Detecting Performance Degradation in Cloud Systems using LSTM Autoencoders

Spyridon Chouliaras and Stelios Sotiriadis

Abstract Cloud computing technology is on the rise as it provides an easy to scale environment for Internet users in terms of computational resources. At the same time, cloud providers manage this demand for computational power by offering a pay per use model for virtualized resources. Yet, it is a challenging issue to administer the variety of different cloud applications and ensure high performance by identifying failures and errors on runtime. Distributed applications are error-prone, and creating a platform to support minimum hardware and software failures is a key challenge. In this work, we focus on anomaly detection of data storage systems, and we propose a solution for detecting performance degradation of cloud deployed systems in real time. We use Long Short-term Memory (LSTM) Autoencoders for learning the normal representations and reconstruct the input sequences. Then, we used the reconstructed errors of the LSTM Autoencoders on unseen time series data to detect abnormal behaviours. We used state-of-the-art benchmarks such as TPCx-IoT and YCSB to evaluate the performance of HBase and MongoDB systems. Our experimental analysis shows the ability of the proposed approach to detect abnormal behaviours in cloud systems.

1 Introduction

Nowadays, cloud computing offers a state-of-the-art technology for organisations that are looking for scalable application development. In a cloud system, applications often deployed as virtual machines (VM) that provide customised technical

Spyridon Chouliaras
Birkbeck, University of London, Malet St, Bloomsbury, London WC1E 7HX, e-mail: `s.chouliaras@dcs.bbk.ac.uk`

Stelios Sotiriadis
Birkbeck, University of London, Malet St, Bloomsbury, London WC1E 7HX, e-mail: `stelios@dcs.bbk.ac.uk`

characteristics, operating systems and software packages. As the demand is on the rise, cloud providers are promising cloud elasticity and scalability on their systems as well as infinite amounts of virtual resources. Cloud scalability is the ability of the system to accommodate larger loads, while cloud elasticity is the ability of the system to scale with loads dynamically [1]. In addition, data are becoming massive in terms of volume, variety and velocity, while different techniques have been implemented in order to deal with this new phenomenon. A NoSQL (meaning 'not only SQL') has come to describe a large class of databases sufficient to deal with massive amounts of data, supporting also non-schema structure and real-time analysis [2]. Such systems can store data from different sources without a specific structure, a characteristic that makes them powerful for different applications such as e-commerce, navigation systems, web services and IoT. However, the robustness and the reliability of cloud systems remains a key problem for information technology companies as cloud systems could suffer from hardware, software and network failures. Therefore, efficient ways on detecting abnormalities are more important than ever, since anomalies in the system can cause performance degradation, unreasonable energy consumption as well as false scaling estimation from automatised algorithms. The latter, indicates the importance of finding techniques in order to detect abnormal cases based on real time usage and develop algorithms able to predict future failures.

We present Long Short-Term Memory (LSTM) Autoencoders for detecting performance degradation in cloud systems. LSTM Autoencoders have been proved effective for time series learning and anomaly detection [3]. The model trained beforehand on normal sequence representations and learned to reconstruct the input sequence with the lowest possible error. Then, the reconstructed error used as a score in order to classify future time series as normal or abnormal. New unseen observations that have lower reconstruction error than a given threshold are being classified as normal while observations that have higher reconstruction error are being classified as abnormal. The threshold value of the reconstructed error has been tuned as a hyper-parameter based on its ability to detect abnormal cases of various systems.

