IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

DETECTION OF ABNORMAL EVENTS IN SMART GRIDS USING AN OPTIMIZED CONVOLUTIONAL LONG SHORT-TERM MEMORY MODEL

[1] Ms. Y. Annie Jerusha, [2] Ms. M. Haswitha

[1] Assistant Professor, [2] UG Scholar

[*]Department of Computer Science and Engineering

Geethanjali Institute of Science and Technology (Autonomous), Nellore, Andhra Pradesh, India

Abstract: The growing dependence of modern power systems on digital communication, smart meters, and automated control has improved the efficiency and reliability of electricity distribution while simultaneously exposing smart grids to abnormal events such as equipment faults, false data injection, electricity theft, and irregular consumption behaviour. Early and accurate detection of such anomalies is essential for maintaining stable, secure, and uninterrupted power supply. This paper presents an optimized Convolutional Long Short-Term Memory (ConvLSTM) model for detecting abnormal events in smart grid systems. The model couples the spatial feature-extraction strength of Convolutional Neural Networks (CNN) with the temporal sequence-learning capability of Long Short-Term Memory (LSTM) networks, enabling it to analyse time-series smart meter readings identified by consumer number (CONS_NO) and labelled as normal or attack (FLAG). The dataset is prepared through a structured pipeline comprising linear interpolation for missing values, quantile-based winsorization for outlier suppression, a three-point moving-average smoothing of consumption sequences, standard scaling, and a custom synthetic oversampling routine to correct class imbalance before an 80:20 stratified train-test split. The ConvLSTM network consists of a one-dimensional convolutional block (128 channels, kernel size 5) with batch normalization and max-pooling, followed by two stacked LSTM layers and a fully connected classification head trained with binary cross-entropy loss and the Adam optimizer. Model hyperparameters are further refined through an automated Optuna-based search across convolutional channels, LSTM hidden size, depth, dropout rates, learning rate, and batch size, with early stopping used to curb overfitting. The trained and optimized system, deployed through an interactive monitoring dashboard, reports an overall test accuracy of 91.35%, with a precision of 92.10% for the normal class, a recall of 91.80% for the attack class, and an F1-score of 91.32%, alongside consistent performance on independent batch evaluations. The results indicate that jointly modelling spatial and temporal dependencies allows the proposed system to reliably distinguish normal consumption from abnormal events, offering a practical pathway toward strengthening the security and resilience of smart grid infrastructure.

Index Terms - Anomaly Detection, Smart Grid, ConvLSTM, Deep Learning, False Data Injection, Energy Theft, Time-Series Analysis, Hyperparameter Optimization, Cyber Security.

I. INTRODUCTION

The conventional electrical grid was built for one-directional power flow from generating stations to consumers. The increasing demand for electricity, the growing share of renewable generation, and the maturing of communication technologies have together driven the transition toward smart grids, in which information technology, automation, and digital control are layered onto the physical power infrastructure. Smart meters, sensors, and automated substations continuously generate data describing energy generation, transmission, and consumption, and this data underpins real-time monitoring, demand management, and improved system reliability.

A defining capability of the smart grid is two-way communication between utilities and consumers, which allows usage patterns to be monitored and adjusted and permits distributed generation, such as rooftop solar, to be integrated into the network. This connectivity, however, also exposes the grid to abnormal events. Equipment malfunctions, sensor faults, sudden and unexplained load variations, false data injection attacks that manipulate measurement data sent to control systems, and energy theft carried out by tampering with meter readings can all degrade billing accuracy, operational stability, and overall grid security. Rule-based and threshold-based detection schemes, which were adequate for simpler grid topologies, struggle to keep pace with the volume, velocity, and complexity of present-day smart grid data streams.

Deep learning offers a way to learn these complex relationships directly from data rather than from hand-crafted rules. Hybrid architectures such as the Convolutional Long Short-Term Memory (ConvLSTM) network are particularly well suited to this task because they combine the spatial pattern-recognition ability of Convolutional Neural Networks (CNN) with the sequence-modelling ability of Long Short-Term Memory (LSTM) networks, allowing both local consumption irregularities and long-range temporal dependencies to be captured within a single model.

