DEVICE CATEGORIZATION FROM ELECTRICAL SIGNALS WITH MACHINE LEARNING

(ELEKTRİK SİNYALLERİNDEN MAKİNE ÖĞRENMESİ İLE CİHAZ KATEGORİZASYONU)

by

TOLGA REİS, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

JUNE 2025

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to all the esteemed faculty members of the Computer Engineering Department at Galatasaray University for their valuable guidance, insightful lectures, and unwavering support throughout my graduate studies. Their academic contributions have significantly shaped my understanding and passion for the field.

I am especially indebted to my thesis supervisor, Asst. Prof. Dr. Ahmet Teoman Naskali, for his continuous encouragement, constructive feedback, and patient mentorship. His expertise and guidance have been instrumental in the successful completion of this work.

I would also like to thank my wife for her endless love, understanding, and motivation during the most challenging times of this journey. Her support has been a constant source of strength.

Finally, I extend my heartfelt appreciation to my mother and father, whose sacrifices and belief in me have always inspired me to aim higher and persevere.

TOLGA REİS

24 June 2025

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The growing need for energy management and sustainability necessitates innovative solutions in the analysis of electrical signals and device recognition. This study presents a novel approach for categorizing devices based on their unique energy consumption patterns using electrical signals. A TinyML-based system has been developed to enable resource-efficient and scalable device recognition on embedded platforms.

The research addresses critical challenges in device recognition systems, such as signal noise, overlapping energy profiles, and real-time processing constraints. By leveraging advanced analytical techniques, the proposed system improves the accuracy and efficiency of device categorization. Additionally, it contributes to the broader adoption of TinyML in energy monitoring systems, providing significant benefits in terms of energy efficiency, fault detection, and sustainability.

This study bridges the gap between academic innovation and practical application, supporting the development of smart home technologies, industrial automation, and intelligent energy management systems. The findings demonstrate the potential of integrating TinyML into energy monitoring frameworks, paving the way for more sustainable and user-friendly solutions.

Keywords : TinyML, Electrical Signal Analysis, Device Recognition, Energy Efficiency, Smart Systems

RÉSUMÉ

La demande croissante pour une gestion énergétique efficace et durable a conduit au développement de solutions innovantes dans l'analyse des signaux électriques et la reconnaissance des appareils. Cette étude propose une nouvelle approche pour la catégorisation des appareils électriques basée sur les signaux électriques. Un système basé sur TinyML a été développé pour permettre une reconnaissance efficace et évolutive sur des plateformes embarquées.

La recherche aborde les défis critiques de la reconnaissance des appareils, notamment le bruit des signaux, les profils énergétiques chevauchants et les contraintes de traitement en temps réel. En s'appuyant sur des techniques analytiques avancées, le système proposé améliore la précision et l'efficacité de la catégorisation des appareils. En outre, il contribue à l'adoption élargie de TinyML dans les systèmes de surveillance énergétique, offrant des avantages significatifs en termes d'efficacité énergétique, de détection des pannes et de durabilité.

Cette étude comble le fossé entre l'innovation académique et l'application pratique, en soutenant les technologies des maisons intelligentes, l'automatisation industrielle et le développement de systèmes de gestion énergétique intelligents. Les résultats démontrent le potentiel d'intégration de TinyML dans les cadres de surveillance énergétique, ouvrant la voie à des solutions plus durables et conviviales.

Mots Clés : TinyML, Analyse des Signaux Électriques, Reconnaissance des Appareils, Efficacité Énergétique, Systèmes Intelligents

ÖZET

Artan enerji yönetimi ve sürdürülebilirlik ihtiyacı, elektrik sinyallerinin analizi ve cihaz tanıma alanında yenilikçi çözümler geliştirilmesini zorunlu kılmaktadır. Bu çalışma, elektrik sinyallerine dayalı olarak cihazların benzersiz enerji tüketim desenlerini sınıflandırmaya yönelik yeni bir yaklaşım sunmaktadır. TinyML tabanlı bir sistem geliştirilmiş olup, gömülü platformlarda kaynak verimli ve ölçeklenebilir cihaz tanıma sağlanmıştır.

Araştırma, cihaz tanıma sistemlerinin karşılaştığı gürültü, örtüşen enerji profilleri ve gerçek zamanlı işleme kısıtlamaları gibi kritik zorlukları ele almaktadır. Gelişmiş analiz tekniklerinden yararlanarak, önerilen sistem cihaz kategorilendirme doğruluğunu ve verimliliğini artırmaktadır. Ayrıca, enerji izleme sistemlerinde TinyML'nin daha geniş çapta benimsenmesine katkıda bulunarak, enerji verimliliği, arıza tespiti ve sürdürülebilirlik açısından önemli faydalar sağlamaktadır.

Bu çalışma, akademik yenilik ile pratik uygulama arasındaki boşluğu doldurarak, akıllı ev teknolojileri, endüstriyel otomasyon ve akıllı enerji yönetim sistemlerinin geliştirilmesini desteklemektedir. Bulgular, enerji izleme çerçevelerine TinyML'nin entegrasyon potansiyelini göstermekte olup, daha sürdürülebilir ve kullanıcı dostu çözümler için bir yol haritası çizmektedir.

Anahtar Kelimeler : TinyML, Elektrik Sinyali Analizi, Cihaz Tanıma, Enerji Verimliliği, Akıllı Sistemler

LIST OF ABBREVIATIONS

ADC	Analog-to-Digital Converter
AFHHM	Additive Factorial Hidden Markov Model
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
BLE	Bluetooth Low Energy
BN	Batch Normalization
CPU	Central Processing Unit
CT	Current Transformer
DAC	Digital-to-Analog Converter
DIN	Deutsches Institut für Normung (German Institute for Standardization)
EM	Energy Meter
ESP32	Espressif Systems Microcontroller Unit
FCN	Fully Convolutional Network
GAP	Global Average Pooling
GPIO	General-Purpose Input/Output
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
I2C	Inter-Integrated Circuit
ILM	Intrusive Load Monitoring
IoT	Internet of Things
ISO	International Organization for Standardization
LAN	Local Area Network
MQTT	Message Queuing Telemetry Transport
NILM	Non-Intrusive Load Monitoring
NPU	Neural Processing Unit
OTA	On-the-air
QAT	Quantization-aware Training
PWM	Pulse-Width Modulation
RAM	Random Access Memory
ReLU	Rectified Linear Unit
ResNet-ID	Residual Network for Time-Series Classification
SE-TSNet	Squeeze-and-Excitation Time-Series Network
SimpleRNN	Simple Recurrent Neural Network
SoC	System on Chip
SPI	Serial Peripheral Interface
TCN	Temporal Convolutional Network
TFLM	TensorFlow Lite for Microcontrollers
TinyML	Tiny Machine Learning

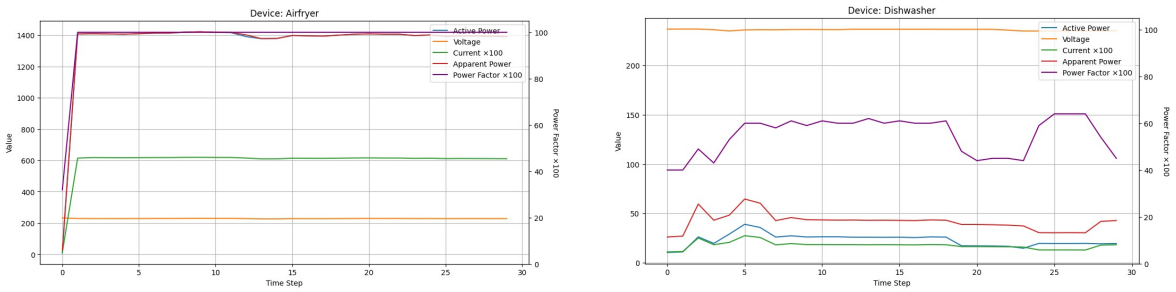
TimeCNN	Time Convolutional Neural Network
TWh	Terawatt-hours
UART	Universal Asynchronous Receiver-Transmitter
Wi-Fi	Wireless Fidelity

1 INTRODUCTION

The analysis of electrical signals has become increasingly vital in modern energy systems, offering insights into consumption patterns, device performance, and system anomalies. In 2024, global electricity consumption experienced a significant surge, increasing by approximately 1,080 TWh, marking a 4.3% rise compared to the previous year (Agency, 2024). This growth is attributed to factors such as economic recovery, increased use of air conditioning due to heatwaves, and the expansion of data centers and electric vehicles.

Household appliances continue to be substantial contributors to residential electricity consumption. Despite advancements in energy efficiency, the proliferation of appliances has led to a steady increase in energy use, particularly in emerging economies. This underscores the importance of detailed electrical signal analysis to optimize energy usage, detect inefficiencies, and inform the development of smarter, more sustainable energy systems.

