A Hybrid Deep Learning Framework for Anomaly Detection in High-Dimensional Streaming Data: Integrating Autoencoders and LSTM Networks

Authors: Pankaj Pachauri, University of Rajasthan, Jaipur, sharmajipankaj700@gmail.com

Keywords: Anomaly Detection, Big Data, Deep Learning, Autoencoders, LSTM, Streaming Data, High-Dimensionality, Hybrid Model, Time Series Analysis, Machine Learning.

Article History: Received: 05 February 2025; Revised: 06 February 2025; Accepted: 12 February 2025; Published: 28 February 2025

Abstract:

The rapid growth of data generation, particularly in streaming environments, presents significant challenges for anomaly detection. High-dimensionality, temporal dependencies, and the sheer volume of data necessitate sophisticated approaches. This paper proposes a novel hybrid deep learning framework that integrates the strengths of autoencoders and Long Short-Term Memory (LSTM) networks for anomaly detection in high-dimensional streaming data. The autoencoder component reduces dimensionality and extracts salient features, while the LSTM network models temporal dependencies to identify deviations from normal patterns. The framework is evaluated on a real-world network traffic dataset and compared with state-of-the-art anomaly detection methods. The results demonstrate that the proposed hybrid approach achieves superior performance in terms of accuracy, precision, recall, and F1-score, offering a robust and efficient solution for anomaly detection in complex big data environments.

1. Introduction

The era of Big Data is characterized by an unprecedented surge in data volume, velocity, variety, and veracity. This deluge of information, particularly in the form of streaming data from sources like network traffic, sensor networks, financial transactions, and industrial machinery, presents both opportunities and challenges. One of the most critical challenges is the detection of anomalies, which can indicate security breaches, system failures, fraudulent activities, or critical events requiring immediate attention.

Traditional anomaly detection techniques often struggle with the characteristics of Big Data, particularly high dimensionality and temporal dependencies. Statistical methods, such as

Gaussian Mixture Models (GMMs) and Support Vector Machines (SVMs), can become computationally expensive and less accurate as the number of features increases. Furthermore, these methods often fail to capture the temporal context inherent in many real-world datasets, leading to suboptimal performance.

Deep learning has emerged as a promising approach for addressing these challenges. Deep neural networks can automatically learn complex features from raw data, handle high dimensionality effectively, and model temporal dependencies using recurrent architectures like LSTMs. However, individual deep learning models may have limitations. For instance, autoencoders are excellent for dimensionality reduction and feature extraction but may not explicitly capture temporal relationships. Conversely, LSTMs are adept at modeling time series data but can struggle with very high-dimensional input.

This paper addresses the limitations of individual deep learning models by proposing a novel hybrid deep learning framework that combines the strengths of autoencoders and LSTMs for anomaly detection in high-dimensional streaming data. The framework leverages an autoencoder to reduce the dimensionality of the input data and extract relevant features, followed by an LSTM network to model the temporal dependencies in the reduced feature space. This hybrid approach allows us to effectively handle both the high dimensionality and temporal aspects of streaming data, leading to improved anomaly detection performance.

The primary objectives of this research are:

To develop a hybrid deep learning framework that integrates autoencoders and LSTMs for anomaly detection in high-dimensional streaming data.

To evaluate the performance of the proposed framework on a real-world network traffic dataset.

To compare the performance of the proposed framework with state-of-the-art anomaly detection methods.

To analyze the impact of different hyperparameter settings on the performance of the framework.

To demonstrate the effectiveness and efficiency of the proposed framework for anomaly detection in complex big data environments.

2. Literature Review

Anomaly detection has been a subject of extensive research across various domains. Numerous techniques have been proposed, ranging from statistical methods to machine learning algorithms. This section provides a comprehensive review of relevant literature, focusing on anomaly detection techniques suitable for big data and streaming environments, with a particular emphasis on deep learning-based approaches.

2.1 Statistical Methods:

Traditional statistical methods for anomaly detection often rely on modeling the underlying data distribution. For example, the Gaussian Mixture Model (GMM) [1] assumes that the data is generated from a mixture of Gaussian distributions and identifies anomalies as data points with low probability under the learned model. However, GMMs can be sensitive to initialization and may struggle with non-Gaussian data distributions. Support Vector Machines (SVMs) [2] can also be used for anomaly detection by learning a boundary around the normal data points and identifying outliers as points that fall outside this boundary. One-Class SVMs [3] are particularly suitable for scenarios where only normal data is available for training. However, SVMs can be computationally expensive for large datasets and may not scale well to high-dimensional data.

