LETS-GZSL: A Latent Embedding Model for Time Series Generalized Zero Shot Learning

Sathvik Bhaskarpandit[1] , **Priyanka Gupta**[1 2] , **Manik Gupta**[1]
[1]BITS Pilani Hyderabad Campus, [2]CVR College of Engineering
{f20191200, p20190501, manik}@hyderabad.bits-pilani.ac.in

Abstract

One of the recent developments in deep learning is generalized zero-shot learning (GZSL), which aims to recognize objects from both seen and unseen classes, when only the labeled examples from seen classes are provided. Over the past couple of years, GZSL has picked up traction and several models have been proposed to solve this problem. Whereas an extensive amount of research on GZSL has been carried out in fields such as computer vision and natural language processing, no such research has been carried out to deal with time series data. GZSL is used for applications such as detecting abnormalities from ECG and EEG data and identifying unseen classes from sensor, spectrograph and other devices' data. In this regard, we propose a Latent Embedding for Time Series - GZSL (LETS-GZSL) model that can solve the problem of GZSL for time series classification (TSC). We utilize an embedding-based approach and combine it with attribute vectors to predict the final class labels. We report our results on the widely popular UCR archive datasets. Our framework is able to achieve a harmonic mean value of at least 55% on most of the datasets except when the number of unseen classes is greater than 3 or the amount of data is very low (less than 100 training examples).

1 Introduction

Amongst the various problems related to handling time series data, time series classification (TSC) proves to be challenging due to presence of temporal ordering in the data [Yu *et al.*, 2021]. A data set $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ is a set of pairs (x_i, y_i) where $x_i = \{x_{i,1}, x_{i,2}, ...x_{i,T}\}$ is a univariate time series, y_i is its label, N is the total number of time series in the dataset and T is the total number of time stamps in a univariate time series. Given a dataset D, the aim of TSC is to output a label $\hat{y}_i$ given a new univariate time series as input [Lei and Wu, 2020].

The challenge of TSC is to capture the temporal patterns that exist in the data and identify the discriminatory features to differentiate between various classes. Several distance-based methods [Abanda *et al.*, 2019], machine learning (ML) [Esmael *et al.*, 2012] and deep learning (DL) [Fawaz *et al.*, 2019] models have been proposed to solve the TSC problem. Since distance-based methods had high computation costs and performed poorly with noisy time series, ML, especially DL methods gained much more focus in research. The features that are selected or extracted for building ML models greatly affect the performance, [Wang *et al.*, 2006] yet DL models provide an advantage of automated feature extraction. Several DL models have been proposed that achieve state of the art results on TSC [Fawaz *et al.*, 2019]. Furthermore, with the rise in number of publicly available datasets, such as the UCR [Dau *et al.*, 2019] and UEA [Bagnall *et al.*, 2018] archives, UCI Machine Learning Repository [Asuncion and Newman, 2007] and Physionet Database [Goldberger *et al.*, 2000], the amount of research in the past few years has grown to quite a large extent. However, there exist two significant shortcomings in many of the existing proposed algorithms that are inherently present in deep learning.

The first shortcoming is that these algorithms assume the availability of a large amount of labeled data for building a classifier. The second shortcoming is that most machine learning and deep learning algorithms assume that the set of class labels belonging to the training and test sets are identical. In a real-world scenario, this might not be the case; say, we have a completely new class of data that we have not seen before. Unlike humans however, computer algorithms require thousands of training examples to identify a new object with a high level of accuracy. Zero shot learning (ZSL) and Generalized ZSL (GZSL) can solve the problem of identifying new classes by transferring knowledge obtained from other seen classes.

In a conventional (ZSL) setting, the set of seen classes in training data and set of unseen classes in test data are mutually disjoint. This is considered an unrealistic scenario and does not reflect standard recognition settings. Most of the time, data from the seen classes is much more abundant than that of unseen classes. It is essential to classify both seen and unseen classes at test time, rather than only unseen classes as in ZSL; such a problem is formulated as a GZSL task. In this research work, our main focus is to provide a GZSL solution for TSC.

ZSL for TSC can be used for applications such as detecting heart abnormalities from ECG data, recognizing unknown classes in human activity recognition, identifying unseen classes from sensor and spectrograph data, etc [Wang]. ZSL and GZSL approaches have been already developed for image classification [Annadani and Biswas, 2018] and text [Alcoforado *et al.*, 2022] classification. To the best of our knowledge, there have not been any methods proposed to solve the problem of ZSL or GZSL for TSC. To fill this gap, our research aims to find a solution to classify time series effectively and alleviate the major difficulties of GZSL. The key contributions of this work are summarised as follows:

1. We propose a new model Latent Embedding for Time Series - GZSL (LETS-GZSL) to solve the problem of GZSL for Time Series Classification (TSC).
2. We test our method on different UCR time series datasets and show that LETS-GZSL is able to achieve a harmonic mean value of at least 55% for most of the datasets.
3. We show that LETS-GZSL generalizes well for seen as well as unseen classes, except when the number of unseen classes are high or the amount of training data is low.