Key electrical metrics such as current, voltage, active power, apparent power, and power factor encapsulate the operational behavior of devices. These parameters form multivariate time-series signals that exhibit appliance-specific variations and can be used to infer their type and operating state. For instance, the energy consumption profile of an airfryer displays sharp and intermittent peaks, whereas a dishwasher demonstrates a more gradual and cyclic load pattern (see Figure 1.1).



(a) Airfryer (b) Dishwasher

FIGURE 1.1 – Time-series profiles of electrical parameters for two different devices over a 30-second window.

Analyzing these metrics facilitates the extraction of device-specific features, which are essential for time series classification models. Sudden current surges at start-up or steady-state voltage fluctuations serve as discriminative indicators that enable device-level identification and operational optimization.

1.1 Importance of Device Recognition Systems

Device recognition systems constitute a critical component of next-generation energy intelligence frameworks by enabling fine-grained, appliance-level, energy consumption monitoring. Unlike traditional aggregate measurement, such systems provide actionable insights into operational patterns, inefficiencies, and anomalies by learning and distinguishing device-specific electrical signatures (Zeifman and Roth, 2011; Klemenjak and Goldsborough, 2016).

— **Operational Efficiency and Demand Optimization :** Accurate identification of high-consumption devices supports strategic decisions such as dynamic load balancing, time-of-use scheduling, and the replacement of inefficient appliances. In industrial domains, it allows peak forecasting, which is vital to maintain grid stability and minimize energy expenditures (Hart, 1992).

— **Fault Detection and Predictive Diagnostics :** Deviations in energy signatures - such as abnormal current spikes or irregular power factor trends - serve as early indicators of equipment degradation or potential failures. Timely classification of such anomalies allows for predictive maintenance strategies, reducing downtime and extending the useful life of equipment (Barker et al., 2012).

— **Smart Infrastructure and IoT Integration :** Embedded systems with device recognition capabilities facilitate real-time automation in smart homes and factories. By integrating with IoT platforms, energy analytics systems can autonomously respond to contextual cues, such as occupancy or environmental changes, thus enabling intelligent feedback control loops (Andrade et al., 2021).

— **Sustainability and Emissions Reduction :** Device-level monitoring allows for granular tracking of standby losses and operational inefficiencies. This supports sustainability goals by minimizing energy waste and aligning with international standards such as ISO 50001. Moreover, such systems can assist prosumers (i.e., consumers who also produce energy) in demand-side participation and tariff-based optimization (Liu et al., 2024).

— **Grid-Aware Responsiveness :** In smart grid contexts, real-time device classification contributes to adaptive demand response strategies. By identifying deferrable loads, the system can mitigate peak demand and support the integration of intermittent renewable energy sources (Kolter and Jaakkola, 2011).

— **User Awareness and Behavioral Shaping :** Personalized feedback based on disaggregated consumption data has been shown to improve user engagement and encourage energy-conscious behaviors. Dashboards featuring appliance-level insights and gamification elements enhance user interaction and promote commitment with energy-saving goals (Fischer, 2008).

— **Security and Monitoring Transparency :** Irregular device behavior—such as night-time activity or unexpected activation—can indicate security breaches or operational misuse. In sensitive environments such as laboratories and data centers, device-level monitoring adds a layer of accountability and supports risk mitigation (Razavi et al., 2018).

1.2 Current Methods and Limitations in Electrical Monitoring Systems

Electrical monitoring systems facilitate the understanding of energy consumption, enable optimization of usage, and support anomaly detection. Despite advances, scalability, real-time analysis, and resource efficiency continue to pose difficulties.

1.2.1 Intrusive Load Monitoring (ILM)

— *Description :* This approach directly measures energy consumption by deploying sensors on individual devices, enabling fine-grained and accurate monitoring.

— *Advantages :* The use of device-specific monitoring enables high accuracy in energy consumption analysis.

— *Limitations :* The high cost and complexity of installation limit the scalability and practicality of large-scale implementations.

1.2.2 Non-Intrusive Load Monitoring (NILM)

— *Description :* This approach utilizes a single measurement point to analyze aggregated energy data and infer individual appliance-level consumption patterns.

— *Advantages :* This approach offers a cost-effective solution that is adaptable to a wide range of application environments.

— *Limitations :* Separating overlapping energy signatures remains difficult under high-noise conditions, limiting classification performance.

1.2.3 Time-Series Analysis

— *Description :* This approach analyzes energy variations over time.

— *Advantages :* Captures dynamic behaviors like peak loads and operational cycles.

— *Limitations :* Computational complexity increases with data volume and resolution.

1.2.4 Machine Learning

— *Description :* Learns patterns from data to improve device identification and anomaly detection.

— *Advantages :* Adapts to new devices and environments.

— *Limitations :* Requires extensive labeled data and computational resources.

1.3 Challenges and Opportunities in Electrical Signal Analysis and Device Recognition

Electrical signal analysis and device recognition offer significant advantages for energy management and monitoring but they also involve a range of technical challenges. Understanding and addressing these challenges is essential for advancing the field and enhancing the effectiveness of current systems.

1.3.1 Key Challenges

— **Noise and Data Contamination :** Environmental noise, power line disturbances, and inter-device interactions can distort electrical signals, making it difficult to extract meaningful information and accurately distinguish between devices.

— **Signal Overlap and Similar Device Profiles :** Many devices, such as coffee machines and toasters, exhibit overlapping energy consumption patterns, which complicates their separation. These similarities necessitate the use of sophisticated analytical methods and advanced feature extraction techniques.

— **Usage Variability :** Device behavior can vary significantly due to load changes, voltage fluctuations, or user-specific usage patterns. For example, the electrical signature of a washing machine changes noticeably across its washing, rinsing, and spinning cycles.

— **Device Aging and Wear :** Over time, devices degrade or undergo maintenance, leading to changes in their energy consumption profiles. Such variations can reduce the accuracy of device recognition systems.

— **Scalability and Cost :** Monitoring systems based on hardware-intensive methods, such as ILM, are often costly and difficult to scale for large installations or heterogeneous environments.

— **Real-Time Constraints and Big Data Processing :** Monitoring a large number of devices generates vast amounts of data, necessitating systems capable of real-time analysis. High-resolution, continuous data streams demand efficient algorithms and optimized computational resources, particularly in resource-constrained environments.

1.3.2 Opportunities

Despite these challenges, there are significant opportunities for innovation and improvement :

— **Advanced Machine Learning Techniques :** Machine learning models, particularly those trained on large datasets, can enhance device recognition accuracy by learning complex patterns in energy consumption data. These models are capable of adapting to new devices and dynamic environments, reducing the need for manual reconfiguration.

— **Innovative Signal Processing Methods :** Techniques such as time-frequency analysis and feature engineering can extract distinctive characteristics from noisy or overlapping signals, thereby improving the differentiation between devices.

— **Integration with Smart Technologies :** Integrating electrical monitoring systems with IoT platforms and smart grids enables dynamic load balancing, facili-

tates demand response strategies, and promotes more efficient energy usage.

— **Cost-Effective and Scalable Solutions :** Emerging technologies, such as non-intrusive monitoring methods and embedded systems, offer more accessible and scalable alternatives to traditional hardware-intensive approaches.

— **Real-Time Decision Making :** Enhanced computational efficiency enables real-time anomaly detection and energy optimization, which are critical for both residential and industrial applications.

By addressing these challenges through innovative approaches in signal processing, machine learning, and system design, electrical monitoring and device recognition systems can achieve improved accuracy, scalability, and efficiency. These advances are expected to play a pivotal role in energy management and sustainability initiatives.

1.4 Research Objectives and Contributions

The primary objective of this research is to accurately identify and categorize electrical devices based on their characteristic signal patterns, using key metrics such as current, voltage, active power, apparent power, and power factor. To enable real-time, on-device inference in resource-constrained environments, the system is developed using TinyML techniques. This approach ensures that the solution remains efficient, scalable, and well-suited for low-power embedded systems.

1.4.1 Research Objectives

The research is driven by the following objectives :

— **Accurate Device Categorization :** Develop a framework to identify and categorize electrical devices based on their unique energy consumption patterns.

— **Energy-Efficient Implementation :** Develop a solution that operates efficiently on embedded platforms with minimal power requirements, ensuring compatibility with real-world applications in smart homes and industrial environments.

1.4.2 Research Contributions

This study provides significant contributions in both academic and practical domains :

- **Innovations in Device Recognition :** Propose novel methods to tackle challenges such as noise, overlapping signal profiles, and scalability, enhancing the accuracy and reliability of device recognition systems.
- **Advancing TinyML Applications :** Demonstrate the application of TinyML in electrical monitoring, showcasing its potential for real-time processing on resource-constrained devices.
- **Support for Energy Management :** Offer solutions that improve energy usage efficiency and reduce operational costs, thereby contributing to sustainability and conservation efforts.
- **User-Centric Smart Systems :** Enable practical implementations in smart homes by providing actionable energy insights and facilitating automated energy optimization.

