Utilizing Unlabeled Data to Detect Electricity Fraud in AMI: A Semisupervised Deep Learning Approach

Tianyu Hu, Student Member, IEEE, Qinglai Guo, Senior Member, IEEE, Xinwei Shen, Member, IEEE, Hongbin Sun, Fellow, IEEE, Rongli Wu, and Haoning Xi

Abstract—As nontechnical losses in power systems have recently become a global concern, electricity fraud detection models attracted increasing academic interest. The wide application of smart meters has offered more possibility to detecting fraud from user’s consumption patterns. However, the performances of existing consumption-based fraud detection models are still not satisfactory enough for practice, partly due to their limited ability to handle high-dimensional data. In this paper, a deep-learning-based model is developed for detecting electricity fraud in the advanced metering infrastructure, namely, the multitask feature extracting fraud detector (MFEFD). The deep architecture has brought MFEFD a powerful ability to handle high-dimensional input, through which consumption patterns inside load profiles can be effectively extracted. Another challenge is that the insufficiency of labeled data has restricted the generalization of existing models since they are mostly based on supervised learning and labeled data. MFEFD is trained in a semisupervised manner, in which multitask training was implemented to combine the supervised and unsupervised training, so that both the knowledge from unlabeled and labeled data can be effectively extracted. MFEFD’s high detection performance, robustness, privacy preservation, and practicability.

Index Terms—Deep learning, electricity fraud detection, multitask learning, nontechnical losses (NTL), semisupervised training.

I. INTRODUCTION

L osses in power system can mainly be categorized into two groups: technical losses and nontechnical losses (NTL) [1]. Technical losses are caused by network resistance or energy transformation process among different forms, which are predictable but inevitable. NTL, namely, commercial losses, are mainly caused by the illegal or dishonest usage of electricity for the purpose of reducing payments, namely, electricity fraud. NTL have recently become a global concern: the total NTL all over the world in 2014 is estimated to reach $85.3 billion [2], and the commercial losses ratios were very high, especially in emerging markets, such as India from 20% to 40%, China at 10%, Brazil from 0.5% to 25% [3]. Electricity fraud has increased the utilities’ expenses (they were ultimately paid by customers through higher electricity prices) as well as the burden and risk of distribution systems [43]. Thus, eliminating the electricity fraud will benefit both customers and utilities.

Electricity fraud can arise from various positions in a distribution system [4].
1) Feeders, e.g., connecting throw-up on a distribution feeder or a transformer.
2) Advanced metering infrastructure (AMI). Recently, the wide application of smart meters and AMI has improved the variety and resolution of available measurements, but it has also made the power system exposed to more attacks. These attacks in AMI have opened new ways for electricity fraud, and they can be launched on various layers [5], [6]: 1) cyber layer, e.g., modifying the firmware or storage of meters, intercept/alter communications, and exhausting memory; 2) physical layer, e.g., reversing, disconnecting or bypassing the meter, and breaking into the meter; and 3) data layer, e.g., reporting zero/negative consumption, cut the report by a given percentage, removing large appliances from measurement, and altering load profiles. In summary, the objective of these attacks in AMI is tampering the load profiles [6].
3) Utility, e.g., arranging billing irregularity help by internal employees, and so on.

In this paper, the focus is put on detecting the electricity fraud only arising in AMI and the fraud arising from feeders or irregularity help by utility employees is beyond this paper’s scope.

In the past, site inspection was the only way to detect electricity fraud. The electricians went to each resident community in person to check the status and connection of each meter, so as to ensure that meters can correctly record the consumption. Recently, developed electricity fraud detection solutions can roughly be categorized into two groups: hardware solutions and nonhardware solutions. Hardware solutions resort to specific metering hardware, infrastructure or equipment to detect fraud, such as specific processor architectures [7], radio frequency identification tags [8], harmonic signal generator [9], and high-frequency signal generators [10]. Different from hardware solutions, nonhardware solutions have instead
put more attention on algorithms or classifiers which can infer the happening or probability of fraud, e.g., state-based models [11]–[13], game-theory-based models [14], [15], and classification-based models [6], [17], [18]. Comprehensive reviews on the state of the art of electricity fraud detection can be found in [4], [6], and [16].

In general, the detection performance of hardware solutions can be better than nonhardware ones, but in the meantime, extra costs are introduced due to the arrangement of this specialized hardware. Thus, before putting a hardware solution into practical use, one might need to make sure that the gains from the resultant improvement of fraud detection accuracy are greater than the costs from this hardware, which has limited the deployment of these hardware solutions to some extent. While for nonhardware solutions, low costs and use of resources are involved [4], which indicates a higher potential for practice. From the perspective of model structures, existing nonhardware solutions can be further categorized into aforementioned state-based, game-theory-based, and classification-based models, as is presented in [4], [6], and [16]. While from the perspective of used data, an alternative categorization can be done, which divides these nonhardware solutions into consumption-based ones and nonconsumption-based ones. Consumption-based solutions are based on the analysis of users’ electricity consumption patterns, and they only need the load profiles of customers and transformers as input. Since the payments for electricity are obtained from load profiles in the billing system, tampering load profiles are actually the ultimate goal of electricity fraud in AMI. Thus, theoretically the existing of fraud and its type can be detected through detailed analysis on the recorded load profiles or consumption patterns, and this is exactly the foundation of most consumption-based fraud detection models. Recently, benefiting from the wide application of smart meters, the resolution of measured load profiles has been greatly improved, which has promoted the emergence of many consumption-based solutions, including peer-to-peer computing (P2P) [17], support vector machines (SVMs) [6], [18], ensemble model (EM) [19], wavelets and classifier combination (WCC) [20], and so on. While different from consumption-based solutions, nonconsumption-based ones need other information beyond load profiles, which can be current [11], power-quality measurements [21], reactive energy and power factor [22], credit worthiness rating [1], number of persons and heavy appliances [23], synchronous voltage and reactive power [13], synchronous line segment current [12], and so on. In general, automatic measuring of consumption is the most fundamental function of smart meters and AMI. Since load profiles are the only measurements that a billing system need, most existing smart meters are not designed to measure other information beyond consumptions, e.g., current, reactive power. Therefore, compared with nonconsumption-based fraud detection models, consumption-based ones have greater potential for practice at the present stage since they can be directly embedded into existing billing systems considering that they share the same data (load profiles).

