2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA) | 978-1-6654-4337-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICMLA52953.2021.00129

Homogeneous Transfer Active Learning for Time Series Classification

Patrick Gikunda
PASTIS Research Group
LIASD, Université Paris 8
Paris, France
kinyuagikunda@gmail.com

Nicolas Jouandeau
PASTIS Research Group
LIASD, Université Paris 8
Paris, France
n@up8.edu

***Abstract*—The scarcity of labeled time-series data is a major challenge in use of deep learning methods for Time Series Classification tasks. This is especially important for the growing field of sensors and Internet of things, where data of high dimensions and complex distributions coming from the numerous field devices has to be analyzed to provide meaningful applications. To address the problem of scarce training data, we propose a heuristic combination of deep transfer learning and deep active learning methods to provide near optimal training abilities to the classification model. To mitigate the need of labeling large training set, two essential criteria – informativeness and representativeness have been proposed for selecting time series training samples. After training the model on source dataset, we propose a framework for the model skill transfer to forecast certain weather variables on a target dataset in an homogeneous transfer settings. Extensive experiments on three weather datasets show that the proposed hybrid *Transfer Active Learning* method achieves a higher classification accuracy than existing methods, while using only 20% of the training samples.**

***Index Terms*—Transfer Learning, Active Learning, Time Series Classification**

I. 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. This type of data is characterized by *time* as the primary axis defined as *Time Series Data* (TSD) and usually attributed by high dimension and complex distributions.

Machine Learning (ML) provides automated abilities for analyzing complex patterns in TSD. The problem is first formulated e.g. into a prediction problem and a model is built on a training dataset composed of input data, labeled with their corresponding classes. The model is then used to estimate the class of test data whose actual class is unknown. To build an accurate model, it is required to find an appropriately abstracted representation of data called features which would contain all the information relevant to the target problem. This process is referred to as *feature engineering*. Feature engineering is a key step in the model building process which is a two-step process of feature extraction and feature selection. The most popular feature learning methods are based on deep learning, e.g., using Deep Neural Networks (DNNs). A DNN consists of an ensemble of artificial neurons organized in a layer-wise fashion. Each neuron is a simple nonlinear computational unit with internal parameters, weights, and biases. Figure 1 presents a 1D Convolving by sliding a filter over the TSD input.

During training, these parameters are optimized so that the model can accurately categories training data into their own classes.

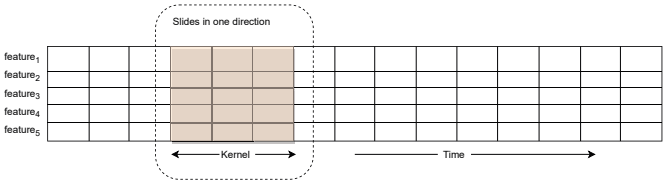

Fig. 1: Convolving on time dimension

In practice however, use of DNNs for TSD face several challenges such as: a) lack of practical technique for the optimisation of hyper-parameters (e.g. activation function, number of layers, number of neurons per layer, etc.); b) requirements in high computational power to train complex models in a reasonable amount of time; c) need for a diverse large quantity of labeled training data, etc. A possible solution to alleviate the training data scarcity problem is to use *Active Learning* (AL) or *Transfer Learning* (TL).

TL refers to techniques that aim at extracting knowledge from a source domain, and using it to improve the learning of a model on a target domain [1]. Data from the source domain can partially compensate the scarcity of data on the target domain by reusing model skill. Deep TL refers to transfer learning applied to DNNs which has become widespread with the rise in popularity of DNNs. For example, this is done by training ConvNets on a large dataset (e.g. ImageNet which contains 1.2M instances with 1000 classes), and then using the ConvNet either by fine-tuning or as fixed feature extractor for the target task. Typically, parameters (weights and biases) of a DNN model pre-trained on a source domain are transferred to another compatible DNN on the target domain. In deep TL, the hypothesis is if the features learned on the source model are useful for the target domain, then the parameters of a DNN pre-trained on the source domain can be used to initialize target model [2].