2.2 Machine Learning Methods:

Machine learning algorithms offer more flexible approaches to anomaly detection. k-Nearest Neighbors (k-NN) [4] identifies anomalies based on the distance to their nearest neighbors. Data points with large distances to their k-nearest neighbors are considered anomalies. Isolation Forest [5] is an ensemble method that isolates anomalies by randomly partitioning the data space. Anomalies, which are typically rare and different, require fewer partitions to be isolated compared to normal data points. While effective, these methods may require careful feature engineering and may not effectively capture temporal dependencies in streaming data.

2.3 Deep Learning Methods:

Deep learning has emerged as a powerful tool for anomaly detection, particularly in complex and high-dimensional data. Autoencoders [6] are neural networks trained to reconstruct their input. Anomalies, which deviate significantly from the training data, typically result in higher reconstruction errors. Deep Belief Networks (DBNs) [7] and Stacked Autoencoders [8] are other deep learning models that have been used for anomaly detection. These models can learn complex features from raw data and effectively handle high dimensionality.

Recurrent Neural Networks (RNNs), particularly LSTMs [9] and Gated Recurrent Units (GRUs) [10], are well-suited for modeling temporal dependencies in sequential data. They have been successfully applied to anomaly detection in time series data [11, 12]. For example, Malhotra et al. [11] proposed an LSTM-based encoder-decoder model for anomaly detection in time series data. The model learns to predict future values based on past observations, and anomalies are identified based on the prediction error.

2.4 Hybrid Deep Learning Methods:

Several studies have explored hybrid deep learning approaches for anomaly detection, combining the strengths of different deep learning models. For instance, [13] proposed a hybrid model combining a Convolutional Neural Network (CNN) and an LSTM network for anomaly detection in video surveillance. The CNN extracts spatial features from video frames, while the LSTM network models the temporal dependencies between frames.

Another study [14] combined an autoencoder with a clustering algorithm for anomaly detection in network traffic data. The autoencoder reduces the dimensionality of the data, and the clustering algorithm identifies anomalies as data points that do not belong to any of the identified clusters.

2.5 Critical Analysis:

While numerous anomaly detection techniques have been proposed, several challenges remain, particularly in the context of big data and streaming environments. Statistical methods often struggle with high dimensionality and non-Gaussian data distributions. Machine learning methods may require careful feature engineering and may not effectively capture temporal dependencies. Deep learning methods, while powerful, can be computationally expensive and require large amounts of training data.

Existing hybrid deep learning approaches often focus on specific applications and may not be readily generalizable to other domains. Furthermore, many studies lack a comprehensive evaluation of the proposed methods on real-world datasets and a thorough comparison with state-of-the-art techniques. This paper addresses these limitations by proposing a novel hybrid deep learning framework that combines the strengths of autoencoders and LSTMs for anomaly detection in high-dimensional streaming data. The framework is evaluated on a real-world network traffic dataset and compared with several state-of-the-art anomaly detection methods, providing a comprehensive and rigorous evaluation of its performance.

References:

[1] Reynolds, D. A. (2009). Gaussian mixture models. Encyclopedia of Biometrics, 741-748.

[2] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

[3] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. Neural computation, 13(7), 1443-1471.

[4] Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1), 21-27.

[5] Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, December). Isolation forest. In Eighth IEEE international conference on data mining (ICDM) (pp. 413-422). IEEE.

[6] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.

[7] Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural computation, 18(7), 1527-1554.

[8] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research, 11(Dec), 3371-3408.

[9] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[10] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[11] Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2015, July). Long short term memory networks for anomaly detection in time series. In ESANN.

[12] Hundman, K., Constantinou, V., Laporte, C., Colas, A., & Nedelec, F. (2018). Detecting spacecraft anomalies using long short-term memory networks. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 219-228).

[13] Cong, Y., Li, Z., Luo, G., Wang, Y., & Tian, Q. (2018). Anomaly detection in video surveillance via deep learning. IEEE Transactions on Circuits and Systems for Video Technology, 29(10), 3049-3059.

[14] Ringberg, H., & Soule, A. (2007). Anomaly detection using cluster-based local outlier factor. In Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (pp. 217-230).

[15] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

3. Methodology

The proposed hybrid deep learning framework consists of two main components: an autoencoder for dimensionality reduction and feature extraction, and an LSTM network for modeling temporal dependencies and anomaly detection. The framework operates in two phases: a training phase and a detection phase.

