, window warping, window cropping, and the Fourier transform specialized in the time and frequency domains, and belong to the inner-augmentation category (Um et al., 2017; Eldele et al., 2021; Yue et al., 2022; Chen et al., 2022). The inter-augmentation category is considered to be ineffective because temporal patterns could be lost after mixing two time-series segments (Iwana & Uchida, 2021). These studies also consider a set of already segmented time series as candidate augmentation targets and assume that the instances in each segment have the same labels. Thus, they are not directly applicable to a *single* continuous time series with *sparse* labels, which is a more practical and challenging setting. Moreover, all of them perturb the target instances, following the common trend of existing data augmentation, whereas our method preserves the target instances.

2.2 Semi-Supervised Learning Based on Augmentations

Semi-supervised learning (SSL) trains a model with both labeled and unlabeled data instances (Van Engelen & Hoos, 2020). Unlabeled data instances are harnessed by two popular approaches: (1) self-supervised learning which is mostly based on contrastive learning or pretext tasks (Chen et al., 2020; Grill et al., 2020; Singh et al., 2021; Yoon et al., 2023) and (2) self-training which produces artificial labels for unlabeled data instances from model predictions (Lee, 2013; Chen et al., 2018; Xie et al., 2020; Pham et al., 2021). We focus on self-training owing to its simplicity and effectiveness demonstrated in recent studies (Yang et al., 2022).

State-of-the-art self-training methods, usually based on *consistency regularization*, force consistency in model predictions from multiple augmentations of a data instance. MixMatch (Berthelot et al., 2019) averages out the predictions from multiple augmentations and sharpens the averaged prediction to reduce the entropy in the pseudo-label. ReMixMatch (Berthelot et al., 2020) generates a sharpened pseudo-label from a weak augmentation and matches it against the predictions from multiple strong augmentations. FixMatch (Sohn et al., 2020) generates a one-hot pseudo-label by choosing a single class above a *fixed* confidence threshold. FlexMatch (Zhang et al., 2021) is a variation of FixMatch, which uses a *dynamic* confidence threshold to adapt to different learning speeds among different classes. Propagation regularizer (Kim & Lee, 2022) also reduces confidence in incorrect predictions to make FixMatch robust to a more sparse label setting. However, all these methods heavily rely on domain-specific augmentation and lack for consideration of time series.

There are several time-series semi-supervised learning methods in the literature, but most of them are based on self-supervised learning. In their settings, a model is first pre-trained with time-series self-supervision and then fine-tuned with initial labels. The examples of time-series self-supervisions are (1) pretext tasks such as forecasting and temporal relation prediction (Jawed et al., 2020; Fan et al., 2021), (2) contrastive learning with the aforementioned time-series augmentation techniques (Singh et al., 2021; Xiao et al., 2022), and (3) clustering results (Singhania et al., 2022). These methods target an already-segmented time series, which cannot deal with continuous time series with sparse labels (Ma et al., 2021; Xu et al., 2022).

3 CrossMatch: Cross-Window Consistency Regularization

3.1 Preliminaries and Problem Setting

Dataset and Model: Let $\mathcal{D} = \mathcal{X} \times \mathcal{Y} = \{(\mathbf{x}_t, y_t) \mid t \in \mathcal{T}\}$ be a time series, where $\mathcal{T}$ is an index set of timestamps, $\mathbf{x}_t \in \mathbb{R}^d$ is a d-dimensional data point at timestamp t, and y_t is a corresponding class label if $\mathbf{x}_t$ is labeled or `null` otherwise. Let $\mathcal{T}_L$ be the index set of *labeled* timestamps and $\mathcal{T}_U$ be the index set of *unlabeled* timestamps, where $\mathcal{T}_L \cup \mathcal{T}_U = \mathcal{T}$ and $|\mathcal{T}_L| \ll |\mathcal{T}_U|$. Here, $\mathcal{T}_L$ is sparse, meaning that its members are few and scattered. In this work, multiple consecutive timestamps, referred to as a *segment instance* (simply, an *instance*), are processed in a batch. An instance $X = \{\mathbf{x}_t \mid t \in [m-w : m+w)\}$ is a list of consecutive $2w$ data points (timestamps) centered at timestamp m, where $[m-w : m+w)$ represents an integer interval from $m-w$ through $m+w-1$. Likewise, $Y = \{y_t \mid t \in [m-w : m+w)\}$ is a set of the corresponding class labels, where $y_t \in \{1, \dots, K\}$ and K is the number of classes. A model f_θ predicts the sequential softmax probabilities of X, i.e., $f_\theta(X) \in [0,1]^{2w \times K}$. The classification loss for training the model given X and Y is formulated as

$$\ell(X, Y) = \frac{1}{2w} \sum_t \mathbf{1}_{y_t \neq \texttt{null}} H(f_\theta(X)_{t,:}, y_t), \quad (1)$$

where $H(\cdot, \cdot)$ is sparse categorical cross-entropy and $\cdot_{t,:}$ means indexing at timestamp t.

Pseudo-Labeling: For each instance X, using the maximum softmax probabilities conditioned on a confidence threshold τ, a pseudo-label $\hat{y}_t$ at each timestamp $t \in [m-w : m+w)$ is derived by

$$\hat{y}_t = \begin{cases} \arg\max_{k \in \{1,\dots,K\}} f_\theta(X)_{t,k} & \text{if } f_\theta(X)_{t,k} > \tau \\ \texttt{null} & \text{otherwise.} \end{cases} \quad (2)$$

A set $\hat{Y} = \{\hat{y}_t \mid t \in [m-w : m+w)\}$ of the pseudo-labels for X is constructed by Equation (2). Then, the classification

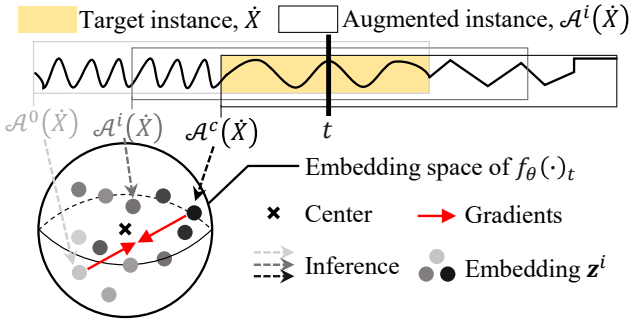


Figure 2. Intuition of the context-attached augmentation.

loss for an instance X and its set $\hat{Y}$ of the pseudo-labels obtained is formulated as $\ell(X, \hat{Y})$ with Equation (1). Note that the pseudo-labels are discarded once the model is updated with the losses calculated from the sampled instances.

Consistency Regularization: Recent consistency regularization methods force consistency between the model outputs of augmentations assuming that the class information is preserved across augmentations. For example, FixMatch (Sohn et al., 2020) matches the pseudo-label from a weak augmentation against the prediction from a strong augmentation. That is, $\ell(X, \hat{Y}) = \frac{1}{2w}\sum_t \mathbf{1}_{\max_k f_\theta(\alpha(X))_{t,k} > \tau} H(f_\theta(\mathcal{A}(X))_t, \hat{y}_t)$, where $\hat{y}_t = \arg\max_k f_\theta(\alpha(X))_{t,k}$, and α and $\mathcal{A}$ represent weak and strong augmentations, respectively. The goal of consistency regularization is to offer informative supervision to update the model by diverse augmentations and reliable pseudo-labels. In FixMatch, pseudo-labels from weak augmentations are reliable due to weak perturbation, and strong augmentations become diverse due to strong perturbation.

3.2 Context-Attached Augmentation

We first define the context-attached augmentation in Definition 3.1 and then discuss its design principle.

Definition 3.1. (CONTEXT-ATTACHED AUGMENTATION) A c-length *context-attached augmentation* $\mathcal{A}$ of a target instance $\dot{X} = \{\mathbf{x}_t | \, t \in [m-w : m+w)\}$ is attaching c surrounding data points of $\dot{X}$ such that $\mathcal{A}^i(\dot{X}) = \{\mathbf{x}_t | \, t \in [m-w-c+i : m+w+i)\}$, where $i \in [0 : c]$.

There are $c+1$ possible choices of context-attached augmentation according to where we put the context (see Figure 2). Choosing augmented instances with a larger difference in the embedding space induces stronger supervision by consistency regularization (Wang et al., 2022; Jiang et al., 2022). Thus, it is preferable to have two augmented instances whose pairwise distance is the farthest in the embedding space. In Theorem 3.2, we show that a pair of augmented instances is more likely to have a larger distance in the embedding space if they have a less number of overlapping data points under the martingale assumption.

Theorem 3.2. (EMBEDDING DISTANCE) *Let $\mathbf{z}^i$ denote an embedding of $\mathcal{A}^i(\dot{X})$ at a certain timestamp $t \in [m-w : m+w)$. Under the martingale embedding with a bounded difference assumption, as the overlap between $\mathcal{A}^i(\dot{X})$ and $\mathcal{A}^j(\dot{X})$ $(i > j)$ decreases, i.e., $(i-j)$ increases, the upperbound of the probability of the Euclidean distance between $\mathbf{z}^i$ and $\mathbf{z}^j$ being farther than ϵ increases. Formally,*

$$\Pr[\|\mathbf{z}^i - \mathbf{z}^j\|_2 \geq \epsilon] \leq 2e^2 e^{-\epsilon^2/2s^2(i-j)}, \tag{3}$$

where $0 \leq j < i \leq c$, and ϵ and s are positive real constants.