These objectives and contributions underscore the novelty and practical relevance of this research, effectively bridging the gap between advancements in machine learning and real-world energy applications.

1.5 Significance of the Research

The significance of this research extends across scientific, societal, and industrial domains, addressing critical challenges in energy management, device recognition, and sustainable technologies.

1.5.1 Scientific Contributions

This research contributes to the advancement of device recognition techniques through :

- **Advanced Analytical Methods :** Utilizing electrical signal metrics to develop robust and efficient device identification systems.
- **Application of Embedded Machine Learning :** Highlighting the feasibility and advantages of TinyML for on-device processing in energy monitoring systems.

1.5.2 Societal Impact

The societal significance of this research is reflected in its potential to enhance energy efficiency and promote sustainable practices :

— **Smart Home Integration :** Enabling advanced automation solutions and providing users with real-time energy insights.

— **Sustainability and Conservation :** Reducing energy wastage and supporting global sustainability goals through improved energy management.

1.5.3 Industrial Applications

This research also holds potential for transforming industrial energy management practices :

— **Enhanced Monitoring Systems :** Improve the functionality of smart meters by delivering device-level energy analytics.

— **Scalable Automation Solutions :** Support industrial energy management and automation processes with efficient monitoring techniques.

— **Cost-Effective Energy Solutions :** Facilitate energy efficiency measures that lower costs while minimizing environmental impact.

By addressing these dimensions, this research underscores its relevance and value in advancing both theoretical understanding and practical applications in the fields of energy systems and machine learning.

1.6 Scope and Limitations

1.6.1 Scope

This study focuses on the development and evaluation of a device recognition system based on the analysis of electrical signals. The scope of the research is structured along the following key dimensions :

— **Device-Level Identification Using Electrical Signatures :** The core objective of this study is to classify and recognize electrical devices by analyzing their

unique energy consumption profiles. These profiles are extracted from characteristic patterns observed in electrical signals including current, voltage, active power, apparent power, and power factor.

— **Selected Electrical Metrics :** The scope of this study is limited to a defined set of electrical parameters - specifically current, voltage, apparent power, active power, and power factor. These metrics are selected due to their effectiveness in capturing the operational behavior of devices and their compatibility with low-power edge computing platforms.

— **Edge-Based Implementation :** The system is designed and evaluated for deployment on a resource-constrained edge device, with a focus on real-time inference using TinyML techniques.

— **Experimental Validation :** The performance of the proposed approach is evaluated using a time-series dataset of electrical measurements collected from multiple household appliances, with particular emphasis on classification accuracy, model size, and inference latency on embedded hardware platforms.

1.6.2 Limitations

While the study aims to contribute to advancements in device recognition and energy monitoring, certain limitations are acknowledged :

— **Focus on Selected Device Groups :** This research is conducted using a predefined set of devices, primarily household appliances, which may not fully capture the diversity of electrical devices encountered in broader real-world settings.

— **Data and Noise Constraints :** The quality and scope of the dataset used in this research are constrained by practical factors such as data collection methods and signal noise, which may impact the accuracy of device recognition.

— **Real-World Deployment Challenges :** Factors such as variability in device operation, environmental noise, and scalability to larger device ecosystems present significant challenges for real-world deployment.

By clearly defining its scope and acknowledging its limitations, this research provides a focused contribution to the field of device recognition while laying the groundwork for future studies to address remaining challenges.

1.7 Structure of the Thesis

This thesis is organized into six main chapters, accompanied by references and appendices, to provide a comprehensive overview of the research problem, methodology, and findings. The structure of the thesis is outlined as follows :

— **Chapter 1 : Introduction** - Presents the background, significance, research objectives, contributions, and key challenges related to the analysis of electrical signals and device recognition systems. This chapter sets the stage for the study by outlining its scope and limitations.

— **Chapter 2 : Literature Review** - Summarizes previous studies in electrical monitoring, device recognition, and related methodologies. This chapter identifies gaps in existing research and underscores the need for innovative approaches such as the one proposed in this thesis.

— **Chapter 3 : Model Formulation** - Provides a detailed explanation of the theoretical framework and mathematical models employed for device recognition. This chapter lays the foundation for the implementation of the proposed system.

— **Chapter 4 : Solution Methodology** - Describes the development and implementation of the TinyML-based system for electrical signal analysis. It covers data acquisition and preprocessing, model training, and the integration of machine learning techniques into resource-constrained environments.

— **Chapter 5 : Computational Studies** - Presents the experimental setup, results, and evaluation of the proposed system. This chapter discusses the key performance metrics and compares the results with those of existing methods.

— **Chapter 6 : Conclusion** - Summarizes the key findings of the research, discusses their implications, and provides recommendations for future work.

The thesis also includes **References** for all cited works and **Appendices** containing supplementary materials, such as detailed datasets and implementation codes. A **Biographical Sketch** of the author is provided at the end of the thesis.

2 LITERATURE REVIEW

In this study, NILM techniques have been utilized to optimize device recognition, building upon the approach proposed in (Abeykoon et al., 2016). Core electrical metrics such as current, voltage, active power, reactive power, and power factor are used to analyze device energy consumption profiles. Data was collected at a frequency of one measurement per second using the Shelly Pro 3EM device, with a particular emphasis on real-time identification methods.

A key advancement over (Abeykoon et al., 2016) is the integration of TinyML for on-device inference. This enables real-time predictions directly on edge devices, enhancing both data privacy and operational efficiency. The lightweight nature of TinyML models allows deployment on memory-constrained hardware, reducing the need for additional sensors and lowering overall system cost, with the Shelly Pro 3EM functioning as a centralized measurement hub.

This study contributes to the field by presenting a scalable, cost-effective NILM solution that leverages TinyML and IoT technologies for improved real-time energy management and device identification.

(Andrade et al., 2021) demonstrated the use of IoT and TinyML technologies for rapid field data analysis. Building on this approach, the present study optimizes TinyML models for real-time device classification based on electrical measurements. Unlike the centralized processing architecture in (Andrade et al., 2021), this study performs data analysis locally on the Shelly Pro 3EM device, eliminating the latency associated with cloud-based processing.

This localized approach provides several key advantages. First, it better addresses energy monitoring requirements by enabling timely and efficient device recognition. Second, through the use of NILM techniques, the system reduces hardware dependency, lowering implementation costs and complexity. Third, it supports broader energy management and sustainability objectives by facilitating efficient, real-time energy usage analysis.

The main contribution of this study lies in advancing the practical application of Ti-

nyML for edge-based energy monitoring. By combining real-time, on-device processing with cost-effective hardware and NILM techniques, it offers a scalable and efficient solution for intelligent energy management systems.

In this study, the MQTT protocol is utilized for transferring electrical measurements collected from the Shelly Pro 3EM device to AWS IoT Core and subsequently to cloud storage. This approach takes full advantage of MQTT's ability to transmit data reliably while consuming minimal energy, as described by (Bajrami et al., 2021). Furthermore, the publish-subscribe model of MQTT has been used to improve the data transmission efficiency of the energy monitoring system, thereby enhancing the overall system performance.

When compared to (Bajrami et al., 2021), this study brings several notable improvements. First, MQTT has been optimized for energy monitoring systems, not just for data transmission. Second, by integrating MQTT with AWS IoT Core, large datasets can be processed at scale, which was not as emphasized in (Bajrami et al., 2021). Third, the transmission processes have been streamlined to lower both the overall cost and complexity of the energy monitoring system.

This study's contribution lies in demonstrating how MQTT, as explored by (Bajrami et al., 2021), can be applied effectively to energy management and device recognition, offering practical insights into its use in real-world energy systems. It showcases the pivotal role of IoT in developing sustainable and efficient energy solutions, providing an optimized infrastructure for broader deployment.

(Chen et al., 2023) focused on the interpretability of machine learning models used in building energy management, highlighting their role in optimizing energy consumption. In a similar vein, this study also uses machine learning to optimize energy consumption and perform effective device classification. However, this study adds an additional layer by incorporating interpretability techniques to better understand energy consumption patterns and device behavior.

Building on (Chen et al., 2023), this study integrates these interpretability methods into the energy monitoring and device classification processes. This allows for deeper insights into how devices operate under varying conditions, providing a more transparent and effective approach to machine learning applications in energy management.

In contrast to (Chen et al., 2023), this study integrates TinyML for real-time energy

analysis and device classification directly on edge devices. This brings about several advantages such as low latency, enhanced data privacy, and improved energy efficiency. Additionally, by operating locally on edge devices, the study reduces dependency on centralized systems, enhancing operational efficiency.

A key contribution of this study is its focus on balancing interpretability and efficiency in real-world applications. By integrating TinyML with interpretable machine learning, it provides a comprehensive solution for both energy optimization and understanding the conditions under which devices operate, offering a novel approach to device classification.

