Deep Neural Networks With Confidence Sampling For Electrical Anomaly Detection

Norman L. Tasfi, Wilson A. Higashino, Katarina Grolinger, Miriam A. M. Capretz
Department of Electrical and Computer Engineering
Western University
London, Ontario, Canada, N6A 5B9
{ntasfi, whigashi, kgroling, mcapretz}@uwo.ca

***Abstract*—The increase in electrical metering has created tremendous quantities of data and, as a result, possibilities for deep insights into energy usage, better energy management, and new ways of energy conservation. As buildings are responsible for a significant portion of overall energy consumption, conservation efforts targeting buildings can provide tremendous effect on energy savings. Building energy monitoring enables identification of anomalous or unexpected behaviors which, when corrected, can lead to energy savings. Although the available data is large, the limited availability of labels makes anomaly detection difficult. This research proposes a deep semi-supervised convolutional neural network with confidence sampling for electrical anomaly detection. To achieve semi-supervised learning, two sub-networks are used: the first performs reconstruction and uses unlabelled data, while the second performs classification with labelled data. The two sub-networks overlap: the encoder parameters are shared between the two. To quantify anomaly detection confidence, a valuable metric in anomaly detection, the network uses a dropout sampling method. The proposed approach has been evaluated with real-world electrical data from systems such as HVAC, lighting, and heat pumps. The results demonstrated the accuracy of the proposed anomaly detection solution.**

***Keywords*—Anomaly Detection, Neural Network, Deep Learning, Confidence Sampling, Semi-supervised Learning, Convolutional Network.**

I. Introduction

An International Energy Agency analysis found worldwide building energy use has grown from 102 exajoules (EJ) in 2000 to 120 EJ in 2012 [1]. The estimated potential global annual savings in building energy use equate to around 53 exajoules (EJ) by 2050 if technology and energy efficient policies are applied [2]. Such technologies include metering which measures and records the resource usage of systems such as HVAC units and lighting at regular intervals.

Building electricity metering provides data that can be analyzed to gain insights into energy usage, manage energy consumption, and identify opportunities for savings. It also enables identification of anomalous patterns, which when corrected, can lead to energy savings. *Anomalous patterns* are defined as deviations from previous historical observations [3]. Because buildings are responsible for a large portion of overall energy consumption and have a large potential for energy savings [2], anomaly detection targeting buildings can effectively reduce energy waste. Major building systems such as HVAC can consume up to 50% of building energy [4] making them an excellent candidate for anomaly detection. However, anomaly detection becomes difficult in practice as many real-world applications have hundreds of metered systems producing large amounts of noisy and sparsely labelled data. If unaided by anomaly detection solutions, this situation becomes overwhelming for building managers because they cannot devote constant attention to identifying anomalies.

One solution is to use machine learning techniques to augment the anomaly detection capabilities of businesses. Such techniques can be applied using one of three forms of learning: supervised learning with provided labels or target values for training, unsupervised learning without labels, or semi-supervised learning, which is a blend of the other two learning methods. In the case of electrical anomaly detection, raw metering data are unlabelled and acquisition of labels is difficult because anomaly events by definition are infrequent. Consequently, the resulting datasets contain few labelled observations [3], a condition referred to here as sparsely labelled.

Prior work using supervised learning sidestepped this labeling issue by training a network to perform a regression task and predict future energy usage values [5], [6]. Classifying an observation in this approach requires a secondary step that compares the difference between actual and predicted values to a threshold. The chosen threshold value dictates the decision boundary for classification of all samples, but generalization is poor if the threshold value is incorrectly chosen. In contrast, when a model learns to perform classification directly using labelled data, a secondary step is not needed and an appropriate boundary is learned for each class, providing better performance. With unsupervised learning, the boundary selection issue also arises because reconstruction error needs to be compared to a threshold [7], [8].

This research proposes a deep semi-supervised convolutional neural network for electrical anomaly detection from streaming data with disjoint training and quantification of prediction confidence. To accommodate sparsely labelled datasets, the proposed network uses a dual architecture comprised of two separate sub-networks each with its own objective: reconstruction or classification. The network parameters are shared between portions of the two sub-networks, enabling disjoint updates as various blends of labelled and unlabelled samples are seen. To quantify prediction confidence, the proposed approach uses dropout sampling method proposed by Gal and Ghahramani [9]. Because the proposed network uses convolutional layers, information is captured from neighbouring observations, enabling contextual anomaly detection. In addition, a preprocessing method for periodic variables ensures that the distance between periodic rollovers is preserved regardless of

N. L. Tasfi, W. A. Higashino, Katarina Grolinger, M. A. M. Capretz, "Deep Neural Networks With Confidence Sampling For Electrical Anomaly Detection", in Proc. of the 3rd International Conference on Smart Data (SmartData-2017).

period and without massive increase in feature dimensionality.

The approach has been evaluated with a large real-world electrical dataset collected from various systems such as HVAC and lighting. The experiments demonstrated that combination of a deep semi-supervised network and periodic preprocessing enables accurate anomaly detection while providing additional confidence information for intelligent decision-making.

This paper is organized as follows: Section II describes related work in the field of anomaly detection using neural networks in the electrical data domain. Section III outlines the approach taken by the research described here. In Section IV, the application and testing of the network on data are described. Finally, Section V describes concluding thoughts and future work.

