Sample-Label View Transfer Active Learning for Time Series Classification

Patrick Kinyua[0000−0001−7962−2168]
and Nicolas Jouandeau[0000−0001−6902−4324]

Université Paris 8, PIF Department
Paris, France
patrick.gikunda@dkut.ac.ke*, n@up8.edu
PASTIS Research Group

Abstract. In many real-world applications, Time Series Data are captured over the course of time and exhibit temporal dependencies that cause two or otherwise identical points of time to belong to different classes or exhibit different characteristic. Although time series classification has attracted increasing attention in recent years, it remains a challenging task considering the nature of data dimensionality, voluminousness and continuous updates. Most of existing Deep Learning methods often depend on hand-crafted feature extraction techniques, that are expensive for real-world time series data mining applications which in addition, require expert knowledge. In practice, training a quality classifier is highly dependent on large number of labeled samples which is mostly inadequate in real-world time series datasets. In this paper, we present a novel Deep Learning approach for time series classification problems, called Transfer Active Learning (TAL) which jointly evaluates informativeness and representativeness of a candidate sample-label pair. TAL learns to map each input into a latent space from both sample and sample-label views which is more effective. For similar tasks, TAL is able to reuse model skill with further reduction on feature extraction costs. Extensive experiments on both classification datasets and real-world prediction tasks demonstrate the efficiency of the proposed approach on exponential reduction of training cost.

Keywords: Transfer Learning · Active Learning · Time series Classification.

1 Introduction

The technological landscape is changing at high speed, with ability to capture, process and disseminate huge amount of time-based data faster than before. Increased demand for smart homes, self-drive cars, autonomous trading algorithms, smart transport network, smart farming, weather forecast etc., uses a datatype that measures how things change over time [1]. Its characterized by data points indexed in *time* order and defined as *Time Series Data* (TSD) . The unique feature of TSD is that the new appended data does not overwrite previous data entry in the database. Therefore, there is ability to track system behavior changes over time as database insert and not update. Collecting such event data can lead to large sized datasets over short period of time. On the other hand, managing or analyzing such huge amount of data for practical world application comes at great performance cost which requires high computation power and complex methods.

Deep Learning (DL) is a well known Machine Learning technique with promising results in many prediction tasks. In particular the Convolutional Neural Network (CNN) which sit at the core of most of the recent breakthroughs in computer vision and data mining related tasks [2]. This is attributed by their ability to learn hierarchies of abstract localized structured data [3]. Despite of the impressive performance in analyzing big data, deep learning methods requires vast amount of labeled data for training. The dependence on large training data necessitates to develop methods for cutting down the classifiers training cost. Research on techniques to reduce training costs include Active Learning (AL), Transfer Learning (TL) and other statistical methods. Despite successful use of DL for big data analysis, it has not widely been used on time series datasets. The reasons for this absence might be: a) it is only recently that DL was proven to work well for TSC and there is still much to be explored in building DL methods for mining TSD [4] and, b) there is a lack of big, general purpose TSC datasets like ImageNet or OpenImages.

TL is used to improve model performance by reusing skills acquired in other related tasks especially where the target task has inadequate training data e.g. a model trained to classify mangoes can be used to classify oranges. Currently TL is used in many real-world DL applications [5]. Training is first done on a ConvNet using a large dataset (e.g. ImageNet which contains 1.2M samples with 1000 classes), and then transferring the ConvNet either by fine-tuning or as fixed feature extractor on the target task. The performance of a classifier on the target task can greatly be improved by using its experience on similar tasks. The assumption is that the source task and the target task can share some hyper-parameters depending on their nature [6]. When the source and target tasks are unrelated, the knowledge transfer from source task may not be useful or even compromise the performance of a target task through a negative transfer.

In DL its considered that the model can maximize the learning performance by allowing it to choose the data points to learn from [7]. AL provides means for iterative selection of data points the model wants to learn from [8]. This means that for a classification task the model will not require all the data for training but instead pick the most effective data-points for training. This is done by evaluating either or both the informativeness or representativeness of a sample. Whereas random and uncertainty sampling are often used, they suffer especially when data has skewed categorical features which can result in selecting non-informative or redundant samples. Classical statistical theory techniques such as entropy and margin are used as an utility to measure sample informativeness, however, they often fails to capture the data distribution information [9]. In this paper, we propose a query selection strategy based on a combination of both uncertainty and information density. The querying strategy is then logically coupled with TL which we now refer as *Transfer Active Learning* (TAL).

