Deep Learning based Detection of Sleeping Cells in Next Generation Cellular Networks

Usama Masood*, Ahmad Asghar†, Ali Imran† and Adnan Noor Mian‡
*Department of Electrical Engineering, Riphah International University, Lahore, Pakistan
†School of Electrical and Computer Engineering, University of Oklahoma, Tulsa, USA
‡Department of Computer Science, Information Technology University, Lahore, Pakistan
Email: usama.masood@riphah.edu.pk, {ahmad.asghar, ali.imran}@ou.edu, adnan.noor@itu.edu.pk

***Abstract*—The growing subscriber Quality of Experience demands are posing significant challenges to the mobile cellular network operators. One such challenge is the autonomic detection of sleeping cells in cellular networks. Sleeping Cell (SC) is a cell degradation problem, and a special case in Cell Outage Detection (COD) because it does not trigger any alarm due to hardware or software problems in the BS. To minimize the effect of such outages, researchers have proposed autonomous outage detection and compensation solutions. State-of-the-art SC detection depends on drive tests and subscriber complaints to identify the effected cells. However, this approach is quickly becoming unsustainable due to rising operational expenses. To address this particular issue, we employ a Deep Learning based framework which uses Minimization of Drive Tests (MDT) functionality introduced in LTE networks. In our proposed framework, MDT measurements are used to train the deep learning model. Anomalies or cell outages in the network can be then quickly detected and localized, thus significantly reducing the duty cycle of self-healing process in SON. In our simulation setup, we also quantitatively compare and demonstrate superior performance of our proposed approach with state of the art machine learning algorithm such as One Class SVM using multiple performance metrics.**

***Index Terms*—Self Organizing Networks (SON), Self-healing, Anomaly Detection, Sleeping Cells, Minimization of Drive Tests (MDT), Deep Autoencoders, One Class SVM**

I. Introduction

Self-x functions [1] in SON are one of the key enablers to meet the stringent requirements of next generation cellular networks, such as high capacity, ultra-low latency, better Quality of Service (QoS) and Quality of Experience (QoE) to the end users while reducing the Total Cost of Ownership (TCO) of the network, which includes CAPital EXpenses (CAPEX) as well as OPerational EXpenses (OPEX). Traditional network troubleshooting techniques are one of the many use cases that have been replaced by the *self-healing* function block in SON, which mainly consists of autonomous cell outage detection (COD) and compensation (COC) methods, without any dependence on drive tests or complaints of the users.

Motivated by this, we propose a Deep learning based approach for detection of *Sleeping Cells*, which is a particular case of Cell Outages, that does not generate any alarm at the Operation and Maintenance (O&M) system due to HW or SW failures at the BS. We leverage MDT functionality specified by 3GPP, to construct a database from user reported measurements containing information regarding network behavior.

Earlier research efforts for fault detection in cellular networks have mostly relied on performance deviation methods [2]. In [3], authors proposed a method for outage detection by analyzing the variations in the visibility relation graph between cells by utilizing NCL reports. Recently, unsupervised learning techniques have also been proposed for automating the process of outage detection. In [4], input embedded measurements are assigned to different classes using fuzzy clustering. Anomalies are then detected by observing the trajectory of embedding measurements. MDT measurements based anomaly detection, initially proposed in [5], used diffusion maps for revealing the hidden structures inside the data and applying k-means clustering for detecting anomalies, whereas [6] compared the performance of Local Outlier Factor (LOF) and One Class Support Vector Machine (OC-SVM) based algorithms for anomaly detection in LTE networks. In [7], the author has provided a complete survey of outage detection techniques in mobile cellular networks.

Our proposed sleeping cells detection approach is different from the traditional techniques in the sense that it uses deep learning to model the normal operating profile of the network by collecting MDT reports from user equipment (UE). The trained model is then able to efficiently detect anomalous behavior in the network at real-time. Furthermore, using the location information stored in the MDT database, every faulty/sleeping cell is localized for autonomous self-healing process in SON. We also quantitatively compare and evaluate the accuracy and average precision of our Deep Autoencoder based Detector with One Class Support Vector Machine based Detector. To our knowledge, no previous study examines the use of deep learning based solution for cell outage / sleeping cell detection in 5G and next generation networks.