(Feng et al., 2020) utilized deep learning techniques for time-series data disaggregation, specifically using a windowing approach to analyze energy consumption patterns. This study adopts a similar time-windowing approach, processing energy data with 30-second windows, which allows for more precise tracking of short-term consumption patterns. Furthermore, overlapping windows are employed to minimize potential information loss.

Unlike (Feng et al., 2020), this study develops a TinyML-based system where device classification is performed directly on edge devices. This eliminates the need for centralized systems, offering advantages like lower latency and improved data privacy. Moreover, the model used in this study is optimized for energy-efficient operation, making it suitable for devices with limited memory.

This study enhances the time-series windowing approach from (Feng et al., 2020) by integrating TinyML on edge devices, making the process more efficient and scalable. The main contribution is the application of TinyML to real-time device identification, providing cost-effective and resource-efficient solutions for energy management systems.

(Hart, 1992) is a pioneering study in the field of NILM, introducing a method to distinguish the energy consumption profiles of appliances using a single point of electrical measurement. Similarly, this study aims to analyze energy consumption patterns to classify devices. Both studies utilize fundamental electrical parameters such as current, voltage, active power, and power factor for device identification.

The NILM approach defined in (Hart, 1992) provided a foundational framework for appliance energy profiling, which significantly influenced the development of energy management systems. In contrast, this study enhances these fundamental NILM prin-

ciples by integrating TinyML, enabling predictions to be performed directly on edge devices in real-time. This shift from centralized to edge-based analysis introduces improvements in latency, data privacy, and operational efficiency, making the system more practical and resource-efficient.

While (Hart, 1992) laid the groundwork for NILM with centralized analysis, this study moves the concept forward by utilizing TinyML to perform localized predictions, enabling real-time device identification. This integration of contemporary machine learning techniques ensures that the system operates efficiently even in resource-constrained environments, thus pushing the boundaries of NILM applications in real-time energy monitoring.

(Klemenjak and Goldsborough, 2016) provided a comprehensive review and outlook on NILM, assessing the current state of appliance recognition technologies in energy management systems. Similarly, this study also builds upon NILM techniques, aiming to classify devices based on their energy consumption patterns.

(Klemenjak and Goldsborough, 2016) highlighted the primary challenges in NILM, such as optimizing data processing and addressing cost efficiency. This study takes a step further by incorporating edge-based inference with TinyML, addressing these challenges in real-time and reducing the need for centralized data processing. The integration of TinyML enables the system to classify devices locally, offering advantages in terms of energy efficiency, reduced latency, and enhanced data privacy.

Whereas (Klemenjak and Goldsborough, 2016) focused on the theoretical and technological advancements in NILM, this study applies these advancements by implementing a TinyML-based system that operates on edge devices. This localized approach enables a more scalable, efficient, and cost-effective solution for energy monitoring systems, offering an innovative contribution to both NILM research and practical applications.

(Kolter and Jaakkola, 2011) proposed approximate inference methods using Additive Factorial Hidden Markov Models (AFHMM) for energy disaggregation processes. Their work provided a significant framework for disaggregating energy consumption data to identify devices. Similarly, this study analyzes energy consumption patterns to classify devices. Both studies rely on advanced modeling techniques for energy disaggregation and device classification processes.

The approach in (Kolter and Jaakkola, 2011) focuses on disaggregating complex energy

consumption patterns, but this study advances their work by integrating modern machine learning techniques and TinyML. The inclusion of 5-second time windows allows for more precise detection of short-term consumption profiles, and the use of TinyML models facilitates predictions on resource-constrained edge devices. This enables real-time, low-latency predictions, which were not achievable in the approach by (Kolter and Jaakkola, 2011).

By combining AFHMM techniques with modern TinyML-based approaches, this study enhances energy disaggregation methods for device classification, improving scalability and cost-effectiveness. The real-time processing capability on edge devices offers an innovative solution that addresses both the theoretical and practical aspects of energy management systems and device identification.

(Lane et al., 2015) introduced DeepX, a software accelerator for low-power deep learning inference on mobile devices. This study focused on optimizing deep learning models for resource-constrained devices while enhancing performance through energy efficiency. Similarly, this study develops a TinyML-based system to optimize energy consumption and perform device classification on edge devices.

(Lane et al., 2015) emphasized the importance of both software and hardware optimizations for low power consumption and accelerated model inference. In this study, the principles of low-power operation are extended to energy monitoring systems by integrating TinyML. The system processes energy data in 5-second windows, making real-time predictions on edge devices more efficient, while also improving the overall energy usage and device classification process.

In contrast to (Lane et al., 2015), which primarily focuses on optimizing deep learning for mobile devices, this study applies TinyML in the context of energy monitoring, making real-time device classification possible with minimal resource use. The study also addresses challenges such as reduced memory capacity and computing power, enabling the system to operate on edge devices without compromising performance.

This study contributes to the literature by applying low-power deep learning techniques from (Lane et al., 2015) to energy monitoring and device classification. By offering a scalable and resource-efficient solution, the study expands the application potential of TinyML for real-time energy analysis, making it a significant contribution to both energy monitoring and device identification fields.

(Lane, 2023) focused on the classification of electrical devices using deep learning techniques. The study utilized deep learning algorithms to analyze energy consumption patterns, aiming to optimize these processes in terms of accuracy and efficiency. Similarly, this study also investigates the energy consumption profiles of devices and performs device classification using machine learning models.

(Lane, 2023) emphasized the power of deep learning in automating the classification of energy data, particularly for large-scale applications. In contrast, this study takes a step further by using TinyML models, deployed on edge devices, to classify devices in real-time. This allows for faster predictions with minimal dependence on centralized systems, offering advantages such as reduced latency and better data privacy.

Whereas (Lane, 2023) focuses on deep learning models in a centralized setting, this study enhances efficiency by utilizing TinyML for localized predictions. This method is optimized for low-resource environments, ensuring that real-time classification and energy analysis can be conducted on devices with memory and processing constraints.

This study provides a new perspective on machine learning for energy monitoring systems by implementing a TinyML-based solution. The shift from centralized to edge-based processing opens up opportunities for scalability, cost-efficiency, and privacy-enhanced solutions in energy management systems.

(Liu et al., 2024) provides a comprehensive review of NILM techniques in the context of smart grids, discussing their applications, challenges, and future directions. The study highlights the importance of NILM in modern energy management systems, focusing on the integration of smart grid technologies and the advancement of energy monitoring methods. Similarly, this study builds upon NILM principles to classify devices based on their energy consumption patterns, contributing to the development of efficient and practical energy management systems.

(Liu et al., 2024) identified challenges such as large-scale data processing and integrating NILM with smart grid infrastructures. This study addresses these challenges by leveraging TinyML, which enables real-time predictions directly on edge devices. This approach reduces dependence on centralized systems, significantly improving scalability, speed, and data privacy while maintaining a low environmental footprint.

Unlike (Liu et al., 2024), which offers a theoretical exploration, this study implements a real-world solution by deploying a TinyML-based system on edge devices. The combi-

nation of small time-window energy analysis and real-time processing improves device classification accuracy, making the system more efficient for use in smart grid environments.

By focusing on the practical implementation of NILM techniques using TinyML, this study advances the application of real-time, localized energy monitoring in smart grid systems, making it more adaptable to energy management needs. It adds value to the existing body of work by bridging the gap between theoretical concepts and actionable, scalable solutions for smart grids.

(Mughal, Mirza and Shafiq, 2020) focuses on hybrid machine learning approaches for appliance identification using low-frequency smart meter data. The study explores the integration of multiple machine learning techniques to improve the accuracy and efficiency of device identification in smart grid systems. Similarly, this study also aims to classify devices based on their energy consumption patterns, utilizing machine learning techniques to address the challenges of accurate and efficient device identification.

(Mughal, Mirza and Shafiq, 2020) emphasizes the cost-effectiveness and scalability of using low-frequency data for appliance classification. This study expands on that by integrating TinyML, enabling real-time, on-device predictions. By deploying the model directly on edge devices, the study reduces reliance on cloud-based systems, cutting down latency and enhancing privacy and efficiency.

While (Mughal, Mirza and Shafiq, 2020) relies on hybrid machine learning models executed in centralized settings, this work advances the idea by localizing device classification with TinyML models. Processing energy data in small, overlapping time windows enhances the system's ability to capture short-term energy usage patterns with greater accuracy.

The contribution of this study lies in combining the hybrid machine learning methods discussed in (Mughal, Mirza and Shafiq, 2020) with TinyML, resulting in a highly scalable, cost-effective, and privacy-conscious solution for real-time energy monitoring in smart grid applications.

(Mughal, Owais and Asim, 2020) presents an IoT-based home appliance management and classification system utilizing deep learning techniques. The study focuses on integrating IoT technologies with deep learning models to efficiently manage and classify appliances in smart home environments. Similarly, this study also uses the IoT and

machine learning to classify devices based on energy consumption patterns, in order to improve the accuracy and scalability of such systems.