2 Related Works

Anomaly detection refers to the process of identifying data points, events or observations that deviate from the normal data distribution. Deep learning based anomaly detection methods have been proposed in many domains [4, 5, 6, 7]. Amongst them, authors proposed LSTM networks as powerful sequence learners that promise a substantial improvement over the conventional Artificial Neural Networks. In extension to this, we propose LSTM based Autoencoders to detect outliers that indicate application performance degradation. LSTM Autoencoders combine the ability of the Autoencoder to extract the most representative information with the advantage of the LSTM to effectively process sequential data. For that reason, LSTM Autoencoders have been applied widely for anomaly detection [8, 9, 10, 11]. In [8] authors proposed an LSTM based Encoder-Decoder scheme for Anomaly detection as an alternative of standard approaches. In their findings, the LSTM Autoencoder learns to reconstruct normal time-series behaviour and by using the reconstruction error effectively detects anomalies from predictable, unpredictable, periodic, aperiodic and quasi-periodic time series. In [9] authors proposed an LSTM Autoencoder network-based method combined with a one-class support vector machine algorithm used for anomaly detection. The LSTM Autoencoder network trained on normal representations and used to calculate a prediction error vector. Then, the one-class Support Vector Machine algorithm used to separate the abnormal observations from normal samples based on these prediction error vectors.

In [10] authors proposed CANolo, an intrusion detection system based on LSTM Autoencoder to identify anomalies in Controller Area Networks. Their framework automatically trained on controller area networks streams to build a model based on the legitimate data sequences. Then the reconstructed error used to detect anomalies between legitimate sequences and simulated attacks. In [11] authors used mobile traffic data and proposed an LSTM Autoencoder to reconstruct mobile traffic samples and an LSTM traffic predictor to predict the traffic of future time instants. In both cases they analyzed the reconstructed and the predicted error to assess if the mobile traffic presents anomalies or not.

In this work, we propose LSTM Autoencoders to reconstruct normal sequence representations of application metric data streamed from MongoDB and HBase systems. Then unseen observations are being classified as normal or abnormal on the fly based on the reconstruction error produced by the LSTM Autoencoder.

3 Methodology

A variety of metrics captured over a period of time from monitoring systems to support our method. Prometheus[1] has been used as a monitoring system for MongoDB[2] and HBase[3] to collect application and system metrics. In addition, Yahoo! Cloud Serving Benchmark (YCSB) [12] and TPCx-IoT [4] workloads used to execute various intensive tasks. Then, under the assumption that YCSB and TPCx-IoT generate repeatable patterns, we created a dataset to train deep learning models that learn normal application behaviours. For that purpose an LSTM Autoencoder has been trained and used for pattern recognition of long-range dependencies and anomaly detection.

[1] `https://prometheus.io/`

[2] `https://www.mongodb.com/`

[3] `https://hbase.apache.org/`

[4] `http://www.tpc.org/tpcx-iot/`

3.1 Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) model is a type of a Recurrent Neural Network that promises a substantial improvement in sequential data with temporal sequences and long-range dependencies [13]. An LSTM model contains special units called memory cells that are organised inside the recurrent hidden layer. Each memory cell contains an input gate, an output gate and a forget gate. The forget gate has been subsequently added in order to enable processing continuous input streams that are not segmented into subsequences [14]. The forget gate gives the ability to LSTM cell to learn to reset itself at appropriate times and prevent the network to break down in situations where time series grow arbitrary through time (e.g. continuous input streams) [15].

An LSTM receives an input sequence $x = (x_1, x_2, x_3, ..., x_T)$ and maps it to an output sequence $y = (y_1, y_2, y_3, ..., y_T)$ by calculating the following equations iteratively from t=1 to T:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f) \tag{2}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \tag{3}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \tag{4}$$

$$m_t = o_t \odot h(c_t) \tag{5}$$

$$y_t = \phi(W_{ym}m_t + b_y) \tag{6}$$

where the i is the input gate that uses the logistic sigmoid function denoted as σ in order to control the information that flows into the cell, f is the forget gate that uses the logistic sigmoid function to decide what information to keep outside the cell state, o is the output gate that uses the logistic sigmoid function to control the information that flows out of the cell, c is the cell activation vectors, m is the output activation vector where $\odot$ is the element-wise product of the vectors, g is the cell input activation function, h is the cell output activation function which is usually the $tanh$ function, ϕ is the output activation function of the output sequence y which is usually the $softmax$ function, W terms denote weight matrices and b terms denote bias vectors.