Although aforementioned consumption-based models are innovative and remarkable, their performances are still not satisfactory enough for practice. For example, they have relatively low (DRs) or high false positive rates, which partly results from their limited ability to handle high-dimensional data. Indeed, as aforementioned, the wide application of smart meters and AMI has improved the variety and resolution of available measurements, but it has also made the power system exposed to more various attacks. Compared to conventional electromagnetic meters which has a simple structure, electronic smart meters, and assorted communication equipments have made it possible for fraudulent users to launch attacks on the whole cyber-physical system, including physical layers, cyber layers, and data layers [5]. The significant expansion of potential fraud approaches has made fraudulent user’s consumption patterns much more complicated [5], which thus has put forward a higher requirement on the time resolution of the data to use for fraud detection since data with high resolution bring more details. For instance, if one wants to build a fraud detection model which takes one week’s load profile with 15-min resolution as input (15 min is a common resolution for smart meters), then the dimension of the input will be 672, which is already very high. Unfortunately, most existing consumption-based fraud detection models are still based on traditional classifiers, such as SVM and decision tree, and it has been revealed that these traditional methods have limited generalization ability on high-dimensionality problems due to their insufficiency to learn complicated functions in high-dimensional spaces [24], which is also called curse of dimensionality. Recently, deep learning has shown very promising performances on several high-dimensional problems, which have resisted the best attempts of the machine learning community for many years, such as image processing, speech recognition, and video processing, which is facilitated by the ability of deep learning to discover intricate structures in high-dimensional data [25], [26]. To overcome the challenge of high dimensionality in electricity fraud detection, a novel deep-learning-based fraud detection model is developed in this paper, namely, the multitask feature extracting fraud detector (MFEFD). MFEFD consists of several deep neural networks which serve for feature extraction and discrimination, respectively. Due to the deep architecture and high capacity of MFEFD, consumption patterns inside the load profiles of honest and fraudulent users can be effectively extracted, thus the performance of this fraud detection approach is significantly improved.

The severe insufficiency of labeled samples is another challenge to overcome for electricity fraud detection. For example, as existing models are mostly based on supervised learning classifiers, such as SVM and decision tree, enough labels indicating whether fraud exists are needed for training them. Unfortunately, labeled samples are rather scarce compared to
unlabeled ones. Due to the high price of labor, large-scale site inspection by electricians for electricity fraud detection is rarely executed. Thus, for most customers, whether they have committed electricity fraud crimes or not are unknown, producing numerous unlabeled samples. Great disparity of the amounts of unlabeled and labeled samples has brought nonnegligible problems: on the one hand, compared with unlabeled samples, insufficiency of labeled ones has limited the generalization of existing fraud detection models (the model can be overfitted when data set is small); on the other hand, huge amount of unlabeled samples which may contain useful information are left unused, resulting in a great waste of data. The proposed MFEFD is trained in a semisupervised manner, and it can make good use of both unlabeled and labeled data simultaneously through unsupervised representative learning and supervised training, respectively. Unsupervised representative learning is a fast-developing training technology, in which a multilayer neural network is trained to reproduce the input, such that the hidden layers can learn a good representation of the input (namely, the feature), and then the learned representation or feature can be further applied in other tasks. Compared to most existing consumption-based fraud detection models which only use labeled data, adding unlabeled data into training set has greatly expanded the information that MFEFD models which only use labeled data, adding unlabeled data into training set has greatly expanded the information that MFEFD can make use of, which has made a significant improvement of detecting performance, statistical strength, and generalization.

In semisupervised learning, how to properly combine the supervised and unsupervised training becomes a new problem. An existing study has used a semisupervised model called transductive SVM (TSVM) [27] to detect electricity fraud, but it has difficulty extending to large amounts of unlabeled data [28], and currently, no algorithm is known to efficiently find a globally optimal solution of TSVM [29]. Other approaches to combine supervised and unsupervised learning include self-training, generative models, cotraining, and graph-based models. For self-training and cotraining, unlabeled samples are labeled and merged into the training set, but the mislabeling of unlabeled data may bring accumulative error. Generative models need the assumption of data distribution, which may cause mismatch from reality. For graph-based models, how to build a proper graph is always a hard problem. Specifically, for neural-network-based models, in addition to these four approaches, two more alternatives have been proposed recently to combine supervised and unsupervised learning, namely, two-stage learning and multitask learning. For two-stage learning, unsupervised and supervised training are implemented successively and separately, which are, namely, unsupervised pretraining and supervised fine-tuning [30]. While for multitask training, unsupervised and supervised training are implemented simultaneously in an alternate manner. Multitask training is a training strategy aiming at combining multiple training tasks, in which the same network is shared among different but relevant training processes for obtaining better generalization and stronger statistical strength. In [31], a deep neural network which had combined supervised classifiers and unsupervised autoencoders through multitask training was proposed for compact document representation. In [32], supervised classification and unsupervised auxiliary tasks were combined through multitask training for predicting interactions between proteins. In [33], semisupervised recursive autoencoders which had combined both supervised and unsupervised training were built for predicting sentiment distributions. Impressed by the superior performances of multitask training in these works, in the proposed MFEFD, we have combined supervised and unsupervised learning through multitask training strategy, so that the MFEFD can not only obtain knowledge from both training processes but also achieve a satisfactory balance between them.

The main contributions of this paper can be summarized as follows.

1) A deep-learning-based model is developed for detecting electricity fraud in AMI, namely, MFEFD. Case studies on different data sets have demonstrated the superiority of MFEFD from the perspectives of high detection performance, satisfactory privacy preservation, high robustness, and low data requirement.

2) The MFEFD is trained in a semisupervised manner, and it can make good use of both unlabeled and labeled data simultaneously. Alternate multitask training is implemented to combine the supervised and unsupervised training of MFEFD, which has properly integrated and balanced the knowledge obtained from supervised and unsupervised learning, equipping MFEFD with high generalization.

The rest of the paper is organized as follows. Section II introduces the MFEFD. Section III introduces the training algorithm of MFEFD. Case studies are introduced in Section IV. Section V draws the conclusion.

II. MULTITASK FEATURE EXTRACTING FRAUD DETECTOR

The structure of MFEFD and its components are introduced in this section, as shown in Fig. 1.

Overall, MFEFD takes the recent 7 days’ load profiles as input and outputs the probabilities of committing fraud crime.
It is joint by two parts: seven feature extraction networks (FENs) and one discrimination network (DN), which are all multilayer neural networks. FEN takes an intraday load profile as input and outputs the extracted features. After feature extraction through FENs, the features are then merged into a vector. Finally, DN takes in the merged vector and outputs the probability of committing fraud crime.

A. Step 1: Feature Extraction Based on Semisupervised Learning

In the first step, FENs of MFEFD map raw load profiles into features. The feature is an abstract and simplified expression of the raw input. It should retain the key information inside the input which may influence the final judgment, while redundant information is excluded. The remove of redundancy results in an information reduction from load profiles to features, which is so-called dimension reduction. In general, features are generated through the feature extraction processes, such as principal component analysis, independent component analysis, and wavelet transform [34]. In MFEFD, multilayer neural networks are utilized to map load profiles to features, which are thus referred as FEN.