2 Literature Survey

In this section we present an overview of the various techniques used for time series classification, followed by various methods adopted for ZSL and GZSL.

2.1 Time Series Classification (TSC) Techniques

One of the earliest and most popular approaches for TSC, often used as a strong baseline for several studies, is the use of a nearest neighbour approach coupled with a distance measure, such as the Euclidean Distance (ED) or Dynamic Time Warping (DTW) and its derivatives [Bagnall *et al.*, 2017; Hsu *et al.*, 2015]. Many approaches also use time series transformations that include shapelet transforms [Bostrom and Bagnall, 2015] interval based transforms [Baydogan and Runger, 2016], etc. However, ensemble techniques were found to outperform their individual counterparts, which led to complex classifiers such as BOSS (Bag of SFA Symbols) [Schäfer, 2015] and HIVE-COTE (Hierarchical Vote Collective of Transformation-based Ensembles) [Middlehurst *et al.*, 2021].

Upon a surge in the popularity of deep learning, several models were proposed that were easier to train than large ensembles. Deep learning models were not only faster to train than large ensembles, but could extract complex features automatically from data. Convolutional neural networks (CNN's) worked well with time series and models such as the Time-CNN [Zhao *et al.*, 2017] were proposed. Recurrent neural networks (RNN) and counterparts such as Long Short Term Networks (LSTM) [Pham, 2021] and Echo State Networks (ESN) [Gallicchio and Micheli, 2017] also performed well. Other approaches included autoencoder-based [Mehdiyev *et al.*, 2017], and attention-based models [Tripathi and Baruah, 2020]

While the aforementioned models are able to efficiently capture patterns in time series, they suffer from the constraint that most deep learning models have: they cannot generalize well to data with previously unseen class labels. In this regard, we discuss some of the existing ZSL and GZSL techniques in the following subsection.

2.2 ZSL and GZSL Techniques

We present an insight into some of the existing methods of ZSL and GZSL for computer vision, the area in which most of such research has been carried out. The 'visual space' is said to comprise the images, and the 'semantic space' the semantic information, usually in the form of manually defined attributes or word vectors [Pourpanah *et al.*, 2020].

Several methods adopt embedding-based approaches, that try to learn a projection function from the visual space to the semantic space (and perform classification in the semantic space) [Chen *et al.*, 2018] or vice versa [Annadani and Biswas, 2018]. However, such projections are often difficult to learn due to the distinctive properties of the two spaces. In this regard, latent embedding models [Zhang *et al.*, 2020] are used which project both visual and semantic features into a latent intermediate space to explore some common properties. Other techniques such as bidirectional projections [Ji *et al.*, 2020], meta-learning [Verma *et al.*, 2019], contrastive learning [Han *et al.*, 2021], knowledge graphs [Bhagat *et al.*, 2021] and attention mechanisms [Huynh and Elhamifar, 2020] have been proposed in relation to embedding-based approaches for ZSL and GZSL.

While purely embedding based approaches are effective in the GZSL paradigm, state of the art results are often achieved with generative approaches, that generate samples for unseen classes from samples of seen classes, and semantic attributes of unseen classes under the transductive GZSL scenario i.e. when the semantic attributes of unseen classes are available during training. Generative modelling alleviates the severity of bias towards seen classes in GZSL and such models tend to have more balanced seen and unseen generalization capabilities. Several generative approaches make use of some sort of generative adversarial network (GAN) [Li *et al.*, 2019], variational autoencoder (VAE) [Schonfeld *et al.*, 2019] or a combination of the two [Narayan *et al.*, 2020]. However, we follow an embedding based approach due to the inductive nature of our GZSL setting i.e. we do not have semantic information of unseen classes during training time. Additionally, generating synthetic time series that are close to the actual distribution of the unseen classes is a difficult task.

We seek to fill the void of GZSL for TSC by proposing a novel method LETS-GZSL, that makes use of latent embeddings and statistical attributes of time series, which is described in detail in section 4.

3 Problem Statement

In this section, we describe formally the setting of conventional ZSL as well as GZSL. We have S number of seen classes in $\mathcal{Y}_s$ and U number of unseen classes in $\mathcal{Y}_u$. The two sets are disjoint, i.e. $\mathcal{Y}_s \cap \mathcal{Y}_u = \emptyset$. The primary goal of ZSL and GZSL is to correctly predict the class label for each time series in the test set, which we describe subsequently.

Let $\mathcal{T} = S + U$ and $\mathcal{Y}_\mathcal{T} = \mathcal{Y}_u \cup \mathcal{Y}_s$. In both conventional and generalized ZSL, all the labelled instances used for training are from the seen classes $\mathcal{Y}_s$. More formally, the training set is $\mathcal{D}_{tr} = \{(x_i, y_i)|\ i = 1, 2, 3, ..., N\}$ where N is the number of time series in the training set, x_i is a univariate time series or a set of samples at different time stamps of the form $\{x_{i,1}, x_{i,2}, ...x_{i,T}\}$, T being the number of timestamps or time series length of x_i, and $y_i \in \mathcal{Y}_s$ is the class label corresponding to the time series x_i. The fundamental difference between ZSL and GZSL lies in the test set. In conventional ZSL, the labels y come from the seen classes' set of labels only, whereas in GZSL, the labels come from both seen and unseen classes' set of labels: $\mathcal{D}_{te} = \{(x_i, Y_i)|\ i = 1, 2, 3, ..., M\}$, where M is the number of time series in the test set, $y_i \in \mathcal{Y}_u$ for conventional ZSL and $y_i \in \mathcal{Y}_{\mathcal{T}}$ for GZSL. We focus on the more realistic GZSL problem for univariate TSC in this research work.