Proof. Under the martingale embedding with a bounded difference assumption, Equation (3) is simply derived by Azuma's inequality (Hayes, 2005). Refer to Appendix A for the complete proof. □

Corollary 3.3. (MAXIMUM DIFFERENCE) *Given a context length c, picking the pair of $\mathcal{A}^0(\dot{X})$ and $\mathcal{A}^c(\dot{X})$ maximizes the upperbound of the probability of the Euclidean distance between their embeddings being farther than ϵ, among all possible pairs of $\mathcal{A}^j(\dot{X})$ and $\mathcal{A}^i(\dot{X})$ $(0 \leq j < i \leq c)$.*

Proof. When $j = 0$ and $i = c$, $(i-j)$ becomes the largest. Then, Equation (3) in Theorem 3.2 concludes the proof. □

By Corollary 3.3, the consistency regularization between the furthest embeddings $\mathbf{z}^0$ and $\mathbf{z}^c$ would make the *whole* embedding space shrink toward its center of a class maximally, compared to other pairs (Van Engelen & Hoos, 2020). Therefore, the effect of consistency regularization can be maximized by picking two augmented instances $\mathcal{A}^0(\dot{X})$ and $\mathcal{A}^c(\dot{X})$. Accordingly, a context of length c is attached on the left and right sides of a target instance, respectively, and thus the two augmented instances, $\mathcal{A}^0(\dot{X}) = X^{\text{left}} = \{\mathbf{x}_t | t \in [m-w-c : m+w)\}$ and $\mathcal{A}^c(\dot{X}) = X^{\text{right}} = \{\mathbf{x}_t | t \in [m-w : m+w+c)\}$, are generated as shown in Figure 3. Each augmented instance is fed to a model, and pseudo-labels are generated from every timestamp of only the target instance (i.e., the overlap between the two augmented instances) by Equation (2). As a result, two pseudo-label sets $\hat{Y}^{\text{left}}$ and $\hat{Y}^{\text{right}}$ for the left and right augmented instances are obtained. *Reliability-weighted mixing* will be introduced in Section 3.3 to merge $\hat{Y}^{\text{left}}$ and $\hat{Y}^{\text{right}}$ while considering the reliability of each pseudo-label based on the left and right side lengths.

Another important factor is the context length c, which directly affects the supervision quality of consistency regularization. If c is too short, then two augmented instances become too similar so that the model barely changes after matching the embeddings of augmented instances (Wang et al., 2022). On the other hand, if c is too long, the embeddings diverge so that the matching would weld the representation of instances from different classes. We empirically study this trade-off and suggest a heuristic in Section 4.4.

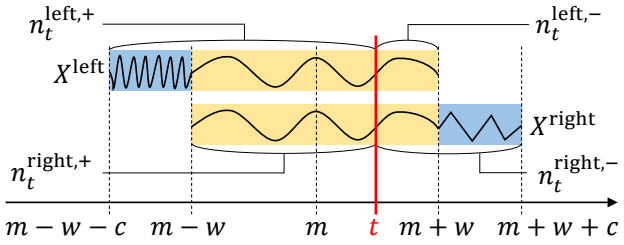


Figure 3. A target instance (yellow) and its context (blue).

3.3 Reliability-Weighted Mixing

Though all pseudo-labels generated for a target instance satisfy the confidence threshold in Equation (2), we treat them differently based on the reliability of a pseudo-label. Our rationale is that the pseudo-label becomes more reliable if (1) the model f_θ receives more data points on the left and right sides of the pseudo-label and (2) the number of data points is balanced between the two sides so that the prediction is not biased to the preceding or succeeding interval. We design a reliability function that follows our rationale and compute reliability of each pseudo-label. Using the reliability as a weight, we mix the pseudo-labels into a single *cross-window label*, which will be matched against two softmax probabilities from both of augmentations.

Let's consider two pseudo-labels $\hat{y}_t^{\text{left}}$ and $\hat{y}_t^{\text{right}}$ generated from $f_\theta(X^{\text{left}})_t$ and $f_\theta(X^{\text{right}})_t$ shown in Figure 3. For each timestamp $t \in [m-w : m+w)$ of a target instance, the length $n_t^{loc,+}$ of the left side and the length $n_t^{loc,-}$ of the right side along each augmented instance are easily calculated by

$$n_t^{loc,+} = \begin{cases} t-m+w+c & \text{if } loc = \text{left} \\ t-m+w & \text{if } loc = \text{right} \end{cases} \quad (4)$$

$$n_t^{loc,-} = \begin{cases} m+w-t & \text{if } loc = \text{left} \\ m+w+c-t & \text{if } loc = \text{right}, \end{cases} \quad (5)$$

where loc indicates the location of context attachment—either $left$ or $right$.

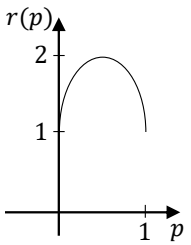


Figure 4. Visualization of Equation (6).

Per our design rationale, the reliability score becomes higher as $n_t^{loc,+}$ and $n_t^{loc,-}$ are larger and $n_t^{loc,+}$ and $n_t^{loc,-}$ are more similar with each other. This requirement can be achieved by exploiting a bell-shaped function,

$$r(p) = \sqrt{2p-p^2} + \sqrt{1-p^2}, \quad (6)$$

where $0 \le p \le 1$. Here, p is a normalization of $n_t^{loc,+}$ with respect to the full length of an augmented instance, i.e., $p^{loc}(t) = n^{loc,+}/(2w+c)$. That is, using Equation (6), we obtain two reliability scores $r(p^{\text{left}}(t))$ and $r(p^{\text{right}}(t))$ for each pseudo-label from the two augmented instances X_{left} and X_{right}. Due to the bell shape in Figure 4, a timestamp with a sufficiently long side length on both sides has an adequately high reliability in an augmented instance. If a side length on either side is too long (i.e., biased to one side), the side length on the other side is too short, and the reliability score is too low. Overall, the reliability score is maximized at the center of a given augmented instance.

Last, in order to treat each pseudo-label differently based on the reliability scores on the two sides, $r(p^{\text{left}}(t))$ and $r(p^{\text{right}}(t))$, the weight of each pseudo-label is assigned as $\beta_t^{loc} = r(p^{loc}(t))/(r(p^{\text{left}}(t)) + r(p^{\text{right}}(t)))$ by normalizing the two reliability scores. The final *cross-window soft label* for the target instances in the left and right instances is

$$\bar{y}_t = \beta_t^{\text{left}} \cdot \texttt{onehot}(\hat{y}_t^{\text{left}}) + \beta_t^{\text{right}} \cdot \texttt{onehot}(\hat{y}_t^{\text{right}}), \quad (7)$$

where the `onehot` function converts a scalar to one-hot encoded vector. In other words, reliability-weighted mixing generates a soft label by adding the two weighted pseudo-labels. Using the cross-window label set and two target instance outputs, the classification loss $\ell_u(X^{\text{left}}, X^{\text{right}}, \bar{Y})$ is computed as

$$\frac{1}{4w} \sum_t \mathbf{1}_{\bar{y}_t \neq \texttt{null}} \big(h(\mathbf{z}_t^{\text{left}}, \bar{y}_t) + h(\mathbf{z}_t^{\text{right}}, \bar{y}_t) \big), \quad (8)$$

where $\mathbf{z} = f_\theta(X)_{t,:}$, h is soft cross-entropy, and $\bar{Y} = \{\bar{y}_t | t \in [m-w : m+w)\}$. By matching a single cross-window soft label to both augmented instances, we can reduce the variance of the label to be predicted, leading to better consistency regularization (Wang et al., 2022). In addition, consistency regularization with soft labeling is more effective than with hard labeling since soft labeling can be robust to erroneous pseudo-labels when at least one pseudo-label is correct (Müller et al., 2019).

3.4 Overall Procedure of CrossMatch

Figure 1 summarizes the overall procedure of CrossMatch using context-attached augmentation and reliability-weighted mixing. Given a time series, CrossMatch first takes a target instance sampled from the time series as an input and generates a pair of augmented instances with context-attached augmentation (see Figure 1(a)). Then, a model f_θ infers the pair of augmented instances X^{left} and X^{right} to produce two independent softmax probabilities. The pseudo-labels respectively generated from the two softmax probabilities are merged ($\bigoplus$) to give cross-window soft labels $\bar{Y}$ with reliability-weighted mixing. Finally, the cross-entropy losses are computed from the cross-window soft labels $\bar{Y}$ as well as the true labels Y and used to update the model f_θ (see Figure 1(b)). These steps are repeated for randomly sampled target instances with varying c until I iterations. CrossMatch in Appendix B details each step.