This paper proposes an optimized ConvLSTM-based system for detecting abnormal events in smart grid energy consumption data, including energy theft, false data injection, and other irregular behaviours. The remainder of the paper is organized as follows. Section II reviews related work on anomaly detection in smart grids. Section III describes the proposed methodology, including data preparation and model architecture. Section IV discusses the implementation details. Section V presents the results and discussion, and Section VI concludes the paper with directions for future work.

II. LITERATURE SURVEY

Anomaly detection in smart grids has progressed from simple statistical and threshold-based monitoring toward machine learning and, more recently, deep learning approaches. Early statistical and rule-based methods flagged deviations from fixed thresholds but were unable to adapt to the complex and evolving attack patterns present in large-scale grid data [1], [3]. Foundational work on deep learning established that multi-layer neural networks can automatically learn hierarchical feature representations from raw data, removing much of the dependence on manual feature engineering that limited earlier classifiers [2].

Recurrent architectures, and Long Short-Term Memory networks in particular, were developed to address the difficulty conventional recurrent networks have in learning long-range temporal dependencies, making them well suited to the sequential nature of smart meter readings [3]. Convolutional architectures, originally popularised for image recognition, were subsequently shown to be effective at extracting local spatial correlations, such as the relationship between neighbouring meters or sensors on the same feeder [6]. The ConvLSTM architecture unifies both ideas by embedding convolutional operations directly within the LSTM cell, and was first shown to be effective for spatiotemporal forecasting problems such as precipitation nowcasting before being adapted to anomaly and intrusion detection tasks [4].

Ensemble and boosted-tree methods such as Random Forest and XGBoost have also been applied to smart grid anomaly classification and offer reasonable accuracy with lower computational cost, but they generally

require carefully engineered input features and treat each observation independently, limiting their ability to model sequential dependence in consumption data [7]. Surveys of anomaly detection more broadly classify abnormal observations into point, contextual, and collective categories, a taxonomy that maps naturally onto smart grid problems such as sudden voltage spikes, coordinated false data injection, and sustained theft patterns [17]. Reviews focused specifically on deep learning for anomaly detection report consistent advantages for hybrid spatiotemporal architectures over single-branch CNN or LSTM models, particularly for high-dimensional, streaming sensor data of the kind generated by smart meters and phasor measurement units [16], [23].

Adam-based optimization is now standard practice for training such networks efficiently, combining adaptive learning rates with momentum to accelerate convergence on large architectures [10]. Practical implementations typically rely on the TensorFlow or PyTorch ecosystems together with NumPy, Pandas, Matplotlib, and Scikit-learn for data handling, visualization, and baseline comparison [11]–[14]. Despite this progress, a recurring gap identified across the literature is the limited attention given to automated hyperparameter tuning and to real-time, continuously updated deployment, with most reported systems trained and evaluated offline on static datasets [3], [21]. Table 1 summarises representative categories of existing approaches and their principal limitations, which together motivate the optimized ConvLSTM-based framework with automated hyperparameter search proposed in this work.

Table 1: Representative Smart Grid Anomaly Detection Approaches

Approach	Description	Limitation
Threshold / rule-based detection	Flags readings that cross predefined statistical limits.	Cannot adapt to evolving or coordinated attack patterns.
Random Forest / XGBoost	Ensemble tree models classify consumption patterns.	Requires manual feature engineering; treats samples independently.
Standalone CNN	Learns spatial correlations among meters or sensors.	Limited ability to capture long-range temporal dependence.
Standalone LSTM	Learns temporal dependence in sequential readings.	Does not directly exploit spatial/local correlations.
Hybrid CNN-LSTM / ConvLSTM	Jointly learns spatial and temporal patterns.	Sensitive to hyperparameter choices; needs systematic tuning.

III. PROPOSED METHODOLOGY

The proposed system detects abnormal events in smart grid data using an optimized Convolutional Long Short-Term Memory (ConvLSTM) model. The overall workflow proceeds through data collection, preprocessing, feature preparation, model training with automated hyperparameter optimization, and anomaly classification, as detailed in the following subsections.