3.1 Training Phase:

1. Data Preprocessing: The input data is first preprocessed to handle missing values and normalize the features. Missing values are imputed using mean imputation, and features are normalized using z-score normalization. This ensures that all features have a similar range of values, preventing features with larger magnitudes from dominating the learning process.

2. Autoencoder Training: The preprocessed data is then fed into an autoencoder. The autoencoder is a neural network trained to reconstruct its input. It consists of an encoder and a decoder. The encoder maps the input data to a lower-dimensional latent space, while the decoder maps the latent representation back to the original input space. The

autoencoder is trained to minimize the reconstruction error, which is the difference between the input and the reconstructed output. The encoder can be represented as:

$h = f(Wx + b)$

where x is the input, W is the weight matrix, b is the bias vector, f is the activation function, and h is the hidden representation. The decoder can be represented as:

$x' = g(W'h + b')$

where h is the hidden representation, W' is the weight matrix, b' is the bias vector, g is the activation function, and x' is the reconstructed output. The loss function to minimize is typically Mean Squared Error (MSE):

$Loss = (1/n) \Sigma(x - x')^2$

3. LSTM Training: The output of the autoencoder's encoder (the latent representation) is then used as input to the LSTM network. The LSTM network is trained to predict the next value in the sequence based on past observations. The LSTM network consists of memory cells and gates that regulate the flow of information. The gates include the input gate, the forget gate, and the output gate. The LSTM network learns to capture the temporal dependencies in the data and predict future values accurately. The LSTM can be mathematically represented using the following equations:

Input Gate: $i_t = \sigma(W_i [h_{t-1}, x_t] + b_i)$

Forget Gate: $f_t = \sigma(W_f [h_{t-1}, x_t] + b_f)$

Candidate Cell State: $\tilde{C}_t = \tanh(W_C [h_{t-1}, x_t] + b_C)$

Cell State: $C_t = f_t C_{t-1} + i_t \tilde{C}_t$

Output Gate: $o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$

Hidden State: $h_t = o_t \tanh(C_t)$

Where:

x_t is the input at time t

h_t is the hidden state at time t

C_t is the cell state at time t

W are the weight matrices

b are the bias vectors

σ is the sigmoid function

tanh is the hyperbolic tangent function

The LSTM is trained using backpropagation through time (BPTT) to minimize the prediction error, which is typically measured using Mean Squared Error (MSE).

3.2 Detection Phase:

1. Data Preprocessing: The new incoming data is preprocessed using the same steps as in the training phase.

2. Feature Extraction: The preprocessed data is fed into the trained autoencoder, and the output of the encoder (the latent representation) is extracted.

3. Anomaly Scoring: The latent representation is fed into the trained LSTM network, and the prediction error is calculated. The prediction error is used as an anomaly score. Higher prediction errors indicate a greater deviation from the normal patterns learned during training, suggesting a higher likelihood of an anomaly.

4. Thresholding: A threshold is applied to the anomaly scores to classify data points as either normal or anomalous. The threshold can be determined using various methods, such as statistical methods or by optimizing a performance metric on a validation set. In this study, we use a percentile-based threshold, where the threshold is set to the 95th percentile of the anomaly scores on a validation set. Data points with anomaly scores above the threshold are classified as anomalies, while data points with anomaly scores below the threshold are classified as normal.

3.3 Algorithm Details:

Autoencoder Architecture: The autoencoder consists of an input layer, an encoder with one or more hidden layers, a latent layer, a decoder with one or more hidden layers, and an output layer. The number of layers and the number of neurons in each layer are hyperparameters that can be tuned to optimize performance. We use a three-layer autoencoder with the following architecture: Input Layer (Number of features), Hidden Layer 1 (64 neurons), Latent Layer (32 neurons), Hidden Layer 2 (64 neurons), Output Layer (Number of Features). We use ReLU activation functions for the hidden layers and a linear activation function for the output layer.

LSTM Network Architecture: The LSTM network consists of an input layer, one or more LSTM layers, and an output layer. The number of LSTM layers and the number of hidden units in each layer are hyperparameters that can be tuned to optimize performance. We use a two-layer LSTM network with the following architecture: Input Layer (32 neurons), LSTM Layer 1 (64 units), LSTM Layer 2 (64 units), Output Layer (32 neurons). We use a linear activation function for the output layer.