II. Related Work

Objective Function: Several studies have used an objective function consisting of a single error function, such as mean square, focused on prediction of future values [5], [6], [10], input reconstruction [7], [8], or classification. Methods using a single error function cannot make use of blends of data, unlabelled and labelled, by their definition. The semi-supervised method proposed here uses an objective function consisting of two error functions, one unsupervised and the other supervised. This enables the network to use all available data, both labelled and unlabelled, which is beneficial in anomaly detection because unlabelled data is plentiful and labelled data is not [11].

The research presented here focuses on performing classification in one step, without a secondary threshold comparison step, which helps generalization and accuracy because the class decision boundary is supported by data. Similar approaches have been used in other domains besides anomaly detection, where the model is updated in one step with a dual objective or is pretrained using unsupervised learning. Dai and Le [12] used supervised and unsupervised recurrent networks to pretrain a network to initialize weights, enabling strong performance in language classification tasks. Ranzato and Szummer [13] used layer-wise pretraining in which each layer performed classification and reconstruction on its inputs. Weston *et al.* [14] suggested several variations of semi-supervised networks that focus on learning the embedding of an input at different positions within a network; the embedding is then used to train a supervised loss function. Unlike existing anomaly detection work, the research presented here uses intermittent supervised updates as labelled data becomes available and unsupervised learning otherwise, meaning that updates do not need to be performed simultaneously. Training in such a disjoint fashion was previously suggested by Weston *et al.* [14].

Confidence Sampling: Previous studies using neural networks cannot provide an estimate of certainty for network predictions. This problem arises primarily because neural networks learn a discriminant function, in which an input is mapped directly to a desired output, making quantification of prediction uncertainty difficult because probabilities play no role [15]. This work uses a technique suggested by Gal and Ghahramani [9], which samples from a network by exploiting the dropout technique. Dropout, a regularization technique proposed by Srivastava *et al.* [16], "drops" a random percentage of units during training. By doing so the network is strongly regularized, combating overfitting common in deep networks, and increasing network generalization. Application of this dropout sampling method makes it possible to understand how confident the network is in its classification. Prediction confidence enables explicit handling of uncertain inputs, in turn supporting additional techniques such as a reject option, improving the model's prediction capabilities, and creating additional labelled samples.

Preprocessing of Periodic Variables: It is common to transform continuous periodic variables such as time of day to 1-hot encoding or linear scale representations. Chae *et al.* [17] transformed time variables into integer values and 1-hot encoding for use in anomaly detection, e.g, $\{1,0,0\}$ for weekdays, $\{0,1,0\}$ for Saturday, and $\{0,0,1\}$ for Sunday. Preprocessing using 1-hot encoding causes an increase in input dimensionality because each possible value of the periodic value requires a dimension. Benedetti *et al.* [6] and Araya *et al.* [7] normalized all values by scaling them to the range [0,1] by dividing each variable by its maximum. Performing such preprocessing causes jumps in "distance" that are not reflective of the actual "distance" between time points. The encoding used in this work, a projection onto the unit circle, correctly preserves distances between points and requires only two additional dimensions per periodic value.

Network Depth: Prior work on anomaly detection has used shallow networks containing fewer than three hidden layers. Mousavian *et al.* [10] used a shallow network with two hidden layers to predict future usage values and to classify a point based on its difference from the real reading. Araya *et al.* [7] and Sakurada and Yairi [8] both used networks to perform input reconstruction where anomaly classification was predicted on the reconstruction error achieved. Araya *et al.* used a deep network with three hidden layers on electrical metering data. Our work in contrast uses deeper networks providing greater representational power and generalization [18]. The increase in representational ability occurs because the network learns a hierarchy of features built upon those from previous layers, which enables extraction of non-local relationships and patterns [19]. In addition, in the context of periodic events, which occur in anomaly detection, Szymanski and McCane [20] found that deep networks learn feature hierarchies that efficiently encode periodic events, a capability that makes deep networks well suited to time series electrical data.

A final consideration is the number of hidden units per layer affecting the width of the network. Previous studies used a small number of units in each layer [8], but our work uses an order of magnitude more units. This increase in units serves a dual purpose: to increase network capacity, and allow usage of higher dropout probabilities.

III. Methodology

Introduced below is the architecture and components of the anomaly detection system. This architecture, in Fig. 1, is comprised of preprocessing, model, and notify components. An explanation of each component's usage and function is detailed below.