The rest of the paper is structured as follows: System model is presented in Section II, which includes the One Class SVM and Deep Autoencoder models for learning the normal network profile and detect faulty cells in the network. Section III presents the simulation configuration, models assessment and evaluation by different performance metrics. Finally section IV concludes the paper.

978-1-5386-4727-1/18/$31.00 ©2018 IEEE

II. System Design

A four step approach is proposed for detection of sleeping cells in cellular networks. It consists of 1) Data collection, 2) Data preprocessing, 3) Data mining and 4) Data visualization. The idea is to model the network during normal operating scenario using MDT measurements and use that model to detect sleeping cells later on.

A. Data Collection

MDT feature for SON was introduced in 3GPP Release 9 [8]. Each MDT measurement sample consists of several KPI's such as latitude, longitude, serving and neighboring cells RSRP and SINR reported by the UE, as specified in Table I. Moreover, time information is appended by the eNB before forwarding it to the Operations and Maintenance system where an MDT database is constructed. The database constructed during normal working scenario of the network act as training data while modeling the normal network behavior, which is then used to detect cell outages during faulty scenario.

TABLE I
Parameters selected from MDT Measurements

Parameters	Description
Serving RSRP	Reference Signal Received Power of serving cell in dBm
Neighbor RSRP	Reference Signal Received Power of three strongest neighbor cells in dBm
Serving SINR	Signal to Interference plus Noise Ratio of serving cell in dB
Neighbor SINR	Signal to Interference plus Noise Ratio of three strongest neighbor cells in dB
Location	Location co-ordinates of a UE
CGI	Cell Global Identity Information of the serving cell

B. Data Preprocessing

Each MDT measurement sample collected is pre-processed before using it for network modeling or testing. Initially data is cleaned by splitting each MDT sample into header and data part. The header contains time, location and cell identity information which is not used for data mining algorithms but for visualization of the results later on. The data part can be represented by a 8-dimensional feature vector as follows:

$$\mathbf{F} = \{RSRP_S, RSRP_{N1}, RSRP_{N2}, RSRP_{N3}, SINR_S, SINR_{N1}, SINR_{N2}, SINR_{N3}\}, \quad (1)$$

where $RSRP_S$ and $SINR_S$ are the serving cell parameters, whereas $RSRP_{Ni}$ and $SINR_{Ni}$ are the ith strongest neighboring cell parameters, i=$\{1,2,3\}$. This 8-dimensional feature vector corresponds to one MDT measurement sample in the training and test databases.

C. Data Mining

To detect cell outages in the network, two state of the art algorithms, One Class Support Vector Machine and Deep Autoencoders are employed for modeling network dynamics and detect anomalies (i.e. sleeping cells).

1) One Class Support Vector Machine based Detector:

To make data mining process faster, simpler and to mitigate the curse of dimensionality, high dimensional dataset is embedded to two dimensions in Euclidean space using Multi-Dimensional Scaling (MDS) method. This critical data transformation step facilitates data modeling and also improves accuracy of anomaly detection algorithms. Moreover, MDS method unlike Principal Component Analysis (PCA), does not assumes linear relationship between the variables and is therefore more suitable for non-linear datasets. The goal of MDS is to map the given data points $\{x_t \in \mathbb{R}^m\}_{t=1}^N$ to their low dimensional embedding points $\{y_t \in \mathbb{R}^n\}_{t=1}^N$ in such a way that the dissimilarity δ_{ij} between x_i and x_j are well approximated by the distance d_{ij} between y_i and y_j. This is an optimization problem and can be done by the minimization of following loss function called Stress:

$$Stress(y_1, .., y_N) = \left(\sum_{i \neq j = 1,..,N} (\delta_{ij} - d_{ij})^2 \right)^{1/2}, \quad (2)$$

where $\delta_{ij} = \|x_i - x_j\|$ and $d_{ij} = \|y_i - y_j\|$. This Stress function is basically the residual sum of squares and the outer square root gives greater spread to small values. The steps in classical MDS algorithm are as follows:

(i) Compute the squared dissimilarity matrix $\Delta^{(2)} = d_{ij}^2$

(ii) Construct a Gram Matrix $\boldsymbol{\mathcal{D}} = -\frac{1}{2}\mathbf{H}^T \Delta^{(2)} \mathbf{H} \in \mathbb{R}^{N \times N}$ by applying double centering to the squared dissimilarity matrix, where $\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}_N \mathbf{1}_N^T$ is the centering matrix

(iii) As $\boldsymbol{\mathcal{D}}$ is power spectral density, we can write it as $\boldsymbol{\mathcal{D}} = \mathbf{X}^T\mathbf{X}$, which after Singular Value Decomposition can be decomposed into $\mathbf{X}^T\mathbf{X} = \mathbf{V}\Lambda\mathbf{V}^T = \mathbf{Y}^T\mathbf{Y}$, where $\mathbf{V} \in \mathbb{R}^{N \times n}$ and $\Lambda \in \mathbb{R}^{n \times n}$

(iv) Solving above equation gives the embedding matrix $\mathbf{Y} = \Lambda^{1/2}\mathbf{V}^T \in \mathbb{R}^{N \times n}$, where $\mathbf{V}$ and Λ are the top n eigenvalues of $\mathbf{X}^T\mathbf{X}$ and their corresponding eigenvectors respectively. That is each column $y_i \in \mathbb{R}^n$ in $\mathbf{Y}$ is an embedding of x_i

In our case $n = 2$ is chosen for easy visual representation of the measurements and to aid the process of outage detection by increasing the variance among the dissimilar measurements, so that the SVM detector can detect network anomalies with higher accuracy and detection rate.

The support vector method for novelty detection described in [9] is different from the traditional two class SVM in the sense that it uses training data from only one class for creating a model and then labels only those measurements from the test dataset as anomalies which are significantly different from the learned model.

Now consider our embedding MDT dataset $\mathbf{F_N}$ in which $y_i \in \mathbb{R}^n$ (where $n = 2$ in our case) are embedding points. SVM will project the data to a high dimensional feature space $\mathbf{S}$ through a non-linear function ϕ by creating a non-linear decision boundary. The hyperplane that is constructed in the feature space determines the margin between the classes and is represented by the equation $\boldsymbol{\omega}^T\mathbf{y} + b = 0$, where $\boldsymbol{\omega} \in \mathbf{S}$ is the norm perpendicular to the hyperplane and $b \in \mathbb{R}$ is the bias of the hyperplane. In One-Class SVM, hyperplane is selected which has the maximum distance from the origin in feature space. This is done by optimally selecting the value of $\boldsymbol{\omega}$ and b through the objective function:

$$\min_{\boldsymbol{\omega},b,\xi_i} \frac{\|\boldsymbol{\omega}\|^2}{2} + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho\,, \tag{3}$$

subject to:

$$(\boldsymbol{\omega}.\phi(y_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0\,.$$

Here ξ_i is the slack variable which creates a soft margin and prevents the classifier from over-fitting by allowing some training data points to lie within the margin, ρ is an offset and $\nu \in (0, 1)$ sets the upper bound on the fraction of margin errors (false positives) and a lower bound on the fraction of support vectors relative to the total number of training examples. This results in a function which returns $+1$ for most of the data points (training data) and -1 elsewhere (outliers). The minimization formulation in Equation 3 after solving for the Lagrange dual results in the following classification rule:

$$f(y) = sgn((\boldsymbol{\omega}.\phi(y_i)) - \rho) = sgn(\sum_{i=1}^{n}\alpha_i K(y_i, y) - \rho)\,. \tag{4}$$

Here α_i are the Lagrange multipliers, patterns y_i with non-zero α_i are the support vectors and $K(y_i, y)$ is the Kernel function, which tricks the SVM classifier by creating non-linear decision boundaries and hence classify non-linear separable datasets also. Popular choices for kernel function are linear, polynomial, sigmoid and Gaussian Radial Base Function (RBF). RBF kernel function, used in our system is:

$$K(y, y^{'}) = exp\left(-\frac{\|y - y^{'}\|^2}{2\sigma^2}\right)\,. \tag{5}$$

Here $\sigma \in \mathbb{R}^n$ is a kernel parameter and $\|y - y^{'}\|$ is the dissimilarity measure. The exact values of the parameters used in our model are selected by grid search using cross validation algorithm explained in next section.