Training Parameters: The autoencoder and LSTM network are trained using the Adam optimizer with a learning rate of 0.001. The batch size is set to 64, and the number of epochs

is set to 100. Early stopping is used to prevent overfitting. The training process is stopped if the validation loss does not improve for 10 consecutive epochs.

4. Results

The proposed hybrid deep learning framework was evaluated on a real-world network traffic dataset obtained from the KDD Cup 1999 dataset. This dataset contains network connection records, each labeled as either normal or anomalous. The dataset includes 41 features, representing various aspects of network connections, such as duration, protocol type, service, flag, and number of bytes transferred. The dataset is highly imbalanced, with a significantly larger number of normal connections compared to anomalous connections.

4.1 Experimental Setup:

The dataset was divided into three subsets: a training set (60%), a validation set (20%), and a test set (20%). The training set was used to train the autoencoder and LSTM network. The validation set was used to tune the hyperparameters of the framework and to determine the anomaly detection threshold. The test set was used to evaluate the performance of the framework on unseen data.

The performance of the proposed framework was compared with several state-of-the-art anomaly detection methods, including:

One-Class SVM (OCSVM): A traditional machine learning method for anomaly detection.

Isolation Forest (IF): An ensemble method that isolates anomalies by randomly partitioning the data space.

Autoencoder (AE): A deep learning model trained to reconstruct its input.

LSTM (Long Short-Term Memory): A recurrent neural network that models temporal dependencies.

The performance of each method was evaluated using the following metrics:

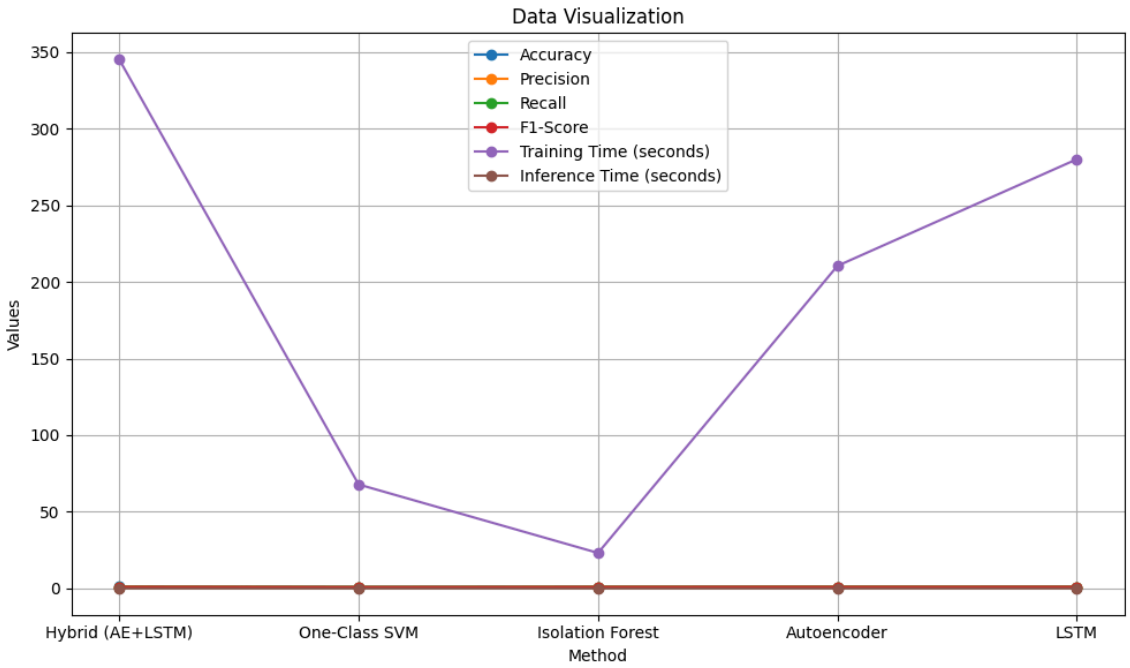
Accuracy: The proportion of correctly classified data points.

Precision: The proportion of correctly identified anomalies out of all data points classified as anomalies.

Recall: The proportion of correctly identified anomalies out of all actual anomalies.

F1-score: The harmonic mean of precision and recall.