Why do we choose to produce weekly consumption patterns? Because we assume that the users’ electricity usage behaviors are weekly periodic, which is also a well-known fact. That is, during a relatively short period such as one month, users’ behaviors in the same days of each week are similar, but behaviors in different days of each week are different. This assumption is rather reasonable, because people’s life is essentially weekly divided, and during a short period, similar things are done in the same days of the week, such as working, attending school, cleaning, and so on. Given this assumption, load profiles of the recent 7 days can satisfactorily represent the user’s current electricity usage regularity. Thus, MFEFD takes load profiles of the recent 7 days as input.

B. Step 2: Fraud Discrimination Based on Supervised Learning

After feature extraction, the second step is to map features to final judgment, namely, the discrimination process. In MFEFD, the discrimination process is also executed by a multilayer neural network, which is referred as DN.

Note that FENs for different days of the week are built and trained independently in MFEFD, such that they can focus more on the difference between honest and fraudulent users rather than the difference among different days of the week. As the only difference among FENs for different days of the week is their training set, in the rest of this paper, they are not differentiated.

Details about how to implement feature extraction and fraud discrimination will be demonstrated in Section III.

III. TRAINING OF MFEFD

The training of MFEFD is divided into two steps. In the first step, FENs are trained through semisupervised training using both labeled and unlabeled samples. In the second step, DN and the whole MFEFD are trained through supervised training on labeled samples.

A. Semisupervised Training of FENs

During the semisupervised training of FENs, both supervised and unsupervised training are implemented. Unsupervised training of FEN only needs intraday load profiles, thus both labeled and unlabeled load profiles are utilized. Supervised training of FEN is implemented using labeled samples and their corresponding labels. To train FENs in both supervised and unsupervised manners, we trained FENs according to the following mother–son strategy: first, replace part of an existing model (we call it mother model) with FEN; then train the mother model such that the FEN inside it can be trained in the meantime.

1) Unsupervised Training Based on Denoising Autoencoder: Denoising autoencoder (DAE) is a multilayer neural network which can be used for unsupervised feature extraction. Its mathematical essence is using the noised input to reproduce the initial input to the greatest extent (so-called denoising), such that in the meantime features hidden inside the initial input can be captured by hidden layers. Given intraday load profiles $X = \{x_1, x_2, \ldots, x_N\}$, $x_i \in \mathbb{R}^d$ as the initial input of DAE and $P_{\text{noise}}$ as the probability that each element of $x_i$ being set to 0, then noising vectors $K = \{k_1, k_2, \ldots, k_N\}$, $k_i = [k_{i1}, k_{i2}, \ldots, k_{id}]$ can be generated, where $k_{ij} \in \{0, 1\}$, $P(k_{ij} = 0) = P_{\text{noise}}$, $j = 1, 2, \ldots, d$. The noised input $X' = \{\hat{x}_1', \hat{x}_2', \ldots, \hat{x}_N'\}$, $\hat{x}_i' \in \mathbb{R}^d$ can be obtained by

$$\hat{x}_i' = x_i \circ k_i \quad (1)$$

where $\circ$ represents the Hadamard product of two vectors. $X'$ is then mapped into features as $F' = \{f_1', f_2', \ldots, f_N'\}, f_i' \in \mathbb{R}^m$ through several hidden layers, where $m$ is the dimension of features

$$f_i' = s(\bar{w}_H \cdot s(\cdots [s(\bar{w}_2 \cdot [s(\bar{w}_1 \cdot \hat{x}_i' + b_1) + b_2]) + b_2] + \cdots) + b_H) \quad (2)$$

where $\bar{w}$ and $b$ are weights and bias, $s$ is the activation function, e.g., sigmoid, tanh, and rectified linear unit (ReLU). $H$ is the number of hidden layers. Afterward, $F'$ is then mapped into the output as $X'' = \{\hat{x}_1'', \hat{x}_2'', \ldots, \hat{x}_N''\}$, $\hat{x}_i'' \in \mathbb{R}^d$, which has the same dimension as $X$

$$\hat{x}_i'' = s(\bar{w}_H \cdot s(\cdots [s(\bar{w}_2 \cdot [s(\bar{w}_1 \cdot \cdot \cdot + b_1)] + b_2] + \cdots) + b_H) \quad (3)$$

The objective of DAE is to reproduce the initial input through minimizing the following loss:

$$\text{Loss}_{\text{DAE}}^i = \|x_i - \hat{x}_i''\|^2 \quad (4)$$

In the unsupervised training of FEN, we take DAE as mother model and replace its former part with FEN, as shown in Fig. 2. As no labels are needed for training DAE, both labeled and unlabeled load profiles can be utilized.

2) Supervised Training Based on Deep Siamese Network: Deep Siamese network (DSN) [35] can be applied as an effective feature extractor trained in a supervised manner. It is composed of two parts: shared feature extractors and a distance measurer. Given the labels of load profile $X$ as $Y = \{y_1, y_2, \ldots, y_N\}$ (where $y_i \in \{0, 1\}$, $y_i = 1$ means that
**Fig. 2.** Structure of DAE.

**Fig. 3.** Structure of (a) DSN and (b) DDN.

$x_i$ is fraudulent), first, the shared feature extractor takes two load profiles as input (denoted as $x_i$ and $x_j$) and outputs the features

$$f_i = s\{w_H \cdot s\{\cdots s\{w_2 \cdot [s(w_1 \cdot x_i + b_1)] + b_2\} \cdots \} + b_H\}$$

$$f_j = s\{w_H \cdot s\{\cdots s\{w_2 \cdot [s(w_1 \cdot x_j + b_1)] + b_2\} \cdots \} + b_H\}. \tag{5}$$

Then, distance measurer outputs the Euclidean distance of features

$$\hat{D}_{i,j} = \|f_i - f_j\|_2 \tag{7}$$

where $\hat{D}_{i,j}$ is the distance. The expected output of DSN is defined as follows:

$$D_{i,j} = \begin{cases} 0, & \text{if } y_i = y_j \\ 1, & \text{if } y_i \neq y_j \end{cases}. \tag{8}$$

The training of DSN is to minimize the mismatch between $\hat{D}_{i,j}$ and $D_{i,j}$. Contrastive loss [35] is used as the loss function

$$\text{Loss}^{i,j}_{\text{DSN}} = (1 - D_{i,j}) \cdot \hat{D}_{i,j} + D_{i,j} \cdot \max[0, 1 - \hat{D}_{i,j}]^2. \tag{9}$$

In the supervised training of FEN, we take DSN as mother model and replace its shared feature extractor with FEN, as shown in Fig. 3(a).