Table 1. Datasets and configurations.

	$\lvert\mathcal{T}\rvert$	Length	#class	d	w	$c_{\max}$	v	I
HAPT	408K	967	6	6	512	256	0.1%	25K
mHealth	343K	2933	12	23	384	256	1.0%	50K
Opportunity	190K	109	17	113	512	64	1.0%	30K

4 Evaluation

4.1 Experiment Setting

Datasets: We use three widely-used benchmark datasets in Table 1. HAPT is a sensor time-series dataset tracking human movements in a laboratory sampled with the frequency of 50Hz (Anguita et al., 2013). mHealth is a similar action recognition dataset recorded with more wearable sensors, such as 3D accelerometers, 3D gyroscopes, 3D magnetometers, and electrodes, whose sampling frequency is 50Hz (Banos et al., 2014). Opportunity is a collection of sensor recordings at 100Hz capturing daily natural human activities with wearable, object, and ambient sensors (Roggen et al., 2010). For each originally fully-labeled dataset, we randomly sample the same number of timestamps for each class and drop the labels from the rest of the timestamps to generate a *sparsely-labeled* time series, leaving multiple labeled time spans located randomly. The sampled timestamps become a labeled timestamp set $\mathcal{T}_L$ for the given ratio of labeled timestamps $v = |\mathcal{T}_L|/|\mathcal{T}|$. Table 1 summarizes the statistics of datasets and the data-specific parameters, specifying the number of timestamps, the average length of a segment with a single class, the number of classes, the dimensionality of a data point, the half length of a target instance w, the maximum context length $c_{\max}$, the default ratio of labeled timestamps v, and the maximum number of iterations I. In particular, $c_{\max}$ is set to be sufficiently smaller than the average length of a segment.

The value of w, half of the length of a target instance, needs to be carefully chosen for each data set. Too large w may include semantically irrelevant data points, while too small w may not give enough temporal information for prediction. Either way leads to incorrect pseudo-labels and ultimately degrades the performance. Thus, we set $2w$, the length of a target instance, as a value lower than the mean length of a label-coherent segment, which could be known in advance or estimated by a given set $\mathcal{T}_L$ of labeled timestamps.

Implementation Details: We use a widely-used multi-stage temporal convolutional network named MS-TCN (Farha & Gall, 2019), which is applicable to our problem setting because it gives softmax probabilities for each timestamp in an instance X. We follow the same hyperparameter and configuration in the original MS-TCN, except the learning rate and an optimizer adjusted for consistency regularization with sparse labels. See Appendix C for more details.

For CrossMatch, we set the confidence threshold τ to 0.95 and the weight of the unlabeled loss λ to 1. The model is first trained without any pseudo-labels, i.e., only using the labeled batches. We start to update a model with pseudo-labels after the number of pseudo-labels in each class for a batch is balanced. Formally, this condition is satisfied when the entropy of the numbers of pseudo-labels per class is above 0.99 for the last 100 iterations; it is enforced to prevent early confirmation bias in consistency regularization (Kim & Lee, 2022). If a data point x_t in an instance X has a true label (i.e., $t \in \mathcal{T}_L$), CrossMatch uses the true label instead of the generated pseudo-label.

For every experiment, we use Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz and NVIDIA RTX 3090. The source code is provided at `https://github.com/kaist-dmlab/CrossMatch`.

Evaluation Metrics: We measure *timestamp accuracy* and *segmental F1 score* with five-fold cross validation and report the average value with standard deviation of five runs. For sequential classifiers such as MS-TCN, timestamp accuracy (denoted as TS accuracy) and segmental F1 score (denoted as F1@25) measure the performance of classification at each timestamp and segment respectively (Li et al., 2021; Kumar et al., 2022). To evaluate pseudo-labeling performance, we report *pseudo-label F1 score* (denoted as PLF) as well. We define each metric in detail in Appendix D.

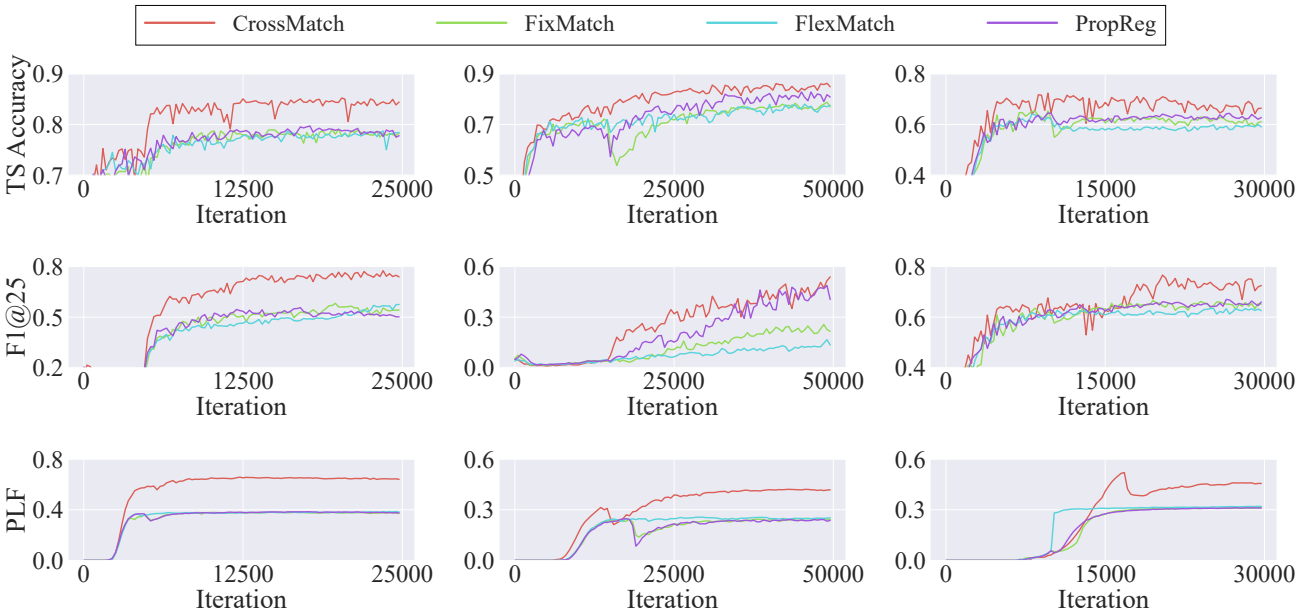
Baselines: We compare CrossMatch with three state-of-the-art consistency regularization methods: FixMatch (Sohn et al., 2020), FlexMatch (Zhang et al., 2021), and ProPReg (Kim & Lee, 2022). As discussed in Section 2, these methods require inner-instance augmentation for consistency regularization. We use two popular time-series augmentations: jittering and scaling, where weak augmentation $\alpha(X) = \mathsf{jittering}(X)$ and strong augmentation $\mathcal{A}(X) = \mathsf{jittering}(\mathsf{scaling}(X))$ (Um et al., 2017). Throughout the experiments, we use the same hyperparameters for all methods except the instance length due to the context-attached augmentation. We also compare CrossMatch with the variants of the compared methods supported by our context-attached augmentation.

4.2 Comparison with State-of-the-Art Methods

Overall Comparison: Table 2 shows the timestamp accuracy and F1@25 score on three datasets with varying label ratios; each value is obtained by averaging over the last 20 iterations for reliable results. Compared with other methods, CrossMatch achieves the best classification performance with a statistical significance of 0.05 using independent (unpaired) t-test for all datasets except HAPT $10\times$. This is mainly because consistency regularization using context-attached augmentation is more informative than inner-instance augmentation used in other consistency regu-

Table 2. Timestamp accuracy and F1@25 score averaged over the last 20 iterations with adjusting the label ratio v ($1\times$, $5\times$, and $10\times$ of a default value). The best values are marked in bold.