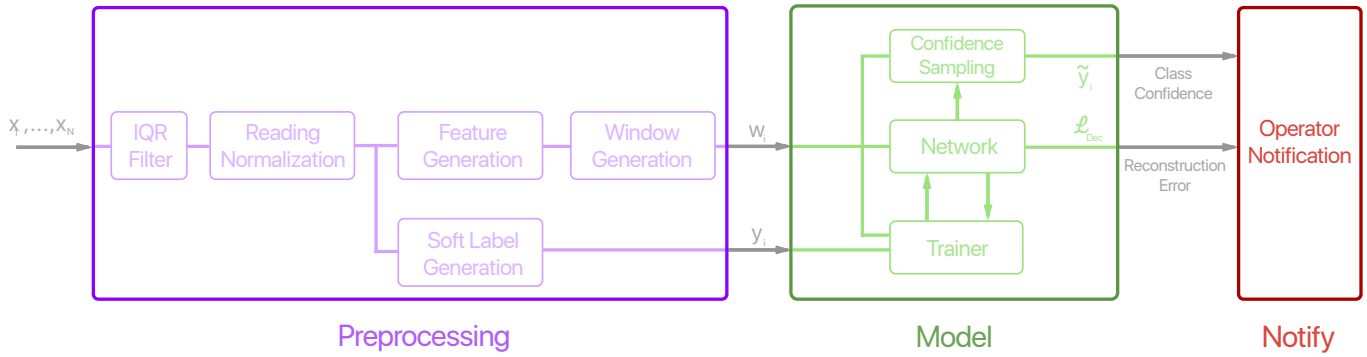

Fig. 1: End-to-end overview of system. $x_1, ..., x_N$ are the raw samples coming in from the monitored electrical systems. The network outputs are used to notify the operator.

A. *Preprocessing*

Preprocessing is performed to exploit the domain knowledge of the system and process specific features, such as time information, helping improve performance of the model. The final product of preprocessing and feature generation can be seen in Table I.

IQR Filter: An interquartile test is performed to remove extreme readings, such as those caused by sensor malfunctions, where the value is unlikely to be output by the system. Such readings are removed if they are above the threshold $Q3 + 1.5IQR$ where $Q3$ is the 75th quantile and IQR is the interquartile range of the data. In addition, simple filtering is performed to remove days where the systems did not record data.

Reading Normalization: The reading values must be normalized into the range $[0, 1]$, on a per-system basis, to minimize the magnitude of gradient updates helping model convergence. This is done by scaling each x_n for $n = 1, .., N$ reading value using Eq. 1, where $\mathbf{X}$ is the dataset and N is the number of raw dataset samples:

$$\tilde{x}_n = \frac{x_n - \min(\mathbf{X})}{\max(\mathbf{X}) - \min(\mathbf{X})} \tag{1}$$

Feature Generation: Each reading value is augmented with first and second difference features between normalized $\tilde{x}_n$ reading values. The difference is calculated as $\tilde{x}_n^{\delta} = \tilde{x}_n - \tilde{x}_{n-1}$ and applied twice: on the normalized data and to the resulting first difference. The two difference features provide information on each inputs value change over time and are similar to a moving first- and second-order derivative approximation.

Features containing periodic variables, such as the timestamp of each sample, are transformed extracting additional information. This transformation involves splitting each timestamp, t_n, into its k constituent, with possible values such as $k = \{month, weekday, weekOfYear, hour, minute\}$, and performing a mapping of each variable to a unit circle representation. This mapping takes the kth constituent's value, $t_{n,k}$, with a known period, p_k, and maps it with the following formula:

$$\hat{t}_{n,k} = \left\{ \cos\left(\frac{2\pi t_{n,k}}{p_k}\right), \sin\left(\frac{2\pi t_{n,k}}{p_k}\right) \right\} \tag{2}$$

where each periodic value $t_{n,k}$ is split into two components representing its x and y locations on the unit circle. This transformation preserves distances between rollovers of periodic variables. To understand its utility consider the following example. Assume a variable with known period $p_k = 60$ is given, and the difference between two values, 59 and 1, needs to be calculated. Taking the algebraic difference produces a result, 58, that ignores the values underlying periodicity. In contrast, when the values are mapped to the unit circle, distance is preserved and the calculation yields, after back-converting, the correct distance of 2.

This periodic mapping requires significantly fewer feature dimensions than other common preprocessing techniques such as 1-hot encoding. Incorporation of time information such as month, weekday, hour, and minute using 1-hot encoding requires an additional 12, 7, 24, and 60 additional features. If all components are included, 103 additional dimensions of very sparse data would be needed per sample. In contrast, the periodic mapping described above simply requires two dimensions per time component, or only eight additional dimensions if all components are included.

TABLE I: Features and description of processed data.

Feature	Description
Sensor Value	Normalized value with range $[0, 1]$
First Difference	Difference of n^{th} and $(n-1)^{st}$ values
Second Difference	Difference of n^{th} and $(n-1)^{st}$ first difference
Month {sin,cos}	Unit circle projection with $p_k = 12$
Weekday {sin,cos}	Unit circle projection with $p_k = 7$
Week Of Year {sin,cos}	Unit circle projection with $p_k = 52$
Hour {sin,cos}	Unit circle projection with $p_k = 24$
Minute {sin,cos}	Unit circle projection with $p_k = 60$

Soft Labels: Data points are assigned soft labels, using simple heuristics, signifying the belief that the particular point is an anomaly. The soft labels are used by the classification network. A soft label is considered "soft" because label values

are chosen to push the classifier closer to a belief of the underlying label. It was found to be most effective to select soft labels that were equidistant from their hard class label. For example, a soft positive label of 0.7 and a soft negative label of 0.3 may be selected for a classification with regular labels of $\{0, 1\}$.

In this study, the following heuristics are used to assign soft positive labels to points:

- If the raw reading value is less than 0.0. This was saved before the values were normalized.
- If the current reading value is greater than the global $99.9th$ quantile.
- If the current reading value is greater than the $99.9th$ quantile of points with the same hour and minute values in the preceding three weeks *and* the variance of this set of points was below a certain threshold.