2) Deep Autoencoder based Detector:

Deep Autoencoders described in [10] are feed-forward multi-layer artificial neural network used for unsupervised learning. Like any other neural network, it consists of an input layer, hidden layers in the middle and then an output layer, but autoencoders are trained to reproduce its input at the output layer. Hidden layers are low-dimensional images or representations of the input data. The part of the network that encodes the input data to its latent representations is called an encoder and the decoder is responsible for reconstructing the input from these representations. Now consider a neural network having an input layer $\mathbf{x} = \mathbf{h}^{(1)} \in \mathbb{R}^m$, k hidden layers $\{\mathbf{h}^{(l)} \in \mathbb{R}^n\}_{l=2}^{k+1}$ where $n \in \mathbb{N}, n < m$ are the hidden units or neurons in each hidden layer and the superscript l denotes the index for each layer, then the latent representation of the hidden layers can be written as:

$$\mathbf{h}^{(l+1)} = \sigma^{(l)}\left(\mathbf{W}^{(l)} \cdot \mathbf{h}^{(l)} + \mathbf{b}^{(l)}\right)\,. \tag{6}$$

Here $\sigma(\cdot)$ is the activation function, $\mathbf{W}$ is the weight matrix, $\mathbf{h}$ is the vectorized input and $\mathbf{b}$ is the bias vector. The popular choices for an activation function include sigmoid $\sigma(x) = (1 + e^{-x})^{-1}$, its variant, hyperbolic tangent $\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ and rectified linear unit (ReLU) $\sigma(x) = max(0, x)$. The output of the first layer will become the input for next layer. Lastly the given input sequence is reconstructed at the output layer after k hidden layers as follows:

$$\hat{\mathbf{x}} = \mathbf{h}^{(k+2)} = \sigma^{(k+1)}\left(\mathbf{W}^{(k+1)} \cdot \mathbf{h}^{(k+1)} + \mathbf{b}^{(k+1)}\right)\,. \tag{7}$$

Autoencoders can be as deep as you want but in our experiment we use $k = 3$ hidden layers to model the input training data. Adding hidden layers allows an autoencoder to learn more complex representations, as do non-linear hidden units or neurons. The number of neurons per layer decreases in each subsequent layer of the encoder and increases back in the decoder. Autoencoders are trained to minimize the reconstruction error, for this purpose, a loss function is used to compute the error between the original and reconstructed input after each forward pass.

Typically in deep learning, for binary input data, Cross Entropy loss function is used, but for real valued input data as ours, Mean Squared Error loss function is used. Further to prevent overfitting on the training data, a regularization term is added to the loss function to apply penalty during the optimization phase. This will result in sparser representations of input data. In our case, L_1 activity regularization is used. Loss function can be therefore written as:

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 + \frac{\lambda}{2}\|\hat{\mathbf{x}}\|_1\,. \tag{8}$$

The error is then back-propagated to update the weight matrices after each forward pass to minimize the reconstruction error. Adam's algorithm for stochastic optimization is used for back-propagation which is more computationally efficient than stochastic gradient descent and it combines the advantages of *Adaptive Gradient (AdaGrad)*, which works well with sparse gradients and *Root Mean Square Propagation (RMSProp)*, which works well in non-stationary settings [11]. At each training iteration i, $\mathbf{M}_i$ and $\mathbf{R}_i$, the exponential moving averages of the gradient and the squared gradient respectively are computed as follows:

$$\mathbf{M}_i = \beta_1 \cdot \mathbf{M}_{i-1} + (1 - \beta_1)\nabla\boldsymbol{\mathcal{L}}_i\,, \tag{9}$$

$$\mathbf{R}_i = \beta_2 \cdot \mathbf{R}_{i-1} + (1-\beta_2)(\nabla \boldsymbol{\mathcal{L}}_i)^2 \,. \tag{10}$$