Method	Ratio	FixMatch		FlexMatch		PropReg		CrossMatch	
		TS Accuracy	F1@25	TS Accuracy	F1@25	TS Accuracy	F1@25	TS Accuracy	F1@25
HAPT	$1\times$	0.78 ± 0.01	0.55 ± 0.04	0.78 ± 0.01	0.54 ± 0.04	0.79 ± 0.01	0.52 ± 0.04	$\mathbf{0.84 \pm 0.01}$	$\mathbf{0.75 \pm 0.02}$
	$5\times$	0.85 ± 0.01	0.51 ± 0.04	0.84 ± 0.01	0.50 ± 0.04	0.82 ± 0.01	0.49 ± 0.04	$\mathbf{0.88 \pm 0.01}$	$\mathbf{0.70 \pm 0.02}$
	$10\times$	$\mathbf{0.96 \pm 0.00}$	0.87 ± 0.01	$\mathbf{0.96 \pm 0.00}$	$\mathbf{0.90 \pm 0.01}$	$\mathbf{0.96 \pm 0.00}$	0.89 ± 0.01	0.95 ± 0.01	0.84 ± 0.02
mHealth	$1\times$	0.77 ± 0.01	0.23 ± 0.01	0.77 ± 0.01	0.13 ± 0.01	0.81 ± 0.01	0.43 ± 0.03	$\mathbf{0.85 \pm 0.01}$	$\mathbf{0.45 \pm 0.02}$
	$5\times$	0.91 ± 0.01	0.65 ± 0.02	0.91 ± 0.01	0.62 ± 0.03	0.91 ± 0.01	0.67 ± 0.03	$\mathbf{0.94 \pm 0.01}$	$\mathbf{0.74 \pm 0.03}$
	$10\times$	0.91 ± 0.00	0.70 ± 0.02	0.90 ± 0.00	0.66 ± 0.02	0.90 ± 0.01	0.70 ± 0.02	$\mathbf{0.95 \pm 0.01}$	$\mathbf{0.84 \pm 0.03}$
Opportunity	$1\times$	0.61 ± 0.04	0.65 ± 0.05	0.59 ± 0.03	0.63 ± 0.05	0.63 ± 0.03	0.65 ± 0.05	$\mathbf{0.67 \pm 0.02}$	$\mathbf{0.73 \pm 0.04}$
	$5\times$	0.73 ± 0.03	0.73 ± 0.04	0.72 ± 0.03	0.75 ± 0.05	0.73 ± 0.03	0.74 ± 0.05	$\mathbf{0.78 \pm 0.02}$	$\mathbf{0.82 \pm 0.03}$
	$10\times$	0.75 ± 0.03	0.75 ± 0.04	0.73 ± 0.03	0.77 ± 0.04	0.75 ± 0.03	0.76 ± 0.04	$\mathbf{0.83 \pm 0.02}$	$\mathbf{0.86 \pm 0.02}$



(a) HAPT. (b) mHealth. (c) Opportunity.

Figure 5. Training curve of the classification performance (the first two rows) and pseudo-labeling performance (the last row) over training iterations in Algorithm 1. The full y-axis is shown in Figure 7 of Appendix E.

larization methods. In particular, CrossMatch exhibits much better performance than others especially when the initial label ratio is low. For instance, with only 0.1% of the labeled timestamps in HAPT, CrossMatch outperforms FixMatch by 20 percentage points (%p), FlexMatch by 21%p, and PropReg by 23%p in F1@25 (see the first row in HAPT of Table 2). Therefore, the performance dominance of CrossMatch indicates the context-attached augmentation with reliability-weighted mixing indeed helps the model select more reliable pseudo-labels even when softmax probabilities fluctuate due to label scarcity and insufficient training.

Training Curve Analysis: Figure 5 shows the training curves of classification and PLF over the entire training iteration. Please refer to Appendix E for the same results with standard deviation and other metrics related to pseudo-labeling. CrossMatch shows much higher performance than the other methods with respect to timestamp accuracy and F1@25 even in the early stage of training, reaching the highest performance in most cases (see the first two rows in Figure 5). This is attributed to its robustness in reliability-weighted mixing in the early stage of consistency regularization and its credibility in enforcing consistency between two augmented instances from context-attached augmentation. For example, as shown in mHealth of Figure 5(b), other methods suffer from the low-quality pseudo-labels at the iteration of around $17,000$ when the warm-up period ends, thereby showing a temporary drop in timestamp accuracy. Although the accuracy recovers gradually, the final accuracy is far behind the accuracy CrossMatch achieves.

Table 3. Timestamp accuracy and F1@25 score averaged over the last 20 iterations with varying context attachment for context-attached augmentation (CAA). The best values are marked in bold.

Variations	Metrics	One-sided CAA	Random-pair CAA	Multiple-2 CAA	Multiple-3 CAA	Multiple-5 CAA	Multiple-7 CAA
HAPT	TS Accuracy	0.72 ± 0.01	0.67 ± 0.02	0.84 ± 0.01	0.85 ± 0.01	0.85 ± 0.01	$\mathbf{0.86 \pm 0.01}$
	F1@25	0.65 ± 0.03	0.44 ± 0.03	$\mathbf{0.75 \pm 0.02}$	0.70 ± 0.02	0.74 ± 0.02	$\mathbf{0.75 \pm 0.04}$
mHealth	TS Accuracy	0.77 ± 0.02	0.75 ± 0.01	0.85 ± 0.01	0.86 ± 0.02	0.86 ± 0.02	$\mathbf{0.87 \pm 0.01}$
	F1@25	0.41 ± 0.04	0.21 ± 0.01	$\mathbf{0.45 \pm 0.02}$	0.40 ± 0.04	0.42 ± 0.04	0.44 ± 0.02
Opportunity	TS Accuracy	0.61 ± 0.07	0.61 ± 0.05	$\mathbf{0.67 \pm 0.02}$	$\mathbf{0.67 \pm 0.05}$	0.70 ± 0.04	0.69 ± 0.02
	F1@25	0.64 ± 0.07	0.68 ± 0.03	0.73 ± 0.04	$\mathbf{0.78 \pm 0.08}$	0.77 ± 0.07	$\mathbf{0.78 \pm 0.03}$

Table 4. Timestamp accuracy and F1@25 score averaged over the last 20 iterations with varying context lengths using CrossMatch. The best values are marked in bold.

$c_{\max}$	$0.25\times$		$0.5\times$		$1.00\times$		$2.00\times$		$4.00\times$	
	TS Accuracy	F1@25	TS Accuracy	F1@25	TS Accuracy	F1@25	TS Accuracy	F1@25	TS Accuracy	F1@25
HAPT	0.65 ± 0.01	0.46 ± 0.02	0.74 ± 0.01	0.63 ± 0.02	$\mathbf{0.84 \pm 0.01}$	$\mathbf{0.75 \pm 0.02}$	0.79 ± 0.02	0.65 ± 0.03	0.81 ± 0.01	0.70 ± 0.02
mHealth	0.69 ± 0.01	0.29 ± 0.02	0.79 ± 0.02	0.37 ± 0.02	$\mathbf{0.85 \pm 0.01}$	$\mathbf{0.45 \pm 0.02}$	0.82 ± 0.01	0.38 ± 0.04	0.77 ± 0.02	0.28 ± 0.03
Opportunity	0.65 ± 0.03	0.72 ± 0.04	$\mathbf{0.69 \pm 0.03}$	$\mathbf{0.73 \pm 0.04}$	0.67 ± 0.02	0.73 ± 0.04	0.66 ± 0.02	0.71 ± 0.03	0.55 ± 0.02	0.56 ± 0.04

The effectiveness of CrossMatch is also supported by its PLF computed from precision and recall (see the last row in Figure 5 and also see Figure 8 in Appendix E). Precision is the ratio of the number of correctly predicted timestamps to the number of pseudo-labeled timestamps, while recall is the ratio of the number of correctly predicted timestamps to the length of a target instance (refer to Appendix D for more details). We averaged PLF over the target instances in a batch. CrossMatch reaches the highest PLF continuously in most cases due to the reliability of our cross-window labels.

4.3 Analysis of Varying Context Attachment

We study the effect of varying the ways of context attachment in the context-attached augmentation. Given all augmented instance candidates, $\mathcal{A}^0(\dot{X}), \mathcal{A}^1(\dot{X}), \ldots, \mathcal{A}^c(\dot{X})$, for consistency regularization, *one-sided CAA* picks either $\mathcal{A}^0(\dot{X})$ (i.e., X^{left}) or $\mathcal{A}^c(\dot{X})$ (i.e., X^{right}), *Random-pair CAA* randomly picks two augmented instances, and *Multiple-N CAA* includes both $\mathcal{A}^0(\dot{X})$ and $\mathcal{A}^c(\dot{X})$ and randomly picks $N-2$ augmented instances; that is, Multiple-2 CAA is the default of CrossMatch. As shown in Table 3, one-sided CAA and Random-pair CAA perform worse than Multiple-N CAA, because of ineffective consistency regularization caused by the lack of variations in their augmented instances. While some cases Multiple-$N > 2$ CAA variations achieve higher performances than Multiple-2 CAA, the performance gain is marginal, i.e., less than 5%p. This result indeed demonstrates that our choice of context attachment in Section 3.2 is practical, considering the computational cost for preparing augmented instances.

4.4 Analysis of Varying Context Lengths

Table 4 summarizes the timestamp accuracy and F1@25 score of CrossMatch by adjusting the default context length $c_{\max}$ (in Table 1) from $0.25\times$ to $4.0\times$. If the context length is too large, augmented instances bear too much perturbation to generate high-quality cross-window labels. On the other hand, if the context length is too small, informative consistency regularization between two augmented instances becomes trivial since they show high similarity. We found out that the optimal context length is highly correlated with the average segment length of each dataset. For instance, the best timestamp accuracy of the Opportunity dataset is achieved with a relatively smaller context length (i.e., 32) than that of mHealth data (i.e., 256) because Opportunity has a much shorter mean segment length; as can be seen in Table 1, Opportunity and mHealth exhibit the shortest and the longest mean segment length among the three datasets. Therefore, the best timestamp accuracy and F1@25 score of each dataset are achieved with different context lengths. Our heuristic to find the best $c_{\max}$ is selecting $c_{\max}$ within $1/5$ and $1/2$ of the average length of segments. The average length of segments in a specific dataset is often available.