4 Our Proposed Method - Latent Embedding for Time Series GZSL

In the following section, we present our framework that uses three trained networks to effectively tackle the problem of GZSL for TSC. Figure 1 shows a flowchart of the various steps taken in the process. We summarize the major steps involved as follows:

1. Time Series Embedding Module: We train a network with contrastive loss to obtain representative embeddings for the time series.
2. Time Series Attribute Vectors: For every time series we obtain a vector containing statistical attributes to help discriminate between classes.
3. Latent Embedding Module: We utilize the time series embeddings and attribute vectors from the first two steps and project them into a latent space.
4. Classifier Module: The latent embeddings are finally fed into a neural network to predict each time series' class label.

4.1 Time Series Embedding Module

We first try to find suitable embeddings for our time series that can aid in classification. Therefore, we first learn an embedding function E that can capture temporal information from the time series and extract robust contextual representations for any granularity. To do so, we adopt the method followed by [Yue *et al.*, 2021] and use contrastive losses to learn representations at various scales. Random cropping is first performed to augment the input time series x_i, which generates two subseries that are subsequently passed into an encoder. The main encoder utilizes an input projection layer, timestamp masking module and dilated CNN module to obtain the representations of the two augmentations, which is depicted in Figure 2. Let $h_{i,t}$ and $h'_{i,t}$ denote the representations for some timestamp t, from these augmentations. Both instance wise contrastive (which views timestamps at the same position, but from different time series in a batch) and temporal wise contrastive (which views timestamps among different positions, but within the same time series) losses are used to learn the embeddings. The temporal contrastive loss is formulated as:

$$\ell_{temporal}(i,t) = -\log \frac{\exp(h_{i,t} \cdot h'_{i,t})}{\sum_{t' \in \Lambda} (exp(h_{i,t} \cdot h'_{i,t}) + \mathbb{1}_{[t \neq t']} \exp(h_{i,t} \cdot h'_{i,t})} \tag{1}$$

where Λ is the set of timestamps within the overlap of the two subseries cropped from a given time series, and $\mathbb{1}[.] \in \{0, 1\}$ is the indicator function that checks if $t \neq t'$.

Additionally, the instance-wise contrastive loss is given by:

$$\ell_{instance}(i,t) = -\log \frac{exp(h_{i,t} \cdot h'_{i,t})}{\sum_{j=1}^{B} (\exp(h_{i,t} \cdot h'_{j,t}) + \mathbb{1}_{[i \neq j]} \exp(h_{i,t} \cdot h_{j,t})} \tag{2}$$

where B denotes the batch size. The overall loss is given by:

$$\mathcal{L}(E) = \mathbb{E}_{i,t}[\ell_{temporal}(i,t) + \ell_{instance}(i,t)] \tag{3}$$

4.2 Time Series Attribute Vectors

Semantic information plays a crucial part in GZSL. It provides extra information about the nature of the seen and unseen classes that helps to establish the relationship between the two. In the field of computer vision, manually defined attributes or textual descriptions of images encoded as word vectors are commonly used as semantic information for performing GZSL [Pourpanah *et al.*, 2020]. However, in the case of time series, we do not have any explicitly given semantic information for either our seen or unseen classes. Instead, we introduce the use of statistical measures to derive an attribute vector $a_i \in \mathbb{R}^K$ for a corresponding time series x_i, that could further help discriminate between the seen and unseen classes. Thus, every attribute vector can be represented as:

$$\boldsymbol{a_i} = [\ f_1(x_i)\ f_2(x_i)\ f_3(x_i)\ ...\ f_K(x_i)\ ] \tag{4}$$

where $f_j\ , j = 1, 2, 3, ..., K$ are different functions applied on x_i and K is the number of such mathematical functions.

4.3 Latent Embedding Module

Once the time series embeddings and attribute vectors are obtained as specified in sections 4.1 and 4.2, they are jointly projected into a latent space. This latent space is the final space in which the classification is performed. We call this module L that learns the latent embedding as the classifier module. The input q_i to L consists of the concatenation of a time series' embedding h_i and its respective attribute vector a_i, i.e. $q_i = h_i \oplus a_i$. A convolutional neural network is trained and the final latent embedding $z_i = L(q_i)$, $z_i \in \mathbb{R}^P$ is learned, where P is the dimension of the latent space.