(Mughal, Owais and Asim, 2020) explored the advantages of combining IoT with deep learning to enhance appliance classification efficiency. This study builds upon the same idea but takes a significant leap by incorporating TinyML for real-time classification directly on edge devices. This approach drastically reduces the need for centralized systems and offers benefits such as improved data privacy, reduced latency, and enhanced energy efficiency.

Where (Mughal, Owais and Asim, 2020) primarily uses deep learning on centralized platforms, this study implements a distributed system powered by TinyML. This localized approach makes the solution more efficient and scalable, addressing the limitations of traditional centralized frameworks.

The contribution of this study is in its practical application of TinyML for localized, real-time device classification. This enhances the scalability and effectiveness of IoT-based energy monitoring systems and provides an innovative solution for home appliance management.

(Ruzzelli et al., 2010) focused on the real-time recognition and profiling of electrical appliances using a single electricity sensor. Their approach to analyzing energy consumption patterns for device recognition in a non-intrusive manner provides a valuable foundation for NILM techniques. Similarly, this study also seeks to classify devices based on their energy consumption profiles, leveraging advanced techniques to enhance classification accuracy and efficiency.

Whereas (Ruzzelli et al., 2010) emphasized using a single sensor for cost-effective device recognition, this study takes the approach a step further by integrating TinyML models. This not only enables local predictions on edge devices but also eliminates the need for centralized systems. This shift results in improved latency, better data privacy, and increased energy efficiency, providing an edge over traditional methods.

While (Ruzzelli et al., 2010) relied on conventional methods for appliance assessment, this study employs modern TinyML techniques for optimized device classification. The use of 5-second time windows, with overlapping data, allows for more precise capturing of short-term energy patterns, thus enhancing the classification accuracy. Moreover, the system's scalability and practicality are significantly improved by deploying TinyML

models on resource-constrained devices.

This study contributes to the literature by combining traditional NILM approaches with cutting-edge edge computing technologies. By integrating real-time recognition with TinyML, it offers a scalable, cost-effective, and privacy-conscious solution for appliance identification and energy monitoring, bridging the gap between conventional methods and modern IoT-based solutions.

(Solatidehkordi et al., 2023) presented an IoT-based system for appliance management using deep learning. This approach integrates IoT frameworks with deep learning to enhance appliance classification in smart environments. Similarly, this study employs IoT technologies combined with machine learning to classify devices, focusing on improving scalability and efficiency.

(Solatidehkordi et al., 2023) highlighted the efficiency gains from integrating IoT with deep learning. This study builds on these principles but significantly enhances the approach by using TinyML to execute device classification directly on edge devices. By eliminating the reliance on centralized systems, this approach reduces latency and energy consumption while also improving data privacy.

Unlike (Solatidehkordi et al., 2023), which uses centralized deep learning models, this study implements a TinyML-based system tailored to edge devices with limited resources. By processing energy data in small, overlapping time windows, the study improves short-term energy pattern detection, leading to more accurate device classification. The TinyML models used are specifically optimized for efficiency on low-memory devices, ensuring scalability.

This study advances the work of (Solatidehkordi et al., 2023) by introducing TinyML to achieve real-time, localized appliance classification. By addressing scalability, data privacy, and energy efficiency, this study provides an innovative approach to energy monitoring systems in smart homes, making IoT-based solutions more practical and cost-effective.

(Vohra et al., 2023) discusses the Parquet format as a high-performance storage solution for large-scale data systems. It highlights the efficiency of Parquet in managing large datasets, particularly in big data environments. Similarly, this study also handles large energy consumption datasets but goes further by combining Parquet storage with TinyML for device classification and energy monitoring.

(Vohra et al., 2023) underscores Parquet's ability to manage large datasets efficiently. Inspired by this, this study integrates Parquet with a TinyML-based approach, enabling real-time device classification and energy monitoring. The use of TinyML in conjunction with Parquet not only enhances storage and data processing but also reduces latency, improves energy efficiency, and ensures data privacy.

In contrast to (Vohra et al., 2023), which primarily focuses on data storage and management, this study uses Parquet for organizing energy data while simultaneously utilizing TinyML models to perform real-time predictions locally on edge devices. This eliminates the need for centralized processing, enhancing system performance with low-latency predictions and optimized energy usage.

The study contributes to the literature by combining the advantages of efficient data storage formats, such as Parquet, with the real-time prediction capabilities of TinyML. This integration provides a more scalable, cost-effective solution for large-scale energy monitoring, offering practical benefits for smart grid and energy management systems.

(Wang et al., 2017) introduced deep neural networks for time-series classification, demonstrating the power of deep learning to classify time-series data without relying on feature engineering. This study adopts a similar focus on time-series energy consumption data but applies machine learning techniques, particularly TinyML, to optimize the classification process.

(Wang et al., 2017) highlighted how deep learning models can be used to automate time-series classification, achieving high accuracy without manual feature extraction. This study builds on this by using TinyML, enabling real-time predictions directly on edge devices. The adoption of TinyML not only reduces latency but also optimizes energy efficiency, making the system more practical for deployment in real-time applications.

Where (Wang et al., 2017) focuses on centralized deep learning models for time-series classification, this study advances the approach by implementing TinyML for localized processing on edge devices. By doing so, it minimizes the reliance on centralized data processing and ensures faster predictions with enhanced data privacy and energy optimization.

This study makes a novel contribution to the field by combining the deep learning methods introduced by (Wang et al., 2017) with TinyML for real-time, localized predictions. It pushes the boundaries of traditional time-series analysis by demonstrating

how machine learning, when applied to edge devices, can enhance energy monitoring systems, providing a scalable, cost-effective, and privacy-preserving solution for device classification in smart environments.

(Zeifman and Roth, 2011) provided an overview of NILM methods, examining device identification and energy monitoring technologies. Their review highlights the potential of NILM for profiling appliances and determining energy consumption patterns. Similarly, this study builds on NILM techniques to classify devices, analyzing energy consumption profiles for accurate device recognition.

In (Zeifman and Roth, 2011), the challenges in profiling appliances using NILM are discussed, along with potential solutions. This study draws on these challenges but takes a step forward by introducing TinyML, enabling real-time device classification on edge devices. This integration offers significant benefits, such as reducing latency, enhancing data privacy, and increasing energy efficiency compared to the more traditional centralized approaches proposed by (Zeifman and Roth, 2011).

While (Zeifman and Roth, 2011) relies on centralized systems for analysis, this study adopts a decentralized, TinyML-based framework, processing data locally on edge devices. This not only improves energy efficiency but also reduces dependence on central systems, making the overall process more streamlined and cost-effective. Moreover, the model is optimized to work on memory-constrained devices, ensuring a wider application.

This research contributes to the field by extending the foundational principles of NILM from (Zeifman and Roth, 2011) into the realm of edge computing. By enabling real-time, low-cost, and efficient device classification, this study offers an innovative solution for scalable energy monitoring systems, facilitating broader adoption of NILM in smart environments.

(Zhang et al., 2021) explored how statistical outlier detection and preprocessing techniques can improve NILM system accuracy by filtering noisy energy consumption data. Inspired by this, the current study also emphasizes data quality but extends the approach by integrating TinyML for real-time, decentralized classification on edge devices—enhancing both latency and privacy beyond what centralized systems offer.

While (Zhang et al., 2021) focuses on statistical preprocessing alone, this work combines such preprocessing with machine learning-based classification using short, overlapping

time windows. This design not only refines the input signal but also enables more precise temporal analysis of device activity in real-world conditions.

Building on the work of (Zhao et al., 2017), who demonstrated the power of CNNs for time-series classification without manual feature engineering, this study adopts CNN-based architectures but targets deployment on resource-constrained microcontrollers using TinyML. This shift from cloud-based to on-device inference directly addresses challenges in latency, scalability, and energy efficiency.

(Zhao et al., 2017) primarily investigated CNNs in a centralized setting. By contrast, the present research demonstrates how such models can be compressed, quantized, and executed efficiently on microcontrollers, paving the way for real-time smart energy applications in constrained environments.

(Zhao et al., 2018) compared various machine learning algorithms for appliance recognition. While their focus was model benchmarking, the current work goes further by showing how suitable architectures can be optimized for edge deployment. This enables autonomous operation without cloud connectivity, significantly reducing energy and bandwidth requirements.

3 MODEL FORMULATION

3.1 Problem Definition

The effective classification of electrical devices using aggregated energy data constitutes a technically complex challenge, particularly within edge computing environments. Although Non-Intrusive Load Monitoring (NILM) offers a foundational methodology for disaggregating household or facility-wide energy consumption into individual device profiles, its practical deployment in real-time embedded systems remains limited due to architectural and computational constraints.

Conventional NILM systems typically rely on offloading raw or preprocessed energy data to cloud-based servers for device-level disaggregation. However, this architecture is suboptimal for latency-sensitive applications or environments where continuous network connectivity cannot be guaranteed. Moreover, cloud-based inference raises potential data privacy concerns and contributes to increased system complexity and operational costs.

