Cluster Distillation: Semi-supervised Time Series Classification through Clustering-based Self-supervision

Manuel Alejandro Goyo
Departamento de Informática
Universidad Técnica Federico Santa María
Santiago, Chile
manuel.goyo@sansano.usm.cl

Ricardo Ñanculef
Departamento de Informática
Universidad Técnica Federico Santa María
Santiago, Chile
jnancu@inf.utfsm.cl

***Abstract*—Time series have always raised great interest among scientists due to their multiple applications in real-world problems. In particular, time series classification using deep learning methods has recently attracted much attention and demonstrated remarkable performance. Unfortunately, most of the techniques studied so far assume that a fully-labeled dataset is available for training, a condition that limits the application of these methods in practice. In this paper, we present *Cluster Distillation*: a technique that leverages all the available data (labeled or unlabeled) for training a deep time series classifier. The method relies on a self-supervised mechanism that generates surrogate labels that guide learning when external supervisory signals are lacking. We create that mechanism by introducing clustering into a *Knowledge Distillation* framework in which a first neural net (the Teacher) transfers its beliefs about cluster memberships to a second neural net (the Student) which finally performs semi-supervised classification. Preliminary experiments in ten widely used datasets show that training a convolutional neural net (CNN) with the proposed technique leads to promising results, outperforming state-of-the-art methods in several relevant cases.**

***Index Terms*—Deep learning, CNN, self-supervision, time series classification**

I. Introduction

A time series is a discrete sequence of numerical values representing the chronological observation of a variable or process of interest. Scientists have widely studied time series due to their wide application in various areas such as economics, fintech, energy, and health studies, among many other cases [1]. As a result, time series analysis is a fundamental task in modern machine learning, and this interest has led to many specialized methods and problems. Among them, time series classification is the task of recognizing a time series as a member of a specific group or category that often reflects a particular motif or pattern of the process under consideration. For instance, in human activity recognition [2], the transitions of sensor measurements over time may allow a classifier to infer the actions (e.g., walking or laying) performed by one or more persons.

Recently, deep learning models have shown remarkable performance in time series classification [3]. This approach first maps time series into a latent space using a deep architecture, i.e., a composition of multiple transformations that disentangles the factors determining the sequence's shape or temporal motif. Then, a simple/traditional classifier infers the classes in the new feature space. Most effort in this area has focused on finding a good architecture to represent time series. Multi-layer perceptrons, recurrent models, convolutional neural networks, attention-based architectures, and even graph-based encoders have been investigated [4]. Unfortunately, these models rely heavily on the availability of a vast set of labeled data to guide the learning process. Except in very particular cases, obtaining fully-labelled datasets for real-world applications requires a significant effort from humans (often specialists), who must carry out the annotation task manually. This requirement significantly increases the resources and time needed to implement a real solution based on deep learning [5].

While annotating data is complex, unlabelled data is often available in vast volumes through effortless human assistance. In practice, a commonly used approach is discarding all the unlabelled data and training the architectures by traditional supervised learning on the annotated dataset. However, research in many fields shows that unlabelled data can provide small but statistically significant performance gains, which are relevant in many applications [6]. For instance, even slight improvements in power management can save substantial amounts of energy or significantly extend the durability of battery-based devices. Thus, semi-supervised methods that leverage labeled and unlabelled data are of great practical value [7].

A well-known principle used to design semi-supervised classifiers is the *Cluster assumption* which presupposes (i) that the data forms clusters in some feature space and (ii) that these clusters have homogeneous labels. Methods based on this assumption use all data to identify the clusters [7] and then propagate the available labels among the clusters. A different approach to building semi-supervised methods combines traditional supervision with self-supervision, i.e., objectives, often based on a secondary pretext task, that supply alternative supervisory signals to guide the learning process [8]–[10]. For instance, predicting the word missing in a text segment helps a sentiment classifier to learn the language structure without explicit annotations. Likewise, the complementary task of sorting patches of the input image could allow an object detector to learn from labeled and unlabelled images. Finally, another popular approach to semi-supervised learning is *Knowledge Distillation*. This technique first trains a model called the *Teacher* using the labeled data. Then, a second model, referred to as the *Student*, is trained to exploit (e.g., to predict) the Teacher's predictions on the unlabelled data.

This paper presents *Cluster Distillation*, a method that combines clustering and Knowledge Distillation to create a self-supervised mechanism that allows a deep time series classifier to learn from labeled and unlabelled data. The procedure works as follows. First, we train a Teacher model using the labeled data. Second, we use the Teacher model to obtain feature representations of all the available data and identify clusters in the Teacher's feature space. Finally, we train a Student model to fit two different objectives simultaneously: (i) to predict the class of the labeled data and (ii) to predict the cluster memberships of the unlabeled instances. The last objective constrains the search for a suitable classifier to those functions that match the modes of the data manifold that the Teacher approximates in the first step. Stated differently, during the training process, the Student can ask the Teacher for pseudo-labels [11], surrogate objectives that guide the learning process in the lack of external annotations. The main contributions of this paper are:

- To our knowledge, we are the first to incorporate clustering into a Knowledge Distillation framework. In this process, we relax the classic Cluster assumption because clusters are not directly used to propagate labels but to regularize the Student's objective function.
- To our knowledge, we are the first to evaluate cluster-based self-supervision in semi-supervised time series classification.