978-1-6654-4337-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICMLA52953.2021.00129

Once transferred, the target DNN is fine-tuned, i.e., re-trained using the target data to adjust the transferred parameters to the problem on the target domain as needed. In deep transfer settings, the ability of a classifier on the target task is based on its experience on similar tasks. The assumption is that the source task and the target task share some (hyper-)parameters but this not always the case [3] especially for heterogeneous tasks. 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. Thus, to ensure safe-transfer of knowledge is very critical to evaluate the similarity of the source and target tasks. In this paper, homogeneous datasets were considered hence no dataset similarity evaluation was undertaken.

AL provides means of iteratively picking data points the model wants to learn from [4]. This means that for a classification task the model will not requires all the data for training but instead pick the most effective data-points for training. This provides means of analyzing vast amount of data with improved efficiency by iteratively select the most informative data instances [5]. In AL various strategies for selecting training samples exist. Majorly, the strategies are based on random sampling or uncertainty sampling. Randomly selecting training instances is inefficient in many situations especially when data has skewed categorical features which can result in selecting non-informative or redundant instances. Better performing strategies use statistical theory such as entropy and margin to measure instance informativeness, however, it often fails to capture the data distribution information [6].

While deep AL and TL are widely used in other domains especially image analysis, they have not reached the same level of maturity for time series prediction tasks mainly because of: a) time-series labeled datasets are rather scarce due to the high cost of the labeling especially for a specific application; b) lack of very large-scale time-series datasets e.g. ImageNet; c) challenge of varying data formats and units of measurement. For instance, some datasets provide sparse time-series containing data points unevenly spaced in time indicating events, while others provide non-sparse time-series consisting of data values evenly spaced in time and sampled at high frequencies; d) data dimensionality consisting of different numbers of channels. For instance, a temperature sensor provides a single channel sequence, while three-axis accelerometers record three channels, each indicating the acceleration on one axis, etc. In this paper, we present an hybrid approach called *Transfer Active Learning* (TAL), for creating an efficient time series classifier.

The rest of the paper is organized as follows: Section II highlights related works on TL and AL on time series analysis; section III presents our proposed TAL method; section IV describes the experimental setup and respective results; section V present our perspective and future interests.

II. Related Works

A. Deep Learning for Time Series Classification

Recurrent Neural Network (RNN) are popular methods for time series forecasting. However, when analyzing large time series datasets they suffer the following limitations; a) they mainly predict an output for each time stamp in the time series [7]. b) when they train from long data series, they suffer vanishing gradient problem [8]. c) they have high computational requirement and hard to parallelize [10]. Successful application of deep learning in various domains motivates researchers to adopt deep learning methods for Time Series Classification (TSC) in an effort to overcome RNN limitations [10]. Wang *et.al* [10] presents a ResNet for TSC problem and validated on the UCR[1] a TSD archive that consists of 85 small-scale univariate time-series datasets covering a wide range of sensor modalities, such as accelerometer data, energy demand, chemical concentration in water, etc.

The first 9 ConvNets layers are followed by a global average pooling layer that averages the time series across the time dimension for reducing the total number of parameters in the model hence avoiding overfitting. In general ResNets are characterized by shortcut residual connection between successive ConvNet layers. On a performance comparative study of DL methods on TSC, ResNet used was found to outperform other 9 methods and the generalization capabilities was attributed to the flexible architectural nature of CNN [11]. The authors of [12] built a CNN based time series classifier from scratch. To circumvent the need of big training data problem they used semi-supervised training method and data augmentation techniques that warped and split the time-series dataset. On a real-world problems DL has been used for spatio-temporal series forecasting problems, such as oceanography and meteorology [13]. On human activity recognition using wearable sensors, DL is slowly replacing the feature engineering approaches by automatically learning the features through back-propagation [14]. On electronic health records, a generative adversarial network with a CNN was trained for risk prediction based on patients historical medical records [15].

B. Transfer Learning for Time Series Classification

AL and TL techniques have been explored much less for time-series data because of the scarcity of training data, and the absence of a large-scale labeled dataset like ImageNet. Past works have attempted to tackle this issue with different degrees of generality. In [1] they re-use data in the source domain to train a model in the target domain through: a) feature representation transfer that finds a feature mapping between the source and target domains, and; b) parameter transfer that transfers parameters from a source model to target model. Several works presents results of parameter transfer. In [16], the results in several scenarios of parameter transfer such as transfer between subjects, datasets, sensor localisation, or modalities were presented. The performances of transfers were better when parameters of the lower layers were transferred.