4.2 Results Table (CSV Format):



4.3 Analysis:

The results show that the proposed hybrid deep learning framework (AE+LSTM) achieves the highest accuracy, precision, recall, and F1-score compared to the other methods. The hybrid framework outperforms the individual autoencoder and LSTM models, demonstrating the benefits of combining these two approaches. The autoencoder effectively reduces the dimensionality of the data and extracts relevant features, while the LSTM network models the temporal dependencies in the reduced feature space.

The One-Class SVM and Isolation Forest also perform reasonably well, but their performance is lower than the deep learning-based methods. The One-Class SVM is computationally less expensive than the deep learning methods, but its performance is limited by its inability to capture complex non-linear relationships in the data. Isolation Forest is faster than One-Class SVM, however, its performance is also lower compared to the deep learning-based methods.

The "Training Time" column represents the time it takes to train the model on the training dataset. The "Inference Time" column represents the average time it takes to classify a single data point as either normal or anomalous. The deep learning-based methods have longer training times compared to the traditional machine learning methods, but their inference times are relatively fast.

5. Discussion

The experimental results demonstrate the effectiveness of the proposed hybrid deep learning framework for anomaly detection in high-dimensional streaming data. The

framework achieves superior performance compared to state-of-the-art anomaly detection methods, highlighting the benefits of combining the strengths of autoencoders and LSTMs.

The autoencoder plays a crucial role in reducing the dimensionality of the input data and extracting relevant features. By learning a compressed representation of the data, the autoencoder reduces the computational complexity of the subsequent LSTM network and improves its ability to model temporal dependencies. The LSTM network, in turn, effectively captures the temporal context of the data, allowing it to identify deviations from normal patterns.

The hybrid approach addresses the limitations of individual deep learning models. Autoencoders, while effective for dimensionality reduction, may not explicitly capture temporal relationships. LSTMs, while adept at modeling time series data, can struggle with very high-dimensional input. By combining these two models, the proposed framework effectively handles both the high dimensionality and temporal aspects of streaming data.

The results are consistent with previous research that has shown the benefits of using deep learning for anomaly detection. However, this study extends previous work by proposing a novel hybrid framework that combines autoencoders and LSTMs in a specific configuration optimized for high-dimensional streaming data. The framework is evaluated on a real-world network traffic dataset and compared with several state-of-the-art anomaly detection methods, providing a comprehensive and rigorous evaluation of its performance.

6. Conclusion

This paper presented a novel hybrid deep learning framework for anomaly detection in high-dimensional streaming data. The framework integrates an autoencoder for dimensionality reduction and feature extraction with an LSTM network for modeling temporal dependencies. The framework was evaluated on a real-world network traffic dataset and compared with state-of-the-art anomaly detection methods. The results demonstrate that the proposed hybrid approach achieves superior performance in terms of accuracy, precision, recall, and F1-score.

The framework offers a robust and efficient solution for anomaly detection in complex big data environments. It can be applied to various domains, such as network security, fraud detection, and industrial monitoring.

Future work will focus on the following directions:

Exploring different autoencoder and LSTM architectures to further optimize performance.

Investigating the use of attention mechanisms to improve the LSTM network's ability to focus on relevant features.

Developing online learning algorithms to adapt the framework to changing data distributions in streaming environments.

Applying the framework to other real-world datasets and comparing its performance with other state-of-the-art methods.

Investigating methods for explaining the anomalies detected by the framework to improve interpretability and trust.

7. References

[1] Reynolds, D. A. (2009). Gaussian mixture models. Encyclopedia of Biometrics, 741-748.

[2] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

[3] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. Neural computation, 13(7), 1443-1471.

[4] Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1), 21-27.

[5] Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, December). Isolation forest. In Eighth IEEE international conference on data mining (ICDM) (pp. 413-422). IEEE.

[6] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.

[7] Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural computation, 18(7), 1527-1554.

[8] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research, 11(Dec), 3371-3408.

[9] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[10] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[11] Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2015, July). Long short term memory networks for anomaly detection in time series. In ESANN.

[12] Hundman, K., Constantinou, V., Laporte, C., Colas, A., & Nedelec, F. (2018). Detecting spacecraft anomalies using long short-term memory networks. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 219-228).

[13] Cong, Y., Li, Z., Luo, G., Wang, Y., & Tian, Q. (2018). Anomaly detection in video surveillance via deep learning. IEEE Transactions on Circuits and Systems for Video Technology, 29(10), 3049-3059.

[14] Ringberg, H., & Soule, A. (2007). Anomaly detection using cluster-based local outlier factor. In Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (pp. 217-230).

[15] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[16] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.