3.2 LSTM Autoencoder for Anomaly Detection

Autoencoder is a fundamental paradigm of unsupervised learning with the ability to reconstruct input features with the least possible amount of distortion [16]. The Autoencoder consists of an encoder and a decoder. The encoder extracts the hidden features from the input data and encodes the information into a learned representation vector. Then, the decoder receives the encoded vector as an input and reconstructs the original sequence with a reconstruction error.

LSTM Autoencoder combines the ability of the Autoencoder to extract the most representative information with the advantage of the LSTM to process sequential data with long-range dependencies. We use normal data to train the LSTM Autoencoder that learns to reconstruct normal univariate time series data while on the contrary produces high reconstruction error on abnormal time series. Thus, the reconstruction error used as a score to classify future data points as normal or abnormal. Figure 1 shows the architecture of the LSTM Autoencoder. We use one LSTM layer for the encoder and one LSTM layer for the decoder. The encoder takes the input sequence $(x_1, x_2, ..., x_n)$ and encodes the information into an encoded feature vector. Then, the encoded feature vector is being used as an input for the decoder layer that tries to reconstruct the original sequence into $(\widehat{x_1}, \widehat{x_2}, ..., \widehat{x_n})$.

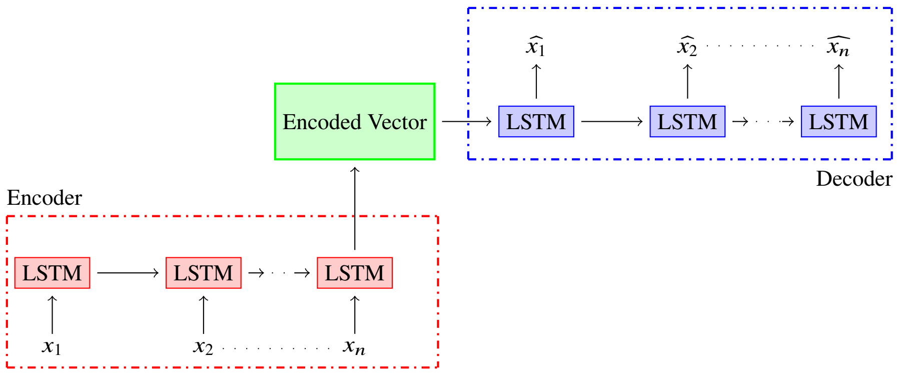

Fig. 1 Univariate Time Series Data Encoding based on LSTM Autoencoder

The LSTM Autoencoder is trained to minimize the Mean Absolute Error (MAE) that measures the average of the absolute differences between the observed and the reconstructed observation as follows

$$MAE = \frac{\sum_{i=1}^{n} |x_i - \widehat{x_i}|}{n} \tag{7}$$

where x_i and $\widehat{x_i}$ are the actual and the reconstructed observation respectively. Then, the reconstruction error is used as a score to detect future outliers. Since the LSTM Autoencoder trained on normal sequences, it produces low reconstruction error on future normal series while on the contrary generates high reconstruction error on abnormal sequences. Thus, new observations are being classified as normal if the reconstruction error is lower than user's threshold or abnormal if it is not.

4 Detecting Performance Degradation in Cloud Systems

In this section, experimental scenarios are being discussed and visualized to demonstrate the effectiveness of our solution. We verify the performance of our method based on YCSB and TPCx-IoT as real world workload benchmarks that execute in MongoDB and HBase system respectively. It includes, (a) the experimental setup and the benchmark analysis, (b) LSTM Autoencoder for anomaly detection of MongoDB system and (c) LSTM Autoencoder for anomaly detection of HBase system.

4.1 Experimental Setup and Benchmark Analysis

We developed the experiments using two infrastructures to support MongoDB and HBase deployment. The first environment consists of a single Virtual Machine (VM) that is running a linux operating system with 2 CPU cores, 8 GB RAM and 512 GB Hard Disk Drive (HDD). The VM hosts MongoDB as a real world NoSQL application system to be monitored by Prometheus. While YCSB workload executes a mix of 50/50 reads and writes, Prometheus collects and stores applications metrics with equally spaced points in time. The second environment consists a cluster of 2 nodes each configured with 2 CPU cores, 8 GB RAM and 256 GB HDD. Each node consists an HBase server that runs on top of Hadoop Distributed File System (HDFS). The TPCx-IoT workload used as representative of activities typical in IoT gateway systems. Similarly, Prometheus collects and stores application metrics in real time.