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

In [17], a transfer approach for CNN was presented. It firstly trains a CNN using labeled data on the source domain and defines a CNN with similar architecture on the target domain. The target CNN is then trained on unlabeled data to minimize the distance between its parameters and the ones of the source CNN. It, however, only works under the assumption that the set of activities on the source and target domains is the same. In [18], an iterative co-training approach using classification models trained on labeled source data to attribute pseudo-labels to unlabeled target data was presented. It works under the assumption that source and target domains contain the same labels. Source and target data are then projected into a common space using the transformation, and classifiers are trained on the projected data to attribute more reliable labels.

In [19], a RNN was trained using data from the UCR. The RNN composed of an encoder and decoder to reproduce its input on its output layer using a subset of 24 datasets of the UCR (source domain). After this pre-training step, the encoder was used as a feature extractor for a Support Vector Machine (SVM) fine-tuned on each of 30 other datasets of the UCR (target domain). The experimental results indicated that data on source domains not necessarily related to the target domain were still useful for achieving state-of-the-art results. Similar recent efforts in [20] to create a pre-trained model for TSD tasks using UCR data. In [11], a method to compute the similarity between source and target datasets to determine the most suitable dataset for transfer was proposed. It assumes that one labeled target and several labeled source datasets are available. For each dataset, the method firstly computes the average of sequences for each class. The barycentre of all class averages is then computed to yield a characteristic sequence of the dataset. The similarity between two datasets is computed using the Dynamic Time Warping distance between their respective characteristic sequences. The source dataset with the lowest distance is then chosen and used to train a DNN. Its weights are finally transferred on the target domain for fine-tuning. Experiments carried out on the 85 datasets of the UCR showed that the transfer yielded better classification performances when the similarity between source and target was higher. Using dissimilarity matrix, Spiegel *et,al* were able to transfer specific training examples from source dataset to target set [21]. Their model was used to predict the wind speed in a farm. First they trained the base network using historical wind-speed data, then validated the model using new data from the farm. Using restricted Boltzmann machines Banerjee *et.al* first pre-trained the model for acoustic phoneme recognition and then fine-tuned for post-traumatic stress disorder diagnosis [22]. In their work, Serra *et.al* use TL to improve the accuracy of deep neural networks for TSC. The CNN was designed with an attention mechanism to encode the time series in a supervised manner before fine-tuning on a target dataset [23].

C. Active Learning for Time Series Classification

Many AL strategies have been proposed for traditional active learning, however, none of them is particularly effective for TSC. Three main sample selection techniques include: a) Uncertainty sampling with an objective to select an instance the classifier is most uncertain about as the most potential instance for labeling [24]; b) Variance reduction aims at minimizing the model error rate by selecting instances with the minimum variance, and c) Expected gradient length that aims at querying instance that causes the maximal change to the current model [4]. AL strategies for TSC range from density estimation to multi-factor methods. The strategies can be categorized into two groups: *population-based* strategies or *pool-based* strategies [25]. In population-based AL, train and test sets are drawn from the same distribution. The assumption here is that train and test data both follow the same conditional distribution. The objective is to find the optimal input density for drawing instances for labeling. In pool-based AL, the objective is to select instances from a pool so that a model trained from them can best classify the remaining instances. Whether population or pool-based, AL is an iterative process [26]. First a base model is built using a small number of labeled training instances, then using utility metrics it selects instances and queries for their labels. The prediction result of a single instance is represented by a vector, whose elements are the posterior probability with respect to each class label. The newly labeled instances are added to the training labeled set and the model is updated. This process iterates until a termination criterion is met, e.g, query budget or number of iterations is exhausted. Based on the number of unlabeled instances to query at each iteration, AL methods can be grouped as either: a) sequence-mode AL- a single instance is queried at a time or; b) batch-mode AL- multiple instances are queried at each iteration [26].