4.4 Classifier Module

The latent embeddings are finally fed through a feedforward neural network C that outputs a vector of logits $r = C(z_i)$ for a given time series. The final probability distribution for a given time series x_i over the S seen classes is given as follows:

$$p_c(x_i) = \frac{\exp(r_c/\tau)}{\sum_{c'=1}^{S} \exp(r_{c'}/\tau)} \tag{5}$$

where τ is the temperature parameter used in temperature calibration [Hinton *et al.*, 2015], which discourages the model from generating overconfident probabilities, which is especially useful since seen class probabilities tend to be much

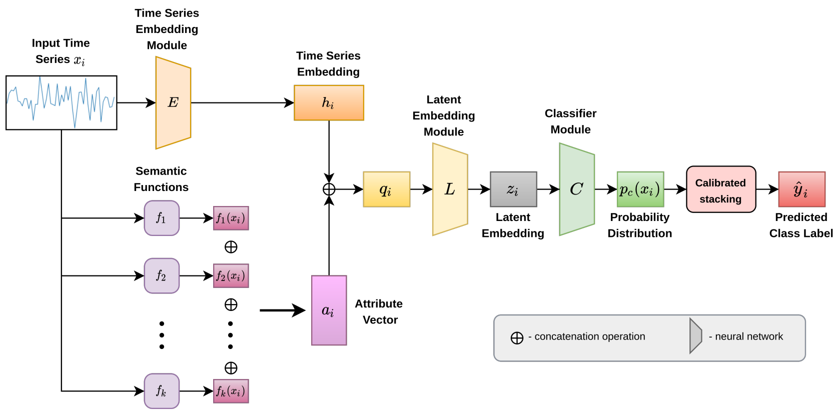

Figure 1: Illustration of the proposed LETS-GZSL method: We learn an embedding function E that maps each time series x_i into an embedding space $h_i = E(x_i)$. We also apply several statistical functions to x_i whose outputs are subsequently concatenated to form the attribute vector a_i. We then concatenate h_i and a_i and project the result into a latent space via the latent embedding module L. The latent embedding z_i is finally passed through a classifier C and calibrated stacking is performed to get the predicted class label $\hat{y}_i$

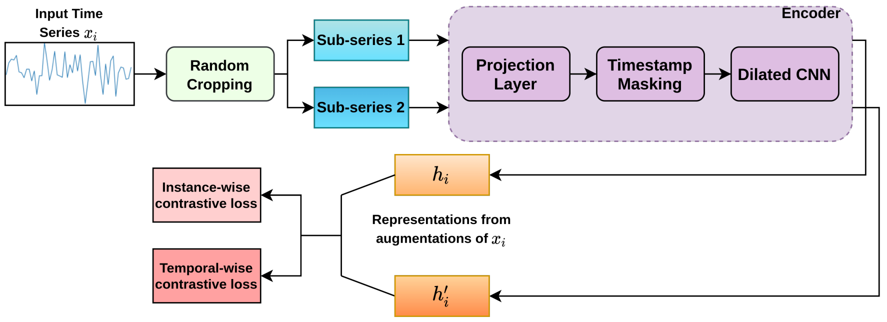

Figure 2: Overview of Time Series Embedding Module E

higher than unseen ones. Additionally, we also use calibrated stacking [Chao *et al.*, 2016] which intuitively reduces the scores of the seen classes. The final predicted class label $\hat{y}_i$ is given as:

$$\hat{y}_i = \underset{c \in \mathcal{Y}_\mathcal{T}}{\arg\max}\ p_c(x_i) - \gamma \mathbb{1}[c \in \mathcal{Y}_s] \quad (6)$$

where γ is a hyperparameter known as the calibration factor and $\mathbb{1}[.] = 1$ if $c \in \mathcal{Y}_s$ and 0 otherwise. An optimal combination of the hyperparameters τ and γ can penalize high seen class probabilities and lead to effective classification of unseen test samples as well.

The training loss for learning networks L and C is the cross-entropy between the probabilities and true labels (in the form of one-hot vectors), given by:

$$\mathcal{L}(C, L) = -\sum_{i=1}^{N}\sum_{c=1}^{S} y_{i,c}\ \log p_c(x_i) \quad (7)$$

The overall architecture of the L and C is depicted in Figure 3.

5 Experimental Details

5.1 Datasets

To test out the effectiveness of our proposed method, we use datasets from the UCR Archive [Dau *et al.*, 2019], which contains several different datasets for univariate time series classification. We choose datasets from different domains that differ in properties such as dataset size (the number of time series in the dataset), number of classes and time series length to study how LETS-GZSL performs with such variations. The datasets used and their properties can be found in Table 1:

Table 1: UCR Archive Datasets' Characteristics

Dataset	Type	Size	Number of Classes	Length
TwoPatterns	Simulated	5000	4	128
ElectricDevices	Device	16637	7	96
Trace	Sensor	200	4	275
SyntheticControl	Simulated	600	6	60
UWaveGestureLibraryX	Motion	4478	8	945
CricketX	Motion	780	12	300
Beef	Spectro	60	5	470
InsectWingbeatSound	Sensor	2200	11	256