Here $\beta_1, \beta_2 \in [0, 1)$ are the decay rates or forgetting factors of $\mathbf{M}_i$, also known as 1st moment (or Mean) and $\mathbf{R}_i$, also known as 2nd moment (or Variance) of the gradient respectively. Both of these moments are initialized as vectors of 0's, resulting in biasness. Therefore, after bias correction, the resulting moment's estimates are as follows:

$$\hat{\mathbf{M}}_i = \frac{\mathbf{M}_i}{1-(\beta_1)^i} \,, \tag{11}$$

$$\hat{\mathbf{R}}_i = \frac{\mathbf{R}_i}{1-(\beta_2)^i} \,. \tag{12}$$

The updated weight matrix after i[th] iteration will be:

$$\mathbf{W}_i = \mathbf{W}_{i-1} - \alpha \frac{\hat{\mathbf{M}}_i}{\sqrt{\hat{\mathbf{R}}_i} + \epsilon} \,. \tag{13}$$

Here $\alpha > 0$ is the learning rate and $\epsilon > 0$ is a small numerical term used to avoid division by zero. After the training phase, the test MDT measurements are predicted using the learned model and their mean squared error is computed. It has been observed that a threshold θ of MSE exists above which a measurement can be classified as anomalous as shown in Figure 4. In our case, threshold θ is chosen to be two standard deviations above the mean of MSE of test measurements.

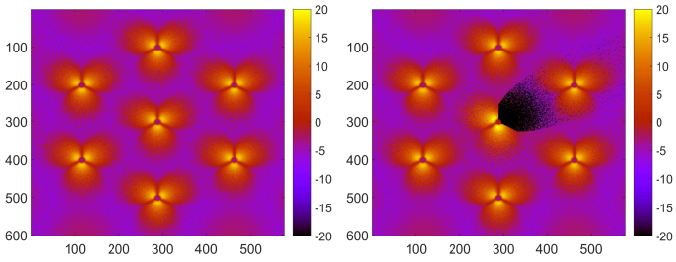
D. Data Visualization

After the anomaly detection phase, every classified measurement is localized on the network topology using the location information stored in header part of the MDT database. This header part was not utilized during the data mining phase and is only used to visualize the sleeping cells after detection. The cell containing the highest number of anomalous measurements can be identified as a sleeping cell.

III. Simulation Results

A. Simulation Configuration

A typical urban macro cell based network topology consisting of 7 cells / 21 sectors is created following 3GPP specifications using a full dynamic network topology simulator by Monte Carlo method. The inter-site distance (ISD) between cells is 1000m whereas the path loss and shadow fading vary with the UE link being LoS or NLoS. Cell association is not changing in this experiment, hence no mobility is considered. The simulation campaign generates the MDT reports from the UE's (consisting of RSRP and SINR of serving and neighboring cells, as explained in Section II-B), for normal as well as outage scenario. To model a sleeping cell during the outage scenario, the antenna gain and transmit power of a BS is attenuated. MDT measurements reported by the users from the normal scenario are used to construct the training dataset, whereas the test dataset is constructed using the reported measurements from outage scenario, which is then used to evaluate the performance of anomaly detection models. The



(a) REM of Normal Scenario (b) REM of Outage Scenario

Fig. 1. (**a**) Radio Environment Map of Normal Scenario (**b**) Radio Environment Map of Outage Scenario where Antenna Gain of one sector of a BS is attenuated to -50 dBi

SINR based Radio Environment Maps are shown in Figure 1. The detailed simulation parameters are listed in Table II.