The authors [27] proposed a procedure to ask experts to label the frequent patterns of a long time series stream. He *et al.* [28] provided a metric that considered both the uncertainty of the classifier, and similarity between time series. Neither methods have exploited the patterns in time series, which is important in saving the labeling cost. A recent study [30] propose use of Nearest Neighbor by adapting shapelet discovery to find the discriminative patterns in the training set, and incrementally update the patterns as new training samples is availed. Further, the study propose a probabilistic model over the instances, patterns and labels that considers both the diversity of label distribution, and patterns of samples. Few works on combination of TL and AL for TSC exist. 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 [26].

III. Transfer Active Learning

The proposed model is made up of two main deep learning techniques namely: a) TL to provide model skill re-use on the target task; b) AL to interactively query and add samples on the

training set by using labeled data to provide information about the class labels or class boundaries while the unlabeled data is used to learn the base data distribution. Before diving into details of mentioned techniques, lets first provide a definition of TSC problem.

Definition III.1. An univariate time series $U_t = [x_1, x_2, ..., x_T]$ is an ordered set of real values. The length of U_t is equal to the number of observable time-points T.

Definition III.2. A multivariate time series $M_t = [U_t^1, U_t^2, ..., U_t^n]$ consists of n observables per time-point with ${U_t}^i \in R^T$.

Definition III.3. A dataset $D = (X_1, Y_1), (X_2, Y_2), ..., (X_N, Y_N)$ is a collection of pairs (X_i, Y_i) where X_i could either be U_t or M_t 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.

We can therefore define, **TSC** as a mapping task from the space of possible time based inputs to a probability distribution over the labels denoted by the following equation:

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

C denotes the convolution result on a univariate time series U_t of length T with a filter w of length l, a bias parameter b and a non-linear function f. Applying several filters on a time series will result in a multivariate time series whose dimensions are equal to the number of filters used. Using the same filter values w and b in ConvNets its possible to find the results 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.

A. Deep Transfer Learning

A simple definition of TL is using knowledge acquired on one task (source task), to solve related task (target task). The more related the tasks, the better suitability of the model skill. Invariably, researchers define TL from different contexts but regardless, the motivation of TL is exploiting knowledge acquired from source setting to improve generalization on target setting with exponentially fewer training examples. Lets begin this section by first defining deep TL.

Definition III.4. A domain can be denoted as $D = \mathcal{X}, P(X)$ with $\mathcal{X}$ representing the feature space and $P(X)$ representing the marginal probability. $X = \{x_i, ..., x_T\}, x_i \in \mathcal{X}$ where x_i represent specific time vector.

Definition III.5. For a given domain, the task can be defined as $T = \mathcal{Y}, P(Y|X)$ with $\mathcal{Y}$ representing the label space and $P(Y|X)$ representing the prediction objective function (learned from instance feature/label pair) where $Y = \{y_i, ..., y_T\}, y_i \in \mathcal{Y}$.

Definition III.6. Given source domain $D_S = \mathcal{X}, P(X)$ and source task $T_S = \mathcal{Y}, P(Y|X)$ then TL is learning target conditional probability distribution $P(Y_T|X_T)$ in D_T.

With the above definitions in mind, a deep learning model Θ is denoted 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.

The two main strategies of deep TL include: a) Using pre-trained models as feature extractors. The objective is to leverage the pre-trained weight to extract features and only the final layer is replaced . b) Fine Tuning pre-trained models. Selective layers are re-trained and others frozen. 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 is scarce, freezing is the best option to avoid overfitting. When the labels are sufficient then fine-tuning is a better choice.