The rest of this paper organizes as follows. Section II reviews related work on self-supervision, semi-supervised learning, and Knowledge Distillation. Our method is motivated and described in Section III. Section IV compares the proposed method with two baseline approaches on ten public datasets. Finally, Section V states the conclusions of our research and some final remarks.

II. Related Work

A. Self-supervision and Clustering

Self-supervised learning techniques train a model using a so-called *pretext task*, a learning objective in which available properties or attributes of the data act as auxiliary targets that improve the learning performance in a primary or downstream task. These self-generated surrogate targets are known as *pseudo-labels*. Well-known examples are *Image Inpainting*, in which the pretext task is restoring missing regions of an input image, and *Jigsaw*, which trains the model to solve puzzles created from the image [12].

Up to our knowledge, the first work to exploit clustering as a pretext task is [13] which introduced the idea with the aim of pre-training data-hungry computer vision models. At each training epoch, the method optimizes the parameters of a convolutional neural network (CNN) and the cluster prototypes of the k-means algorithm. Experiments on object detection and image classification show that this approach leads to better image representations in domains where annotations are scarce. More recently, [14] proposed *Clusterfit*, another clustering-based self-supervised method that performs clustering only once using a CNN pre-trained in a supervised fashion. Finally, this method fits a different neural net to learn the cluster assignments. Although closely related to our method, Clusterfit seeks to improve generalization in transfer learning tasks with a large gap between the source and target domains. However, the method is not studied in semi-supervised learning scenarios and does not include a double objective in the second phase.

B. Self-supervision for Semi-Supervised Learning

In the vein of *Image Inpainting*, [10] exploits the temporal dynamics underlying time series to create a pretext task in which the model has to forecast the next steps of an incomplete sequence. Semi-supervised classification is achieved by optimizing a CNN to simultaneously solve the pretext task and the main task of predicting the categories of the labeled time series. More recently, [8] adapts *Jigsaw* [12] to the time series domain. This method divides the unlabeled sequences into two segments characterizing the " past" and " future" of each time series. By sampling these segments, one can obtain pseudo-labels: *positive examples* are those in which " past" and " future" match, and *negative examples* are those in which the segments belong to different time series. For semi-supervised learning, a CNN optimizes two different objectives: for labeled data, the net has to predict the category of the time series, and for unlabelled data, the model has to predict the pseudo-label.

STCN [9] is a time series clustering method that exploits labeled data to optimize feature extraction and clustering simultaneously. In the first stage, it uses all the data to train an Echo State Network that has to forecast one step of the input time series. In the second phase, it uses the features obtained during the first stage to perform clustering. In contrast to an ordinary clustering algorithm, it constrains the cluster assignments to align well with the classes of the labeled data. Although related to our approach, we must note that this method is devised and validated for semi-supervised clustering and not semi-supervised classification.

C. Knowledge Distillation

Knowledge Distillation (KD) is a deep learning framework relying on two neural networks, one commonly called the *Teacher* and one called the *Student*. The original method [15] first trains the Teacher as usual. Then, it trains the Student for the task of interest but adds a regularizer to its objective function such that the Student's predictions imitate the Teacher's responses. If the Teacher is a cumbersome (perhaps deep) model that generalizes well, KD can transfer this generalization ability to a smaller and more efficient Student. There are many extensions and applications of this idea. For instance, [16] proposed *Relational Knowledge Distillation*. This technique enforces the feature representations learned by the Student to be consistent with the internal representations learned by the Teacher. To this end, it computes the distances and angles among triplets of training points and adds that loss to the Student's objective function. [17] presented a probabilistic variant of KD that transfers knowledge from the Teacher to the Student by matching probability distributions computed on pairs of training examples.

Recently, [18] proposed an adaptation of KD for semi-supervised learning. They first use supervised learning to train the Teacher on the labeled data. Then, it trains the Student to predict the actual labels on the labeled data but also to imitate the Teacher's predictions for the unlabelled data. It is worth noting that the model uses the Teacher's probability distribution on the classes as pseudo-labels, which allows the Student to deal with a smoother optimization landscape.

III. Proposed Method

This section presents *Cluster Distillation* in detail. Consider a labelled dataset $D_l = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m))\}$, where each $x_i \in X$ is a real-valued time series of length T, i.e., $x_i = [x_{i,1}, x_{i,2}, \cdots, x_{i,T}]$ with $x_{i,j} \in \mathbb{R}$, and $y_i \in Y \equiv [0,1]^K$ is a one-hot vector encoding the class of x_i. Furthermore, consider an unlabelled dataset $D_u = \{\tilde{x}_1, \tilde{x}_2, \dots \tilde{x}_n\}$ containing other n time series $\tilde{x}_i = [\tilde{x}_{i,1}, \tilde{x}_{i,2}, \cdots, \tilde{x}_{i,T}]$, produced by the same input distribution $p(x)$ on X, but for which we do not have any label. In this context, we want to train an algorithm that can correctly classify novel time series $x \sim p(x)$ exploiting both D_u and D_l.