3) **Supervised Training Based on Day Discrimination Network:** Another way to train an FEN in a supervised manner is constructing a day DN (DDN) as the mother model of FEN. The DDN is constructed by putting one extra neuron on the top of the FEN which indicates fraud or not, as is illustrated in Fig. 3(b). In DDN, the intraday load profile is first mapped into a feature vector by the FEN

$$f_i = s\{w_H \cdot s\{\cdots s\{w_2 \cdot [s(w_1 \cdot x_i + b_1)] + b_2\} \cdots \} + b_H\}. \tag{10}$$

Then, the feature vector is further mapped into a predicted label (scalar) by the neuron on the top

$$\hat{y}_i = \sigma(w_D \cdot f_i + b_D) \tag{11}$$

where $\sigma$ represents the sigmoid function. The loss of DDN from $\{x_i, y_i\}$ is defined as the cross-entropy loss

$$\text{Loss}^{i}_{\text{DDN}} = -[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]. \tag{12}$$

4) **Multitask Training of FEN:** Both supervised and unsupervised training have shortcomings. For example, the objective of DAE is to extract the main information inside load profiles, but there is no guarantee that the removed information is dispensable; the objective of DSN and DDN is to find the key information which is decisive for determining the labels, but overfitting tends to happen if only supervised learning is implemented, because only a few labeled samples are available. Therefore, supervised and unsupervised training of FEN are combined in this paper to overcome the above-mentioned shortcomings. For example, FEN is trained through alternate multitask learning using both DAE and DSN/DDN. In this paper, two combinations are considered for building FEN, namely, DAE + DSN and DAE + DDN. For conciseness, we only take DAE + DSN as the example to demonstrate the detailed training algorithm of FENs in the following text, and the cases for DAE + DDN can be obtained by replacing DSN with DDN.

Multitask learning is a training strategy through which the same network is shared and trained by different but relevant tasks so that the shared network can simultaneously obtain knowledge from all of them. The loss function of FEN during multitask training is as follows:

$$\text{Loss}_{\text{FEN}} = \frac{1}{L + U} \sum_{i=1}^{L+U} \text{Loss}^{i}_{\text{DAE}} + \frac{\mu}{\sum_{i,j=1}^{L} 1} \sum_{i,j=1}^{L} \text{Loss}^{i,j}_{\text{DSN}}. \tag{13}$$

where labeled load profiles and their labels are denoted as $(x_i, y_i), i = 1, \ldots, L$, unlabeled load profiles are denoted as $x_i, i = L + 1, \ldots, L + U$. Without loss of generality, $\mu$ was set as 1.0. Stochastic gradient descent is used during multitask training. It first partitions the whole training set evenly into many minibatches. Then, the trainable parameters are updated based on the gradient of loss on each minibatch. During multitask training of FEN, the training of DAE and DSN is implemented alternately, whose pseudocode is shown in Algorithm 1. As FENs are trained in both supervised and unsupervised manners, they can not only preserve the main information inside load profiles but also catch the key features which are decisive for fraud detection.
Algorithm 1 Alternate Multi-Task Training of the FEN

**Input:** $x_i, i = L + 1, \ldots, L + U$ the unlabeled intra-day load profiles $(x_i, y_i), i = 1, \ldots, L$ the labeled intra-day load profiles and labels $M_{E_{DAE}}$ the maximal epochs of training DAE $M_{E_{DSN}}$ the maximal epochs of training DSN

**Output:** $\theta$ parameters of the FEN

1. Partition all the load profiles into $M_u$ mini-batches
2. Partition labeled load profiles and labels into $M_l$ mini-batches
3. Initialize the FEN and its parameters $\theta$;
4. Build a DAE which takes FEN as its former part
5. Build a DSN which takes FEN as its shared feature extractor
6. Initialize count variables $C_{DAE} \leftarrow 0, C_{DSN} \leftarrow 0, c_{DAE} \leftarrow 0, c_{DSN} \leftarrow 0$
7. Initialize flag variables $flag_{DAE} \leftarrow 1, flag_{DSN} \leftarrow 1$
8. While $C_{DAE} \leq M_{E_{DAE}}$ or $C_{DSN} \leq M_{E_{DSN}}$ do
   9. If $flag_{DAE} = 1$ do
      10. Copy $\theta$ to the former part of DAE
      11. Get average loss of DAE on the $c_{DAE}$th mini-batch as $Loss_{DAE}^{c_{DAE}}$
      12. Make a gradient step to minimize $Loss_{DAE}^{c_{DAE}}$
      13. $c_{DAE} \leftarrow c_{DAE} + 1$
      14. If $c_{DAE} = M_u$ do
         15. $c_{DAE} \leftarrow 0$
      16. $C_{DAE} \leftarrow C_{DAE} + 1$
      17. Get the average loss of DAE on validation set as $Loss_{DAE}^{v_{DAE}}$
      18. If $Loss_{DAE}^{v_{DAE}}$ hasn’t decreased during the latest 10 epochs do
         19. $flag_{DAE} \leftarrow 0// early-stopping to avoid over-fitting
      20. End if
      21. If $C_{DAE} = M_{DAE}$ do
         22. $flag_{DAE} \leftarrow 0// reaching the maximal epochs of DAE
      23. End if
      24. End if
   25. End if
   26. If $flag_{DSN} = 1$ do
      27. Copy $\theta$ to the shared feature extractor of DSN
      28. Get average loss of DSN on the $c_{DSN}$th mini-batch as $Loss_{DSN}^{c_{DSN}}$
      29. Make a gradient step to minimize $\mu \cdot Loss_{DSN}^{c_{DSN}}$
      30. $c_{DSN} \leftarrow c_{DSN} + 1$
      31. If $c_{DSN} = M_l$ do
         32. $c_{DSN} \leftarrow 0$
      33. $C_{DSN} \leftarrow C_{DSN} + 1$
      34. Get the average loss of DSN on validation set as $Loss_{DSN}^{v_{DSN}}$
      35. If $Loss_{DSN}^{v_{DSN}}$ hasn’t decreased during the latest 10 epochs do
         36. $flag_{DSN} \leftarrow 0// early-stopping to avoid over-fitting
      37. End if
      38. End if
      39. If $C_{DSN} = M_{DSN}$ do
         40. $flag_{DSN} \leftarrow 0// reaching the maximal epochs of DSN
      41. End if
      42. End if
      43. End while
      44. Return $\theta$

B. Supervised Training of DN and the Whole MFEFD

Multitask training has only brought knowledge to FENs, leaving DN still untrained after random initialization. Now if we train the whole MFEFD directly, large gradient updates triggered by the randomly initialized DN will wreck the learned parameters in FENs. Thus, to protect FENs, a two-step training procedure is applied to train DN and the whole MFEFD after multitask training, namely, domain adaption and fine-tuning.

1) Training DN Through Domain Adaption: Domain adaption can protect part of the network through freezing its parameters. After multitask training, FENs are first frozen and thus only DN is trainable so that the large gradient updates triggered by DN will not damage FENs. Through domain adaption, DN can be trained into an appropriate status while the knowledge in FENs can be completely reserved.

2) Fine-Tuning the Whole MFEFD: After domain adaption, MFEFD is finally trained with all the parts trainable, namely, the fine-tuning. As domain adaption has trained MFEFD into a satisfactory status, fewer epochs and less learning rate are needed during fine-tuning.