4.5 Extensions with Additional Augmentations

We further investigate three possible extensions: (1) CrossMatch to combine the two inner-instance augmentations (+IA) of jittering and scaling, (2) the variant of existing consistency regularization methods to combine our proposed context-attached augmentation (+CAA), and (3) the variant of existing consistency regularization methods to use the cut-based augmentation (+CUT). For the first variant, we perform jittering and scaling before applying the context-attached augmentation. For the second variant, we modify the existing methods in support of the context-attached augmentation. The two augmented instances (the left and right instances in Figure 3) are respectively treated as weakly and strongly augmented instances; the instance with a higher

Table 5. Timestamp accuracy and F1@25 score of CrossMatch and the compared methods with various augmentation methods. The best values are marked in bold.

Variations	Metrics	CrossMatch	FixMatch		FlexMatch		PropReg	
		+IA	+CAA	+CUT	+CAA	+CUT	+CAA	+CUT
HAPT	TS Accuracy	**0.88 ± 0.01**	0.62 ± 0.02	0.78 ± 0.01	0.64 ± 0.02	0.81 ± 0.01	0.68 ± 0.01	0.81 ± 0.01
	F1@25	**0.77 ± 0.02**	0.39 ± 0.03	0.66 ± 0.02	0.49 ± 0.03	0.67 ± 0.02	0.45 ± 0.02	0.69 ± 0.02
mHealth	TS Accuracy	**0.89 ± 0.01**	0.76 ± 0.04	0.74 ± 0.02	0.82 ± 0.01	0.76 ± 0.02	0.77 ± 0.03	0.75 ± 0.02
	F1@25	**0.67 ± 0.03**	0.35 ± 0.04	0.42 ± 0.04	0.31 ± 0.03	0.38 ± 0.03	0.30 ± 0.03	0.47 ± 0.04
Opportunity	TS Accuracy	**0.75 ± 0.02**	0.49 ± 0.02	0.35 ± 0.01	0.36 ± 0.01	0.34 ± 0.02	0.45 ± 0.01	0.39 ± 0.02
	F1@25	**0.85 ± 0.03**	0.53 ± 0.03	0.43 ± 0.01	0.37 ± 0.02	0.40 ± 0.02	0.45 ± 0.02	0.42 ± 0.01

reliability score becomes the weakly augmented instance, and that with a lower reliability score becomes the strongly augmented instance. For the third variant, weak augmentation eliminates 1% of data points in an instance, while strong augmentation eliminates 10%.

Table 5 summarizes the performances of the extensions of compared methods for all datasets. Note that we use the default label ratios of 0.1%, 1.0%, and 1.0% for the datasets, respectively. The extension of CrossMatch (+IA) shows additional performance gains of 4–8%p in timestamp accuracy and 2–22%p in F1@25 compared with the original CrossMatch. However, in general, the extensions of existing methods (+CAA) rather suffer from a significant performance drop because they failed to reliably deal with diverse augmented instances. The cut-based augmentation (+CUT) also does not improve existing consistency regularization methods in most cases, which reaffirms that inner-instance augmentation alone is not suitable for time-series. This result demonstrates that CrossMatch can be further enhanced with additional inner-instance augmentation techniques (e.g., jittering and scaling) under the proposed consistency regularization framework.

4.6 Discussion on Versatility and Validity

Context-attached augmentation in CrossMatch can be applied to any recent, popular architectures such as temporal convolutional networks and Transformers, where information is conveyed *bidirectionally* and output sequential softmax probabilities (Farha & Gall, 2019; Chen et al., 2022). If a model is unidirectional (e.g., from left to right as in the vanilla RNN and WaveNet), the context attached to the left side of a target instance would be only meaningful (Oord et al., 2016); thus, CrossMatch needs to consider a target instance and its augmented instance with only the left context in order to accommodate a unidirectional model.

One may argue that our augmentation will not work when a context has different labels than the target instance. However, such cases may occur very infrequently (i.e., only 6.4% of the target instances in the mHealth dataset) with a proper setting of the hyper-parameter $c_{\max}$, the maximum length of a context, because of temporal coherence inherent in time series. Even though a context contains different labels, our augmentation will work well in general. It is known that a set of transition patterns tend to appear repeatedly in many time-series datasets (Roggen et al., 2010; Stein & McKenna, 2013; Banos et al., 2014); for example, suppose that a transition pattern, *standing up* $\rightarrow$ *walking*, repeats in a human activity recognition dataset. Then, a different label (e.g., *standing up*) in the context is definitely helpful to predict the label (e.g., *walking*) in the target instance. Although the transition patterns do not repeat, there is a safety net, where pseudo-labels are not generated for uncertain target instances by confidence thresholding in CrossMatch. Therefore, CrossMatch hardly suffers from such transition between a target instance and its context.

5 Conclusion

In this paper, we propose CrossMatch, a novel consistency regularization framework with context-attached augmentation, for classifying time series with sparse labels. CrossMatch adds contexts to a target instance on either its left or right sides to generate two augmented instances with a maximal difference. Then, the inference of the target instance is regularized by reliability-weighted mixing of the two pseudo-labels from the two augmented instances. Our extensive experiments demonstrate that CrossMatch achieves considerably higher accuracy than state-of-the-art consistency regularization methods—by up to 23p% even with only 0.1% of labels. CrossMatch introduces a new direction of data augmentation for sequential data and has the potential to be applied to various time-series applications.

Acknowledgements

This work was partly supported by Samsung Electronics Co., Ltd. (IO201211-08051-01) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-00862, DB4DL: High-Usability and Performance In-Memory Distributed DBMS for Deep Learning).

References

Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J. L., et al. A public domain dataset for human activity recognition using smartphones. In *ESANN*, pp. 437–442, 2013.

Banos, O., Garcia, R., Holgado-Terriza, J. A., Damas, M., Pomares, H., Rojas, I., Saez, A., and Villalonga, C. mHealthDroid: A novel framework for agile development of mobile health applications. In *AAL Workshop*, pp. 91–98, 2014.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019.

Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., and Raffel, C. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *ICLR*, 2020.

Chen, D.-D., Wang, W., Gao, W., and Zhou, Z.-H. Tri-net for semi-supervised deep learning. In *IJCAI*, pp. 2014–2020, 2018.

Chen, M., Wei, F., Li, C., and Cai, D. Frame-wise action representations for long videos via sequence contrastive learning. In *CVPR*, pp. 13801–13810, 2022.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, pp. 22243–22255, 2020.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. AutoAugment: Learning augmentation strategies from data. In *CVPR*, 2019.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. RandAugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*, pp. 18613–18624, 2020.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Eldele, E., Ragab, M., Chen, Z., Wu, M., Kwoh, C. K., Li, X., and Guan, C. Time-series representation learning via temporal and contextual contrasting. In *IJCAI*, pp. 2352–2359, 2021.

Fan, H., Zhang, F., Wang, R., Huang, X., and Li, Z. Semi-supervised time series classification by temporal relation prediction. In *ICASSP*, pp. 3545–3549, 2021.

Farha, Y. A. and Gall, J. MS-TCN: Multi-stage temporal convolutional network for action segmentation. In *CVPR*, pp. 3575–3584, 2019.

Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.-Y., Cubuk, E. D., Le, Q. V., and Zoph, B. Simple copy-paste is a strong data augmentation method for instance segmentation. In *CVPR*, pp. 2918–2928, 2021.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent – a new approach to self-supervised learning. In *NeurIPS*, pp. 21271–21284, 2020.

Hansen, L. P., Roberds, W., and Sargent, T. J. Time series implications of present value budget balance and of martingale models of consumption and taxes. In *Rational Expectations Econometrics*, pp. 121–161. CRC Press, 2019.

Hayes, T. P. A large-deviation inequality for vector-valued martingales. *Combinatorics, Probability and Computing*, 2005.

Ho, S.-S. A martingale framework for concept change detection in time-varying data streams. In *ICML*, pp. 321–327, 2005.

Iwana, B. K. and Uchida, S. An empirical survey of data augmentation for time series classification with neural networks. *Plos One*, 16(7):e0254841, 2021.

Jawed, S., Grabocka, J., and Schmidt-Thieme, L. Self-supervised learning for semi-supervised time series classification. In *PAKDD*, pp. 499–511, 2020.

Jiang, Y., Li, X., Chen, Y., He, Y., Xu, Q., Yang, Z., Cao, X., and Huang, Q. Maxmatch: Semi-supervised learning with worst-case consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Kim, N.-r. and Lee, J.-H. Propagation regularizer for semi-supervised learning with extremely scarce labeled samples. In *CVPR*, pp. 14401–14410, 2022.

Kumar, S., Haresh, S., Ahmed, A., Konin, A., Zia, M. Z., and Tran, Q.-H. Unsupervised action segmentation by joint representation learning and online clustering. In *CVPR*, pp. 20174–20185, 2022.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. In *ICLR*, 2017.