A. General Idea

In a nutshell, our method is an extension of Knowledge Distillation for semi-supervised time series classification. As in previous approaches to this task, we first train the Teacher using supervised learning on D_l and then train the Student on $D_l \cup D_u$ to optimize two goals simultaneously: (i) to fit the labels in D_l and (ii) imitate the Teacher's responses on D_u. However, the novelty of our approach is in the way of measuring the latter goal. Instead of focusing on the Teacher's responses, we focus on the Teacher's beliefs about the input data. More formally, if $p_t(y|x)$ ($p_s(y|x)$) denotes the conditional distribution learned by the Teacher (Student) on the label set Y, and $p_t(x)$ ($p_s(x)$) denotes the marginal distribution learned by the Teacher (Student) on the input space X, we do not constrain $p_s(y|x)$ to be close to $p_t(y|x)$, but $p_s(x)$ to be close to $p_t(x)$. To achieve this goal, we represent $p_t(x)$ using a clustering algorithm Ψ on the Teacher's feature space and then constrain the Student to approximate the cluster assignments made by the Teacher. More specifically, we equip the Student with the following regularization term:

$$L(f_t(x), f_s(x)) = \sum_{x \in D_u} \ell(\Psi_t(f_t(x)), \Psi_s(f_s(x))) \quad (1)$$

where f_t, f_s denote Teacher's and Student's internal representations, respectively, Ψ_t is a function determining the cluster assignments made by the clustering algorithm Ψ for a point $f_t(x)$, Ψ_s is a function capable of categorizing each the features delivered by the Student network, and $\ell : Y \times Y \rightarrow \mathbb{R}$ is an appropriate loss function (e.g. cross-entropy).

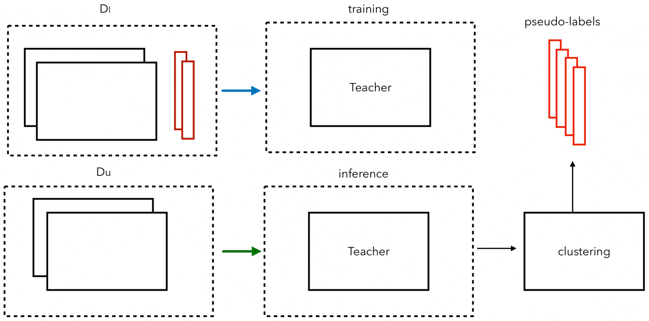

Fig. 1. Teacher training and pseudo-label generation. At the top, we see that the Teacher is trained using only the labeled data. At the bottom, we see how pseudo-labels are generated for unlabeled data.

Below we show how to train the Teacher and the Student in more detail.

B. Teacher Training

Our approach first trains a Teacher network that will transmit knowledge about the data distribution to a second (Student) network by learning a feature (vector) representation of the time series. As illustrated in Fig. 1 (top), we train this net using the fully-labeled dataset D_l. To transfer the feature representation, we need to identify two main blocks: a convolutional block $f_s : X \rightarrow \mathbb{R}^S$, and a fully-connected block $g_s : \mathbb{R}^S \rightarrow \mathbb{R}^K$. The convolutional block extracts high-level features from raw time series data. The fully-connected bloc predicts the class of an input time series using the attribute representation $f_s(x)$. If we denote by θ the trainable parameters of f_s and by γ the corresponding parameters of g_s, the Teacher network $f_s \circ g_s$ optimize the following objective function:

$$\min_{\theta, \gamma} \frac{1}{m} \sum_{x_i \in D_l} \ell(y_i, g_t(f_t(x_i))) \quad (2)$$

where ℓ is the cross-entropy loss function. In this way, we perform traditional supervised training with the data available with labeling. Note that the convolutional block learns an embedding that represents each time series in a low-dimensional feature space, in this case in $S-$dimensional space.

C. Clustering The Latent Space

Before training the Student network, we need a way to summarize or represent the knowledge about the input space captured by the Teacher. As shown in Fig. 1 (bottom), we propose to address this task by clustering the images of all the time series under the Teacher's latent space. That is, we construct the dataset $f_t(D) = \{f_t(x_i) : x_i \in D_l \cup D_u\}$ and apply a clustering algorithm on $f_t(D)$.

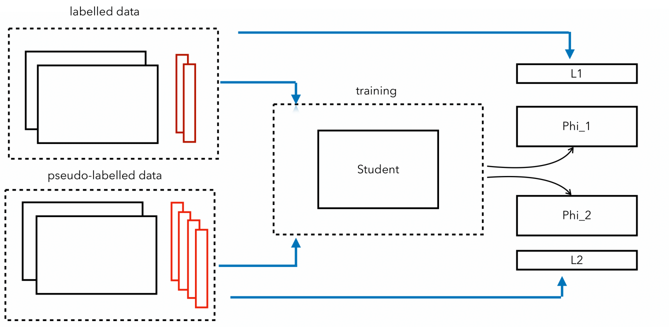

Fig. 2. The Student network takes advantage of the Teacher's knowledge about the data distribution to guide learning on data for which we do not have labels.