A. Data Collection and Dataset Description

The dataset consists of smart meter energy consumption records, with each record identified by a consumer number (CONS_NO) and a binary label (FLAG) indicating whether the corresponding consumption sequence is normal or anomalous. Each record additionally contains a sequence of time-indexed numerical consumption readings that form the temporal features used by the model. This structure allows the system to treat every consumer's record as an ordered time series suitable for sequence modelling.

B. Data Preprocessing

Several preprocessing stages are applied before training. Non-numeric or missing readings within each sequence are first coerced to numeric form and corrected using linear interpolation along the time axis, with any remaining gaps resolved through backward and forward filling followed by median imputation. Extreme outliers are then suppressed using quantile-based winsorization, in which each feature is clipped to the 0.5th and 99.5th percentile range to limit the influence of abnormal spikes without discarding the underlying data points. A lightweight three-point moving-average filter is subsequently applied along each sequence to smooth short-term noise while preserving the broader temporal structure. Standard scaling is then used to normalize all features to zero mean and unit variance. Finally, because attack and theft instances are typically under-represented relative to normal consumption, a custom synthetic oversampling routine is applied to the minority class until both classes are balanced, reducing the risk of the model becoming biased toward the majority class.

C. Feature Selection

The time-series consumption readings associated with each consumer form the primary feature set used for anomaly detection, together with the consumer identifier (CONS_NO) used for record tracking and the FLAG attribute used as the supervised label. Since the ConvLSTM architecture learns spatial and temporal representations directly from the raw sequence, manual feature engineering beyond the preprocessing steps described above is intentionally minimized, allowing the convolutional and recurrent layers to identify the most informative patterns automatically.

D. Data Splitting

After balancing, the dataset is partitioned into training and testing subsets using an 80:20 stratified split, ensuring that the proportion of normal and anomalous samples is preserved in both partitions. The resulting sequences are reshaped into a (samples, time-steps, 1) tensor format suitable for input into the convolutional and recurrent layers of the network.

E. Model Architecture: ConvLSTM

The proposed network begins with a one-dimensional convolutional block that applies 128 filters of kernel size 5 over each input sequence, followed by a ReLU activation, batch normalization, max-pooling, and dropout regularization to extract local spatial patterns while controlling overfitting. The convolutional output is permuted and passed into two stacked LSTM layers with a hidden size of 128, which learn temporal dependencies across the smoothed and pooled sequence. The final hidden state from the LSTM stack is passed through a dropout layer and a fully connected head consisting of a 128-to-512 linear layer with ReLU activation and dropout, followed by a final linear layer that outputs a single logit. A sigmoid function, applied implicitly through the binary cross-entropy loss during training, converts this logit into the probability that the input sequence represents an anomaly.

To move beyond a fixed architecture, the model hyperparameters are further refined through an automated search using the Optuna framework. The search explores the number of convolutional channels, the LSTM hidden size and number of layers, the size of the fully connected hidden layer, the convolutional kernel size, dropout rates for the convolutional, recurrent, and fully connected stages, the learning rate, and the batch size, with each candidate configuration trained for a fixed number of epochs and evaluated on a held-out validation split. A median-pruning strategy is used to terminate unpromising trials early, and the configuration achieving the lowest validation loss across the search is retrained for the full training schedule to obtain the final optimized model. Table 2 summarizes the principal hyperparameters explored during optimization.

Table 2: Hyperparameters Explored During Optimization

Parameter	Search Range / Value	Model Component
Convolutional channels	64, 128, 256	CNN block
Kernel size	3, 5, 7	CNN block
LSTM hidden size	50, 100, 150, 200	LSTM block
Number of LSTM layers	1 to 3	LSTM block
Fully connected hidden size	256, 512, 1024	Output head
Dropout rate (conv / LSTM / FC)	0.1-0.5 / 0.1-0.5 / 0.2-0.6	Regularization
Learning rate	1e-5 to 1e-3 (log scale)	Adam optimizer
Batch size	32, 64, 128	Training loop

F. Loss Function and Optimization

Binary cross-entropy loss, computed directly from the output logits for numerical stability, is used to measure the discrepancy between predicted and actual labels. The Adam optimizer updates network weights using adaptive learning rates and momentum, and a reduce-on-plateau learning rate scheduler lowers the learning rate when validation loss stops improving. Gradient clipping limits the magnitude of parameter updates to maintain training stability, and early stopping, governed by a patience window on validation loss, halts training once further epochs no longer yield improvement, preventing overfitting and reducing unnecessary computation.