Lee, D.-H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop*, pp. 896, 2013.

Li, Z., Abu Farha, Y., and Gall, J. Temporal action segmentation from timestamp supervision. In *CVPR*, pp. 8365–8374, 2021.

Ma, F., Zhu, L., Yang, Y., Zha, S., Kundu, G., Feiszli, M., and Shou, Z. SF-Net: Single-frame supervision for temporal action localization. In *ECCV*, pp. 420–437, 2020.

Ma, Q., Zheng, Z., Zheng, J., Li, S., Zhuang, W., and Cottrell, G. W. Joint-label learning by dual augmentation for time series classification. In *AAAI*, pp. 8847–8855, 2021.

Moltisanti, D., Fidler, S., and Damen, D. Action recognition from single timestamp supervision in untrimmed videos. In *CVPR*, pp. 9915–9924, 2019.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? In *NeurIPS*, pp. 4694–4703, 2019.

Neufeld, A. and Sester, J. A deep learning approach to data-driven model-free pricing and to martingale optimal transport. *IEEE Transactions on Information Theory*, 2022.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Park, J. Y. and Whang, Y.-J. A test of the martingale hypothesis. *Studies in Nonlinear Dynamics & Econometrics*, 9 (2), 2005.

Pham, H., Dai, Z., Xie, Q., and Le, Q. V. Meta pseudo labels. In *CVPR*, pp. 11557–11568, 2021.

Rizve, M. N., Duarte, K., Rawat, Y. S., and Shah, M. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. In *ICLR*, 2021.

Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A., et al. Collecting complex activity datasets in highly rich networked sensor environments. In *INSS*, pp. 233–240, 2010.

Shin, Y., Yoon, S., Kim, S., Song, H., Lee, J.-G., and Lee, B. S. Coherence-based label propagation over time series for accelerated active learning. In *ICLR*, 2022.

Shorten, C. and Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

Singh, A., Chakraborty, O., Varshney, A., Panda, R., Feris, R., Saenko, K., and Das, A. Semi-supervised action recognition with temporal contrastive learning. In *CVPR*, pp. 10389–10399, 2021.

Singhania, D., Rahaman, R., and Yao, A. Iterative contrast-classify for semi-supervised temporal action segmentation. In *AAAI*, pp. 2262–2270, 2022.

Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., Cubuk, E. D., Kurakin, A., and Li, C.-L. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, pp. 596–608, 2020.

Stein, S. and McKenna, S. J. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *UbiComp*, pp. 729–738, 2013.

Um, T. T., Pfister, F. M., Pichler, D., Endo, S., Lang, M., Hirche, S., Fietzek, U., and Kulić, D. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *ICMI*, pp. 216–220, 2017.

Van Engelen, J. E. and Hoos, H. H. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

Wang, X., Fan, H., Tian, Y., Kihara, D., and Chen, X. On the importance of asymmetry for siamese representation learning. In *CVPR*, pp. 16570–16579, 2022.

Wen, Q., Sun, L., Yang, F., Song, X., Gao, J., Wang, X., and Xu, H. Time series data augmentation for deep learning: A survey. In *IJCAI*, pp. 4653–4660, 2021.

Xiao, J., Jing, L., Zhang, L., He, J., She, Q., Zhou, Z., Yuille, A., and Li, Y. Learning from temporal gradient for semi-supervised action recognition. In *CVPR*, pp. 3252–3262, 2022.

Xie, Q., Dai, Z., Hovy, E., Luong, T., and Le, Q. Unsupervised data augmentation for consistency training. In *NeurIPS*, pp. 6256–6268, 2020.

Xu, Y., Wei, F., Sun, X., Yang, C., Shen, Y., Dai, B., Zhou, B., and Lin, S. Cross-model pseudo-labeling for semi-supervised action recognition. In *CVPR*, pp. 2959–2968, 2022.

Yang, X., Song, Z., King, I., and Xu, Z. A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–20, 2022.

Yoon, S., Shin, Y., Lee, J.-G., and Lee, B. S. Multiple dynamic outlier-detection from a data stream by exploiting duality of data and queries. In *SIGMOD*, pp. 2063–2075, 2021.

Yoon, S., Meng, Y., Lee, D., and Han, J. SCStory: Self-supervised and continual online story discovery. In *TheWebConf*, pp. 1853–1864, 2023.

Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., and Xu, B. TS2Vec: Towards universal representation of time series. In *AAAI*, pp. 8980–8987, 2022.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. CutMix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, pp. 6023–6032, 2019.

Zhang, B., Wang, Y., Hou, W., Wu, H., Wang, J., Okumura, M., and Shinozaki, T. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. In *NeurIPS*, pp. 18408–18419, 2021.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

A Proof of Theorem 3.2

We prove Theorem 3.2 in Section 3.2. Let $\mathcal{A}^i(\dot{X})$ be the context-attached augmented instances from Definition 3.1. A feature extractor $f_\theta(\cdot)_t$ generates an embedding for a timestamp t within the target instance, i.e., $\mathbf{z}^i = f_\theta(\mathcal{A}^i(\dot{X}))_t$, where $\mathbf{z}^i \in \mathbb{R}^k$ and k is the dimensionality of the embedding.

Comparing $\mathbf{z}^i$ and $\mathbf{z}^{i+1}$, their inputs share $2w+i-1$ timestamps and differ only at the timestamp $t = m + w + i - 1$. The data point at that timestamp would not deviate from the previous data points used to infer $\mathbf{z}^i$ due to temporal coherence. Thus, the next embedding would change in the vicinity centered at the current embedding (if we know it in advance), like a random walk, although the global distribution of embeddings is not known. This behavior leads to the *martingale* assumption in the embedding sequence, which is frequently used in time-series analysis (Ho, 2005; Hansen et al., 2019; Neufeld & Sester, 2022). In addition, the range of vicinity would also be limited because the change comes from only a single input timestamp.

To justify the martingale embedding assumption, let us conduct a *martingale hypothesis test* (Park & Whang, 2005) on the embedding sequence generated by a trained model of CrossMatch. The null hypothesis of the martingale hypothesis test for a specific dimension of $\mathbf{z}^i$ is

$$\mathrm{H}_0 : \Pr(\mathbb{E}[z^{i+1}|z^i] = z^i) = 1 \text{ where } z^i \in \mathbb{R}. \tag{9}$$

A Kolmogorov-Smirnov type statistic is given by

$$KS(z) = \sup_{\delta \in \mathbb{R}} \left| \frac{1}{\sqrt{c+1}} \sum_{i=0}^{c} (z^{i+1} - z^i) \mathbf{1}_{z^i \leq \delta} \right|. \tag{10}$$

The maximum value of $KS(z)$ from every dimension of the softmax probabilities or penultimate layer output is 1.36, 1.45, and 1.43 for the HAPT, mHealth, and Opportunity datasets, respectively, when δ is searched from -10 to 10 with a step size of 0.001. With the significance level set to be 0.05, the critical value of $KS(z)$ is 2.338 (Park & Whang, 2005), which is much higher than these values that we computed. Therefore, we do *not* reject the null hypothesis for every dimension of the embedding sequence, meaning that the embedding vector sequence is martingale.

When the embedding (vector) sequence is martingale and its consecutive differences are bounded, for a pair of consecutive embeddings $\mathbf{z}^i$ and $\mathbf{z}^{i+1}$, the following properties hold by definition: (1) $\mathbb{E}[\mathbf{z}^{i+1}|\mathbf{z}^i] = \mathbf{z}^i$, and (2) $\|\mathbf{z}^{i+1} - \mathbf{z}^i\|_2 \leq s$, where s is a sufficiently small positive real constant. Then, the following variant of Azuma's inequality holds for $0 \leq j < i \leq c$, and see (Hayes, 2005) for the detailed derivation.

$$\Pr[\|\mathbf{z}^i - \mathbf{z}^j\|_2 \geq \epsilon] \leq 2e^2 e^{-\epsilon^2/2s^2(i-j)}. \tag{11}$$

Equation (11) says that the upperbound for the probability that the Euclidean distance between two embeddings $\mathbf{z}^i$ and $\mathbf{z}^j$ is larger than a positive real constant ϵ increases as the corresponding augmented instances $\mathcal{A}^i(\dot{X})$ and $\mathcal{A}^j(\dot{X})$ are more distant in terms of $(i-j)$.