- **Hard Assigments.** For simplicity, the first variant of the method we consider in this paper relies on k-means clustering [19] with $k = 2K$, i.e., we find twice the number of concepts in the original label set Y. The motivation for this decision is that we need a characterization of the Teacher's feature space $f_t(X) = \{f_t(x) : x \in X\}$ which is sufficiently detailed but also well correlated with the true target concepts in Y. To represent the assignments of each possible data point x to the clusters found by k-means, we can define a function $\Psi_t : X \rightarrow [0,1]^{2K}$ such that the i-th component of $\Psi_t(x)$ is 1 if $f_t(x)$ is assigned to the cluster i and 0 otherwise.
- **Soft Assigments.** The second variant of our method relies on a Mixture of Gaussians (MoE) [20] with $k = 2K$ components. The motivation for this decision is to keep the simplicity of the clustering algorithm but allow smooth assignments. Some data points may lie on the boundary between two latent concepts, and probabilistic assignments, such as those found by MoE, also distill that uncertainty to the Student. Note that the the i-th component of the cluster assignment function $\Psi_t : X \rightarrow [0,1]^{2K}$ computes now the probability that x is assigned to the cluster i in the Teacher's latent space.

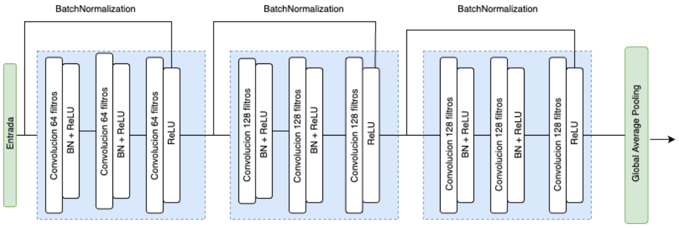

Fig. 3. Diagram of the Residual Convolutional Network architecture.

D. Student Training

After obtaining clusters to summarize the distribution of the Teacher's feature space, we want to distill that knowledge to the Student network and, simultaneously, learn to classify the time series into the original label set Y. The architecture of the Student network is similar to that of the Teacher network: it has a convolutional block f_s followed by two fully-connected blocks g_s and Ψ_s. The convolutional block $f_s : X \rightarrow \mathbb{R}^S$ extracts features from the time series just as before. However, we have two fully-connected blocks this time because the Student has two different objectives. Block $g_s : \mathbb{R}^S \rightarrow \mathbb{R}^K$ serves to predict a category in the original label set Y. Block $\Psi_s : \mathbb{R}^S \rightarrow \mathbb{R}^{2K}$, in contrast, serves to predict the cluster assignments made by the clustering algorithm in the Teacher's latent space. As shown in Fig. 2, we train the Student network using all the available data $\mathcal{D} = D_u \cup D_l$. If a time series has a label, that label is used to measure the Student's prediction error, as usual. However, if a time series x_i does not have a label, the pseudo-label $p_i = \Psi_t(x_i)$ will be used to guide learning (soft or hard version, depending on the variant). More specifically, the Student's loss function is the linear combination of two different losses:

$$L = \alpha L_1 + (1-\alpha) L_2 \tag{3}$$

where

$$L_1 = \sum_{x_i \in D_l} \ell(y_i, g_s(f_s(x_i))) \, , \tag{4}$$

$$L_2 = \sum_{x_i \in D_u} \ell(\Phi_t(f_t(x_i)), \Phi_s(f_s(x_i))) \ , \tag{5}$$

ℓ is the cross-entropy loss, and α is a parameter that weights the proportion of unlabeled data in the entire dataset. Therefore, the Student's objective is double. First, to minimize the divergence of its predictions about $p(y|x)$ from the ground-truth data about $p(y|x)$. Second, to minimize the divergence of its beliefs about $p(x)$ from the Teacher's beliefs about $p(x)$.

E. Arquitectures

As our model deals with time series data, it is essential to highlight the architecture used to implement the neural nets.

- **Convolutional Block**: inspired by Self-distillation [21] we implement the mappings f_s and f_t as twin networks with independent parameters. We adopt the Residual architecture proposed in [3]. It has three residual blocks followed by a global average pooling [22]. Each residual block is composed of three convolutional layers with a skip connection, i.e., the output of the block is the sum of the original input with the result of the three convolutions. These residual blocks allow gradients to flow easily from the output to the input layers. The number of filters for all the convolutions is 64 for the first block and 128 for the last two. The activation function is always ReLU [23], which is preceded by a BatchNormalization layer [24]. At each block, the filter size for the convolutions is $8, 5$, and 3, respectively. In Fig. 3, we summarize the convolutional architecture.
- **Dense blocks**: to implement the fully-connected blocks, we used a single layer with the required number of output neurons (K or $2K$, see above) and a softmax activation function.

IV. Experiments

This section presents the results of experiments conducted to evaluate the performance of our model in scenarios of label scarcity and compare it to the current state of the art.

A. Datasets & Implementation Details

We used ten public datasets, widely used in previous studies, that are part of the UCR time-series repository [25]. Table I summarizes the names and attributes of each dataset. *DistalPhal* is an abbreviation of *DistalPhalanxOutlineAgeGroup*.