Both domain adaption and fine-tuning of MFEFD are supervised training, thus only labeled load profiles and their labels are used.

IV. CASE STUDIES

Real-world-data-based case studies are presented in this section, which have shown that adding unlabeled samples into training set has greatly improved the performance of MFEFD. Caffe was used to simplify the coding works [36].

A. Evaluation Indexes

Two evaluation indexes are used in this paper, namely, false positive rate (FPR) and DR. Given the number of false positive (positive means fraudulent), false negative, true positive, and true negative samples as FP, FN, TP, and TN, respectively, then FPR and DR can be obtained through

$$FPR = \frac{FP}{TN + FP} \times 100\% \quad (14)$$

$$DR = \frac{TP}{TP + FN} \times 100\%. \quad (15)$$

B. Data Description

MFEFD is tested on the data set issued by Irish Smart Energy Trial (ISET) [37], which includes half-hourly smart...
meter measurements of over 5000 Irish families and businesses covering 535 days. As customers who participated in the trial agreed to take part in the research, it is a reasonable assumption that all samples come from honest users [6]. Among the 5000 families, we randomly chose 3000 out of them as unlabeled data. Then, for the left 2000 ones, we randomly chose 1000 of them as fraudulent samples and modified their intraday load profiles according to the fraudulent sample generation methods in [6]. For example, for each customer that was chosen to be a fraudulent sample, all his/her intraday load profiles were modified according to a fraudulent sample generation method which is randomly chosen from the following six.

Given an intraday load profile as \( x = \{x_1, x_2, \ldots, x_{48}\} \), then:

1) \( h_1(x_t) = ax_t, a = \text{random}(0.1, 0.8); \)
2) \( h_2(x_t) = \beta_t x_t \)

\[
\beta_t = \begin{cases} 
0 & \text{start\_time} < t < \text{end\_time} \\
1 & \text{else} 
\end{cases}
\]

\( \text{start\_time} = \text{random}(0.47-\text{min\_Off\_Time}) \)
\( \text{end\_time} = \text{start\_time} + \text{duration} \text{ min\_Off\_Time}\)

3) \( h_3(x_t) = \gamma_t x_t, \gamma_t = \text{random}(0.1, 0.8); \)
4) \( h_4(x_t) = \gamma_t \cdot \text{mean}(x), \gamma_t = \text{random}(0.1, 0.8); \)
5) \( h_5(x_t) = \text{mean}(x); \)
6) \( h_6(x_t) = x_{48-t}. \)

The physical interpretations of these six methods are as follows [6]: \( h_1() \) multiplies all the samples by the same randomly chosen value; using \( h_2() \) the smart meter does not send its measurements or sends zero for a random duration; \( h_3() \) multiplies each meter reading by a different random number; \( h_4() \) and \( h_5() \) orderly report a factor and the exact value of the average of readings over the day. \( h_6() \) reverses the order of readings. \( h_5() \) and \( h_6() \) represent attacks against load control mechanisms in which the price of electricity varies over different hours of the day; while the total amount of electricity usage stays the same, the usage is reported to happen during the low-tariff periods. In summary, the final proportions of unlabeled, fraudulent, and honest samples were 60%, 20%, and 20%, respectively. All the unlabeled samples were randomly divided into two parts: 2100 ones in the unlabeled training set and 900 ones in the unlabeled validation set. All the labeled samples were randomly divided into three parts: 1000 in the labeled training set, 300 in the labeled validation set, and 700 in the labeled testing set. DAE was trained using the load profiles in both unlabeled and labeled training sets. The labeled training set was also used for training DSN and DDN, domain adaption, and fine-tuning. The labeled testing set was used for performance evaluation. The unlabeled and labeled validation sets were used for judging whether over-fitting happened (see Algorithm 1 in Section III-A.4) as well as structure determination (see Sections IV-C.1 and IV-D.1).

### C. Multitask Training of FENs

1) **Determination of FEN's Structure**: Neural networks’ performance can be significantly influenced by their structures.

FENs for different days of the week were built independently, and their structures were determined through grid search strategy. To simplify the complexity of grid-searching, following roles were formulated.

1) \( s \) was set as ReLU to reduce the computation complexity.
2) all the layers in FENs were fully connected.

Then, all the remaining undetermined factors were chosen from given ranges: the number of layers in FENs from \( \{3, 4, 5, 6, 7\} \), the size of each layer from \( \{4, 8, 16, 32, 64\} \). Structures achieving the minimum loss on validation sets after multitask training were chosen as optimal structures. Other parameters were chosen by experience or by try and error, namely, learning rate was set as 0.001, \( M_{\text{DSN}} \) and \( M_{\text{DAE}} \) were both set as 1000.

2) **Performance of Multitask Training**: To test FENs’ feature extraction ability, t-distributed stochastic neighbor embedding (t-SNE) [38] was applied to visualize load profiles and their extracted features. T-SNE can map high-dimensional vectors into low-dimension ones, making the structure of the data set more intuitive. Here, all the labeled load profiles in the training set and their features extracted by FENs were mapped to two-dimension vectors by t-SNE, and they were then plotted in Fig. 4. For all the images displayed in Fig. 4, the red points represent benign samples while the blue ones mean fraudulent. As shown in Fig. 4(a-1)–(g-1), t-SNE of load profiles with different labels overlapped with each other, which means that the structure of load profiles as well as their relationships were not clear. However, as shown in Fig. 4(a-2)–(g-2) and (a-3)–(g-3), after multitask training, features with different labels can be mostly separated. This is because FENs had successfully captured the differences between honest and fraudulent load profiles, which were enabled by supervised training through taking DSN/DDN as mother model. In addition, the \( \text{Loss}_{\text{DAE}} \) of each FEN (see Algorithm 1) after multitask training is listed in Table I, which shows that all \( \text{Loss}_{\text{DAE}} \) were rather small, indicating that most information in the load profiles had been transferred into the features. Thus, in summary, multitask learning had brought both the merits of supervised and unsupervised training to FENs.

Moreover, as shown in Fig. 4, even after multitask training, some benign and fraudulent samples still overlapped. It means that, on the one hand, judging a user is benign or not just from one day’s load profile is not accurate enough, thus, in this paper, a week’s load profiles were used instead; on the other
hand, only FEN cannot give a satisfactory result for fraud detection, which thus verifies the necessity of DN.

D. Training DN and Fine-Tuning the Whole MFEFD

1) Determination of DN’s Structure: The structure of DN was also determined using grid search strategy, namely, the number of layers was chosen from \{1, 2, 4, 6\}, the size of each layer from \{16, 32, 64, 128\}. To simplify the complexity of grid-searching, following roles were formulated.

1) \(s\) was set as sigmoid.
2) all the layers in DN were fully connected.

As the inputs of MFEFD are load profiles of 7 successive days, for each consumer in the labeled set, totally 528 samples were assembled. After domain adaption and fine-tuning, the structure achieving the minimum cross-entropy loss on the validation set was chosen as the optimal structure of DN.