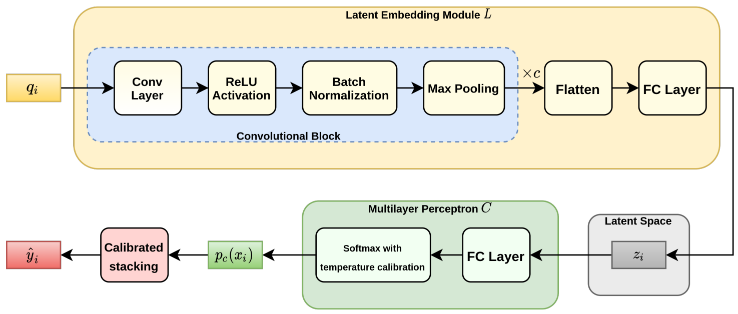

Figure 3: Architecture of the Latent Embedding Module and Classifier Module

5.2 Validation Scheme

Whereas in classical machine learning and deep learning, datasets are divided into train, validation and test sets sample-wise, in ZSL and GZSL, they are divided class-wise. Originally, we keep $\lceil \frac{2U}{3} \rceil$ classes as training and validation classes and the rest as test classes. 20% of the total samples belonging to the training and validation classes are kept aside as the seen test set. The samples from the test classes are kept as the unseen test set. To arrive at the optimal hyperparameters, we require a validation set, which we create by splitting the $\lceil \frac{2U}{3} \rceil$ classes again, same as before. Half the classes are kept for training and the other half for validation. 20% of the total samples belonging to the training classes are kept aside as the seen validation set, and the remaining as the unseen validation set. The division of classes is chosen so as to keep the number of samples in the validation and the test sets roughly the same.

The validation scheme [Le Cacheux *et al.*, 2019], is further shown diagrammatically in Figure 4 for a better understanding. Essentially, we are given the samples as shown in red and yellow during training time, for which we need to split appropriately into training and validation sets, to correctly predict the samples as shown in green during test time. Therefore, the LETS-GZSL model is first trained on the training set, and hyperparameters are tuned based on the metrics evaluated on the seen validation and unseen validation sets. The model is finally retrained on the train, seen and unseen validation sets, and the final results are reported on the seen and unseen test sets.

5.3 Evaluation Metrics

Under the GZSL scenario, we measure the top-1 accuracy on the seen and unseen classes, denoted as acc_s and acc_u respectively. Another metric we measure is the widely popular harmonic mean, which is able to measure the inherent bias towards seen classes and is defined as follows:

$$H = \frac{2 \times acc_s \times acc_u}{acc_s + acc_u} \tag{8}$$

We also report the area under seen unseen curve (AUSUC), introduced by [Chao *et al.*, 2016]. There is an intrinsic tradeoff between the seen and unseen accuracy of a GZSL model and the AUSUC metric measures the tradeoff between the two. The AUSUC metric is calculated by varying the value of γ in equation 6, noting the seen and unseen accuracies and then calculating the area under the curve given by the two quantities. Models with higher values of AUSUC achieve better balanced performance.

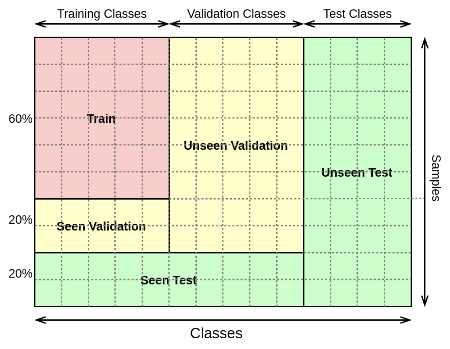

Figure 4: Validation Scheme: Each row represents number of samples and column represent number of classes. Note that the number of samples in the three sets may be different.

5.4 Implementation Details

We first find out the best hyperparameters for our embedding module for a given dataset. The main hyperparameters for E include the output dimensions of the representative embeddings, hidden dimension of the encoder, number of residual blocks in the encoder and batch size for training. The batch size is set to a larger value of 128 as suggested by Yue *et al.* [2021] to incorporate enough negative samples for calculating the instance-wise contrastive loss. The remaining hyperparameters are decided via a random search. We use an Adam optimizer [Kingma and Ba, 2014] with 10^{-3} as the initial learning rate to arrive at the optimal weights.

To compute the attribute vector a_i for each time series, we use statistical measures that include mean, median, max, argmax, min, argmin, skew, kurtosis and approximate entropy.

The latent embedding module L consists of convolutional blocks which is subsequently flattened and fed into a fully connected (FC) layer of the required latent embedding dimension P. The latent dimensions, number of convolutional blocks (denoted by c), filter sizes, number of filters, pool size and other hyperparameters of L are tuned for each dataset separately due to the varying lengths and nature of the time series. The hyperparameters of the classifier module C are also tuned simultaneously. We utilize the Tree of Parzen Estimators algorithm [Bergstra *et al.*, 2011] while using AUSUC as the metric to choose the optimal hyperparameters.