We performed the experiments on a computer with an Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz, 32GB of RAM, and a GeForce GTX 1080 Ti GPU. The clustering algorithm for our method's first (hard) variant is the k-means algorithm implemented in scikit-learn [26]. As for the second variant (soft pseudo-labels), the Gaussian Mixture algorithm of scikit-learn [26] was employed.

All the code required to reproduce the experiments has been made available at Cluster Distillation.

B. Methods & Experimental Setup

We compare our method with the semi-supervised approach discussed in section II and recently proposed in [18]. In addition, we consider a fully-supervised baseline to evaluate the benefits of using unlabelled data for learning. We must note that the latter approach combined with the residual architecture described in Section III is the method presented in [3] as state of the art after comparing nine deep learning algorithms on 85 time series classification datasets. In all the tables of this section, we refer to the semi-supervised baseline as *SemiSup-Distillation* and the fully-supervised as *Sup-ResNet*.

To assess the algorithms' robustness in label scarcity, we train and evaluate the models at varying levels of supervision $\rho = n/(n+m)$, the ratio of labeled examples in the training set. Starting from $\rho = 0.5$ (50% labeled examples), we stress the algorithms reducing ρ till a 9% of supervision. The latter is the minimum we can use to guarantee that the training set of the smallest datasets includes at least one example per class.

C. Results

Tables II, III, IV, V, VI, VII, VIII, IX, X, XI present the accuracy of the different methods in each of the ten datasets. To obtain these values, we ran each of the algorithms ten times with different random seeds and reported the mean and standard deviation of the scores. We want to note that precision and recall were also measured and were always well correlated with accuracy. We do not present these results for space constraints. In each table, the columns 50%, 25%, 12.5%, 9% represent different levels of supervision ρ as defined above.

The results of the different algorithms were analyzed using the Friedman test to assess the (null) hypothesis that they were statistically equivalent, considering the variance throughout the different repetitions of an experiment. In cases where the Friedman test showed significant differences between the groups at the 5% level, we applied a Nemenyi post hoc test to assess the differences between all pairs of methods. A p-value of 5% or less was considered statistically significant. We colored our method's results for easier comparison using the following criteria. The color green indicates that our method's performance is found to be statistically superior to both *Sup-ResNet*'s and *SemiSup-Distillation*'s performance. The color blue indicates our method's accuracy is statistically higher than that of *Sup-ResNet*. The color brown indicates that this value is statistically lower than the accuracy achieved by *Sup-ResNet*. Finally, the color red indicates that our method's performance is statistically lower than *Sup-ResNet* and *SemiSup-Distillation* simultaneously. The absence of color means that the statistical tests revealed no significant differences between the methods.

Dataset Name	Training Set Size	Test Set Size	Series Length	Number of Classes
ArrowHead	36	175	251	3
Car	60	60	577	4
Coffee	28	28	286	2
DistalPhal	400	139	80	3
ECG200	100	100	96	2
Ham	109	105	431	2
Herring	64	64	512	2
Meat	60	60	448	3
Trace	100	100	275	4
Wine	57	54	234	2

TABLE I
Datasets used in the experiments and their properties.

The table II summarizes the results obtained on the *Arrow Head* dataset. We can see that, for all levels of supervision studied, the proposed methods outperform the *Sup-ResNet* method, which cannot take advantage of unlabeled data. In addition, we can see that, in most cases, our methods also outperform the semi-supervised baseline [18]. The more salient cases are those in which the percentage of labeled data is small (12.5%, 9%). In these scenarios, the proposed methods obtain margins of around 10 and 7 points of accuracy over the *Sup-ResNet* and *SemiSup-Distillation*, respectively, an advantage that the statistical tests reveal to be statistically significant in both cases. This pattern seems to be confirmed in the tables III and IV, which compare the performance of the different methods on the datasets *Car* and *Coffee*, respectively. The methods obtain similar results in these datasets when labeled data is abundant. However, when labels become more scarce (columns 12.5% and 9%), the two variants of the proposed method outperform the others by a significant margin, with advantages ranging from around 8 to 12 points of accuracy. Indeed, the statistical tests reveal significant differences in these cases, both when comparing against *Sup-ResNet* and when comparing against the semi-supervised baseline.

The results corresponding to the dataset *Distal Phalanx (Dist.Phal)* are summarized in Table V. In this case, we can see that the proposed method obtains higher accuracies than the Sup-ResNet method in all cases, except in the scenario with the lowest availability of labels (column 9%). In this case, the statistical tests reveal significant differences between the proposed method and both baselines. When we compare the proposed method with the semi-supervised method, we observe similar results in most cases, except in the previously

	Method	50%	25%	12.5 %	9%
ArrowHead	Sup-ResNet [3]	0.7131 (0.0249)	0.6382 (0.0395)	0.5782 (0.0435)	0.4051 (0.0363)
	SemiSup-Distillation [18]	0.7771 (0.0276)	0.6548 (0.0339)	0.5891 (0.0204)	0.4331 (0.0197)
	Proposed - Hard Variant	0.7617 (0.017)	0.6605 (0.0265)	0.6188 (0.0295)	0.5062 (0.0379)
	Proposed - Soft Variant	0.768 (0.0179)	0.6771 (0.0295)	0.6148 (0.039)	0.5005 (0.0435)

TABLE II

ARROWHEAD AVERAGE ACCURACY TABLE.