Soft negative labels were assigned to points:

- If the current reading value is roughly equal, $\pm\epsilon$ where ϵ is a very small number, to the mean of points with the same hour and minute values in the preceding three weeks *and* the variance of this set of points was above a certain threshold.

Window generation: The samples are created by collecting sliding windows $w_{i,j}$ of $j = 1, ..., J$ observations and soft labels $y_{i,j}$. The experiments in this work use $J = 23$ observation per window, covering a 2 hour duration, with six observations skipped between each window. The skip is chosen to reduce the similarity between window samples. The label values over each window must be a valid probability distribution where y_i must satisfy $y_{i,j} \in [0, 1]$ and $\sum_{j=1}^{J} y_{i,j} = 1$. To enforce this constraint if any soft labelled observations $y_{i,j}$ exist within a window w_i normalization is performed over all J observations in $y_{i,j}$, producing a valid probability distribution. Each observation label y_i, after normalization, indicates a prior belief of the presence or absence of an anomaly in the jth observation of w_i. If the window contains no labels, a $y_i = nil$ label is assigned to the window. Therefore, each sample within the generated dataset, denoted by w_i, consists of a sliding windows of samples, each with corresponding label value y_i. Each individual window is stacked into one matrix $\mathbf{W} \in \mathbb{R}^{N \times J \times F}$ and each label into one matrix $\mathbf{Y} \in \mathbb{R}^{N \times J}$ where N is the number of samples, J is the number of observations in the window, and F are the number of generated features.

B. Model

Network: The proposed network uses a dual architecture comprised of two separate sub-networks, each with its own objective: reconstruction or classification. This enables the training of deep networks on sparsely labelled datasets. The encoder parameters are shared between the two objectives, enabling disjoint updates as various blends of observations are seen and making it possible to train deep expressive networks. The network architecture, as seen in Fig. 2, is comprised of separate sub-networks: encoder, decoder, and classifier. The encoder sub-network, $Enc(w_i; \theta_E)$, learns a representation

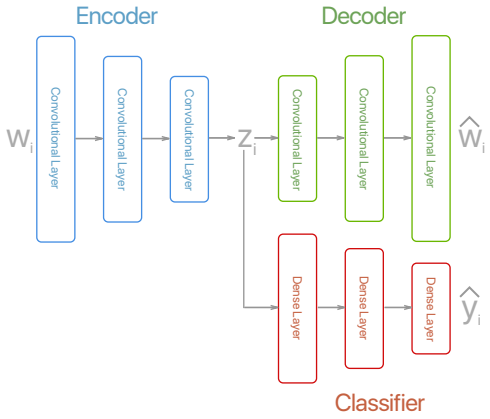

Fig. 2: Diagram of network structure. The network uses two losses: Class Prediction and Reconstruction Input. The encoder, decoder, and classifier sub-networks are shown by the blue, green and red blocks respectively.

z_i which is an embedding of w_i. The decoder sub-network, $Dec(z_i; \theta_D)$, reconstructs w_i given the compressed representation z_i. Finally the classifier sub-network, $Class(z_i; \theta_C)$, predicts which observations $1, .., J$ within $w_{i,j}$ are anomalous. Each sub-network Enc, Dec, and $Class$ is parameterized by θ_E, θ_D, and θ_C respectively. The compressed representation z_i must contain the information needed to perform *both* the classification and reconstruction tasks.

Training: The model is trained using backpropagation to minimize a dual loss function. The loss function, seen in Eq. 3, is a weighted sum of the reconstruction and classification error for each sub-network.

$$\mathcal{L}(w_i, y_i; \theta) = \gamma \mathcal{L}_{Recon}(w_i, \hat{w}_i; \theta_D) + (1-\gamma)\mathcal{L}_{Class}(y_i, \hat{y}_i; \theta_C) \tag{3}$$

In situations where the observation is unlabelled, eg. (w_i, nil), only the encoder and decoder sub-networks (EDNs), $Enc(w_i; \theta_E)$ and $Dec(z_i; \theta_D)$, are used for parameter updates. If the observation is labeled all sub-networks are updated: $Enc(w_i; \theta_E)$, $Dec(z_i; \theta_D)$, and $Class(z_i; \theta_C)$. The ability to train portions of the network independently is extremely beneficial because most anomaly detection datasets contain few labelled samples. The natural expression for classification and reconstruction errors are given, respectively, by cross entropy (Eq. 4) and mean square error (MSE, Eq. 5) functions:

$$\mathcal{L}_{Class}(y_i, \hat{y}_i; \theta_C) = \frac{1}{N} \sum_{i=1}^{N} H(y_{i,j}, \hat{y}_{i,j}) \tag{4}$$

$$\mathcal{L}_{Recon}(w_i, \hat{w}_i; \theta_D) = \frac{1}{N} \sum_{i=1}^{N} \{w_i - \hat{w}_i\}^2 \tag{5}$$

where $H(y_j, \hat{y}_j)$ is the cross entropy function

$$H(y_i, \hat{y}_i) = -\sum_{j=1}^{J} y_{i,j} log \hat{y}_{i,j} \tag{6}$$

$\hat{w}_i$ and $\hat{y}_i$ are outputs of the $Dec(.)$ and $Class(.)$ networks for training tuple (w_i, y_i).