The rest of the paper is organized as follows: Section 2 highlights related works on TL and AL; section 3 presents our proposed Transfer Active Learning method; section 4 describes the experimental setup and respective results are presented in 5; section 6 presents our perspective and future interests.

2 Related Works

Time Series Classification (TSC) is a mapping task from the space of possible time based inputs to a probability distribution over the labels which can either be: a) an univariate time series $X = [x_1, x_2, ..., x_T]$ which is an ordered set of real values with the length of X being equal to the number of observable time-points T or; b) multiple time series $X = [X^1, X^2, ..., X^M]$ which consist of M observable per time-point with $X^i \in R^T$. We now define a time series dataset $D = (X_1, Y_1), (X_2, Y_2), ..., (X_N, Y_N)$ as a collection of pairs (X_i, Y_i) where X_i could either be univariate or multivariate with Y_i as its corresponding label. For a dataset containing K classes, the label vector Y_i is a vector of length K where each element $j \in [1, K]$ is equal to 1 if the class of X_i is j and 0 otherwise.

While the literature has sufficient works in which AL or TL is applied to TSD, a combination of the two remain comparatively unexplored. We attribute this to the fact that its only recent that TSC big datasets have began shaping up. In real-world TSC settings, the key motivation for use of AL is the dearth of training data. To account for lack of, or sparse training data, different techniques are used either together with user-intervention settings or a fully-model based. Considering the cost of training an effective model, methods such as random selection may result in models with unrepresentative truth of the actual data distribution. Therefore, it is particularly important to select samples that provide a distinct view of how the different classes are separated in the data with regions of greater uncertainty often sampled to define the decision boundaries [10].

Recurrent Neural Network(s) (RNN) are among the main methods used for time series forecasting. The recurrent based methods suffer the following limitations in the TSC tasks: a) they mainly predict an output for each time stamp in the time series [11]; b) when they train from long data series, they suffer vanishing gradient problem [12]; c) they have high computational requirement and hard to parallelize [13]. Successful application of DL in various domains motivates researchers to adopt different strategies to overcome RNN limitations in TSC tasks [4]. Majorly, these strategies are aimed at reducing the training data annotation cost using AL and transfer of model skill among tasks using TL.

Effectiveness AL strategies are based on sample selection techniques which include: a) uncertainty sampling with an objective to select an sample the classifier is most uncertain about as the most potential sample for labeling [14]; b) variance reduction aims at minimizing the model error rate by selecting samples with the minimum variance; c) expected gradient length that aims at querying sample that causes the maximal change to the current model [8]. The most common selection technique is the uncertainty sampling which considers the most uncertain sample for labeling [15]. In all cases the focus is to develop classification models with good generalization performance on unseen samples in the problem domain.

In representativeness-based approach the queried samples tend to provide a theoretical resemble of the overall distribution of the input space better. Unlike error-based methods which tend to improve the error behavior on the aggregate or uncertainty-based methods that query samples from most unknown regions, representativeness-based methods tend to avoid outlier-like samples by considering dense regions [15]. These methods combine both heterogeneity and representativeness behavior of the queried sample from the unlabeled set. Multiple approaches can be combined for query selection in active learning [16]. Such approaches exhibit: a) informative properties of the queried samples either close to the decision boundary of the learning model or far away from existing labeled samples in order to bring new knowledge about the feature space; b) representative behavior of the queried samples which should be less likely to be an outlier data and should be representative of the input space [16]. Natarajan and Laftchiev in their work combined TL and AL methods to predict personal thermal comfort. The method leverages domain knowledge from prior users and an AL strategy for new users that reduces the necessary size of the labeled dataset. When tested on real dataset from five users, their method achieves a 70% reduction in the required size of the labeled dataset as compared to the fully supervised learning approach [17]. The authors of [18] created an informativeness metric that considers the characteristics of time series data for defining instance uncertainty and utility. Their experiments on UCR archive presents a 50% reduction of training data. Using min-max approach the authors of [19] demonstrates viability of AL to reduce the annotation cost during training.