	Method	50%	25%	12.5 %	9%
Car	Sup-ResNet [3]	0.9 (0.0351)	0.735 (0.0308)	0.6366 (0.0281)	0.33 (0.0219)
	SemiSup-Distillation [18]	0.9066 (0.0195)	0.7366 (0.0233)	0.5783 (0.0272)	0.3566 (0.0409)
	Proposed - Hard Variant	0.905 (0.0193)	0.7433 (0.014)	0.7116 (0.0437)	0.4533 (0.0489)
	Proposed - Soft Variant	0.91 (0.0224)	0.7416 (0.018)	0.7033 (0.0291)	0.4583 (0.0263)

TABLE III

CAR AVERAGE ACCURACY TABLE

	Method	50%	25%	12.5 %	9%
Coffee	Sup-ResNet [3]	1 (0)	0.9607 (0.0459)	0.7964 (0.0338)	0.8607 (0.0393)
	SemiSup-Distillation [18]	1 (0)	0.9892 (0.0172)	0.85 (0.0225)	0.8607 (0.0263)
	Proposed - Hard Variant	0.9821 (0.0252)	1 (0)	0.8785 (0.0184)	0.9428 (0.0698)
	Proposed - Soft Variant	0.9821 (0.0252)	1 (0)	0.8785 (0.0184)	0.9428 (0.0698)

TABLE IV

COFFEE AVERAGE ACCURACY TABLE

	Method	50%	25%	12.5 %	9%
Dist.Phal	Sup-ResNet [3]	0.7172 (0.0244)	0.69 (0.0127)	0.6676 (0.0172)	0.6848 (0.0213)
	SemiSup-Distillation [18]	0.7402 (0.0141)	0.7553 (0.0107)	0.6812 (0.0059)	0.705 (0.0126)
	Proposed - Hard Variant	0.7438 (0.0176)	0.7273 (0.0141)	0.6733 (0.0136)	0.6597 (0.014)
	Proposed - Soft Variant	0.733 (0.0213)	0.7258 (0.0193)	0.6899 (0.0204)	0.651 (0.0128)

TABLE V

DISTALPHALANXOUTLINEAGEGROUP AVERAGE ACCURACY TABLE

	Method	50%	25%	12.5 %	9%
ECG200	Sup-ResNet [3]	0.88 (0.0253)	0.795 (0.0347)	0.741 (0.0128)	0.742 (0.0175)
	SemiSup-Distillation [18]	0.885 (0.0279)	0.804 (0.0107)	0.75 (0.0081)	0.75 (0.0081)
	Proposed - Hard Variant	0.866 (0.0171)	0.801 (0.0166)	0.738 (0.0063)	0.734 (0.0150)
	Proposed - Soft Variant	0.871 (0.026)	0.8 (0.0133)	0.741 (0.0073)	0.734 (0.0150)

TABLE VI

ECG200 AVERAGE ACCURACY TABLE

mentioned scenario. Although the differences fluctuate in a tight interval of around 3 and 5 points of accuracy, this advantage of the baselines contradicts the previously observed pattern. Certainly, this result could be due to particular attributes of the dataset. Indeed, contrary to what we can expect, the baselines improve when we reduce the amount of labeled data from 12.5% to 9%. Alternatively, we may attribute this result to the training set size, which is the largest of all the cases considered in this work. Given greater data availability, a supervision level of 9% could generate enough ground-truth labels to give the baselines an advantage over our technique, based on artificial, possibly noisy pseudo-labels. Finally, it is worth mentioning that in this dataset, the *hard* variant of the proposed method exhibits a slight advantage over the *soft* version. Unlike the results in Table II, III and IV, the *soft* version degrades more strongly in one case and improves less substantially in another.

With less clarity and significance, we observe the situation described for *Distal Phalanx* also in the datasets *ECG200* and *Ham*. In the case of *ECG200*, the Table VI shows that the proposed method exhibits, in some cases, a small nominal disadvantage compared to the others (between ≈ 0.3 and ≈ 1.5 points of accuracy). We must note, however, that, in addition to being very small, these differences are statistically significant only in one scenario (column 12.5%) and only for one of the two variants studied in this paper (the *hard* version). For *Ham*, the Table VII shows that no method consistently outperforms the others. In one case, *Sup-ResNet* outperforms both variants of the proposed method with statistically significant margins. Both baselines outperform the *soft* variant in another case. As in *Distal Phalanx*, we observe that these situations occur in scenarios with a greater scarcity of labels. However, we should also note that, as in *Distal Phalanx*, the fully-supervised method *Sup-ResNet* improves its performance when we reduce the groud-truth labels from 25% to 12.5%. The latter is a completely unexpected result that suggests a quite particular composition of the training set. As in *Distal Phalanx*, the training sets in *ECG200* and *Ham* are also larger than those corresponding to the Tables II, III and IV. The *Trace* dataset is also small. However, as shown in the X table, the results of the different methods are quite similar, with the vast majority of values close to perfect accuracy 1. In only one case, the statistical tests find a significant difference (column 25%) disfavoring the *soft* variant.