TABLE II
Simulation Parameters

Parameter	Values
Cellular Layout	7 Macrocell sites
Sectors	3 sectors per BS
Inter Site Distance	1000m
User Distribution	Uniform Random Distribution
Distance dependant Path Loss	$L_{LoS}[dB]$ = 22log D + 34.02, $L_{NLoS}[dB]$ = 39.1log D + 19.56 where D is distance in meters
Slow Fading SD	4 dB (LoS) and 6 dB (NLoS)
BS Transmit Power	46 dBm (Normal Scenario) 0 dBm (SC Scenario)
Antenna Gain	17 dBi (Normal Scenario) -50 dBi (SC Scenario)
Carrier Frequency	2 GHz
Bandwidth	10 MHz

B. Models Assessments and Evaluation

For training and assessing the performance of our anomaly detection models, different algorithms and performance metrics are used. Algorithm 1 explains the training phase of One Class SVM, in which Grid Search and Cross Validation techniques are employed to maximize the performance of the model. Initially a parameter search space is defined for regularization constant ν and kernel parameter σ. In our experiment, logarithmic grid for hyper-parameter optimization is used to search a bigger search space quickly. ν is varied from $\log_{10}[-5:-1]$ and σ is varied from $\log_{10}[-4:1]$. N-fold cross validation (N=10 in our case) is then employed on every unique combination (ν, σ) iteratively to find the optimal hyper-parameter pair with highest performance estimate. This combination is used to train the OCSVM model and Figure 2 shows the classification of test dataset in Euclidean Space.

For training Deep Autoencoder based neural network, as explained in Algorithm 2, 4 fully connected layers with 4, 2,

Algorithm 1 One Class SVM Training Algorithm

Input embedding dataset $\mathbf{F_N}, \nu_{min}, \nu_{max}, \sigma_{min}, \sigma_{max}$
for $\nu = \nu_{min} : \nu_{max}$: **do**
  for $\sigma = \sigma_{min} : \sigma_{max}$: **do**
    Partition the dataset $\mathbf{F}_N$ randomly into n sets
    for $i = 1, 2, ..., n$: **do**
      Train the SVM model using $\mathbf{F}_{n-1}$ and then validate its performance θ_i on $\mathbf{F}_i$
    end for
    Compute average performance θ_{av} over n sets
  end for
end for
Train the model on the parameter ν and σ values corresponding to highest value of θ_{av}

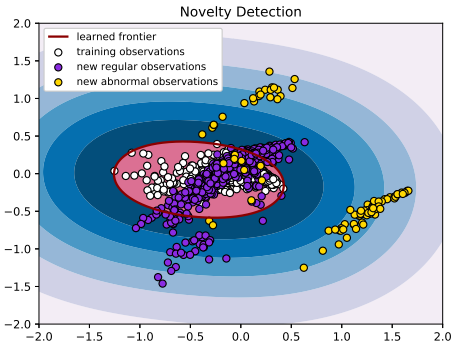


Fig. 2. MDT Measurements in Euclidean Space classified by One Class SVM based Detector

2 and 8 neurons respectively are used, where first 2 layers are used for encoder and last 2 for decoder part. Hyperbolic tangent activation function is used for Layer 1 and 3 whereas ReLU function is used for Layer 2 and 4 in our network. Additionally L_1 activity regularization is used to prevent overfitting on the training data. Training dataset is trained over 100 epochs with mini-batch updates of $batchsize = 32$. Mini-batch updates are typically used because of memory constraints in large datasets and only a subset (32 measurement in our case) is used for each parameter update. Therefore 100 epochs/iterations are performed, so that our algorithm can converge to an acceptable level of accuracy. Initially the weights are randomly set and after one complete forward pass, error is back-propagated to update the weight matrices using Adam's algorithm. The hyper-parameters are initialized with typical default values of $\mathbf{M_0} = 0$, $\mathbf{R_0} = 0$, $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, which are automatically optimized with each iteration to reduce the model loss, as shown in Figure 3. Furthermore, it can be seen from Figure 4, that the reconstruction error of anomalous measurements in test dataset is significantly higher than that of normal measurements.