6 Results and Discussion

We evaluate the LETS-GZSL method on the datasets mentioned in section 5.1 and discuss the key results and observations. The various evaluation metrics recorded across the datasets are tabulated in table 2. To obtain the AUSUC values and plots, we vary γ from -1 to 0 using an interval of 0.05, then vary γ from 0 to 1, with an interval of 0.001, noting down the seen and unseen accuracies for each value.

Table 2: Evaluation metrics of LETS-GZSL on various UCR datasets

Dataset	AUSUC	acc_s	acc_u	H
TwoPatterns	95.233	94.504	88.815	91.571
ElectricDevices	55.749	60.971	55.323	58.010
Trace	52.699	43.333	76.000	55.195
SyntheticControl	46.028	50.000	68.500	57.805
UWaveGestureLibraryX	44.671	58.944	51.636	55.049
CricketX	27.328	50.427	39.487	44.291
Beef	25.000	10.000	100.000	18.181
InsectWingbeatSound	18.907	30.625	35.500	32.882

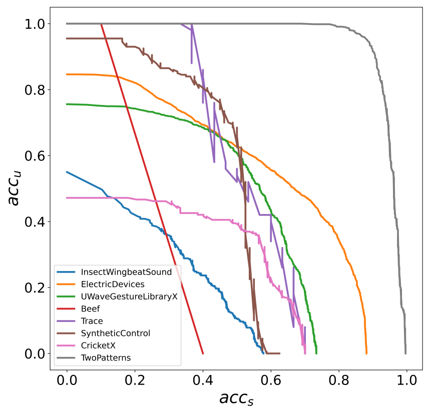

Figure 5: AUSUC curves for LETS-GZSL evaluation on various datasets

From the AUSUC values, we notice that LETS-GZSL is able to achieve good accuracies with both seen and unseen classes on most of the datasets. However, there are two factors that greatly influence the AUSUC. We discuss them below and provide plausible explanations for their effects.

The first factor is the number of classes, which has an inverse relationship to the AUSUC. Noticeably, although the harmonic mean is relatively high for datasets even with a large number of classes, the AUSUC is low in such cases, as can be seen in the CricketX and InsectWingbeatSound datasets. This implies that for some value of γ and τ LETS-GZSL can efficiently decrease the bias towards seen classes, yet lacks the intrinsic ability to balance the seen and unseen accuracy. We attribute this to the fact that our model can learn from seen class examples easily, but has no 'efficient mechanism' to distinguish between unseen class examples themselves. This is further demonstrated by the fact that calibrated stacking penalizes all unseen class scores by an equal amount.

The second factor is the amount of data, i.e. size of the dataset. Scarcity of labelled data has always posed a problem for deep learning models and we concur that the same problem applies to our framework. LETS-GZSL is not able to generalize to new seen class samples by simply seeing a couple of such samples during training. In this scenario, the predictions are almost equal to random guesses. Therefore, we notice from Figure 5 that the AUSUC curves of smaller datasets are noisier as compared to larger ones. From Table 2 it is evident that the TwoPatterns dataset, which is 25 times larger than the Trace dataset, has a much higher AUSUC despite the two having the same number of classes. Although the Beef dataset has only 5 classes, with 2 for training and validation each and one for testing, due to its extremely small size, the AUSUC and acc_s values are very low.

We also notice another interesting observation in the values of the harmonic mean for the various datasets. For datasets with smaller number of classes or limited data, the maximum value of H is achieved when the seen class scores are penalized heavily and $acc_s < acc_u$. For most datasets with relatively larger number of classes, acc_s tends to be greater than acc_u. For example, in the Beef dataset, $acc_s = 10\%$, while $acc_s = 100\%$. This could be due to the insufficient amount of data for recognizing seen classes themselves, as well as the ability to classify all the unseen class test examples correctly by simply using a large value of γ, since there exists only one unseen test class.

We further conduct ablation studies that can be found in the supplementary section to study how each component affects the model performance. Additionally, we also study performance against a deep learning baseline: the Long Short Term Memory-Fully Convolutional Network (LSTM-FCN) [Karim *et al.*, 2019], which shows the superiority of our model.

7 Conclusions and Future Work

In this paper, we have proposed the use of a latent embedding model for the purpose of GZSL for TSC. The proposed LETS-GZSL model uses contrastive loss to learn embeddings for time series data and projects it simultaneously along with statistical attributes to a latent space. A classifier is trained on these latent embeddings, with temperature calibration and calibrated stacking to aid in classification. We have shown that LETS-GZSL is able to generalize well to both seen and unseen class example, except when the number of unseen classes are high or the amount of training data is low.

In the future, techniques to solve the GZSL problem for TSC in the above mentioned cases of large number of unseen classes or limited amount of seen class data, can be explored. In this direction, meta learning methods and generative approaches can be employed to offset the above problem as well as class imbalance problems. Such techniques for multivariate TSC are also yet to be experimented with.

References

Amaia Abanda, Usue Mori, and Jose A Lozano. A review on distance based time series classification. *Data Mining and Knowledge Discovery*, 33(2):378–412, 2019.