The Tables VIII, IX and XI present the average accuracy of the different methods on the datasets *Herring*, *Meat* and emphWine, respectively. These results align better with those reported in the Tables II, III and IV. The proposed method ob-

	Method	50%	25%	12.5 %	9%
Ham	Sup-ResNet [3]	0.7247 (0.0343)	0.6733 (0.0399)	0.6942 (0.0292)	0.6552 (0.02409)
	SemiSup-Distillation [18]	0.7257 (0.0219)	0.7314 (0.0272)	0.6438 (0.0206)	0.6895 (0.025)
	Proposed - Hard Variant	0.7466 (0.0201)	0.7257 (0.0223)	0.62 (0.0192)	0.66 (0.66)
	Proposed - Soft Variant	0.7476 (0.0262)	0.7257 (0.0223)	0.6304 (0.0272)	0.6514 (0.0229)

TABLE VII
HAM AVERAGE ACCURACY TABLE

	Method	50%	25%	12.5 %	9%
Herring	Sup-ResNet [3]	0.6093 (0.0448)	0.5062 (0.0362)	0.5125 (0.047)	0.4687 (0.0429)
	SemiSup-Distillation [18]	0.6312 (0.0461)	0.5187 (0.0218)	0.5578 (0.0361)	0.4265 (0.0255)
	Proposed - Hard Variant	0.6046 (0.0255)	0.5593 (0.0205)	0.564 (0.0214)	0.475 (0.0246)
	Proposed - Soft Variant	0.6312 (0.0235)	0.539 (0.0355)	0.5812 (0.0273)	0.4625 (0.4625)

TABLE VIII
HERRING AVERAGE ACCURACY TABLE

	Method	50%	25%	12.5 %	9%
Meat	Sup-ResNet [3]	0.8766 (0.16)	0.7483 (0.1117)	0.6833 (0.0293)	0.64 (0.025)
	SemiSup-Distillation [18]	0.925 (0.037)	0.78 (0.1563)	0.5916 (0.139)	0.6033 (0.1044)
	Proposed - Hard Variant	0.905 (0.0824)	0.83 (0.0443)	0.6916 (0.0453)	0.7316 (0.0595)
	Proposed - Soft Variant	0.8733 (0.1849)	0.84 (0.0417)	0.6883 (0.0465)	0.75 (0.0458)

TABLE IX
MEAT AVERAGE ACCURACY TABLE

	Method	50%	25%	12.5 %	9%
Trace	Sup-ResNet [3]	1 (0)	1 (0)	0.999 (0.0031)	1 (0)
	SemiSup-Distillation [18]	1 (0)	0.993 (0.0221)	1 (0)	1 (0)
	Proposed - Hard Variant	1 (0)	0.995 (0.0084)	0.998 (0.0063)	1 (0)
	Proposed - Soft Variant	1 (0)	0.984 (0.0096)	0.999 (0.0031)	0.983 (0.0176)

TABLE X
TRACE AVERAGE ACCURACY TABLE

	Method	50%	25%	12.5 %	9%
Wine	Sup-ResNet [3]	0.6962 (0.0654)	0.6277 (0.0883)	0.5814 (0.0895)	0.5037 (0.0918)
	SemiSup-Distillation [18]	0.7518 (0.0392)	0.687 (0.0678)	0.5444 (0.034)	0.5703 (0.0774)
	Proposed - Hard Variant	0.7259 (0.0622)	0.7222 (0.0359)	0.5629 (0.0304)	0.6444 (0.0299)
	Proposed - Soft Variant	0.7148 (0.0677)	0.6222 (0.0886)	0.5666 (0.0217)	0.624 (0.0641)

TABLE XI
WINE AVERAGE ACCURACY TABLE

tains nominal advantages over both alternatives in the vast majority of cases. However, these are not statistically significant, especially in scenarios with greater availability of ground-truth labels (first columns from left to right). In *Herring*, the Table VIII shows that both variants of the proposed method obtain significant improvements over the Sup-ResNet model and the semi-supervised baseline when considering a 25% of direct supervision. When traditional supervision drops to 12.5%, the *soft* variant is the only one to obtain significant margins over the *Sup-ResNet* method. Both the *hard* variant and the semi-supervised baseline nominally improve over the classical method, but these differences are not statistically conclusive. When label scarcity increases to the maximum (column 9%), the *hard* variant outperforms both the semi-supervised baseline and the classic method. In the *Meat* dataset, table IX shows that while the *hard* variant has an average accuracy of around 9 points over *Sup-ResNet* and around 13 points over *SemiSup-Distillation*, only the *soft* variant outperforms both techniques in a statistically significant way. Finally, the Table XI shows that in the dataset *Wine*, both variants of the proposed method improve over both alternatives when considering a 9% of supervision. In addition to being statistically significant, note that these differences occur again in the scenario with the greatest scarcity of ground-truth labels.

D. Summary

(win-loss) Variants	50%	25%	12.5%	9%
Sup-ResNet [3]	(1-0)	(2-2)	(4-0)	(5-2)
SemiSup-Distillation [18]	(0-0)	(0-1)	(1-1)	(4-2)
(win-loss) Hard Variant	50%	25%	12.5%	9%
Sup-ResNet [3]	(1-0)	(3-0)	(2-2)	(5-1)
SemiSup-Distillation [18]	(0-0)	(0-0)	(1-1)	(4-1)

TABLE XII
SUMMARY TABLE OF WINS AND LOSSES OF THE SOFT AND HARD METHOD AT THE DIFFERENT LEVELS OF SUPERVISION.