From a computational standpoint, existing NILM approaches often rely on resource-intensive models that demand high memory bandwidth and significant processing power. These characteristics render them unsuitable for real-time deployment on low-power microcontrollers or edge devices, where energy efficiency and low-latency inference are critical design constraints.

This research formulates the problem as a time-series classification task executed locally on the edge using a compact, low-latency machine learning model. The goal is to enable real-time device recognition based on short sequences of electrical measurements—such as current, voltage, apparent power, active power, and power factor—sampled at regular intervals.

The proposed problem formulation is required to satisfy the following key constraints :

— Operate within the memory and compute limitations of embedded devices.

— Maintain classification accuracy comparable to cloud-based NILM systems.

— Ensure low inference latency for real-time responsiveness.

— Avoid dependence on constant internet connectivity.

Accordingly, the core problem addressed in this study is the design of a TinyML-compatible NILM framework capable of performing multivariate time-series classification of device states directly on edge hardware. This formulation enables both high responsiveness and privacy-preserving deployment, representing a paradigm shift from centralized architectures to autonomous, local energy intelligence.

3.2 Assumptions

For the effective operation of the proposed TinyML-based NILM system, several key assumptions are introduced. These assumptions are essential for defining the scope and limitations of the model, and they help establish the conditions under which the model can be deemed valid. The primary assumptions are as follows :

— **Accurate Energy Consumption Measurements :** It is assumed that the EM device provides accurate, reliable, and consistent energy consumption data. This data includes essential electrical parameters such as current, voltage, active power, apparent power, and power factor. The reliability of these measurements is crucial for the classification process as they form the basis for identifying appliance usage patterns.

— **Known Appliance Profiles :** The model assumes that the energy consumption profiles of the appliances are pre-established or available in the training dataset. These profiles are based on the typical operational patterns of each appliance under normal conditions. The model relies on these known profiles to match energy consumption data with corresponding devices for accurate classification.

— **Sufficient Training Data :** A key assumption is that a sufficient amount of labeled training data is available for training the TinyML models. This data should contain accurate appliance labels and corresponding energy consumption patterns. The accuracy of the appliance classification heavily relies on the quality and quantity of this training data, ensuring that the model can generalize well to new, unseen data.

— **Edge Device Resource Constraints :** The model assumes that the TinyML models deployed on edge devices are optimized for low memory and computational power, as these devices are typically resource-constrained. The performance

of the system depends on the ability to execute the machine learning models efficiently within the limited computational and memory capacity available on these edge devices.

— **Real-time Data Processing :** It is assumed that real-time data processing is achievable with the edge devices capable of handling incoming energy consumption data and performing device classification with minimal latency. The edge devices must be sufficiently powerful to process the 5-second time window data in real-time, enabling timely predictions without significant delays.

— **No Significant Interference from Other Devices :** It is assumed that the energy consumption data from different devices is not significantly affected by electrical interference from other appliances. This assumption implies that the model can reliably distinguish between the consumption patterns of various devices. If devices operate on the same electrical circuit and cause substantial electrical noise or disturbances, the model's ability to correctly classify individual devices may be compromised.

These assumptions are critical to the operation and applicability of the proposed system. They help define the operational boundaries of the model and provide a basis for evaluating its performance under different real-world conditions. While these assumptions may not hold universally, they provide a clear framework for understanding the scope and limitations of the model in its current form.

3.3 Model Components

The proposed NILM framework consists of modular components designed to work in coordination for real-time electrical device classification using TinyML. These components and their high-level functions are summarized below :

— **Sensing Unit :** Captures real-time electrical signal data such as current, voltage, and power metrics from the monitored environment.

— **Data Aggregation and Structuring :** Processes raw measurements into structured time-series datasets suitable for training and inference. This includes time windowing, feature calculation, and formatting.

— **TinyML Classifier :** A lightweight machine learning model trained to recognize appliances from structured energy profiles. It operates directly on edge devices

with constrained memory and compute resources.

— **Inference Engine :** Executes the trained model locally to generate device classification predictions in real time, without relying on cloud connectivity.

— **Communication Module (Optional) :** Transmits classification results or raw/processed data to a central server for logging, analysis, or remote control functionalities.

Each component is optimized to operate in resource-constrained settings and contributes to the end-to-end objective of achieving responsive, low-cost, and privacy-preserving device recognition.

4 SOLUTION METHODOLOGY

The proposed solution presents a comprehensive methodology for real-time electrical device classification using time-series signals on embedded platforms. It integrates cloud-based infrastructure for scalable data management with TinyML-based inference at the edge to support low-latency, energy-efficient, and privacy-preserving applications. This hybrid architecture is specifically tailored for smart energy monitoring scenarios and builds upon NILM principles while eliminating many of the limitations associated with cloud based systems.

4.1 System Overview and Design Rationale

The system consists of three fundamental layers, each responsible for a distinct function in the end-to-end pipeline :

- **Sensing and Acquisition Layer :** Responsible for collecting high-fidelity electrical signal measurements in real-time using the Shelly Pro 3EM device.
- **Data Management and Processing Layer :** Hosted in the cloud (AWS), this layer is tasked with efficient data handling, storage, preprocessing, and dataset preparation for model training.
- **Inference and Deployment Layer :** Consists of a microcontroller-based device, which executes pre-trained, quantized machine learning models in real-time using TinyML.

This layered design ensures modularity, scalability, and robustness, allowing improvements at one layer without disrupting others. The decision to split functionality across edge and cloud components was made to balance the trade-offs between computational load, inference speed, data privacy, and storage overhead.

4.2 Data Collection Framework

The data collection framework developed in this study enables efficient structuring of electrical signal data, supports real-time inference, and ensures optimized model

execution. It is designed to operate in coordination with both edge-level monitoring and cloud-based processing, allowing seamless data flow between components. This integration facilitates reliable and scalable device classification suitable for real-world energy monitoring applications.

4.2.1 Setup

The data collection setup consists of a Shelly Pro 3EM energy monitoring device, which performs real-time acquisition of electrical parameters, and an ESP32 microcontroller, which executes on-device inference using TinyML models. In this architecture, AWS cloud services, specifically IoT Core, Lambda, and S3, are used for reliable data transmission, real-time processing, and structured storage.

The measurement data from the Shelly Pro 3EM is transmitted to AWS IoT Core, where it is routed to Lambda functions for preprocessing and formatting. The processed data are then stored in AWS S3 as structured datasets, which are later used for model training and performance evaluation. The entire setup is designed to support a continuous and responsive data pipeline from acquisition to inference. A visual representation of the system architecture is provided in Figure 1.1b.

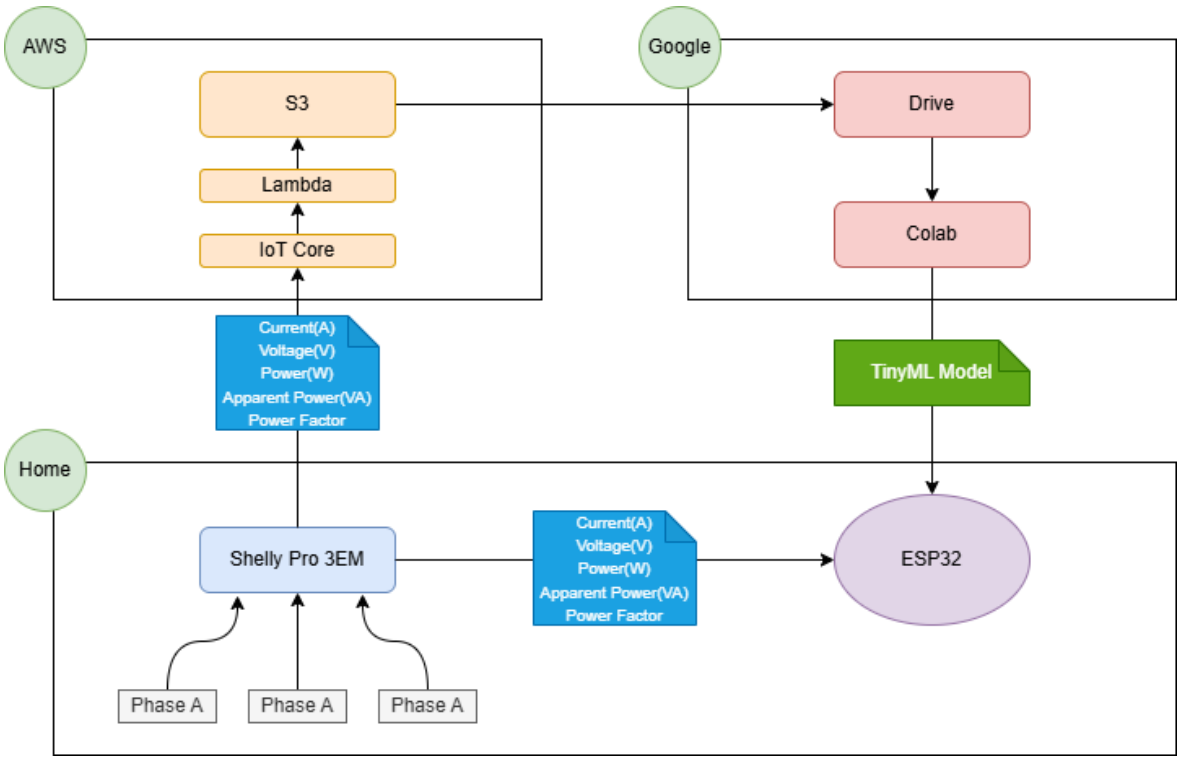


FIGURE 4.1 – System Design

4.2.2 Data Acquisition

The Shelly Pro 3EM device is utilized for real-time acquisition of electrical parameters necessary for appliance classification. Designed for both residential and industrial use, it is a three-phase energy meter capable of measuring a comprehensive set of electrical parameters at a user-configurable sampling frequency of 1 Hz. These parameters are critical for distinguishing between different appliance types based on their power consumption profiles.