The contribution of each loss function to network parameters is controlled by γ as seen in Eq. 3, where the valid range is $[0, 1]$. If γ is 0.0 network parameters will be adjusted only by classification losses. Conversely, for $\gamma = 1.0$ classification loss has no impact on parameters and only reconstruction loss causes change. Values between the range of $[0, 1]$ shift the impact each loss has on network parameters such that we dedicate more network capacity towards one loss function.

Algorithm 1 Training algorithm for network

Require: $\mathbf{W}_{train} = \{w_1, ..., w_N\}$ and $\mathbf{Y}_{test} = \{y_1, ..., y_N\}$
Require: α: learning algorithm parameters
Require: ρ: percent of units to drop
Require: γ: capacity of network used for reconstruction.
  $\theta_E, \theta_D, \theta_C \leftarrow init$
  $N \leftarrow$ number of samples
  for $i := 1$ to N **do**
    $z_i \leftarrow Enc(w_i; \theta_{E_t})$
    $\hat{w}_i \leftarrow Dec(z_i; \theta_{D_t})$
    $\mathcal{L}_{D_i} \leftarrow \gamma \frac{1}{2} \Sigma_{j=1}^{J} \{w_{i,j} - \hat{w}_{i,j}\}^2$
    $\mathcal{L}_{C_i} \leftarrow 1$
    if $y_i \neq nil$ **then**
      $\hat{y}_i \leftarrow Class(z_i; \theta_{C_t}, \rho)$
      $\mathcal{L}_{C_i} \leftarrow (1.0 - \gamma) \left\{ - \sum_{j=1}^{J} y_{i,j} \log \hat{y}_{i,j} \right\}$
    end if
    $\theta_{C_{t+1}} \leftarrow \theta_{C_t} - \alpha_C \frac{\partial \mathcal{L}_{C_i}}{\partial \theta_C}$
    $\theta_{D_{t+1}} \leftarrow \theta_{D_t} - \alpha_{EDN} \frac{\partial \mathcal{L}_{D_i}}{\partial \theta_D}$
    $\theta_{E_{t+1}} \leftarrow \theta_{E_t} - \alpha_{EDN} \left[\frac{\partial \mathcal{L}_{D_i}}{\partial \theta_E} + \frac{\partial \mathcal{L}_{C_i}}{\partial \theta_E} \right]$
  end for

The training algorithm is detailed in Algorithm 1 where α_C is the learning rate used for the classifier sub-network, α_{EDN} is the learning rate used for the EDN sub-network, and ρ is the percentage of units dropped per batch update.

Confidence Sampling: The classifier sub-network was trained using dropout, which enable the usage of dropout sampling proposed by Gal and Ghahramani [9]. Dropout sampling enables quantification of network prediction confidence by performing several forward passes, determined by a parameter S, with dropout enabled, while holding the hidden representation z_i constant. The resulting output sample set, $\tilde{y}_i = \{\tilde{y}_{i,1}, ..., \tilde{y}_{i,S}\}$, provides access to both the predictive mean and variance of our networks output. Estimates of the mean and variance of $Class(z_i; \theta_C)$ were calculated as follows:

$$\tilde{y}_{i,s} \sim Class(z_i; \theta_C) \tag{7}$$

$$E[\tilde{y}_i] \approx \frac{1}{S} \sum_{s=1}^{S} \tilde{y}_{i,s} \tag{8}$$

$$Var[\tilde{y}_i] \approx \tau^{-1} I_D + \frac{1}{S} \sum_{s=1}^{S} \tilde{y}_{i,s}^T \tilde{y}_{i,s} \tag{9}$$

where $\tilde{y}_{i,s}$ is the networks output given $z_i = Enc(w_i; \theta_E)$, w_i is an observation, I_D is an D-dimensional identity matrix and τ is the inverse precision calculated using the following formula:

$$\tau = \frac{\rho l^2}{2N\lambda} \tag{10}$$

where l is chosen as the prior frequency of the data and λ is the amount of L_2 regularization applied on the network parameters. Algorithm 2 describes the sampling algorithm. This can be loosely viewed as sampling various sub-networks of $Class(z_i; \theta_C)$ to determine the confidence that z_i, an encoded representation of w_i, is anomalous. With a measurement of network confidence, the subsequent decision-making process can be augmented by a reject option. This option requires human intervention only for low-confidence observations, which provides the possibility of adding labelled samples to the network over time in an online fashion.

Algorithm 2 Sampling algorithm for network

  Given: $\mathbf{X}_{test} = \{w_1, ...w_N\}$
  Given: $\theta_E, \theta_D, \theta_C$: learned parameters
  Given: ρ: dropout percent used during training
  $N \leftarrow$ number of samples
  $S \leftarrow$ number of time forward passes from network
  $\tau \leftarrow \frac{\rho l^2}{2N\lambda}$
  for $i := 1$ to N **do**
    $\hat{z}_i \leftarrow Enc(w_i; \theta_E)$
    $\tilde{y}_i = \{\}$
    for $s := 1$ to S **do**
      $\tilde{y}_{i,s} \sim Class(z_i; \theta_C)$: sample with dropout
    end for
    $\mu_i \leftarrow \frac{1}{S} \sum \tilde{y}_{i,s}$
    $\sigma_i \leftarrow \tau^{-1} I_D + \frac{1}{S} \sum_{s=1}^{S} \tilde{y}_{i,s}^T \tilde{y}_{i,s}$
  end for

C. Notify

Finally, the operator is notified when an anomaly has occurred using the classifier sub-network output and the EDN reconstruction error. As the EDN is a convolutional Autoencoder performing dimensionality reduction, the assumption that correlated data can be reduced to a lower dimensional subspace is used. On this subspace normal and anomalous data are substantially different [8], meaning that high reconstruction error indicates the presence of an anomaly.