Alexandre Alcoforado, Thomas Palmeira Ferraz, Rodrigo Gerber, Enzo Bustos, André Seidel Oliveira, Bruno Miguel Veloso, Fabio Levy Siqueira, and Anna Helena Reali Costa. Zeroberto–leveraging zero-shot text classification by topic modeling. *arXiv preprint arXiv:2201.01337*, 2022.

Yashas Annadani and Soma Biswas. Preserving semantic relations for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7603–7612, 2018.

Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery*, 31(3):606–660, 2017.

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

Mustafa Gokce Baydogan and George Runger. Time series representation and similarity based on local autopatterns. *Data Mining and Knowledge Discovery*, 30(2):476–509, 2016.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

PK Bhagat, Prakash Choudhary, Kh Singh, et al. A novel approach based on fully connected weighted bipartite graph for zero-shot learning problems. *Journal of Ambient Intelligence and Humanized Computing*, 12(9):8647–8662, 2021.

Aaron Bostrom and Anthony Bagnall. Binary shapelet transform for multiclass time series classification. In *International conference on big data analytics and knowledge discovery*, pages 257–269. Springer, 2015.

Wei-Lun Chao, Soravit Changpinyo, Boqing Gong, and Fei Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *European conference on computer vision*, pages 52–68. Springer, 2016.

Long Chen, Hanwang Zhang, Jun Xiao, Wei Liu, and Shih-Fu Chang. Zero-shot visual recognition using semantics-preserving adversarial embedding networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1043–1052, 2018.

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.

Bilal Esmael, Arghad Arnaout, Rudolf K Fruhwirth, and Gerhard Thonhauser. Improving time series classification using hidden markov models. In *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*, pages 502–507. IEEE, 2012.

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, 2019.

Claudio Gallicchio and Alessio Micheli. Deep echo state network (deepesn): A brief survey. *arXiv preprint arXiv:1712.04323*, 2017.

A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000. Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

Zongyan Han, Zhenyong Fu, Shuo Chen, and Jian Yang. Contrastive embedding for generalized zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2371–2381, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Che-Jui Hsu, Kuo-Si Huang, Chang-Biau Yang, and Yi-Pu Guo. Flexible dynamic time warping for time series classification. *Procedia Computer Science*, 51:2838–2842, 2015.

Dat Huynh and Ehsan Elhamifar. Fine-grained generalized zero-shot learning via dense attribute-based attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4483–4493, 2020.

Zhong Ji, Hai Wang, Yanwei Pang, and Ling Shao. Dual triplet network for image zero-shot learning. *Neurocomputing*, 373:90–97, 2020.

Fazle Karim, Somshubra Majumdar, and Houshang Darabi. Insights into lstm fully convolutional networks for time series classification. *IEEE Access*, 7:67718–67725, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yannick Le Cacheux, Hervé Le Borgne, and Michel Crucianu. From classical to generalized zero-shot learning: A simple adaptation process. In *International Conference on Multimedia Modeling*, pages 465–477. Springer, 2019.

Yuxia Lei and Zhongqiang Wu. Time series classification based on statistical features. *EURASIP Journal on Wireless Communications and Networking*, 2020(1):1–13, 2020.

Jingjing Li, Mengmeng Jing, Ke Lu, Zhengming Ding, Lei Zhu, and Zi Huang. Leveraging the invariant side of generative zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7402–7411, 2019.

Nijat Mehdiyev, Johannes Lahann, Andreas Emrich, David Enke, Peter Fettke, and Peter Loos. Time series classification using deep learning for process planning: A case from the process industry. *Procedia Computer Science*, 114:242–249, 2017.

Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. Hive-cote 2.0: a new meta ensemble for time series classification. *arXiv preprint arXiv:2104.07551*, 2021.

Sanath Narayan, Akshita Gupta, Fahad Shahbaz Khan, Cees GM Snoek, and Ling Shao. Latent embedding feedback and discriminative features for zero-shot classification. In *European Conference on Computer Vision*, pages 479–495. Springer, 2020.

Tuan D Pham. Time–frequency time–space lstm for robust classification of physiological signals. *Scientific reports*, 11(1):1–11, 2021.

Farhad Pourpanah, Moloud Abdar, Yuxuan Luo, Xinlei Zhou, Ran Wang, Chee Peng Lim, and Xi-Zhao Wang. A review of generalized zero-shot learning methods. *arXiv preprint arXiv:2011.08641*, 2020.

Patrick Schäfer. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505–1530, 2015.

Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8247–8255, 2019.

Achyut Mani Tripathi and Rashmi Dutta Baruah. Multivariate time series classification with an attention-based multivariate convolutional neural network. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

Vinay Kumar Verma, Dhanajit Brahma, and Piyush Rai. A meta-learning framework for generalized zero-shot learning. *arXiv preprint arXiv:1909.04344*, 2019.

Wei Wang. Sensor-based human activity recognition via zero-shot learning.

Xiaozhe Wang, Kate Smith, and Rob Hyndman. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*, 13(3):335–364, 2006.