3 Transfer Active Learning

3.1 Transfer Learning

The TSC problem introduced in this paper is a semi-supervised classification problem. The two main strategies of deep TL include: a) using pre-trained models as feature extractors where objective is to leverage the pre-trained weight to extract features and only the final layer is replaced and; b) fine tuning pre-trained models by retraining some selected layers and freezing others. The question of weather to freeze pre-trained layers or use them as fixed feature extractor is determined by the size of available labeled set in the target settings that can be used for training. Under normal circumstance, when the labels in the target task are scarce, freezing is the best option to avoid overfitting. When the labels are sufficient then fine-tuning is a better choice. A DL model Θ with parameters and hyper-parameters can be represented as $Y \approx \Theta\vartheta(\theta|D)$ where θ are parameters and ϑ are hyper-parameters. D is training data and Y is class labels. The objective is to find estimate of parameters θ that optimizes some loss function L. The model performance based on loss function is dependent on ϑ, this implies that the parameters are also dependent on the hyper-parameters. The parameters are learnt during training, but hyper-parameters are set of initial model variables set before start of training and they include: number and size of the ConvNet layers, learning rate, weight initialization, etc. A general TSC problem can be expressed with the following equation:

$$C_t = f(w \times X_{t-l/2:t+l/2} + b) | \forall t \in \{1, T\} \tag{1}$$

where C denotes the result of a convolution on a univariate time series X of length T with a filter w of length l, a bias parameter b and a final non-linear function f. To learn multiple discriminative features filters are applied on individual univariate time series. Using the same filter values w and b in ConvNets, we are able to find the result for all time stamps $t \in [1, T]$. This is possible by using weight sharing that enables the model to learn feature detectors that are invariant across the time array.

3.2 Sample Selection

We consider actively selecting samples in batches, where the selection must be constrained by some budget. Let x_i represents a sample and y_i where $y_i \in \{1, ...K\}$ represents the class label for x_i, class in D we use the class probability estimator to compute the estimate of a label. In order to avoid the problem of generalization of unseen samples and to learn an accurate model, we present a robust approach that uses both uncertainty and correlation utility. Since the boundary class regions are often those in which samples of multiple classes are present, they can be characterized by class label uncertainty or disagreements between different learners. However, this may not always be the case, because samples with greater uncertainty are not always representative of the data, and may lead to the selection of outlier data points. This situation is especially likely to occur in datasets that are very noisy. In order to address such issues, some models focus directly on the error itself, or try to find samples that are representative of the underlying data. Based on the application, analyst goal, and data distribution, different strategies have different tradeoffs and work differently.

Uncertainty Measure In uncertainty sampling settings, the model attempts to select samples it is most uncertain about or what has not been seen so far. Application of the approach range from simple binary classification problem using Bayes classifier to multivariate classification using deep neural networks [20]. For the binary classification the naive Bayes classifier is normalized to ensure that the predicted probabilities sum up to 1. Therefore, the entropy objective function $En(X)$ for the binary class ($k = 2$) problem should be minimized and can be defined as follows:

$$En(\bar{X}) = \sum_{i=1}^{k} p_i - 0.5 \tag{2}$$

Great entropy value indicate greater uncertainty therefore the objective function should be maximized. In case of imbalanced data the uneven distributed classes are often associated with cost of misclassification denoted by w_i. Each probability p_i is replaced by a value proportional to $p_i \cdot w_i$, with the constant of the proportionality being determined by the probability values summing to 1. Given a label space Y the uncertainty measure f_u of independent and identically distributed sample considering the features and the label expressed as:

$$f_u(x) : \begin{cases} D^U \to R, & \text{(i) features view} \\ (D^U \times Y) \to R, & \text{(ii) features-label view} \end{cases} \tag{3}$$

to a real number space R. In 3: (i) the sample features are only considered in computing the sample uncertainty while in, (ii) a combination of sample label and features is considered in computing sample uncertainty. The three common metrics used to define uncertainty includes least confidence, sample margin and entropy. Least confidence consider the class label with the highest posterior probability with an objective function to decrease the error rate. Sample margin considers the first two most probable class labels using an objective function of decreasing the error rate. One major deficiency of both least confidence and sample margin is that they does not consider the output distributions for the remaining class labels in the set. Entropy considers class label over the whole output prediction distributions with an objective function to reduce the log-loss. Using entropy the uncertainty of an sample x_i in D^U can be defined as:

$$f_u(x_i) = \underset{i}{\operatorname{argmax}} - \sum_i P(y_i|x_i) \log P(y_i|x_i) \tag{4}$$

where $P(y_i|x_i)$ denotes posterior probability of the sample x_i being a member of the ith class, which ranges over all possible labels y_i. For a binary classification task, the most potential samples are the ones with equal posterior probability with respect to all possible classes.