4.2 Performance Anomaly Detection of MongoDB System

In this section, we evaluate the performance of LSTM Autoencoder to detect anomalies in MongoDB system based on application throughput. Figure 2 shows application throughput under YCSB workload execution. The first wavelet, illustrates the load phase of YCSB workload where 200,000 records loaded in the database. The second wavelet shows the run stage where YCSB uses a mix of 50% read and 50% write operations. In total, 5 load and 5 run stages have been used alternately to create 10 representative wavelets. Figure 2 suggest that executing YCSB workload with the same configurations generate a repeatable application throughput pattern. Consequently, the LSTM Autoencoder learns to reconstruct normal throughput representations by minimizing the loss function, that is, the mean absolute error. The reconstruction error has been used as a score to detect future abnormalities in the throughput.

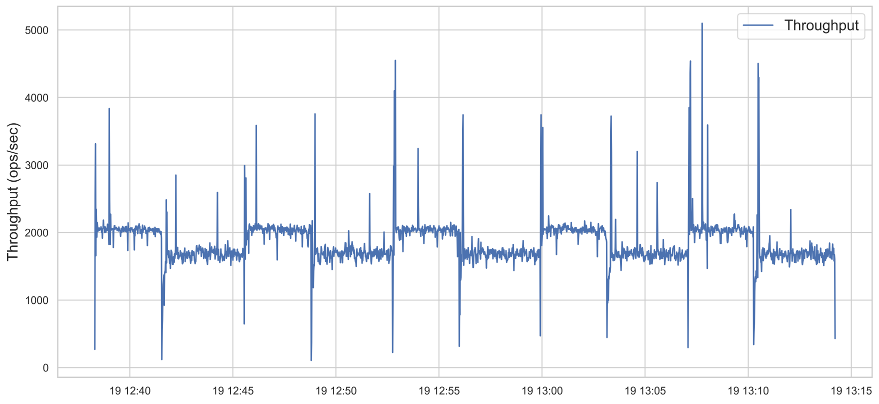

Fig. 2 Throughput of MongoDB on YCSB workload.

To demonstrate the effectiveness of our method, we used Stress linux package to introduce additional intensive tasks that impact application performance. Figure 3 shows a continuity of Figure 2 with two additional wavelets that represent the load and the run stage of YCSB workload. While YCSB executed the load stage under normal conditions (11th wavelet), in the run stage (12th wavelet) we execute Stress package as an additional intensive task. The latter, impacts the application throughput as it creates abnormal patterns shown in Figure 3.

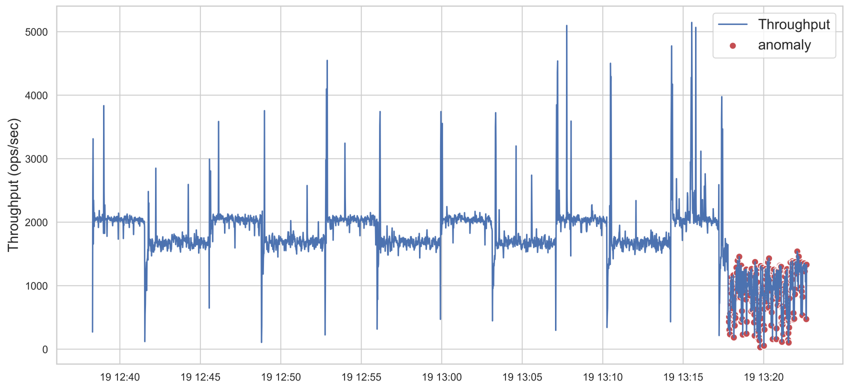

Fig. 3 Throughput of MongoDB on YCSB workload with Stress.