The performance of our anomaly detection models is evaluated using AUROC and AUPRC performance metrics. AUROC is the Area Under the Receiver Operating Characteristic curve plot, also known as detection accuracy, which is basically the

Algorithm 2 Deep Autoencoder Training Algorithm

Given N input sequences $\mathbf{x}$, k hidden layers, Epoch = N_E, Mini-Batch Size = N_B,
for $h = 1, 2, ..., N_E$: **do**
  for $i = 1, 2, ..., N/N_B$: **do**
    for $l = 2, ..., k+1$: **do**
      Using N_B input samples, compute activations $\mathbf{h}_i^{(l)}$ for l^{th} hidden layers using Eq. (6)
    end for
    Reconstruct input $\hat{\mathbf{x}}_i$ at output layer using Eq. (7) and complete a feed forward pass
    Find reconstruction error through Loss function $\boldsymbol{\mathcal{L}}_i(\mathbf{x}_i, \hat{\mathbf{x}}_i)$ using Eq. (8)
    for $l = k+1, k, k-1, ..., 2$: **do**
      Backpropagate the error by first estimating 1^{st} moment $\mathbf{M}_i^{(l)}$ and 2^{nd} moment $\mathbf{R}_i^{(l)}$ of the gradient of Loss function using Eq. (9) and (10)
      Perform Bias correction on $\mathbf{M}_i^{(l)}$ and $\mathbf{R}_i^{(l)}$ and compute $\hat{\mathbf{M}}_i^{(l)}$ and $\hat{\mathbf{R}}_i^{(l)}$ using Eq. (11) and (12)
      Update the weight matrix $\mathbf{W}_i^{(l)}$ using Eq. (13)
    end for
  end for
end for

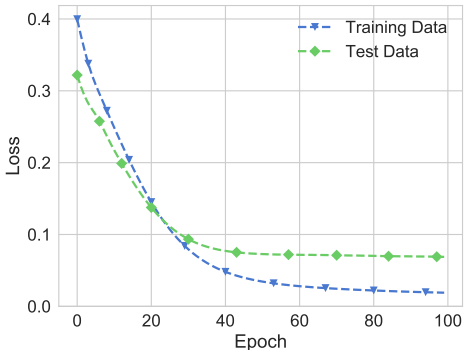


Fig. 3. Model Loss over each Epoch

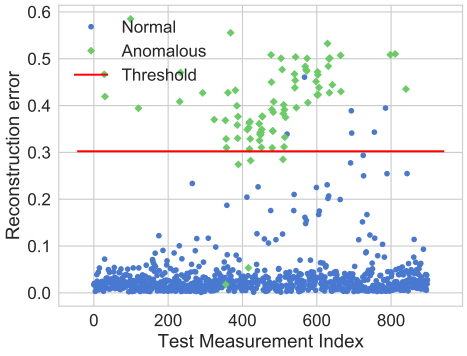
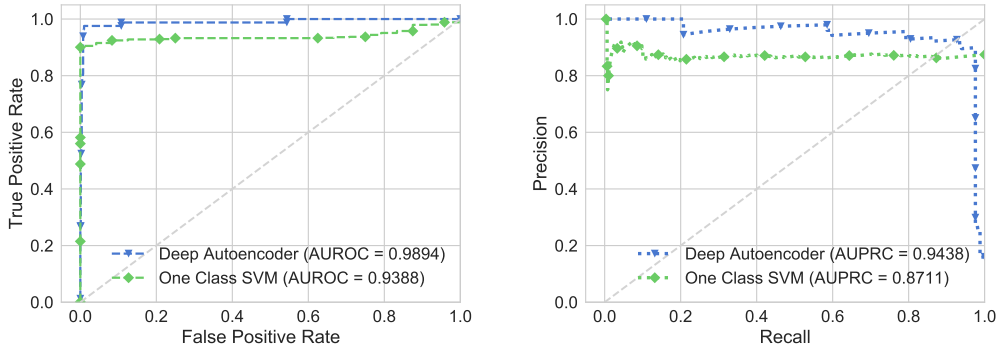