IV. IMPLEMENTATION

A. Software Requirements

The system is implemented in Python (version 3.10 or above). The deep learning components are built using the PyTorch framework, with Optuna used for automated hyperparameter search. NumPy and Pandas handle numerical computation and dataset manipulation, Scikit-learn provides preprocessing utilities and evaluation metrics, and Matplotlib supports visualization. The trained model is served through an interactive Streamlit-based dashboard, and Visual Studio Code is used as the development environment.

B. System Architecture

The system architecture comprises six interconnected components. Smart Meter Data Collection acts as the input layer, gathering consumption sequences identified by consumer number. The Data Processing module performs cleaning, interpolation, outlier suppression, smoothing, normalization, and class balancing. The Feature Extraction stage prepares the resulting sequences for the learning model. The ConvLSTM Model Training component fits the network to the prepared data and, through the Optuna search, selects an optimized configuration. The Anomaly Detection stage applies the trained model to classify incoming sequences as normal or anomalous, generating an attack probability and threat level. Finally, the Performance Evaluation component measures accuracy, precision, recall, and F1-score and feeds the results into the visualization and alerting components of the dashboard.

C. Model Implementation

The ConvLSTM network is implemented as a PyTorch module. The convolutional sub-module defines a Conv1d layer with 128 output channels and a kernel size of 5, followed by ReLU activation, batch normalization, max-pooling, and dropout. The pooled feature sequence is permuted to a batch-first format and passed to two stacked LSTM layers with hidden size 128. The final hidden state is projected through a fully connected head to a single output logit, and a threshold of 0.5 on the corresponding sigmoid probability is used during inference to assign the normal or anomalous class. The model checkpoint and

the fitted scaler are persisted so that the same preprocessing transformation can be applied consistently at inference time.

D. Testing

The system is validated through a layered testing strategy consisting of unit testing of individual components such as data cleaning, scaling, and model loading; integration testing of the combined preprocessing-to-prediction pipeline; system testing of the complete end-to-end application; and acceptance testing against the stated functional requirements. Table 3 summarizes representative test cases exercised during development.

Table 3: Representative Test Cases

Test ID	Test Scenario	Expected Result	Status
TC_01	Dataset loading and column validation	CONS_NO and FLAG columns present	Pass
TC_02	Missing value handling	No residual null values after interpolation	Pass
TC_03	Outlier suppression and scaling	Features clipped and standardized	Pass
TC_04	Class balancing	Equal sample counts for normal and attack classes	Pass
TC_05	Train-test split	Stratified 80:20 split without data loss	Pass
TC_06	Model training convergence	Validation loss decreases and early stopping triggers	Pass
TC_07	Single-record prediction	Correct normal/anomaly classification with probability score	Pass

V. RESULTS AND DISCUSSION

The optimized ConvLSTM model is evaluated on the held-out 20% test split using accuracy, precision, recall, and F1-score. The deployed application reports an overall classification accuracy of 91.35%, a precision of 92.10% for the normal consumption class, a recall of 91.80% for the attack class, and an F1-score of 91.32%, indicating that the model maintains a balanced trade-off between correctly identifying anomalies and avoiding false alarms.

Batch evaluations on independent consumption datasets further corroborate this behaviour. On the first evaluation batch, the system achieved 85.00% accuracy, correctly classifying 85 of the records, with 45 instances flagged as attacks and 55 identified as normal patterns. On a second, independently sampled batch, the system achieved 80.00% accuracy, correctly classifying 80 records, with 62 instances flagged as attacks and 38 as normal. These batch-level results are somewhat lower than the held-out test accuracy, which is consistent with the expected variation in performance when the model is applied to new, previously unseen consumption batches rather than the original stratified test partition.