During target training, first the parameters's are initialized using previous task weights $\Theta \leftarrow \vartheta\theta$. After the weight's initialization, a forward pass through the model is applied using the function $f(\theta, x_i)$ and the output of an input x_i is computed. The output is a vector whose components are the estimated probabilities of x_i belonging to each class. The model's prediction loss is computed using a cost function, for example the negative log likelihood. Then, using gradient descent, the weights are updated in a backward pass to propagate the error. Thus, by iteratively taking a forward pass followed by backpropagation, the model's parameters are updated in a way that minimizes the loss on the training data. During testing, the model is tested on unseen data which is and a forward pass on this unseen input followed by a class prediction. The prediction corresponds to the class whose probability is maximum. For this case a categorical cross entropy is applied as the loss function denoted as:

$$L(U_t) = -\sum_{j=1}^{K} Y_j \log \hat{Y}_j \tag{2}$$

with $L(U_t)$ denoting the loss when classifying the input time series U_t, $\hat{Y}_j$ denoting probability of U_t class Y equal to class j out of K classes in the dataset. For batch wise training loss can be defined using the following equation:

$$J(\Omega) = \frac{1}{N} \sum_{n=1}^{N} L(U_t^n) \tag{3}$$

B. Sample Selection

The key objective in active learning is to develop algorithms with precise queries that maximize the accuracy of classifi-

cation or prediction task. Since the boundary class regions are often those in which instances 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 instances with greater uncertainty are not representative of the data, and may sometimes lead to the selection of unrepresentative outliers. This situation is especially likely to occur in data sets 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. We consider actively selecting instances in batches, where the selection must be constrained by some budget. Let x_i represents an instance and y_i where $y_i \in \{1, ...K\}$ represents the class label for x_i, $D = D^L \cup D^U$, D^L denotes labeled instances where $D^L = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, D^U denotes unlabeled instances where $D^U = \{(x_1, ?), (x_2, ?), ..., (x_n, ?)\}$, and Θ to denotes model defined by model parameters. For label space Y with K classes in D the class probability estimator is used to compute the class estimate of a label.

1) Uncertainty Measure: In uncertainty sampling settings, the model attempts to select instances it is most uncertain about or what has not been seen so far. The Three major metrics used to define uncertainty include; 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 instance x_i in D^U can be defined as:

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

where $P(y_i|x_i)$ denotes posterior probability of the instance 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 instances are the ones with equal posterior probability with respect to all possible classes. For the binary classification the 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 is minimized as follows:

$$En(\bar{X}) = \sum_{i=1}^{k} P - 0.5 \tag{5}$$

2) Correlation Measure: Majorly, AL query strategies use the uncertainty metric measure to evaluate the utility of a independent and identically distributed instance. However, when developing efficient AL methods, it is important to consider instance distribution information. The instance diversity information aids in querying most representative instances. This approach significantly improves the query performance while avoiding selecting outlier instances. In this paper we focus on exploiting the pairwise similarities of instances, therefore the informativeness of an instances is weighed by average similarity to its neighbors. Let x_i and x_j be a pair of instances. Given a label space the correlation measure (f_c) between a pair of instances x_i and x_j can be defined as:

$$f_c(x) = \frac{1}{D^U} \sum_{x_j \in D^U / x_i} f_c(x_i, x_j) \tag{6}$$

The value of $f_c(x_i)$ represents the instance density of x_i in the unlabeled set. The larger the value, the more densely an instance is correlated with others. A low value of the correlation measure indicates an outlier instance which should not be considered for labeling.

3) Most Informative and Representative sample: To select the most informative and representative samples in a distribution, a heuristic combination of correlation and uncertainty is done. The most effective instance to label by the current model can be expressed as:

$$x^* = \underset{i}{\operatorname{argmax}}(f_u(x) \cdot f_c(x)) \tag{7}$$

Algorithm 1: Transfer Active Learning (TAL)

input : labeled data $D_{train}, D_{test} \leftarrow D^L$, available D^U, budget m
$\Theta \leftarrow \vartheta\theta$ initialization;
while $m \neq 0$ **do**
  for *each* $x_i \leftarrow D^U$ **do**
    $u \leftarrow f_u(x)$, $c \leftarrow f_c(x)$;
    Select $\hat{x} \leftarrow \underset{i}{argmax}(u \cdot c)$;
    Predict class $\hat{y} \leftarrow \Theta(\hat{x})$;
    Update labeled set $D_{train} \leftarrow \hat{x}_t, \hat{y}_t$;
    Compute query loss $L_{train} \leftarrow L(\hat{y}, y)$;
  for $t = [1, T]$ **do**
    Get batch from D-train $x_t, y_t \leftarrow D_{train}$;
    Get train loss on each batch $L \leftarrow L(\Theta(x_t), y_t)$;
    $\theta \leftarrow$: Update parameters;
  Get batch from D-test $x, y \leftarrow D_{test}$;
  Get test loss on test batch $\theta \leftarrow L(\Theta(x), y)$;
  $m \leftarrow m - 1$;
output: Output Θ