Fig. 4. Reconstruction Error of Test MDT Measurements by Deep Autoencoder



(a) ROC Curve Plot (b) PR Curve Plot

Fig. 5. (**a**) Receiver Operating Characteristic Curve and (**b**) Precision Recall Curve of Deep Auto-encoder and One Class SVM based Detectors

true positive rate of the classifier plotted against the false positive rate at different threshold levels, whereas, AUPRC is the Area Under the Precision Recall Curve, also known as average precision, which is specifically useful for evaluating classifiers on highly imbalanced datasets such as ours, where the majority of test measurements only belong to one class (normal measurements in our case). Here precision is basically the probability of a measurement classified as positive to actually being positive, whereas Recall is just the true positive rate. The precision-recall curve basically highlights the performance differences lost in ROC curves. Figure 5 shows that Deep Autoencoder outperforms One Class SVM, as it has higher AUROC and AUPRC.

TABLE III
PERFORMANCE COMPARISON FOR ANOMALY DETECTION MODELS

Models	**AUROC Score**	**AUPRC Score**
Deep Autoencoder	0.99	0.94
One Class SVM	0.94	0.87

IV. CONCLUSION

This paper has presented an approach for autonomic detection of sleeping cells using deep learning technique. The conventional methods for outage detection are time-consuming as they involve drive tests and complaints from the customers, hence our proposed autonomic framework will significantly reduce the duty cycle of self-healing function in SON. In the proposed framework, MDT reports collected from the end users are used to construct a database, which after pre-processing is fed to our data mining framework. Deep Autoencoders are used to train the normal network model. Anomalous measurements in test dataset can then be identified based on their higher reconstruction error than normal ones. We then compare the performance of Deep Autoencoder and One Class SVM based detector for sleeping cell detection. Our experiments have shown that Deep Autoencoder based detector has higher detection accuracy (AUROC) and average precision (AUPRC) as compared to One Class SVM, as listed in Table III. Ultimately, sleeping cells can be visualized using the location information stored in each MDT measurement. The proposed sleeping cell detection framework can be extended to also identify other performance degradation problems in the network.

V. ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant Numbers 1559483, 1619346 and 1730650. For further information about these projects please visit www.AI4Networks.com.

REFERENCES

[1] O. G. Aliu, A. Imran, M. A. Imran, and B. Evans, "A survey of self organisation in future cellular networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 336–361, 2013.

[2] B. Cheung, S. Fishkin, G. Kumar, and S. Rao, "Method of monitoring wireless network performance," Mar. 23 2006, uS Patent App. 10/946,255.

[3] C. M. Mueller, M. Kaschub, C. Blankenhorn, and S. Wanke, "A cell outage detection algorithm using neighbor cell list reports," in *International Workshop on Self-Organizing Systems*. Springer, 2008, pp. 218–229.

[4] Q. Liao and S. Stanczak, "Network state awareness and proactive anomaly detection in self-organizing networks," in *Globecom Workshops (GC Wkshps), 2015 IEEE*. IEEE, 2015, pp. 1–6.

[5] F. Chernogorov, J. Turkka, T. Ristaniemi, and A. Averbuch, "Detection of sleeping cells in lte networks using diffusion maps," in *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*. IEEE, 2011, pp. 1–5.

[6] A. Zoha, A. Saeed, A. Imran, M. A. Imran, and A. Abu-Dayya, "Data-driven analytics for automated cell outage detection in self-organizing networks," in *Design of Reliable Communication Networks (DRCN), 2015 11th International Conference on the*. IEEE, 2015, pp. 203–210.

[7] A. Asghar, H. Farooq, and A. Imran, "Self-healing in emerging cellular networks: Review, challenges and research directions," *IEEE Communications Surveys & Tutorials*, 2018.

[8] *Technical Specification Group Radio Access Network. Study on Minimization of drive-tests in Next Generation Networks*, 3GPP, 12 2009, version 9.0.0.

[9] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, 2000, pp. 582–588.

[10] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.