Single-record analysis within the dashboard illustrates the model's probability-based decision process. For a sequence subsequently confirmed as an attack, the model assigned an attack probability of 99.9%, correctly classifying the record as anomalous with a high threat level and a recommended block action. For a sequence confirmed as normal consumption, the model assigned a low attack probability of 20.9%, correctly classifying the record as normal with a low threat level and a recommended allow action. These examples demonstrate that the model produces well-separated probability estimates for clearly normal and

clearly anomalous patterns, supporting its use for graded risk assessment rather than a purely binary decision.

Taken together, the results show that combining convolutional spatial feature extraction with LSTM-based temporal modelling, refined through automated hyperparameter optimization, allows the proposed system to detect both sudden and gradually developing abnormal events in smart grid consumption data with consistently high accuracy across both held-out test data and independent batch evaluations.

VI. CONCLUSION

This paper presented an optimized Convolutional Long Short-Term Memory (ConvLSTM) model for detecting abnormal events, including energy theft and false data injection, in smart grid energy consumption data. By combining convolutional layers for spatial feature extraction with LSTM layers for temporal pattern learning, and by refining the resulting architecture through an automated Optuna-based hyperparameter search, the proposed system captures complex spatiotemporal patterns more effectively than conventional rule-based or single-branch machine learning approaches. A structured preprocessing pipeline comprising interpolation, outlier winsorization, temporal smoothing, standard scaling, and synthetic class balancing further improves data quality prior to model training.

The trained and optimized model achieved a test accuracy of 91.35%, with corresponding precision, recall, and F1-score values above 91%, and maintained consistent performance of 85.00% and 80.00% accuracy across two independent batch evaluations. These results confirm the practical effectiveness of the proposed approach for identifying abnormal smart grid behaviour. Future work will focus on extending the model with attention mechanisms and GRU-based variants, incorporating continual or online learning for real-time monitoring, applying explainable AI techniques to improve operator trust, and extending the framework to multi-class attack classification and edge deployment for low-latency detection.

REFERENCES

[1] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network anomaly detection with the restricted Boltzmann machine," Neurocomputing, vol. 122, pp. 13-23, Dec. 2013.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, pp. 436-444, May 2015.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[4] X. Shi et al., "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in Advances in Neural Information Processing Systems, 2015, pp. 802-810.

[5] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA: MIT Press, 2016.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proc. NIPS, 2012.

[7] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. ACM SIGKDD, 2016.

[8] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, pp. 85-117, 2015.

[9] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in Proc. EMNLP, 2014.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. ICLR, 2015.

[11] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in Proc. OSDI, 2016.

[12] W. McKinney, "Data structures for statistical computing in Python," in Proc. SciPy, 2010.

[13] J. D. Hunter, "Matplotlib: A 2D graphics environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

[14] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[15] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow. Sebastopol, CA: O'Reilly Media, 2019.

[16] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," arXiv preprint arXiv:1901.03407, 2019.

[17] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Computing Surveys, vol. 41, no. 3, pp. 1-58, 2009.

[18] H. He, D. Janicke, and Q. Cao, "Anomaly detection using deep learning," IEEE Access, 2017.

[19] Y. Zhang, B. Wallace, and X. Liu, "Time series classification with deep learning," arXiv preprint arXiv:1708.03679, 2017.

[20] Y. Bengio, "Learning deep architectures for AI," Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1-127, 2009.

[21] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A survey of deep learning methods for cyber security," Information, vol. 10, no. 4, p. 122, 2019.

[22] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive dataset for network intrusion detection," in Proc. Military Communications and Information Systems Conf. (MilCIS), 2015.

[23] G. Pang, C. Shen, L. Cao, and A. van den Hengel, "Deep learning for anomaly detection: A review," ACM Computing Surveys, vol. 54, no. 2, pp. 1-38, 2021.

[24] Z. Zhou, B. Wang, M. Guo, and Y. Zhang, "Deep learning-based anomaly detection in cyber-physical systems," IEEE Access, vol. 6, pp. 67373-67380, 2018.

[25] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," IEEE Access, vol. 5, pp. 21954-21961, 2017.