Correlation Measure Majorly, AL query strategies use the uncertainty metric measure to evaluate the utility of a independent and identically distributed sample. However, when developing efficient AL methods, it is important to consider sample distribution information. The sample diversity information aids in querying most representative samples. This approach significantly improves the query performance while avoiding selecting outlier samples. Different algorithms for exploiting sample information exists. Majorly these algorithms are used in a multi-label learning tasks when an sample has more than one label. This setting is ideal for mining tasks on samples with complex structure. In this paper we focus on exploiting the pairwise similarities of samples, therefore the informativeness of a sample is weighed by average similarity to its neighbors. Let x_i and x_j be a pair of samples. To cope with the drawback of uncertainty based selection, we consider the diversity by evaluating the correlation of the samples.

Algorithm 1 Deep Transfer Active Learning (TAL)

Input: labeled data D^L, available D^U **Parameter**: θ, loss L, budget m **Hyper-parameters**: ϑ
Output: Θ

```
 1: $D_{train}, D_{test} \leftarrow D^L$
 2: $\Theta \leftarrow \vartheta\theta$ Initialize model
 3: while $m \neq 0$ do
 4:     for each $x_i$ in $D^U$ do
 5:         $u \leftarrow f_u(x)$
 6:         $c \leftarrow f_c(x)$
 7:         Select sample $\hat{x} \leftarrow \underset{i}{\text{argmax}}(u \cdot c)$
 8:         Predict class $\hat{y} \leftarrow \Theta(\hat{x})$
 9:         Update labeled set $D_{train} \leftarrow \hat{x}_t, \hat{y}_t$
10:         Compute query loss $L_{train} \leftarrow L(\hat{y}, y)$
11:     end for
12:     for $t = [1, T]$ do
13:         Get batch from D-train $x_t, y_t \leftarrow D_{train}$
14:         Get train loss on each batch $L \leftarrow L(\Theta(x_t), y_t)$
15:         $\theta \leftarrow$: Update parameters
16:     end for
17:     Get batch from D-test $x, y \leftarrow D_{test}$
18:     Get test loss on test batch $L_{test} \leftarrow L(\Theta(x), y)$
19:     $\Theta \leftarrow$: Update model
20:     $m \leftarrow m - 1$
21: end while
22: return $\Theta$
```

Given a label space Y the correlation measure f_c between a pair of samples x_i and x_j can be defined as:

$$f_c(x_i) = \frac{1}{D^U} \sum_{x_j \in D^U / x_i} f_c(x_i, x_j) \tag{5}$$

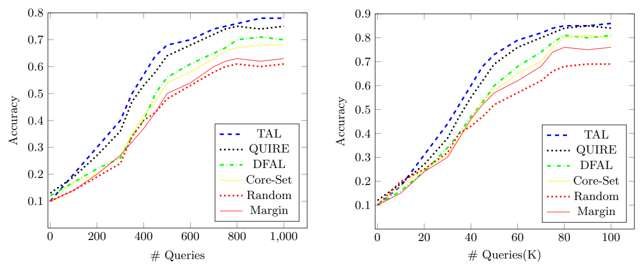
The value of $f_c(x_i)$ represents the sample density of x_i in the unlabeled set and $f_c(x_i, x_j)$ represents the mean of the correlation for all $j \neq i$. The larger the value, the more densely a sample is correlated with others. A low value of the correlation measure indicates an outlier sample which should not be considered for labeling.