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 $f_c(x)$ for each candidate sample x_i is computed. Then using a heuristic combination, the most informative instance instance is selected for labeling. After that, the new labeled instance is directly added to the training set D^L to update the model. Training on new data proceed in batch mode while computing loss for every batch.

IV. Experimental Setup

In this section, we empirically study the proposed method for multivariate time series on three real-world datasets. The experiments were carried out on NVidia K80 GPU, which provided up to 12.0X speedup compared to standard CPU. The experiments were implemented using Pytorch. To show competence of proposed method, comparative experiments were run on pre-trained and un-trained settings. Compared AL methods include: a) **Random** selection of samples in the distribution; b) **Margin** selection based on the sample distance to the hyperplane; c) **QUIRE** margin based with mini-max viewpoint [30]; d) **DFAL** adversarial with smallest perturbation [31], and ; e) **Core-Set** non-uncertainty based AL method [32].

A. Network Architecture

For experiments ResNet architecture was considered following two reasons: a) the architecture has been adopted in other recent time series classifications [33]; b) it performs comparably well in a large number of cases [11]. This is because skip-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. In the proposed network the main hyper-parameters are 4 residual modules, 8×32 kernel size and 128 filters. For all the convolution and dense layers L2 regularization is used, 10^{-2} learning rate, categorical cross-entropy is used as loss function. Accuracy is used as a performance metric by recording training loss and performance reporting on the test set. Sigmoid function is used as a decision boundary to return a probability value between 0 and 1. Training batch size was set to 64 with and testing batch size set to 128 with 100 epochs on each round of training.

B. Datasets

Homogeneous transfer was considered with pre-training done with three multivariate time series datasets namely: a) RAUS - Rainfall in Austrialia[2] - dataset contains daily weather observations from various Australian weather stations for a period of 10 years (with over 110K samples); b) MeteoNet[3] - a meteorological dataset developed and made available by METEO FRANCE, the French national meteorological service. The dataset temporal coverage range 2016-2018 (with over 60M samples) and spatial coverage being North-West region of France. The dataset contains full time series over 500 ground stations measuring pressure, temperature, humidity, wind direction and speed, dew point and precipitation, recorded every 6 min; c) KenCentralMet - Kenya Meteorological Department[4] daily weather observations covering Central Kenya for a period of 3 years from 2012-2014 (with over 100K samples).

For RAUS dataset, data was first segregated into categorical and numerical variables. The 6 categorical variables include: Location, WindGustDir, WindDir9am, WindDir3pm, RainToday and RainTomorrow. There are two binary categorical variables i.e. RainToday and RainTomorrow with RainTomorrow being the target variable. For categorical the date variable 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 was performed on all variables while adding dummy variables on missing data. The other data preprocessing include removing of outliers and splitting training and testing data at 0.2. Median imputation of missing data was done and in order to avoid overfitting imputation was done over the training set and then propagated to the test set. For missing categorical values, input was done with most frequent value. Similarly, for MeteoNet and Central Kenya dataset, one hot encoding was performed on all categorical variables while adding dummy variables on missing data with precipitation being the binary target variable.

Deep Learning Models	Accuracy %	Recall	F1-Score
Random	61.68	61.33	62.58
Margin	60.57	64.08	60.32
QUIRE	63.45	67.45	61.78
DFAL	65.87	63.45	65.43
Core-Set	59.87	61.48	57.56
TAL	**67.45**	**69.21**	**65.56**

TABLE I: Median performance during the first 100 number of queries on each of the three datasets

Table I shows the classification accuracy during the first 100 number of queries on each of the three datasets. Random and Margin approaches tend to yield decent performance when the number of queries is minimal. But, as the number of queries increases, this simple approach loses its edge and often is not as effective as the other active learning approaches. Both Core-set and DBAL are performing well at the beginning of the learning stage. As the number of queries increases, we observe their performance become less competitive. The performance of QUIRE is mixed. It works well on RAUs and KenCentralMet datasets, but performs less competitive on MeteoNet. We attribute the this to the fact that unlabeled data structures may not be consistent. TAL is the most competative method among the compared methods. This is attributed to its simple yet effective hybrid strategy of actively selecting most effective training samples from the distribution.