2) Training DN Through Domain Adaption: During domain adaption, learning rate was set as 0.001, maximal training epoch was set as 2000. Performance of the optimal-structure MFEFD on the testing set after domain adaption is presented in Table II, which shows that even without fine-tuning, domain adaption had equipped MFEFD with strong discriminating power.

3) Fine-Tuning MFEFD: During fine-tuning, learning rate was set as 0.0001, maximal training epoch was set as 1000. Performance of the optimal-structure MFEFD on the testing set after fine-tuning is presented in Table II. As shown in Table II, for DAE + DSN, fine-tuning had increased the DR by 4.87% and decreased FPR by 1.04%, while for DAE + DDN, fine-tuning had increased the DR by 5.24% and decreased FPR by 1.36%, which were significant improvements, indicating that fine-tuning had made FEN and DN cooperate better with each other.

E. Evaluate the Robustness of MFEFD

1) Evaluate the Robustness Against Sampling Rate Variation: The sampling rate of smart meters varies with different countries and utilities. For example, it varies from quarterly to hourly among different provinces in China. To evaluate the robustness and generalization of MFEFD, three more sampling rates were further tested, which were hourly, two-hourly, and four-hourly.

First, hourly, two-hourly, and four-hourly load profiles were generated by downsampling all the initial load profiles in the testing set through averaging on corresponding time periods; then MFEFD was tested using the downsampled samples and their labels for each sampling rate. As the dimension of FENs’ input is 48 (namely, half-hourly), all the downsampled load profiles must be upsampled before being sent to FENs. Three interpolation methods were used for upsampling, namely, the nearest, linear, and cubic spline interpolation. Performances of MFEFD are summarized in Table III, which

<table>
<thead>
<tr>
<th>MFEFD</th>
<th>DAE=DSN</th>
<th>DAE=DDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR, FPR</td>
<td>DR, FPR</td>
<td></td>
</tr>
<tr>
<td>After domain adaption</td>
<td>90.17% 6.18%</td>
<td>92.03% 6.92%</td>
</tr>
<tr>
<td>After fine-tuning</td>
<td>95.04% 5.14%</td>
<td>97.27% 5.56%</td>
</tr>
</tbody>
</table>

Fig. 4. T-SNE of load profiles and their features. (a)-(g) t-SNE for Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday, respectively. (*−1), (*−2), and (*−3) represent the t-SNE of intraday load profiles, features of DAE + DSN and features of DAE + DDN, respectively.
shows that MFEFD was rather robust against sampling rate variation, and cubic spline was the most suitable upsampling method in this case. For example, for sampling rates not lower than two-hourly, the DRs were all higher than 80%, while FPRs were all lower than 15% (using cubic spline). Considering that a smart meter’s sampling rate is usually higher than two-hourly, MFEFD’s performances were rather acceptable for practice. Moreover, as MFEFD’s superiority can still be guaranteed under low sampling rates, user’s privacy can be effectively protected [6].

2) Evaluate the Robustness Against Holidays: To evaluate the robustness of MFEFD against irregular consumption patterns caused by holidays, we took three important holidays in Ireland as examples and tested the performances of MFEFD on them. The three chosen holidays were New Year’s Day, St. Patrick’s Day, and Christmas Day. First, we picked out all the intraday load profiles of these holidays in the testing set and tested the performances of DDN on them. Then, we picked out all the weekly load profiles which contained these holidays in the testing set and tested the performances of MFEFD on them. The results are illustrated in Fig. 5, which shows that the irregularity caused by holidays did influence the performances of MFEFD. For example, both the DRs and FPRs as shown in Fig. 5 are worse than the ones evaluated on the whole testing set (shown in Table II). Among the three chosen holidays, irregularity caused by Christmas Day had the greatest impacts on fraud detection, which resulted in the lowest DRs and highest FPRs. However, on the whole, the performances of MFEFD on these holidays were still rather promising, namely, the DRs were all over 80% while FPRs all under 10%. This is probably because taking a weekly load profile as input had equipped MFEFD with satisfying robustness against the irregularity caused by a single day (such like a holiday), which can be inferred from the fact that DDN had performed much worse than MFEFD on these holidays. Thus, unlike MFEFD, only taking an intraday load profile as input had made DDN easily affected by a holiday, which resulted in worse performances.

F. Investigate the Effect of Unlabeled Data on MFEFD

To investigate the effect of unlabeled data on MFEFD, we built new MFEFDs and trained them using different numbers of unlabeled samples. Their performances on the testing set are presented in Fig. 6, which shows that a huge number of unlabeled samples had greatly improved MFEFD’s performance. For example, as the number of unlabeled samples increases, DR and FPR increases and decreases fast, respectively. This is because MFEFD, as a deep neural network essentially, is rather sensitive to the size of its training set. As enlarging training set is one of the most efficient ways to improve neural network’s performance and overcome overfitting, increasing training set size through adding in huge volume of unlabeled data had brought plenty of valuable information on the distribution of intraday load profiles to MFEFD, resulting in a significant improvement of MFEFD’s performance, statistical strength, and generalization.

G. Evaluate the Effectiveness of Multitask Training

To evaluate the effectiveness of alternate multitask training, four more training strategies on FEN were implemented: taking DAE + DSN as the example, two single-task strategies, namely, only-DAE and only-DSN; two pretraining fine-tuning strategies, namely, first DAE then DSN (DAE-DSN) and first DSN then DAE (DSN-DAE). All the other changeable factors are kept the same with multitask training, and results after fine-tuning are shown in Table IV.

As shown in Table IV, alternate multitask learning had the best performances among all the training strategies. For example, it had the lowest FPRs and highest DRs. That is because: 1) multitask training had brought FENs both the merits of supervised learning and unsupervised representative learning. Both the main and key information inside input can be effectively extracted, thus it outperformed single-task
TABLE IV
Evaluation of Multitask Training

<table>
<thead>
<tr>
<th></th>
<th>FEN Training</th>
<th>DAE=DSN</th>
<th>DAE=DDN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FPR</td>
<td>DR</td>
<td>FPR</td>
</tr>
<tr>
<td>Only-DAE</td>
<td>11.70%</td>
<td>88.28%</td>
<td>11.71%</td>
</tr>
<tr>
<td>Only-DSN/DDN</td>
<td>13.03%</td>
<td>85.54%</td>
<td>11.53%</td>
</tr>
<tr>
<td>DAE-DSN/DDN</td>
<td>9.17%</td>
<td>89.83%</td>
<td>9.82%</td>
</tr>
<tr>
<td>DSN/DDN-DAE</td>
<td>9.82%</td>
<td>91.91%</td>
<td>10.32%</td>
</tr>
<tr>
<td>Multi-task</td>
<td>5.14%</td>
<td>95.04%</td>
<td>5.56%</td>
</tr>
</tbody>
</table>