We effectively detect throughput degradation by using LSTM Autoencoder reconstruction error. Since the LSTM Autoencoder trained to reconstruct normal throughput patterns, it produces high reconstruction error on throughput values that deviate from the normal data distribution. Then we classify throughput observations as normal or abnormal based on a predefined application threshold. If the reconstructed error of a given observation surpasses the application threshold, then the observation is tagged as abnormal. The application threshold consists a tuning parameter based on application functionality. If the application is resilient in throughput fluctuations, a high threshold value is suggested to detect only extreme outliers. On the other hand, if the application is sensitive on throughput changes, a lower threshold will be less forgiving and more values will be classified as abnormal.

4.3 Performance Anomaly Detection of HBase System

In this section, we demonstrate the ability of the LSTM Autoencoder to detect abnormal values in IoT sensor data. We deployed an HBase cluster with 2 nodes while TPCx-IoT benchmark used to model sensor data generated by power substations. Figure 4 shows the throughput of the HBase cluster with 2 nodes while we execute TPCx-IoT workload. In total the TPCx-IoT workload generated 8 wavelets that used as a representative normal input sequence for the LSTM Autoencoder. Then, the LSTM Autoencoder learns to reconstruct the input sequence with the lowest possible error rate. We use the reconstruction error as a score in order to classify future throughput values as normal or abnormal.

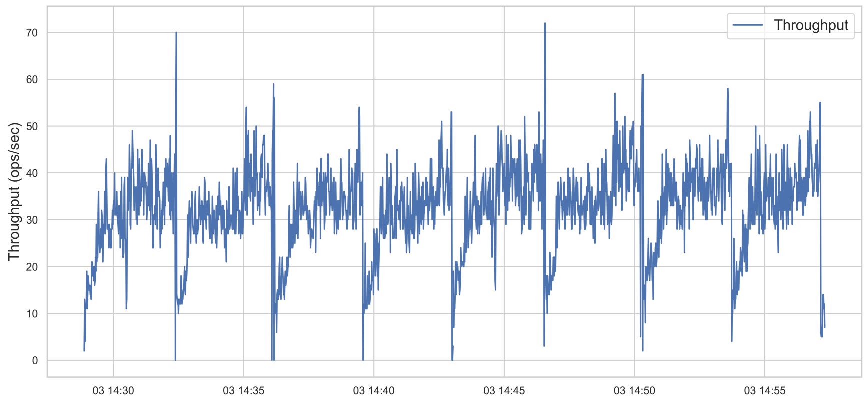

Fig. 4 Throughput of HBase on TPCx-IoT workload.

Figure 5 is a continuity of Figure 4 that shows the execution of the TPCx-IoT workload while at the same time network delays introduced in the system. We demonstrate the ability of the LSTM Autoencoder to detect abnormal throughput values by artificially injecting two network delays of 200ms into the network system. The first delay injected between 14:58 and 15:00 while the second delay injected between 15:07 and 15:09. As a result, the throughput of the system plummets to zero between the time window of the injected network delay.

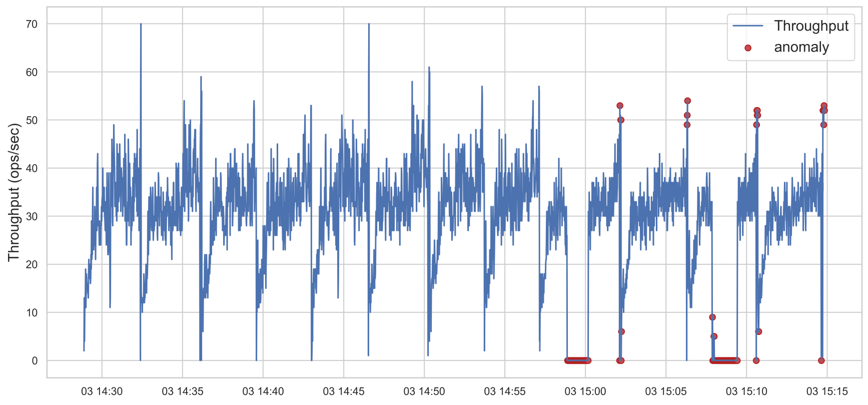

Fig. 5 Throughput of HBase on TPCx-IoT workload with Network Delay.