The primary electrical features captured by the Shelly Pro 3EM are summarized in Table 4.1.

TABLE 4.1 – Electrical Parameters Measured by Shelly Pro 3EM

Parameter	**Symbol / Unit**	**Description**
Voltage	V (Volts)	Instantaneous line voltage measured between phase and neutral
Current	I (Amperes)	Instantaneous current measured via external CTs
Active Power	P (Watts)	Real power consumed by the device, calculated as $P = V \cdot I \cdot \cos(\theta)$
Apparent Power	S (VA)	Total power drawn from the source, calculated as $S = V \cdot I$
Power Factor	PF (Unitless)	Ratio of active power to apparent power, $PF = \cos(\theta)$
Energy	kWh	Cumulative energy consumption over time
Frequency	Hz	Supply frequency of the measured voltage

Current is measured using dedicated CTs clamped onto each phase line—denoted as CTA, CTB, and CTC—with an optional transformer on the neutral line (CTN). These CTs detect the magnetic field generated by the current and produce a low-voltage AC signal. This signal is digitized using onboard ADCs.

Voltage measurements are obtained directly through phase-to-neutral connections using voltage divider circuits, which also feed into the ADCs. Internally, the device samples current and voltage at high frequency (typically in the kilohertz range), computes derived quantities such as power and power factor, and down-samples the result to a 1 Hz reporting frequency.

The following key electrical metrics are computed internally :

$$P = V \times I \times \cos(\theta) \quad (4.1)$$

$$S = V \times I \quad (4.2)$$

$$PF = \frac{P}{S} = \cos(\theta) \quad (4.3)$$

These parameters offer a rich description of device behavior and enable accurate modeling of energy consumption dynamics.

After local computation, the device transmits structured data packets in real time to AWS IoT Core using the MQTT protocol. The data is then routed through AWS Lambda functions for preprocessing and stored in AWS S3 for long-term analysis and model training. This pipeline ensures low-latency access to clean, time-aligned datasets and supports scalable data acquisition across multiple devices and environments.

4.2.3 Data Transmission and Storage

Real-time measurement data is processed using AWS Lambda functions and stored in AWS S3 to enable long-term preservation and efficient retrieval. Upon arrival, the raw data is first parsed and cleaned by Lambda functions triggered via AWS IoT Core events. The processed entries are then saved as structured records in cloud storage.

To organize the incoming stream, a scheduled aggregation function groups the data into daily datasets. This structuring enhances accessibility and reduces overhead during model training and evaluation phases. The system includes an adaptive aggregation strategy that dynamically modifies the data organization schema based on the activity levels of monitored devices. When device usage is intense, finer-grained resolution is preserved ; during idle periods, data is coarsely summarized to reduce redundancy.

The resulting datasets are both time-aligned and lightweight, facilitating high-speed retrieval for offline processing and machine learning model development. This design ensures a scalable, storage-efficient infrastructure suitable for large-scale deployment scenarios.

4.2.4 Data Processing for Model Training

For the training phase, labeled datasets are retrieved from AWS S3 cloud storage and processed using a cloud-based Jupyter notebook environment (Google Colab). The labeling procedure involves the association of known device usage intervals with corresponding time-series segments. These labels are applied in a structured `DataFrame` format, where each time interval is annotated with the identifier of the active electrical device.

Once labeling is completed, the raw time-series data is segmented into non-overlapping windows of 30 seconds, with each window containing 30 consecutive data points sampled at 1 Hz. This results in a three-dimensional dataset structure, where each sample is represented as a 30×5 matrix, corresponding to 30 time steps and five electrical features : current, voltage, active power, apparent power, and power factor as illustrated in Table 4.2.

TABLE 4.2 – Structure of a Single Time-Series Sample

Time Step	**A**	**V**	**W**	**VA**	**PF**
t_1	I_1	V_1	P_1	S_1	PF_1
t_2	I_2	V_2	P_2	S_2	PF_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_{30}	I_{30}	V_{30}	P_{30}	S_{30}	PF_{30}

This preprocessing stage is essential for transforming the raw measurements into a supervised learning format compatible with deep learning models. The labeled and segmented dataset serves as the foundation for subsequent training, validation, and evaluation procedures.

4.2.5 Model Training and Deployment

Once the dataset is preprocessed and labeled, deep learning models are trained within a cloud-based development environment (Google Colab). The selected architectures are specifically designed for time-series classification tasks in the context of electrical device identification. Each model is trained using the segmented input windows, with the corresponding device labels serving as target outputs in a supervised learning framework.

After the training phase is complete, the resulting models are optimized for deployment on resource-constrained hardware. This optimization process includes model compression and post-training quantization using the TensorFlow Lite framework. The primary goal of quantization is to reduce the model's memory footprint and computational load by converting 32-bit floating point weights and activations into 8-bit integer representations. This enables efficient inference on embedded devices with limited processing power and memory capacity.

Following optimization, the model is converted into a TinyML-compatible format and deployed to an ESP32 microcontroller. During operation, the ESP32 receives real-time energy measurement data from the Shelly Pro 3EM device. The incoming data is processed locally through the embedded neural network, which performs device classification without requiring connection to a cloud server.

This edge-based inference architecture significantly reduces system latency, eliminates reliance on continuous network connectivity, and improves overall efficiency. By shifting decision-making to the edge, the system achieves real-time responsiveness and enhanced scalability, making it suitable for widespread deployment in intelligent energy monitoring applications.

4.2.6 Summary of Data Flow

The complete data flow of the proposed system begins with real-time measurement acquisition using the Shelly Pro 3EM device. Electrical parameters are sampled at 1 Hz and transmitted to the cloud via AWS IoT Core using a lightweight, event-driven messaging protocol. Upon reception, the data is processed by AWS Lambda functions, which format and store the information in structured datasets on AWS S3.

For model training, historical data is retrieved from AWS S3 and transferred to Google Drive, where it is accessed and processed using Google Colab. In this environment, the dataset is labeled, segmented into fixed time windows, and preprocessed for supervised learning. Deep learning models are trained on the processed data and subsequently optimized through quantization and conversion into a TinyML-compatible format using TensorFlow Lite.

The optimized model is deployed to an ESP32 microcontroller, which performs real-

time inference on live energy data received from the Shelly Pro 3EM. By embedding the classification process directly on the edge device, the system minimizes latency and reduces dependency on external cloud computation.

This hybrid framework—combining cloud-based data storage and preprocessing with on-device inference—enables efficient, scalable, and privacy-preserving electrical device classification in real-world environments.

4.3 Methodology

4.3.1 Dataset Description

The dataset used in this study comprises a total of 42,353 time-series sequences, where each sequence corresponds to a 30-second observation window of electrical measurements collected from various household appliances. The primary objective of the dataset is to facilitate supervised learning for time-series classification, particularly for appliance recognition based on characteristic patterns of power consumption.

Structurally, the dataset is organized as a three-dimensional tensor with a shape of $(42353, 30, 5)$, where :

— The first dimension (42353) represents the total number of samples.

— The second dimension (30) denotes the number of time steps per sample, corresponding to 30 consecutive seconds of measurements at a sampling rate of 1 Hz.

— The third dimension (5) includes the five recorded electrical features : current, voltage, active power, apparent power, and power factor.

Each time-series sample is associated with a unique class label that identifies the operating appliance during the recorded interval. The labels are stored in a separate one-dimensional array of shape $(42353,)$ and are encoded as integers corresponding to distinct appliance categories.

The dataset includes nine appliance classes, summarized in Table 4.3.

TABLE 4.3 – Class Distribution in the Appliance Dataset

Label	Appliance Class	Number of Samples
0	Airfryer	8943
1	Blender	940
2	Blow Dryer	2540
3	Coffee Machine	1864
4	Dishwasher	13,015
5	Dryer	9690
6	Oven	2555
7	Shaker	490
8	Tea Machine	2316

This distribution highlights a class imbalance, which was taken into account during model training and evaluation. The labeled dataset serves as the foundation for developing deep learning models aimed at real-time electrical device classification in resource-constrained environments.