B Overall Training Algorithm

Algorithm 1 describes how CrossMatch works in time-series consistency regularization. For each iteration, there are two steps for batch initialization: center timestamp sampling from $\mathcal{T}_L$ and $\mathcal{T}_U$ (Line 2) and context length sampling from a uniform distribution (Line 3). Note that we assign q as $w + c$ after the sampling (Line 4). From labeled center timestamps, instances are sliced from $\mathcal{X}$ with length $2q$ to construct a labeled batch, $\mathcal{X}_l$ and $\mathcal{Y}_l$, where $\mathcal{X}_{[m-q:m+q)} = \{\mathbf{x}_t \mid t \in [m-q : m+q)\}$ and $\mathcal{Y}_{[m-q:m+q)} = \{y_t \mid t \in [m-q : m+q)\}$ (Line 6). The classification loss for a labeled batch $\mathcal{L}_l$ is computed and averaged over the batch (Line 7). From unlabeled center timestamps, instances are sliced with context-attached augmentation that generates a pair of instances whose length is $w+q$ (Line 9). The target instance of each instance in an augmented unlabeled batch $\mathcal{X}_u$ is then pseudo-labeled using a confidence threshold τ (Line 10). CrossMatch softens the pseudo-labels with reliability weighting across two augmented instances with different contexts, transforming two pseudo-labels into a single cross-window label shared across the instances (Line 11). The classification loss with cross-window labels is computed for each pair of augmented instances and averaged over the batch (Line 12). Finally, the losses for labeled and unlabeled batches are then integrated into a single loss $\mathcal{L}$ with a hyperparameter λ, and the model f_θ is updated using its gradient (Line 13).

Algorithm 1 Time-series consistency regularization with CrossMatch

Input: A time series with initial labels $\mathcal{D}$, labeled timestamp set $\mathcal{T}_L$, unlabeled timestamp set $\mathcal{T}_U$, labeled batch size B_l, unlabeled batch size B_u, maximum number of iterations I, a model f_θ, confidence threshold τ, loss weight λ, learning rate η, half length of a target instance w, maximum context length $c_{\max}$.

Output: Final model f_θ.

for each iteration up to I **do**
  $\mathcal{T}_l \leftarrow$ Sample B_l timestamps from $\mathcal{T}_L$; $\mathcal{T}_u \leftarrow$ Sample B_u timestamps from $\mathcal{T}_U$;
  $c \leftarrow$ Sample a context length from Uniform$(2, c_{\max})$;
  $q \leftarrow w + c$;
  /** Loss computation for labeled batch **/
  $\mathcal{X}_l \leftarrow \{\mathcal{X}_{[m-q:m+q)}, m \in \mathcal{T}_l\}$; $\mathcal{Y}_l \leftarrow \{\mathcal{Y}_{[m-q:m+q)}, t \in \mathcal{T}_l\}$;
  $\mathcal{L}_l = \frac{1}{B_l} \sum_{X \in \mathcal{X}_l, Y \in \mathcal{Y}_l} \ell(X, Y)$; // See Equation (1)
  /** Loss computation for unlabeled batch **/
  $\mathcal{X}_u \leftarrow \{(\mathcal{X}_{[m-q:m+w)}, \mathcal{X}_{[m-w:m+q)}), m \in \mathcal{T}_u\}$; // See Section 3.2
  $\hat{\mathcal{Y}}_u \leftarrow$ PseudoLabeling$(\mathcal{X}_u, f_\theta, \tau)$; // See Equation (2)
  $\bar{\mathcal{Y}} \leftarrow$ ReliabilityWeighting$(\hat{\mathcal{Y}}_u)$; // See Section 3.3
  $\mathcal{L}_u = \frac{1}{B_u} \sum_{(X^{\text{left}}, X^{\text{right}}) \in \mathcal{X}_u, \bar{Y} \in \bar{\mathcal{Y}}} \ell_u(X^{\text{left}}, X^{\text{right}}, \bar{Y})$;
  $\mathcal{L} \leftarrow \mathcal{L}_l + \lambda \mathcal{L}_u$; $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$;
end for
return f_θ;

C Detailed Training Setup

As described in implementation details of Section 4.1, we use MS-TCN as the backbone sequential classifier (Farha & Gall, 2019). It can classify each data point in an instance X, generating sequential softmax probabilities at each timestamp. MS-TCN has four stages, and each stage is composed of eleven dilated convolution layers and a softmax output layer. The first stage takes a subsequence of the whole time series and outputs a softmax probability distribution at each timestamp. After the first stage, every stage is fed with softmax probabilities and then outputs another softmax probabilities. For all datasets, we use the same training hyperparameters and classifier as listed in Table 6. We set the labeled batch size as 4 and the unlabeled batch size as 8, use an SGD optimizer with the momentum and Nesterov method. The initial learning rate is 0.005 and is scheduled with a cosine decay function. In "Scheduling," the symbol i denotes a current iteration during training, and the symbol I denotes the total number of iterations.

Table 6. Training hyperparameters.

Stage	Layer	B_L	B_U	Optimizer	Momentum	Nesterov	η	Scheduling
4	11	4	8	SGD	0.9	True	0.005	$cos(\frac{7\pi i}{I})$

D Detailed Evaluation Metrics

Timestamp accuracy is the ratio of the number of timestamps with correctly predicted labels to the total number of timestamps in a times series, which is computed as

$$\text{TS accuracy} = \frac{1}{|\mathcal{T}_{\text{test}}|} \sum_{t \in \mathcal{T}_{\text{test}}} \mathbf{1}_{y_t = \hat{y}_t}.$$

Segmental F1 score is a performance measure for judging whether a classifier outputs *correct* and *coherent* labels for consecutive timestamps. First, *segmental precision* and *segmental recall* are computed by counting the number of correct matches between a predicted segment set $\hat{\mathbb{Y}}$ and the true segment set $\mathbb{Y}$ with a Jaccard similarity threshold (e.g., 0.25),

$$\text{Segmental Precision} = \frac{1}{|\hat{\mathbb{Y}}|} \sum_{\hat{Y} \in \hat{\mathbb{Y}}} \sum_{Y \in \mathbb{Y}} \mathbf{1}_{\text{Jaccard}(\hat{Y}, Y) > 0.25} \quad \text{and} \quad \text{Segmental Recall} = \frac{1}{|\mathbb{Y}|} \sum_{\hat{Y} \in \hat{\mathbb{Y}}} \sum_{Y \in \mathbb{Y}} \mathbf{1}_{\text{Jaccard}(\hat{Y}, Y) > 0.25}.$$

Then, *segmental F1 score* is computed as $F1@25 = \frac{2*\text{precision}*\text{recall}}{\text{precision}+\text{recall}}$.

At each iteration, we measure *pseudo-labeling precision* and *pseudo-labeling recall* from each timestamp of the target instances in a batch, which are computed as

$$\text{PL Precision} = \frac{\text{the number of correct PLs}}{\text{the number of PLs}} \quad \text{and} \quad \text{PL Recall} = \frac{\text{the number of correct PLs}}{\text{the number of timestamps}}.$$

To count the number of correct pseudo-labels, we examine whether a pseudo-label exists for each timestamp and, if so, compare it with the true label. When there is no pseudo-label at any timestamp, we simply define the pseudo-labeling precision as 0, because it is undefined otherwise due to division by zero. Likewise, *pseudo-labeling F1 score* is PLF $= \frac{2*\text{PL precision}*\text{PL recall}}{\text{PL precision}+\text{PL recall}}$. After a few iterations, the entire time series is expected to be used by pseudo-labeling the sampled target instances.

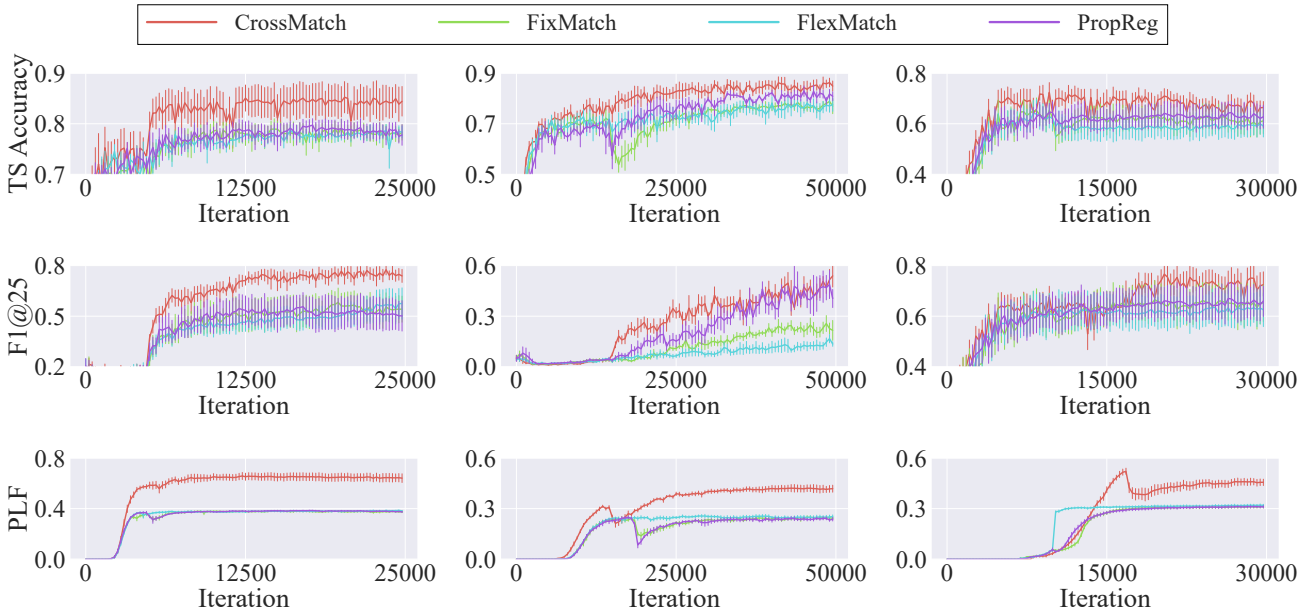
E Detailed Experiment Results

Figure 6 is a detailed version of Figure 5, with standard deviation error bars added.