Wennian Yu, Il Yong Kim, and Chris Mechefske. Analysis of different rnn autoencoder variants for time series classification and machine prognostics. *Mechanical Systems and Signal Processing*, 149:107322, 2021.

Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. *arXiv preprint arXiv:2106.10466*, 2021.

Lei Zhang, Peng Wang, Lingqiao Liu, Chunhua Shen, Wei Wei, Yanning Zhang, and Anton Van Den Hengel. Towards effective deep embedding for zero-shot learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(9):2843–2852, 2020.

Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017.

A Ablation Studies

We perform ablation studies to measure the impact of the LETS-GZSL components on the performance of the overall framework. More specifically, we remove the embedding module E and attribute vectors of the time series a_i one at a time and record the metrics on the datasets. We also give plausible reasons for the observations.

Table 3: Results without Embedding Module E

Dataset Name	AUSUC	acc_s	acc_u	H
TwoPatterns	80.628	84.455	73.290	81.791
ElectricDevices	53.768	54.904	48.977	51.771
Trace	49.933	50.613	41.526	45.621
Synthetic Control	44.102	65.004	47.722	55.038
UWaveGestureLibraryX	42.051	56.809	49.134	52.693
CricketX	25.447	48.560	37.334	42.213
Beef	14.216	12.125	20.375	15.202
InsectWingbeatSound	18.565	28.725	27.036	27.854

Table 4: Results without Stastical Attributes

Dataset Name	AUSUC	Acc_s	Acc_u	H
TwoPatterns	93.136	93.852	86.908	90.246
ElectricDevices	54.343	58.075	53.162	55.510
Trace	50.664	45.734	72.153	55.983
Synthetic Control	45.093	49.607	65.398	56.418
UWaveGestureLibraryX	43.589	57.216	50.574	53.69
CricketX	26.147	47.997	38.353	42.636
Beef	20.702	8.750	80.000	15.774
InsectWingbeatSound	17.841	29.119	32.675	30.790

On observing the results of the ablation studies we notice that the performance is not as good as that of the entire LETS-GZSL model.

In our first experiment, we remove the embedding module E and instead concatenate the raw input time series with the statistical features into the classifier module. From table 3 we notice that there is a major drop in performance in doing so. All the metrics, i.e., AUSUC, seen accuracy, unseen accuracy and harmonic mean are significantly lesser. This is possibly due to the inability of our model to discriminate between seen and unseen classes with raw input data, i.e. the time series. The embedding module has extracted meaningful representative features from the time series for the classifier module to discriminate between the classes. Additonally, the seen accuracy is greater than the seen accuracy for almost all the datasets.

In our second experiment, we utilize the embedding module E but do not concatenate any statistical features. The time series representations are directly fed into the classifier module. Here we notice that there is a much smaller dropoff in performance. For most datasets, this dropoff is around by 1 or 2 units in AUSUC. Interestingly, the decrease in unseen accuracy is more that that of the seen accuracy. We hypothesize that is due to the ability of the model to identify seen classes with relative ease even with just the embedding module. However discriminating between unseen classes themselves poses a bigger challenge and the statistical features further help distinguishing between seen and unseen classes.

B Comparison with Deep Learning Baseline: LSTM-FCN

In this section, we study the performance of our model against another deep learning model, the Long Short Term Memory - Fully Convolutional Network [Karim *et al.*, 2019]. We train the model using the scheme mentioned in section 5.2, utilize calibrated stacking and evaluate the metrics. Table 5 shows the results of training the LSTM-FCN.

Table 5: Evaluation of LSTM-FCN

Dataset Name	AUSUC	acc_s	Acc_u	H
TwoPatterns	52.651	66.973	37.302	47.916
ElectricDevices	24.822	47.749	32.886	38.948
Trace	10.666	6.666	44.00	11.578
Synthetic Control	23.800	28.750	17.000	21.736
UWaveGestureLibraryX	32.413	50.418	39.982	44.492
CricketX	13.670	36.752	27.692	31.585
Beef	11.214	20.145	12.590	15.495
InsectWingbeatSound	11.577	32.500	17.000	22.323

From the table we observe that the model does not perform well for GZSL and our LETS-GZSL model is far superior. Here we hypothesize that this is due to the fact that our embedding module with contrastive loss E is better able to capture meaningful information at various time scales, than the LSTM-FCN. Similar to our ablation studies in section A, the lack of time series attribute vectors which boosts LETS-GZSL's performance is also a reason why the LSTM-FCN performs worse.

C Symbol Table

Table 6: Various symbols used and their meaning

Symbol	Meaning
x_i	univariate time series
S	number of seen classes
U	number of unseen classes
$\mathcal{T}$	total number of classes
$\mathcal{Y}_s$	set of seen classes
$\mathcal{Y}_u$	set of unseen classes
$\mathcal{Y}_\mathcal{T}$	set of all classes
E	time series embedding module
h_i	time series embedding of x_i
a_i	attribute vector for x_i
q_i	concatenation of h_i and a_i
L	latent embedding module
z_i	latent embedding of x_i
C	classifier module
τ	temperature parameter
γ	calibration factor