TABLE V
Comparison with Models Having the Same Data Requirements

<table>
<thead>
<tr>
<th></th>
<th>DAE=DSN</th>
<th>DAE+DDN</th>
<th>TSVM</th>
<th>EM</th>
<th>SVM</th>
<th>WCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR</td>
<td>95.04%</td>
<td>97.27%</td>
<td>90.21%</td>
<td>83.48%</td>
<td>80.54%</td>
<td>81.46%</td>
</tr>
<tr>
<td>FPR</td>
<td>5.14%</td>
<td>5.56%</td>
<td>9.38%</td>
<td>11.73%</td>
<td>10.47%</td>
<td>12.85%</td>
</tr>
<tr>
<td>DR</td>
<td>94.85%</td>
<td>98.45%</td>
<td>89.94%</td>
<td>84.20%</td>
<td>78.65%</td>
<td>80.73%</td>
</tr>
<tr>
<td>FPR</td>
<td>5.39%</td>
<td>5.43%</td>
<td>9.64%</td>
<td>11.09%</td>
<td>11.93%</td>
<td>12.69%</td>
</tr>
<tr>
<td>DR</td>
<td>93.78%</td>
<td>96.78%</td>
<td>89.29%</td>
<td>82.42%</td>
<td>79.37%</td>
<td>81.30%</td>
</tr>
<tr>
<td>FPR</td>
<td>6.02%</td>
<td>6.42%</td>
<td>10.17%</td>
<td>11.25%</td>
<td>10.01%</td>
<td>12.74%</td>
</tr>
<tr>
<td>Time</td>
<td>2.3 h</td>
<td>1.5 h</td>
<td>5.6 h</td>
<td>4.6 h</td>
<td>618.3 s</td>
<td>1.2 h</td>
</tr>
</tbody>
</table>

Note: ‘t’ means testing set; ‘t’ means training set; ‘u’ means validation set.

strategies such as only-DAE and only-DSN/DDN and 2) during multitask learning, the training of DSN/DDN and DAE was implemented alternately, so their effects on FEN were balanced. However, for DAE-DSN/DDN strategy, as DSN/DDN actually took the training result of DAE as its initial status, it may destroy the knowledge learned in DAE, thus the DAE-DSN/DDN strategy had a strong bias toward DSN/DDN ultimately. Similarly, DSN/DDN-DAE strategy had a strong bias toward DAE. Thus, multitask learning strategy had achieved the best balance between DAE and DSN/DDN, and knowledge from both DAE and DSN/DDN can be effectively protected.

H. Comparative Evaluation With Existing Models

1) With Models Having the Same Data Requirements: In this part, the MFEFD is compared with several existing models which have the same data requirements with MFEFD (namely, only load profiles of users). These models are TSVM [27], EM [19], SVM [18], WCC [20]. We have implemented these competing models on our data set, and their performances on the testing set, as well as the consumed times during training are listed in Table V.

Table V shows that: 1) every model had performed very close on the training and validation sets, which means overfitting did not occur; 2) The performances of MFEFD and TSVM were much better than the other models’. Since the MFEFD and TSVM are semisupervised models, both the information in unlabeled and labeled data can be utilized during training them, which has equipped them with better generalization ability; 3) the MFEFD had achieved the best performances, which was facilitated by deep neural networks’ powerful ability to handle high-dimensional data. Comparing the performances of MFEFD and other models on the training set, one can infer that this powerful ability may partly result from MFEFD’s huge model capacity. In general, a machine learning model will perform best when its capacity is appropriate in regard to the true complexity of the task as well as the amount of training data [24], and underfitting tends to happen if its capacity has been set too small. As shown in Table V, the performances of MFEFD on the training set were obviously better than all the other models’, which means that other models were essentially underfitted. While for MFEFD, the multilayer architectures of FENs and DNs have equipped it with a greater capacity, which was proven more appropriate for this case than other low-capacity models; and 4) the training times of DAE + DSN and DAE + DDN were only 2.3 and 1.5 h, respectively, which were much shorter than the TSVM’s and the EM’s. This is because we have utilized graphics processing units (two NVIDIA GTX980 graphics cards) to accelerate the training of MFEFD. Considering that the requirement on the timeless of electricity fraud detection is usually not so high, the training times of MFEFD are already acceptable for practice.

Class imbalance is another very important issue in the domain of fraud detection since usually honest samples are much more than fraudulent ones in the training set [6], [39]. To investigate the effects of class imbalance on MFEFD, we have regenerated several new data sets with different rates of fraudulent users in the labeled data and also rebuilt these models and trained them on these data sets. Their performances on testing sets are presented in Fig. 7, which shows that the MFEFD was rather robust against the class imbalance. For example, as the rates of fraudulent samples decreased, both the DRs and FPRs of all the models had decreased since these models were gradually dominated by the honest samples and thus tended to give negative judgments. While, even when the fraudulent sample rate was very low (e.g., 10%), the DRs of MFEFD were still satisfying (above 80%), which was a rather promising result for practice.

2) With Models Having Higher Data Requirements: In this part, the MFEFD is compared with several existing models which have higher data requirements than MFEFD. They are consumption pattern-based energy theft detector [6], P2P computing [17], and decision tree coupled SVMs (DTCSVMs) [23]. Due to the limitation of our available data, we were not able to implement these models, but we found their performance in corresponding papers as follows.

Table VI shows that MFEFD had achieved very competitive performances on both DR and FPR, while it had the lowest...
The other three groups were utilized in this test, which consist of 67, 100, and 100 time series, respectively. Time series in the first group are real traffic measurements from Yahoo!’s web services, while series in the other groups are all synthesized.

1) Yahoo! S5 Data Set: The Yahoo! S5 data set is composed of 367 time series in total, each of which consists of around 1500 hourly points together with corresponding time stamps and labels indicating anomaly or not. The whole data set is divided into four groups by its providers, which contain 67, 100, 100, and 100 time series, respectively. Time series in the first group are real traffic measurements from Yahoo!’s web services, while series in the other groups are all synthesized. The real-time stamps of each series in the first group have been replaced with ordinal numbers from one to its length, leading to the nonavailability of the exact time for each measurement (i.e., time of the day, the day of the week), which makes it impossible to implement MFEFD on this group. Thus, data in this test was finally built into a much smaller structure to reduce. As the size of Yahoo! S5 data set (16 MB) is much smaller than that of the ISET data set (623 MB), the MFEFD parameters and less training time.