Figure 7 is an expanded version of Figure 5, containing the entire y-axis for TS accuracy and F1@25.

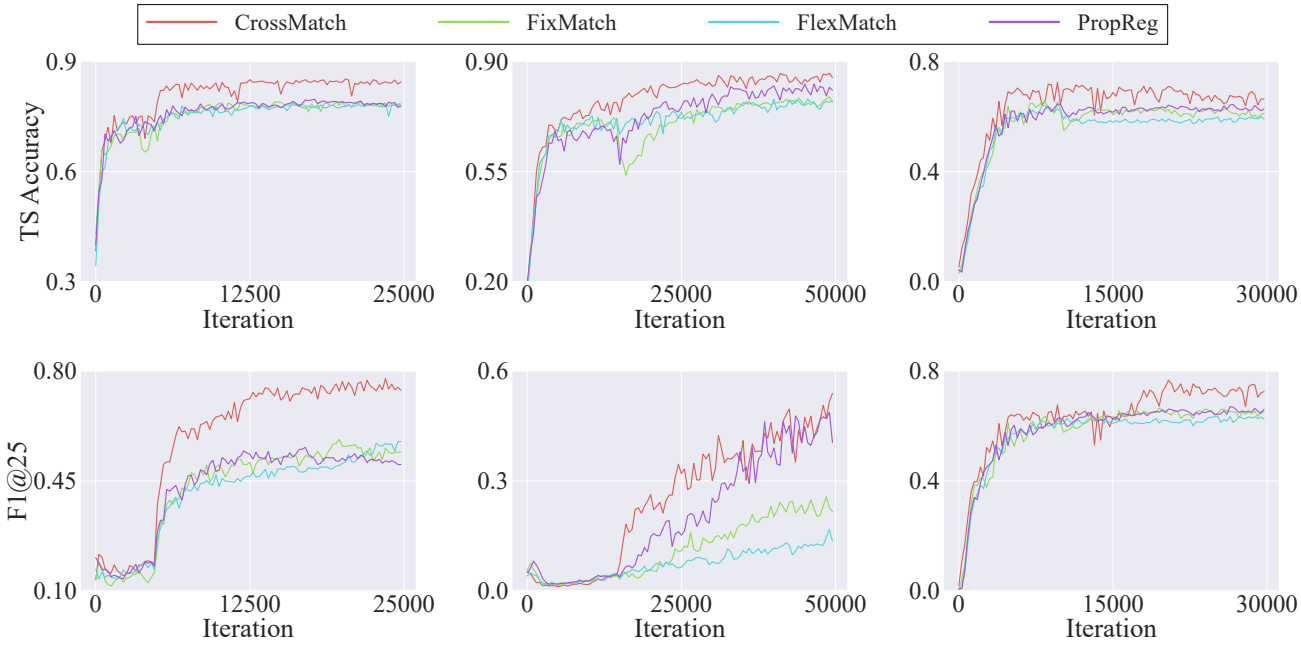
Figure 8 shows the pseudo-labeling metrics such as pseudo-labeling precision, pseudo-labeling recall, and the total number of pseudo-labels.

Figure 9 shows the accuracy convergence trend when the context length is varied as a factor or multiple of a given length c, illustrating the accuracy curves during the training iterations.



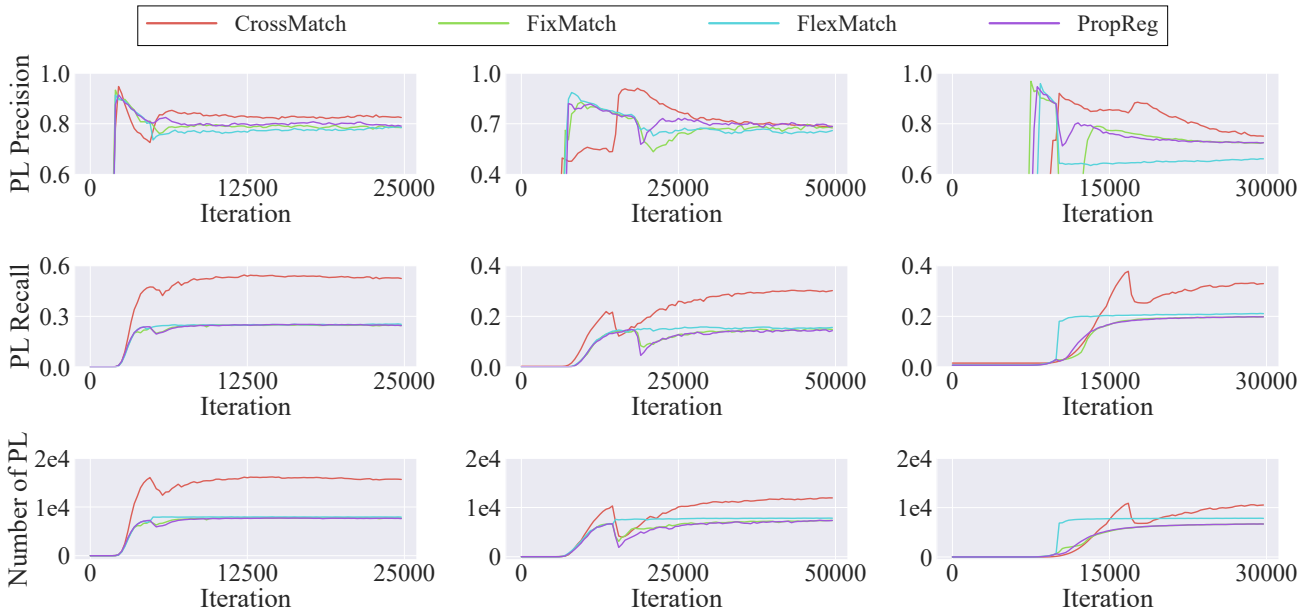
(a) HAPT. (b) mHealth. (c) Opportunity.

Figure 6. Training curve of the classification performance (the first two rows) and pseudo-labeling performance (the last row) over training iterations in Algorithm 1. Error bars represent the standard deviation of five runs.



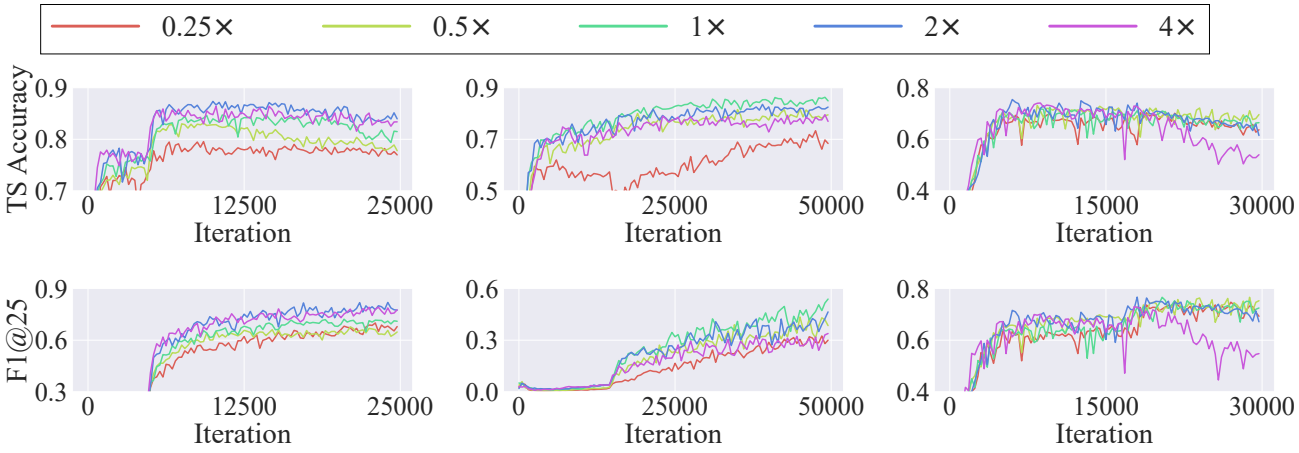
(a) HAPT. (b) mHealth. (c) Opportunity.

Figure 7. Training curve of the classification performance over training iterations in Algorithm 1, showing the entire range of each metric.



(a) HAPT. (b) mHealth. (c) Opportunity.

Figure 8. Pseudo-labeling precision, pseudo-labeling recall, and the number of pseudo-labels of CrossMatch compared with those of FixMatch, FlexMatch, and PropReg.



(a) HAPT. (b) mHealth. (c) Opportunity.

Figure 9. Classification accuracy of CrossMatch for varying the context length $x \times c$.