Selecting the most informative and representative samples in a distribution is very critical for improving the generalization performance of the classifier. To do this we integrate correlation and uncertainty values together. The most effective sample to label by the current model can be expressed as a product of $f_u(x_i)$ and $f_c(x_i)$. As the uncertainty of a sample increases its potential for being selected for labeling increases. To do this we can rank the selected samples based on the utility value f_i: with the top ranked samples in each subset being the most effective samples to label. The query strategy discussed implemented in this paper is based on product utility $f_u(x_i) \cdot f_c(x_i)$. In Algorithm 1. learning starts from a small labeled set D^L with initialized parameters θ and hyper-parameters ϑ for the model Θ and proceeds sequentially. For each iteration of AL, uncertainty $f_u(x)$ and correlation $f_i(x)$ for each candidate sample x_i are computed. Then using a heuristic combination, the most informative sample is selected for labeling. After that, the new labeled sample is directly added to the training set D^L and the model is updated. Classification on new data proceeds in batch mode while computing loss for every iteration. Specifically classification is now presented as a mapping function from the feature space F to the class label space Y which can be expressed as $P(x) : F \mapsto Y$. To improve the model performance, we use a reward utility function for a single task of labeling an sample x.

$$R(Y, x) = \sum_{y} p(Y = y|x) r(p, y|x) \tag{6}$$

where p represents the posterior probability of sample x belonging class y with $R()$ as a reward function. This formula accumulates the reward on each possible label y.

4 Experimental Setup



(a) Rain in Australia (b) Flight Cancellation and Delay data

Fig. 1. Classification accuracy comparison on two real-world multivariate time-series datasets

To validate the effectiveness of the proposed approach, experiments were performed on both real-world application dataset for Rainfall in Austrialia & Flight Delay and Cancellation and 30 multivariate UEA Time Series classification [1] datasets. ResNet architecture was used to implement the experiments for two reasons: a) it has been adopted in other recent time series classifications [21]; b) it performs comparably well in a large number of cases [22]. This is because

[1] http://www.timeseriesclassification.com/

skip-connections or residual connections are very efficiently with deeper networks by allowing gradient flow directly through the bottom layers. The residual connections allow skipping of multiple layers within deeper neural network [23]. In the experiment settings, the network main hyper-parameters are 4 residual modules, 8×32 kernels and 128 filters. For all convolutional and dense layers L2 regularization is used with 10^{-1} learning rate and categorical cross-entropy used as loss function. Accuracy is used as a performance metric by recording training loss and performance reporting on the testset. A sigmoid function was used as a decision boundary to return a probability value between 0 and 1. The boundary is set to $p \leq 0.5$ for no rain and $p \geq 0.7$ for rain. All experiments were implemented in python with scikit-learn, PyTorch and on NVidia K80 GPU. We further test our model on transfer skill using the 30 multivariate time-series datasets. For each of the 30 datasets, we randomly partition it into two subsets, 70% training samples and 30% testing samples. From the training set, 10% is sampled as labeled set and 100 samples are actively selected in every iteration. To ensure uniform fine-tuning, only the last layer parameters are adjsuted to match the target classes. In the end we ended up with total of 60 pre-trained models i.e. 30 pair for each of the UCR dataset with Rainfall dataset and for Flight dataset respectively.

The selected pre-trained models were then used to learn a more challenging real-world problem using a multivariate rainfall in Austrialia[2] and flight information[3] datasets. The objective is to predict whether or not it will rain tomorrow. The dataset contains daily weather observations from numerous Australian weather stations. We then empirically demonstrate that combining TL and Al greatly improves performance. Firstly, we test the proposed method on two real-world prediction datasets. On the Rainfall data we segregate it into categorical and numerical variables. The date variable is denoted by Date column with 1000 total data points. The 4 categorical variables include: Location, WindGustDir, WindDir9am and WindDir3pm. There are two binary categorical variables i.e. RainToday and RainTomorrow with RainTomorrow being the target variable. For categorical the date variable that has the highest cardinality of 3436 labels, we performed some feature engineering to deal with high cardinality problem. To do this we parse the date coded as object into datetime format. Then one hot encoding is performed on all variables while adding dummy variables on missing data. The other data pre-processing include removing of outliers and splitting the data at 80% for training and 20% for testing. To avoid overfitting, imputation was done over the training set and then propagated to the test set. For missing categorical values, assumed input was done with most frequent value.