2) Implementing MFEFD and Benchmarking Models: The building and training of FENs, DN, and the whole MFEFD as well as the determination of their structures were all done following the aforementioned procedures. As for benchmarking models, in addition to TSVM and SVM, another two state-of-the-art anomaly detection models which had also utilized the Yahoo! S5 data set were implemented, namely, the C-LSTM [41] which is based on deep learning, and the ES-CVAE [42] which is based on probabilistic inference. For C-LSTM, we adjusted the dimension of its input from the original 60 to 168 (24 × 7) for the ease of performance comparing with MFEFD, and all the remaining factors (e.g., structures and hyperparameters) were kept the same with [41]. The ES-CVAE works differently from C-LSTM and MFEFD, that is, it finally gives labels not on profiles but on all the points, which means its results are label series. Again for the ease of performance comparing with MFEFD, we have merged every week’s label series (predicted by ES-CVAE) according to the following rule: once anomaly appears in the (predicted) weekly label series, then its corresponding weekly profile is labeled as anomaly; if no anomaly appears in the profile, then its profile is labeled as normal.

Table VII shows that: 1) the performances of MFEFD and TSVM were better than the other models’ on the whole. Since the MFEFD and TSVM are semisupervised models, both the information in unlabeled and labeled data can be utilized during training them, which has equipped them with better generalization ability; 2) the MFEFD had achieved the best performances, which was facilitated by its deep architecture (compared to the TSVM, SVM, and ES-CVAE) as well as the semisupervised multitask training strategy (compared to the C-LSTM); and 3) compared to the results on ISET data set in Table V, the training time of MFEFD has been greatly reduced. As the size of Yahoo! S5 data set (16 MB) is much smaller than that of the ISET data set (623 MB), the MFEFD in this test was finally built into a much smaller structure to avoid overfitting, which thus indicates much fewer trainable parameters and less training time.

4) Effect of Unlabeled Data in the Yahoo! S5 Data Set: Similarly, how unlabeled data affect the performance of MFEFD

<table>
<thead>
<tr>
<th>Models</th>
<th>DR</th>
<th>TP</th>
<th>Accuracy</th>
<th>Data required</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAE+DSN</td>
<td>95.04%</td>
<td>5.14%</td>
<td></td>
<td>Load profiles of customers</td>
</tr>
<tr>
<td>DAE+DDN</td>
<td>97.27%</td>
<td>5.56%</td>
<td></td>
<td>Load profiles of customers and transformers</td>
</tr>
<tr>
<td>CPBETD</td>
<td>94%</td>
<td>11%</td>
<td></td>
<td>Load profiles of customers and transformers</td>
</tr>
<tr>
<td>P2P</td>
<td>96%</td>
<td>9%</td>
<td></td>
<td>Load profiles of customers, number of persons and heavy appliances, etc.</td>
</tr>
<tr>
<td>DTSCVM</td>
<td>--</td>
<td>5.12%</td>
<td>92.50%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Models</th>
<th>DR</th>
<th>TP</th>
<th>Accuracy</th>
<th>Data required</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAE+DSN</td>
<td>94.15%</td>
<td>2.03%</td>
<td></td>
<td>Load profiles of customers</td>
</tr>
<tr>
<td>DAE+DDN</td>
<td>94.27%</td>
<td>2.71%</td>
<td></td>
<td>Load profiles of customers and transformers</td>
</tr>
<tr>
<td>TSVM</td>
<td>90.15%</td>
<td>4.24%</td>
<td></td>
<td>Load profiles of customers</td>
</tr>
<tr>
<td>SVM</td>
<td>76.08%</td>
<td>6.98%</td>
<td></td>
<td>Load profiles of customers</td>
</tr>
<tr>
<td>C-LSTM</td>
<td>90.74%</td>
<td>4.34%</td>
<td></td>
<td>Load profiles of customers</td>
</tr>
<tr>
<td>ES-CVAE</td>
<td>79.19%</td>
<td>2.31%</td>
<td></td>
<td>Load profiles of customers</td>
</tr>
</tbody>
</table>

## Table VI

### Comparison With Models Having Higher Data Requirements

<table>
<thead>
<tr>
<th>Models</th>
<th>DR</th>
<th>FPR</th>
<th>Accuracy</th>
<th>Data required</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAE+DSN</td>
<td>95.04%</td>
<td>5.14%</td>
<td></td>
<td>Load profiles of customers</td>
</tr>
<tr>
<td>DAE+DDN</td>
<td>97.27%</td>
<td>5.56%</td>
<td></td>
<td>Load profiles of customers and transformers</td>
</tr>
<tr>
<td>CPBETD</td>
<td>94%</td>
<td>11%</td>
<td></td>
<td>Load profiles of customers and transformers</td>
</tr>
<tr>
<td>P2P</td>
<td>96%</td>
<td>9%</td>
<td></td>
<td>Load profiles of customers, number of persons and heavy appliances, etc.</td>
</tr>
<tr>
<td>DTSCVM</td>
<td>--</td>
<td>5.12%</td>
<td>92.50%</td>
<td></td>
</tr>
</tbody>
</table>
was also investigated on the Yahoo! S5 data set. For example, we built new MEFEDs and trained them using different numbers of unlabeled samples, and their performances on the testing set are demonstrated in Fig. 8. Similar to Fig. 6, Fig. 8 also shows that increasing the number of unlabeled samples had obviously improved MEFED’s performance. The reason for this phenomenon has been presented in Section IV-F.

V. CONCLUSION

In this paper, a semisupervised deep-learning-based fraud detection model, namely, MEFED, is developed. Deep structure and high nonlinearity equip MEFED with powerful feature extraction ability. MEFED is trained in a semisupervised manner, in which both unlabeled and labeled data can be utilized, which overcomes the difficulties of shortage of data resources about fraud detection on power distribution system. Multi-task alternate training is applied for training MEFED, in which both supervised and unsupervised training are implemented. Real-world-data-based case studies on different data sets have shown that adding unlabeled data into training set has greatly improved the performance of MEFED. Comparisons among MEFED and other state-of-the-art models have demonstrated the superiority of MEFED from the perspectives of high detection performance, satisfactory privacy preservation, high practicability, and low data requirement.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful advice, especially on the proposal of DAE + DDN as well as the case study on the Yahoo! S5 data set. They would also like to thank the Electric Ireland and Sustainable Energy Authority of Ireland for their great efforts in collecting and maintaining the ISET data set and Dr. H. Xiang for his efforts on preprocessing this data set.

REFERENCES


Tianyu Hu (S’17) received the B.S. degree from the School of Electrical Engineering, Shandong University, Jinan, China, in 2015. He is currently pursuing the Ph.D. degree with the Tsinghua-Berkeley Shenzhen Institute, Shenzhen, China.

Qinglai Guo (SM’14) was born in Jilin City, Jilin, China, in 1979. He received the B.S. degree from the Department of Electrical Engineering, Tsinghua University, Beijing, China, in 2000, and the Ph.D. degree from Tsinghua University in 2005. He is currently an Associate Professor with Tsinghua University. His current research interests include smart grids, cyber-physical systems, and electrical power control center applications.

Haonong Xi received the B.Eng. degree in transportation from Central South University, Changsha, China, in 2017. She is currently pursuing the M.E. degree in electrical engineering with Tsinghua University, Beijing.

Her current research interests include intelligent transportation systems and traffic flow model.