Deep Learning Models	Accuracy %	Recall	F1-Score
QUIRE	63.45	67.45	61.78
Pre-trained QUIRE	63.45	67.45	61.78
DFAL	65.87	63.45	65.43
Pre-trained DFAL	65.87	63.45	65.43
Core-Set	59.87	61.48	57.56
Pre-trained Core-Set	59.87	61.48	57.56
TAL	67.45	69.21	65.56
Pre-trained TAL	**67.45**	**69.21**	**65.56**

TABLE II: Median transfer learning performance during the first 100 number of queries

In order to maintain a homogeneous transfer, 6 features were considered from the 3 datasets with precipation as the

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

[3]https://meteonet.umr-cnrm.fr/

[4]https://meteo.go.ke/

target variable for all sets. This way, leveraging the similarity between the datasets was possible without further pairwise comparison of the data. MeteoNet was selected as the picked as source dataset while RAUS and KenCentralMet as target datasets. The result of two transfer tasks were recorded as presented in table II. The results shows pre-trained models leverage on target task better than un-trained models with TAL taking a competitive lead on performance.

V. Conclusion

In this paper, we proposed a novel deep Transfer Active Learning method for time series classification. We showed that its possible to use deep learning methods to discovery the discriminative patterns of the multivariate time series data. Then we defined a heuristic combination of informativeness and representative metrics for selecting training samples. Further we demonstrated the relevance of model skill transfer in homogeneous settings. In the experiment, we validated our method on a three time series datasets. The results showed that our proposed hybrid method is most competitive. In future, we plan to perform an adaptive transfer active learning on multivariate time series classification.

References

[1] Cook, D., Feuz, K.D. and Krishnan, N.C., Transfer learning for activity recognition: A survey. Knowledge and information systems, vol. 36(3), pp.537–556, September 2013.

[2] Kornblith, S., Shlens, J. and Le, Q.V., Do better imagenet models transfer better?. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2661–2671 2019.

[3] Raina, R., Battle, A., Lee, H., Packer, B. and Ng, A.Y., June. Self-taught learning: transfer learning from unlabeled data. In Proceedings of the 24th international conference on Machine learning, pp. 759–766), 2007.

[4] Settles, B., Active learning literature survey, 2009.

[5] Gal, Y., Islam, R. and Ghahramani, Z., July. Deep bayesian active learning with image data. In International Conference on Machine Learning, pp. 1183–1192, 2017.

[6] Fu, Y., Zhu, X. and Li, B., A survey on instance selection for active learning. Knowledge and information systems, 35(2), pp.249–283, 2013.

[7] Langkvist, M., Karlsson, L. and Loutfi, A., A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters, Vol. 42, pp.11–24, 2014.

[8] Pascanu, R., Mikolov, T. and Bengio, Y., Understanding the exploding gradient problem. CoRR, abs/1211.5063, vol. 2(417), 2012.

[9] Pascanu, R., Mikolov, T. and Bengio, Y., May. On the difficulty of training recurrent neural networks. In International conference on machine learning, pp. 1310–1318, 2013.

[10] Gamboa, J.C.B., Deep learning for time-series analysis. arXiv preprint arXiv:1701.01887, 2017.

[11] Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L. and Muller, P.A., Deep learning for time series classification: a review. Data mining and knowledge discovery, Vol. 33(4), pp.917–963, 2017.

[12] Le Guennec, A., Malinowski, S. and Tavenard, R., September. Data augmentation for time series classification using convolutional neural networks. In ECML/PKDD workshop on advanced analytics and learning on temporal data, 2016.