Operator Notification: The combination of the networks output, classification confidence values and reconstruction error over a window of samples, is converted to binary output. Eq. 11 intuitively captures this conversion:

$$\hat{y}(w_i) = \begin{cases} 1 & \begin{array}{c} \mathrm{Var}[\tilde{y}_i] < \mathrm{Var}[\tilde{y}_{train}] \,\& \\ \mathrm{E}[\tilde{y}_i] > \mathrm{E}[\tilde{y}_{train}] \,\& \\ \mathcal{L}_{Recon}(w_i) > \mathrm{E}[\mathcal{L}_{Recon}(X_{train})] \end{array} \\ 0 & otherwise \end{cases} \tag{11}$$

Notification occurs when the joint model has low predictive variance, high predictive mean, and high reconstruction error.

Ideally each window would be presented to the operator screen for anomalies, with predictive mean and variance superimposed over the window. This of course would not scale as the operator would need to inspect many samples, so notification are gated based on the networks predictive variance.

IV. Experimental Results and Discussion

The evaluation of this work used data provided by Powersmiths [21], a Canadian based company, specialized in Internet connected devices used to record and transmit information from electrical systems. The captured information, primarily from building systems, is used to help businesses manage environmental footprints and reduce electrical waste. The dataset is comprised of two years of electrical data, recorded at five minute intervals, from buildings in the same geographical location.

A typical sample from the data was comprised of a timestamp and a sensor value. After preprocessing, the dataset contained 664053 rows with 23 observations and 13 features per window. Soft labelling heuristics generated 91k labelled points. As the percentage of labelled points were higher than most practical scenarios involving anomaly detection, a subset was sampled reducing the percentage of labelled samples from 13% to 3%. This resulted in 19.9k labelled samples. Finally the dataset was split into training, test, and validation sets, each respectively containing 80%, 15%, and 5% of the dataset. The number of labelled samples were kept proportional to each set percentage.

To evaluate this work the following anomaly models were chosen: a Fully Connected AutoEncoder (FCAE), a Convolutional AutoEncoder (CAE) the modernized version of the FCAE, and a Support Vector Machine (SVM). Each model was configured as follows:

- FCAE contained two sets of three layers, representing the encoder and decoder, with 299, 16, and 4 neurons respectively. The decoder neurons were 8, 16, and 299. Where 299 is the product of the input window dimensions, $J * F$.
- CAE used the same architecture as the EDN sub-network as seen in Table II.
- SVM used $C = 1.0$, a RBF kernel, and a η of $\frac{1}{13}$ where η is used instead of the γ notation common in the SVM literature.

The models used for comparison could not be trained using the previously generated labels so a binary classification label dataset was derived. The derived set was assigned a positive class label of 1 if any value within a label window y_i was above random, which in this case was defined as $1.0/J$ where J is the window size. Furthermore, as the FCAE & CAE cannot be trained directly to do classification, the MSE function was minimized instead and used a threshold function on the output of each model $\mathcal{M}$ on sample i:

$$\hat{y}(x_i) = \begin{cases} 1 & \mathcal{M}(x_i) > \mathrm{E}[\mathcal{M}(\hat{X}_{train})] + \frac{1}{2}\sqrt{\mathrm{Var}[\mathcal{M}(\hat{X}_{train})]} \\ 0 & otherwise \end{cases} \tag{12}$$

For all model performances to remain comparable the binary cross entropy function was used: $L(\hat{y}, y) = -ylog(\hat{y}) - (1-y)log(1-\hat{y})$.

A. Implementation details

The implementation was completed in Python using Theano [22] and ran on a server with Intel Core i7 CPU at 3.50GHz, 32GB RAM, and 2 NVIDIA GeForce GTX 980 GPUs. The EDN was constructed using 3 convolutional layers with decreasing filter and stride size, as detailed in Table II. Filter sizes of $[5, 1]$ and $[3, 1]$ were used because incorporation of temporal information was needed along the J dimension and filters of shape $[M, 1]$ act only along this dimension. The classifier sub-network used four fully connected layers of: 512, 256, 128, 23 units each, spanning from input z to output $\hat{y}$. All layers used the Leaky ReLu activation function with leakiness of 0.01. The weight matrices of the network were initialized with Glorot uniform initialization [23].

TABLE II: Encoder and Decoder sub-network settings.

Layer	Num. Filters	Filter size	Stride
Encoder L1	64	5,1	2,1
Encoder L2	32	3,1	1,1
Encoder L3	16	3,3	1,1
Decoder L1	16	3,3	1,1
Decoder L2	32	3,1	1,1
Decoder L3	64	5,1	2,1

The EDN used input corruption and magnitude penalties for regularization. The encoder's input was corrupted with 0.1 standard deviation of added zero-mean Gaussian noise and a magnitude penalty on z was used with weight 9e-5. Typically parameters of the encoder and decoder networks are tied as a regularization technique [24]. Empirically it was found that tying parameters heavily adjusted the encoder parameters towards the reconstruction task which decreased the encoders capacity used for classification. This interference could not be corrected unless the γ parameter was set very close to zero. The experiments used untied parameters with $\gamma = 0.2$.