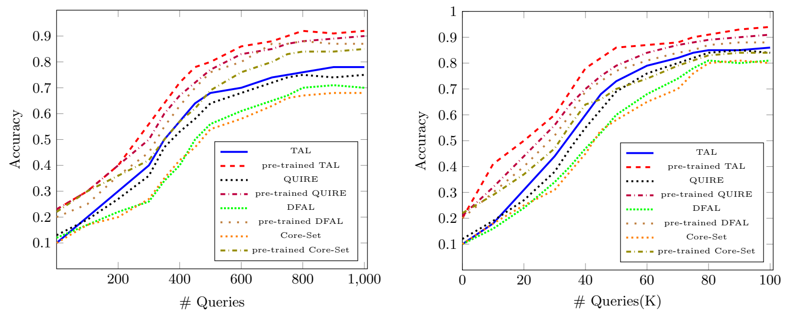
A set of 100 and 10000 samples was randomly selected as initial annotated samples for rainfall and flight datasets respectively. On flight information dataset, similar to above data preparation was carried out. The predicted probabilities of k classes are *delay. divert* and *cancelation* which take the form $p_1, ...p_k$ based on current labeled examples. Since $k = 3$ the entropy is defined

[2] https://www.kaggle.com/jsphyg/weather-dataset-rattle-package
[3] https://www.kaggle.com/usdot/flight-delays

as follows:

$$En(\bar{X}) = -\sum_{i=1}^{3} p_i . \log(p_i) \tag{7}$$

The data contain 100000 data samples spread over 9 categories. Each sample is linked to the cause which includes: cancellation reason, air-system delay, security delay, late-aircraft delay and weather delay.



(a) ElectricDevices (b) MelbournePedestrian

Fig. 2. Classification accuracy variation in percentage over the original accuracy and on pre-trained models. In (a) classification is done on Rain dataset using a model pre-trained on ElectricDevices dataset. Similarly, in (b) we compare the performance on Flight dataset using a model pretrained with MelbournePedestrian and (c) on FordB datasets

5 Results

We compare our proposed method with the following active learning methods in the experiments: a) Random selection; b) Margin based sampling; c) proposed method; d) QUIRE - a method inspired by the margin based active learning from the mini-max viewpoint with emphasize on selecting unlabeled instances that are both informative and representative [16]; e) DFAL method that selects unlabeled samples with the smallest perturbation. The distance between a sample and its smallest adversarial example better approximates the original distance to the decision boundary [24]; f) Core-Set for non-uncertainty based AL method [25]. Figure 1 shows classification accuracy of different AL methods with varied number of queries on each dataset. As expected the baseline methods are not as effective as hybrid methods. Both DFAL and Core-Set approach can outperform the Random and Margin methods but are worse than hybrid methods. Random and Margin approach have little improvement on both datasets. The performance of DFAL and Core-Set is impressive at the beginning, but loses the edge as the querying goes on. The proposed method performs 4% better than

QUIRE on both Rain and Flight datasets. We attribute the better performance on the ability to jointly evaluate sample informativeness and representativeness based on sample-label pair. In addition, the potential contribution of the current sample candidate is incorporated on the strategy for the subsequent query. Figure 2 presents the comparison on skill transfer using three best performing datasets. Its evident that skill transfer significantly reduce the training cost while achieving the same or better performance as compared to using un-trained model.

6 Conclusion

In this paper, we propose a novel Active Learning approach that jointly evaluates informativeness and representativeness for multivariate time series classification problems. Observing the structured nature of time series data, we propose a bi-objective method based on sample-label pair views which is considered more effective in reducing annotation cost. Learning is enhanced by Transfer Learning to further lower the learning cost by reusing model skill among tasks. One key phenomena observed with many UCR datasets is network overfitting which we attribute to the small size of the datasets. Since deep networks are highly dependent on large amount of training data, generating synthetic data can help mitigate the data size challenge. Also its important to note that with emerging time series big data repositories, the challenge of data size is a hot topic for many researchers. In the future, we will study other hybrid transfer active learning approaches.