[13] Ziat, A., Delasalles, E., Denoyer, L. and Gallinari, P., November. Spatio-temporal neural networks for space-time series forecasting and relations discovery. In 2017 IEEE International Conference on Data Mining, pp. 705–714, 2017.

[14] Nweke, H.F., Teh, Y.W., Al-Garadi, M.A. and Alo, U.R., Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. Expert Systems with Applications, Vol. 105, pp.233–261, 2018.

[15] Che, Z., Cheng, Y., Zhai, S., Sun, Z. and Liu, Y., November. Boosting deep learning risk prediction with generative adversarial networks for electronic health records. In 2017 IEEE International Conference on Data Mining, pp. 787–792, 2017.

[16] Morales, F.J.O. and Roggen, D., September. Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations. In Proceedings of the ACM International Symposium on Wearable Computers, pp. 92–99, 2016.

[17] Khan, M.A.A.H., Roy, N. and Misra, A., March. Scaling human activity recognition via deep learning-based domain adaptation. In IEEE international conference on pervasive computing and communications, pp. 1–9, 2018.

[18] Wang, J., Chen, Y., Hu, L., Peng, X. and Philip, S.Y., March. Stratified transfer learning for cross-domain activity recognition. In IEEE international conference on pervasive computing and communications, pp. 1–10, 2018.

[19] Malhotra, P., TV, V., Vig, L., Agarwal, P. and Shroff, G., TimeNet: Pre-trained deep recurrent neural network for time series classification. arXiv preprint arXiv:1706.08838, 2017.

[20] Kashiparekh, K., Narwariya, J., Malhotra, P., Vig, L. and Shroff, G., July. ConvTimeNet: A pre-trained deep convolutional neural network for time series classification. In International Joint Conference on Neural Networks, pp. 1–8, 2019.

[21] Hu, Q., Zhang, R. and Zhou, Y., Transfer learning for short-term wind speed prediction with deep neural networks. Renewable Energy, Vol. 85, pp.83–95, 2016.

[22] Banerjee, D., Islam, K., Xue, K., Mei, G., Xiao, L., Zhang, G., Xu, R., Lei, C., Ji, S. and Li, J., A deep transfer learning approach for improved post-traumatic stress disorder diagnosis. Knowledge and Information Systems, Vol. 60(3), pp.1693–1724, 2019.

[23] Serrà, J., Pascual, S. and Karatzoglou, A., Towards a Universal Neural Network Encoder for Time Series. In CCIA, pp. 120–129, 2018.

[24] Lewis, D.D. and Gale, W.A., A sequential algorithm for training text classifiers. In SIGIR'94, pp. 3–12, Springer, 1994.

[25] Sugiyama, M. and Nakajima, S., Pool-based active learning in approximate linear regression. Machine Learning, Vol. 75(3), pp.249–274, 2009.

[26] Ranganathan, H., Venkateswara, H., Chakraborty, S. and Panchanathan, S., Deep active learning for image classification. In IEEE International Conference on Image Processing, pp. 3934–3938), 2017.

[27] Hao, Y., Chen, Y., Zakaria, J., Hu, B., Rakthanmanon, T. and Keogh, E., Towards never-ending learning from time series streams. In Proceedings International Conference on Knowledge discovery and data mining, pp. 874-882, 2013.

[28] He, G., Duan, Y., Li, Y., Qian, T., He, J. and Jia, X., Active learning for multivariate time series classification with positive unlabeled data. In IEEE 27th International Conference on Tools with Artificial Intelligence, pp. 178-185), 2017.

[29] 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, 2017.

[30] Huang, S.J., Jin, R. and Zhou, Z.H., Active learning by querying informative and representative examples. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 36(10), pp.1936–1949, 2014.

[31] Ducoffe, M. and Precioso, F., Adversarial active learning for deep networks: a margin based approach. arXiv preprint arXiv:1802.09841, 2018.

[32] Sener, O. and Savarese, S., Active learning for convolutional neural networks: A core-set approach. arXiv preprint arXiv:1708.00489, 2017.

[33] Wang, Z., Yan, W. and Oates, T., 2017, May. Time series classification from scratch with deep neural networks: A strong baseline. In International joint conference on neural networks, pp. 1578-1585, 2017.