The classifier sub-network used dropout and L2 regularization on all dense layers with a dropout rate of 0.25 and L2 weight of 1e-4. Additionally, the EDN network acts as a strong regularizer on the classifier sub-network. The network was trained for 50 epochs on the training dataset and batch size of 128 using the Adam optimizer [25]. A learning rate of 3e-5 was used for the classifier sub-network and 1e-5 for the EDNs.

B. Results and Discussion

The performance of our models on two datasets were examined: the *Test dataset* and a small set of samples containing *Generated Anomalies*. For the generated dataset the effectiveness of having confidence information presented and the models ability to identify anomalies were discussed.

Test dataset: The loss functions, measured in bits, signifies how effectively each model captures the underlying class

TABLE III: Classification entropy of experiments.

Model	Test Entropy
Random	13.78
FCAE	11.43
EDN	7.94
CAE	6.45
SVM	3.82
Proposed model	**3.08**

distribution. A lower value indicates better performance. The results on the test dataset, containing 2.98k labelled samples, are found in Table III. The performance of *Random*, where a class is randomly assigned to a window w_i, was 13.78. This acts as a baseline for which all models must outperform to provide useful anomaly detection capabilities.

From Table III, it is quickly apparent the *Proposed Model* outperforms all models by a healthy margin. The SVM, though unable to perform localization, showed excellent performance on the dataset. The authors strongly believe the utilization of a distributed training environment with larger volumes of data would allow the proposed model to use deeper networks with increased performance and generalization.

Notable is the performance difference between the EDN and CAE as both models share the exact same architecture. This result is expected as the EDN had parameter updates from the classifier sub-network that adjust the common parameters in the encoder network. The FCAE model outperforms the Random baseline by a smaller than expected factor. The poor performance of the FCAE is most likely because the network is unable to use temporal information present within each window. This seems even more likely when the number of parameters available to the FCAE and CAE, which uses temporal information, are compared. The FCAE has almost an order of magnitude more free parameters than the CAE: 9792 and 1312 respectively.

Generated Anomalies: After inspection, reoccurring periods of low usage, where the majority of reported values are near zero, were found within the dataset. These periods typically were weekends and time periods between 9pm-4am. Therefore a handful of samples from these periods had anomalous signals added to the reading values, where 'anomalous' signals are defined as impulse or gradual increase in usage over time for a small duration. Examples of the impulse and gradual increase can be seen, with the proposed model output, in Fig. 3 and Fig. 4 respectively.

In Fig. 3 an impulse anomaly and the models response was plotted where the predictive mean and predictive variance are shown on the top portion of the plot. It is clear that the model has roughly localized where the injected anomaly has been placed at $t = 75$. The model also has several smaller peaks at time $t = \{15, 30, 40, 60, 90, 110\}$ which are likely due to noise from sampling. Upon inspection of the confidence bands, shown as the area around the lines, the smaller peaks have larger bands indicating the model had high uncertainty in these localizations. This is in contrast to the impulse anomaly at $t = 75$ which has narrow, nearly non-existent, confidence

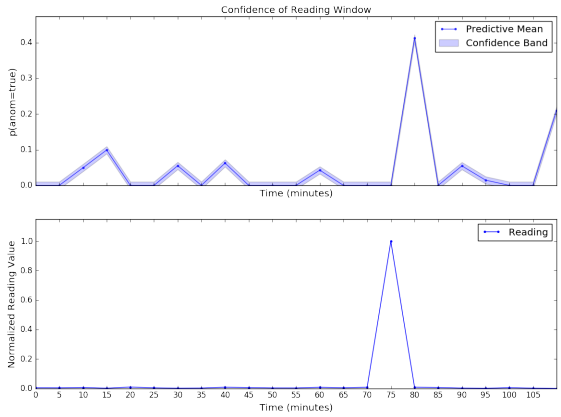

Fig. 3: Confidence plot of an impulse.

band around the point.

Fig. 4 depicts a longer duration anomaly being added. Again the model correctly identifies this anomaly and has the same smaller peaks in predictive mean. Through the examination of the confidence bands all but the prediction at $t = 55$ can be considered as accurate.

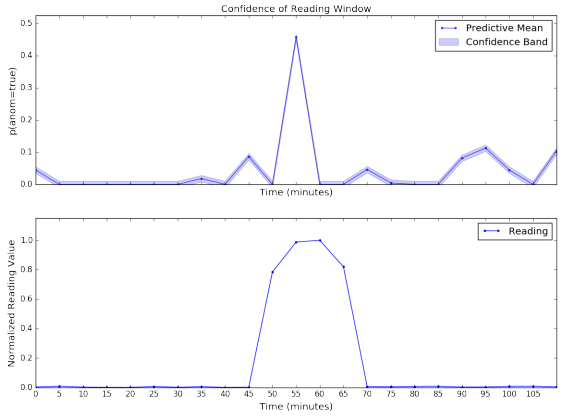

Fig. 4: Confidence plot of gradual anomaly over a short duration.