As Figure 5 shows LSTM Autoencoder successfully detects abnormal cases including the throughput values between the network delay time windows. As discussed earlier, we classify throughput observations as normal or abnormal based on a predefined application threshold. Thus, observations with a reconstruction error higher than the threshold were tagged as abnormal. The threshold has been tuned according to application functionality to detect values that significantly vary from the normal distribution.

5 Conclusion

In this study, we presented LSTM Autoencoders for detecting performance degradation in cloud systems. We evaluated the ability of our method by using various NoSQL systems and real world workload benchmarks. YCSB workload used to execute read and write operations in MongoDB that generate repeatable throughput patterns. Additionally, to show the ability of our method to adapt in various applications, we used an HBase cluster and executed TPCx-IoT benchmark that models sensor data. In our findings, LSTM Autoencoders successfully detected performance outliers in both systems under different scenarios such as intensive system tasks and network system delays. The LSTM Autoencoders trained on normal sequence representations to minimize the reconstruction error. Then, a threshold application parameter used as an indicator to classify future values as normal or abnormal.

Our method detects abnormal application behaviours under various tasks. We believe that our accurate and low overhead anomaly detection tool introduces a real time approach by using application metrics, as strong indicators of application overall performance. Detecting degradation of application performance increases the reliability of cloud systems and promises strong monitoring tools to cloud system administrators.

References

1. Stelios Sotiriadis, Nik Bessis, Cristiana Amza, and Rajkumar Buyya. Vertical and horizontal elasticity for dynamic virtual machine reconfiguration. *IEEE Transactions on Services Computing*, (99):1–1, 2016.
2. Venkat N Gudivada, Dhana Rao, and Vijay V Raghavan. Nosql systems for big data management. In *2014 IEEE World congress on services*, pages 190–197. IEEE, 2014.
3. Zeineb Ghrib, Rakia Jaziri, and Rim Romdhane. Hybrid approach for anomaly detection in time series data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.
4. Arnamoy Bhattacharyya, Seyed Ali Jokar Jandaghi, Stelios Sotiriadis, and Cristiana Amza. Semantic aware online detection of resource anomalies on the cloud. In *2016 IEEE international conference on cloud computing technology and science (CloudCom)*, pages 134–143. IEEE, 2016.
5. Spyridon Chouliaras and Stelios Sotiriadis. Real time anomaly detection of nosql systems based on resource usage monitoring. *IEEE Transactions on Industrial Informatics*, 2019.
6. Ritesh K Malaiya, Donghwoon Kwon, Jinoh Kim, Sang C Suh, Hyunjoo Kim, and Ikkyun Kim. An empirical evaluation of deep learning for network anomaly detection. In *2018 International Conference on Computing, Networking and Communications (ICNC)*, pages 893–898. IEEE, 2018.
7. Cheng Feng, Tingting Li, and Deeph Chana. Multi-level anomaly detection in industrial control systems via package signatures and lstm networks. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 261–272. IEEE, 2017.
8. Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.
9. HD Nguyen, Kim Phuc Tran, S Thomassey, and M Hamad. Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management. *International Journal of Information Management*, page 102282, 2020.
10. Stefano Longari, Daniel Humberto Nova Valcarcel, Mattia Zago, Michele Carminati, and Stefano Zanero. Cannolo: An anomaly detection system based on lstm autoencoders for controller area network. *IEEE Transactions on Network and Service Management*, 2020.
11. Hoang Duy Trinh, Engin Zeydan, Lorenza Giupponi, and Paolo Dini. Detecting mobile traffic anomalies through physical control channel fingerprinting: A deep semi-supervised approach. *IEEE Access*, 7:152187–152201, 2019.
12. Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 143–154, 2010.
13. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
14. Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.
15. Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.
16. Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49, 2012.