In this part, we want to summarize our experimental results. In table XII, we count the number of datasets on which our method was found statistically better than a baseline (wins) and the number of datasets on which a baseline outperformed our algorithm by a statistically significant margin (losses). We observe that the proposed methods fail to beat the baselines in all the datasets and scenarios considered in this section. However, we observe that the balance is positive regardless of the level of supervision, demonstrating the robustness of our approach. As discussed in a recent survey [7], semi-supervised methods often perform better than their supervised counterparts in specific cases, and the potential performance degradation is generally much more significant than the improvement. Therefore, the fact that in label-rich scenarios (columns 50% and 25%), ties and wins are much more frequent than losses is an important result. Moreover, as discussed above, the scenarios with a low percentage of supervision (columns 12.5% and 9%) in which our algorithms are outperformed correspond to datasets in which the baselines perform better using less ground-truth labels (see for instance table V and VII). Except for those atypical cases, we can posit that our method dominates in scenarios where ground-truth annotations are scarce. Indeed, the balance improves with more frequent wins and less frequent ties.

V. Conclusions

This paper presented Cluster Distillation, a semi-supervised method for time series classification. This method requires training a neural net (Teacher), detecting clusters in its feature space, and then training a second neural net (Student) with a regularized objective function. Our approach differs from standard Knowledge Distillation in that it does not train the Student to imitate the Teacher's beliefs about the class distribution but to preserve the clusters detected in the Teacher's feature space. To our knowledge, we are the first to evaluate clustering in a Knowledge Distillation framework and clustering as a pretext task for semi-supervised time series classification.

The method was evaluated on ten public datasets, considering different levels of supervision. In addition, we compared our approach with a state-of-the-art deep learning architecture for supervised time series classification and a recently proposed adaptation of Knowledge Distillation for semi-supervised time series classification. Our experiments show that the proposed method fails to beat the baselines in all the datasets and scenarios. However, we conclude that the balance is positive regardless of the level of supervision, with a more significant advantage in problems where ground-truth labels are scarce. Overall, results demonstrate that Cluster Distillation is robust, provides competitive results, and deserves further analysis.

Acknowledgements. The first author acknowledges financial support from the National Agency for Research and Development (ANID) Chile / PhD. Scholarship Program 2021.

References

[1] Robert H Shumway, David S Stoffer, and David S Stoffer. *Time series analysis and its applications*, volume 3. Springer, 2000.

[2] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Computing Surveys (CSUR)*, 54(4):1–40, 2021.

[3] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, mar 2019.

[4] Gabriel Spadon, Shenda Hong, Bruno Brandoli, Stan Matwin, Jose F Rodrigues-Jr, and Jimeng Sun. Pay attention to evolution: Time series forecasting with deep graph-evolution learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5368–5384, 2021.

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[6] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. Cluster kernels for semi-supervised learning. *Advances in neural information processing systems*, 15, 2002.

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

[8] Haoyi Fan, Fengbin Zhang, Ruidong Wang, Xunhua Huang, and Zuoyong Li. Semi-supervised time series classification by temporal relation prediction. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3545–3549, 2021.

[9] Qianli Ma, Sen Li, Wanqing Zhuang, Sen Li, Jiabing Wang, and Delu Zeng. Self-supervised time series clustering with model-based dynamics. *IEEE Transactions on Neural Networks and Learning Systems*, 32(9):3942–3955, 2021.

[10] Shayan Jawed, Josif Grabocka, and Lars Schmidt-Thieme. Self-supervised learning for semi-supervised time series classification. In Hady W. Lauw, Raymond Chi-Wing Wong, Alexandros Ntoulas, Ee-Peng Lim, See-Kiong Ng, and Sinno Jialin Pan, editors, *Advances in Knowledge Discovery and Data Mining*, pages 499–511, Cham, 2020. Springer International Publishing.

[11] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:4037–4058, 2021.

[12] Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, and Hamed Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *Proceedings of the IEEE CVPR*, pages 9359–9367, 2018.

[13] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*, 2018.

[14] Xueting Yan, Ishan Misra, Abhinav Gupta, Deepti Ghadiyaram, and Dhruv Mahajan. Clusterfit: Improving generalization of visual representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6509–6518, 2020.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *stat*, 1050:9, 2015.

[16] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019.

[17] Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[18] Mauricio Orbes-Arteaga, Jorge Cardoso, Lauge Sørensen, Christian Igel, Sebastien Ourselin, Marc Modat, Mads Nielsen, and Akshay Pai. Knowledge distillation for semi-supervised domain adaptation, 2019.

[19] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8):1295, 2020.

[20] Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

[21] Linfeng Zhang, Chenglong Bao, and Kaisheng Ma. Self-distillation: Towards efficient and compact neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4388–4403, 2021.

[22] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[23] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323. PMLR, 11–13 Apr 2011.

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[25] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.