References

1. Gikunda, P. K. and Jouandeau, N. *State-of-the-art Convolutional Neural Networks for Smart Farms: A Review*, In Intelligent Computing, pp. 763–775, Springer (ICIC 2019).
2. Szegedy, C., *et al. The Inception Architecture for Computer Vision*, In Conference on Computer Vision and Pattern Recognition, pp. 2818–2826, IEEE (CCVPR 2016).
3. Krizhevsky, A., Sutskever, I. and Hinton, G.E., *ImageNet Classification with Deep Convolutional Neural Networks*, In Communications of the ACM, vol. 60(6), pp. 84–90, ACM (2017).
4. Torres, J. F., *et al. A Deep Learning for Time Series Forecasting: A Survey*, Big Data, 9(1), pp. 3–21, Mary Ann Liebert Inc. (2021).
5. Gavves, E., *et al. Active Transfer Learning with Zero-Shot Priors: Reusing Past Datasets for Future Tasks.* In International Conference on Computer Vision, pp. 2731–2739, IEEE (ICCV 2015).
6. Raina, R., *et al. Self-taught learning: transfer learning from unlabeled data.* In International Conference on Machine Learning, pp. 759–766, ACM (2017).
7. Fu, Y., Zhu, X. and Li, B. *A survey on instance selection for active learning.* Knowl Inf Syst 35, pp. 49–283 (2013). https://doi.org/10.1007/s10115-012-0507-8
8. Settles, B., *Active Learning Literature Survey*, In Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009).

9. Li, X. and Guo, Y. *Adaptive active learning for image classification.* In Conference on Computer Vision and Pattern Recognition, pp. 859–866, IEEE (CCVPR 2013).
10. Lewis, D.D. and Catlett, J. *Heterogeneous uncertainty sampling for supervised learning.* In Machine Learning Proceedings, pp. 148–156 (1994).
11. Langkvist, M., Karlsson, L. and Loutfi, A. *A review of unsupervised feature learning and deep learning for time-series modeling.* In Pattern Recognition Letters, vol. 42, pp. 11–24 (2014).
12. Pascanu, R., Mikolov, T. and Bengio, Y. *Understanding the exploding gradient problem.* In CoRR, abs/1211.5063, vol. 417 (2012).
13. Pascanu, R., Mikolov, T. and Bengio, Y. *On the difficulty of training recurrent neural networks.* In International Conference on Machine Learning, pp. 1310–1318, ACM (2013)
14. Lewis, D.D. and Gale, W.A. *A sequential algorithm for training text classifiers.* In ACM-SIGIR Conference on Research and Development in Information Retrieval, pp. 3–12, Springer (1994).
15. Settles, B. and Craven, M. *An analysis of active learning strategies for sequence labeling tasks.* In Conference on Empirical Methods in Natural Language Processing, pp. 1070–1079 (2008).
16. Huang, S.J., Jin, R. and Zhou, Z.H. Active learning by querying informative and representative examples. In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36(10), pp. 1936–1949 (2014).
17. Natarajan, A. and Laftchiev, E, *A Transfer Active Learning Framework to Predict Thermal Comfort*, In International Journal of Prognostics and Health Management, vol. 10(3), pp. 1–13 (IJPHM 2019).
18. Peng, F., Luo, Q. and Ni, L.M. *ACTS: an active learning method for time series classification.* In International Conference on Data Engineering, pp. 175–178 (ICDE 2017).
19. Gikunda, P. K., and Jouandeau, N. *Cost-Based Budget Active Learning for Deep Learning.* In 9th European Starting AI Researchers' Symposium co-located with 24th European Conference on Artificial Intelligence, vol. 2655 (ECAI 2020).
20. Lewis, D.D. and Catlett, J. *Heterogeneous Uncertainty Sampling for Supervised Learning.* In Machine Learning proceedings, pp. 148–156 (1994).
21. Wang, Z., Yan, W. and Oates, T. May. *Time series classification from scratch with deep neural networks: A strong baseline.* In International Joint Conference on Neural Networks, pp. 1578–1585 (IJCNN 2017).
22. Fawaz, H.I., *et al. Deep learning for time series classification: a review.* In Data Mining and Knowledge Discovery, Vol 33(4), pp. 917–963 (2019).
23. He, K., Zhang, X., Ren, S. and Sun, J. *Deep residual learning for image recognition.* In Conference on Computer Vision and Pattern Recognition, pp. 770–778 (CVPR 2016).
24. Deng, Y., Chen, K., Shen, Y. and Jin, H. *Adversarial Active Learning for Sequences Labeling and Generation.* In International Joint Conference on Artificial Intelligence, pp. 4012–4018 (IJCAI 2018). https://doi.org/10.24963/ijcai.2018/558
25. Sener, O. and Savarese, S. *Active Learning for Convolutional Neural Networks: A Core-Set Approach.* In 6th International Conference on Learning Representations, (ICLR 2018).