Of additional interest, in both Fig. 3 and Fig. 4 the detection of an anomaly is slightly offset by a tick or two of the x-axis, approximately 5-10 minutes. This phenomena is most likely the result of the larger 1-d convolutional filters. The first convolutional layer filter, of size $[5, 1]$, is sensitive to features in an area of 5 ticks and the convolution of this filter results in some temporal information being lost, hence the inexact location of anomalies. A possible solution to this might involve adjustment of the filter dimension, filter stride or number of convolutional filters in the EDN.

As an additional consideration a single sample from the dataset, with a manually verified anomaly, is seen in Fig. 5. The model assigns a higher probability, around time $t = 90 - 105$, with a tighter confidence bound indicating higher certainty that an anomaly occurred. Inversely, at time $25 - 40$, there is an increase in the probability with a wider confidence bound indicating uncertainty of the networks prediction.

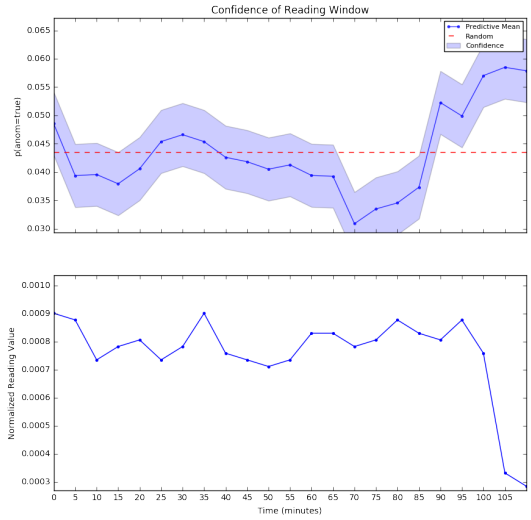

Fig. 5: Model confidence in predictions

V. Conclusion and Future Work

In this research, a methodology for training deep neural networks was presented. This method uses disjoint training enabling the training of deep neural networks for anomaly detection on sparsely labelled data. Additionally the ability to quantify network confidence was presented allowing deeper understanding of observations that the network classifies as anomalous. The results show that this methodology can accurately identify multiple anomalies within a contextual window. Moreover, the detection ability of this methodology outperforms other models such as a SVM and Autoencoder variants. Future works will explore improving the efficiency of this methodology allowing a smaller dataset. Additionally the impact of soft labeling and various schema on modeling performance will be explored.

Acknowledgement

This research was supported in part by an NSERC CRD at Western University (CRDPJ 453294-13). In addition, the authors would like to acknowledge the support provided by Powersmiths.

References

[1] IEA, "World energy statistics and balances (database)," 2014.
[2] ——, "Building energy performance metrics," 2015.
[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.
[4] L. Pérez-Lombard, J. Ortiz, and C. Pout, "A review on buildings energy consumption information," *Energy and buildings*, vol. 40, no. 3, pp. 394–398, 2008.
[5] J.-S. Chou and A. S. Telaga, "Real-time detection of anomalous power consumption," *Renewable and Sustainable Energy Reviews*, vol. 33, pp. 400–411, 2014.
[6] M. Benedetti, V. Cesarotti, V. Introna, and J. Serranti, "Energy consumption control automation using artificial neural networks and adaptive algorithms: Proposal of a new methodology and case study," *Applied Energy*, vol. 165, pp. 60–71, 2016.
[7] D. B. Araya, K. Grolinger, H. F. ElYamany, M. A. Capretz, and G. Bitsuamlak, "Collective contextual anomaly detection framework for smart buildings," in *Neural Networks (IJCNN), 2016 International Joint Conference on*, IEEE, 2016, pp. 511–518.
[8] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, ACM, 2014, p. 4.
[9] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," *ArXiv preprint arXiv:1506.02142*, 2015.
[10] S. Mousavian, J. Valenzuela, and J. Wang, "Real-time data reassurance in electrical power systems based on artificial neural networks," *Electric Power Systems Research*, vol. 96, pp. 285–295, 2013.
[11] M. M. Masud, C. Woolam, J. Gao, L. Khan, J. Han, K. W. Hamlen, and N. C. Oza, "Facing the reality of data stream classification: Coping with scarcity of labeled data," *Knowledge and information systems*, vol. 33, no. 1, pp. 213–244, 2012.
[12] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 3061–3069.
[13] M. Ranzato and M. Szummer, "Semi-supervised learning of compact document representations with deep networks," in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 792–799.
[14] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*, Springer, 2012, pp. 639–655.
[15] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.
[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[17] Y. T. Chae, R. Horesh, Y. Hwang, and Y. M. Lee, "Artificial neural network model for forecasting sub-hourly electricity usage in commercial buildings," *Energy and Buildings*, vol. 111, pp. 184–194, 2016.
[18] Y. Bengio and O. Delalleau, "On the expressive power of deep architectures," in *Algorithmic Learning Theory*, Springer Berlin Heidelberg, 2011, pp. 18–36.
[19] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1798–1828, 2013.
[20] L. Szymanski and B. McCane, "Deep networks are effective encoders of periodicity," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 10, pp. 1816–1827, 2014.
[21] *Powersmiths international*, http : / / powersmiths . com/, Accessed: 2017-03-03.
[22] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *ArXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688.
[23] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks.," in *Aistats*, vol. 9, 2010, pp. 249–256.
[24] Y. Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ArXiv preprint arXiv:1412.6980*, 2014.