4.3.2 Data Preprocessing

The raw energy consumption data collected from the Shelly Pro 3EM device undergoes a series of preprocessing steps prior to being used for model training. These steps include data segmentation, cleaning and normalization, and label transformation via one-hot encoding. The aim of preprocessing is to structure the dataset in a format that is both computationally efficient and suitable for multi-class classification tasks.

Dataset Splitting : To ensure reliable model development and fair performance assessment, the dataset is divided into three distinct subsets : training, validation, and test sets. The training set, consisting of 27,105 samples, is used to learn the model parameters. The validation set includes 6,777 samples and is employed during training to fine-tune hyperparameters and monitor overfitting. The remaining 8,471 samples form the test set, which is strictly reserved for final evaluation and reflects the model's generalization ability on unseen data. This partitioning strategy enables robust experimentation by separating model learning, tuning, and assessment phases. The sample

distribution across these subsets is illustrated in Figure 4.2.

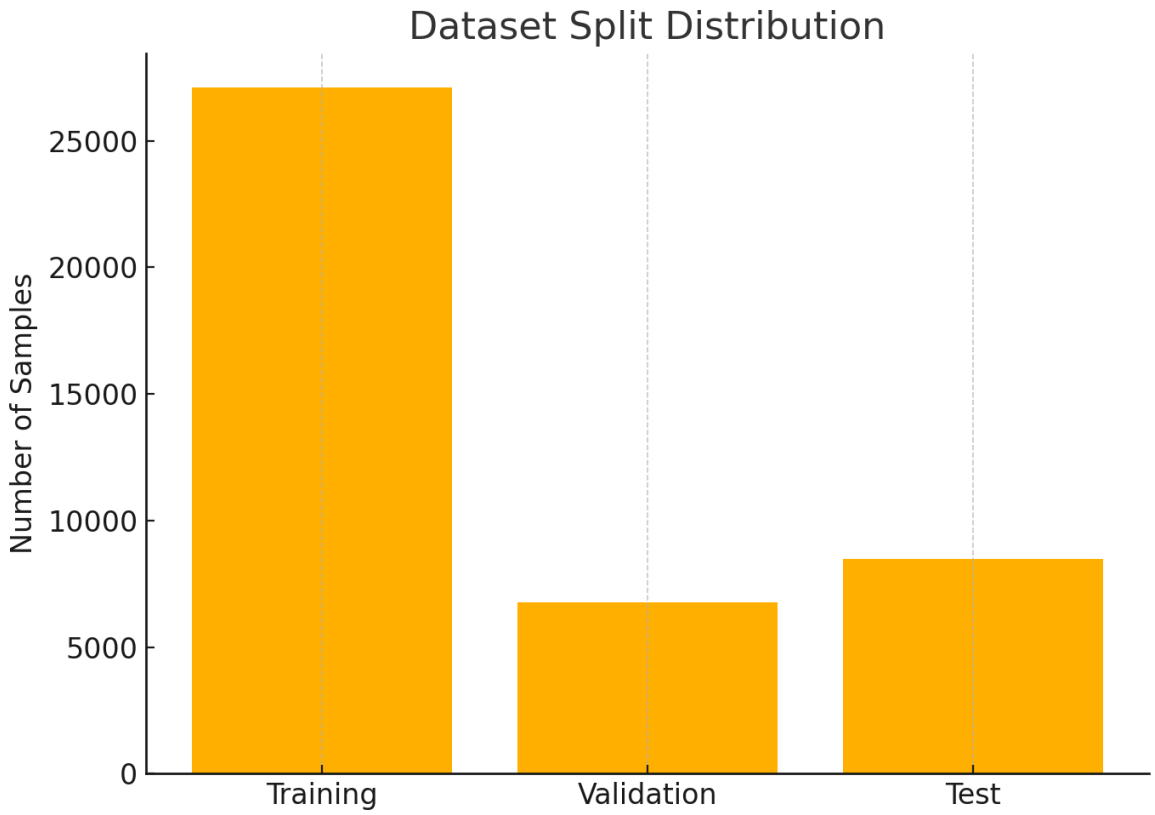


FIGURE 4.2 – Distribution of dataset across training, validation, and test sets.

Cleaning and Normalization : Each sample in the dataset represents a 30-second window of multivariate time-series data. To maintain consistency and numerical stability across features, all input sequences undergo a cleaning process. Any missing values, as well as positive or negative infinite values, are replaced with zero. Subsequently, Min-Max normalization is applied to rescale all numerical features into the range $[0, 1]$. This transformation ensures that no single feature dominates the learning process and that gradient-based optimizers converge more effectively during training.

One-hot Encoding : The categorical class labels, originally represented as integer values (ranging from 0 to 8), are transformed into one-hot encoded vectors. Each class is encoded as a binary vector of length 9, with a value of 1 at the index corresponding to the correct appliance class and 0 elsewhere. This encoding is essential for training deep learning models with categorical output layers, particularly those using softmax activation and categorical cross-entropy loss.

4.3.3 Model Architectures

In this study, six different deep learning models are evaluated for the task of time-series-based appliance classification. Each model is designed to capture distinct temporal and feature-level patterns in multivariate electrical signal data. The models differ in terms of architectural complexity, parameter count, and suitability for TinyML deployment. Their core designs are outlined below.

Fully Convolutional Network. The FCN model comprises three sequential one-dimensional convolutional layers, each followed by batch normalization and ReLU activation to ensure stable training and effective feature extraction. The first convolutional layer uses 128 filters with a kernel size of 8 to capture long-range temporal dependencies. The second layer contains 256 filters with a kernel size of 5, refining mid-level features, while the third layer uses 128 filters and a kernel size of 3 to extract fine-grained local patterns.

Instead of using fully connected layers, FCN employs Global Average Pooling (GAP), which significantly reduces the number of trainable parameters while retaining temporal representations. This architecture, originally proposed by (Wang et al., 2017) as a strong baseline for time-series classification, has been widely adopted due to its balance between performance and computational efficiency.

A final dense layer with 9 units and softmax activation is used for classification. The total parameter count is 270,985 (1.03 MB), with 269,961 trainable and 1,024 non-trainable parameters due to batch normalization. The detailed architecture of the FCN model is presented in Table 4.4.

TABLE 4.4 – Fully Convolutional Network (FCN) architecture

Layer	**Type**	**Configuration**	**Output**
1	Input	$(30, 5)$	$(30, 5)$
2	Conv1D + BN + ReLU	128 filters, kernel size 8, padding='same'	$(30, 128)$
3	Conv1D + BN + ReLU	256 filters, kernel size 5, padding='same'	$(30, 256)$
4	Conv1D + BN + ReLU	128 filters, kernel size 3, padding='same'	$(30, 128)$
5	GAP	–	(128)
6	Dense (Softmax)	9 units (appliance classes)	(9)

Time Convolutional Neural Network. TimeCNN consists of three convolutional layers, each integrated with batch normalization and ReLU activation. The first layer applies 64 filters with a kernel size of 3 to capture short-range dependencies. The second layer, with 128 filters and a kernel size of 5, targets mid-level features, while the third layer uses 256 filters and a kernel size of 7 to extract long-range patterns. This hierarchical convolutional structure has been effectively utilized for classifying variable-length time series in prior work by (Sawada et al., 2022).

The model incorporates Global Average Pooling (GAP) to compress the temporal features before classification, minimizing parameter count while maintaining representational power. A final dense layer with 9 units and softmax activation produces class probabilities. TimeCNN has a total of 275,849 parameters (1.05 MB), with 274,953 trainable and 896 non-trainable due to batch normalization layers. The detailed architecture of the TimeCNN model is presented in Table 4.5.

TABLE 4.5 – Time Convolutional Neural Network (TimeCNN) architecture

Layer	**Type**	**Configuration**	**Output**
1	Input	$(30, 5)$	$(30, 5)$
2	Conv1D + BN + ReLU	64 filters, kernel size 3, padding='same'	$(30, 64)$
3	Conv1D + BN + ReLU	128 filters, kernel size 5, padding='same'	$(30, 128)$
4	Conv1D + BN + ReLU	256 filters, kernel size 7, padding='same'	$(30, 256)$
5	GAP	–	(256)
6	Dense (Softmax)	9 units (appliance classes)	(9)

Residual Network for Time-Series Classification. ResNet-ID employs residual connections to mitigate the vanishing gradient problem and preserve learned representations across deeper layers. Such architectures have been shown to be highly effective in time-series classification tasks, as extensively discussed by Fawaz et al. (Fawaz et al., 2019). The model begins with a convolutional layer (64 filters, kernel size 3), followed by batch normalization and ReLU activation. It includes three residual blocks, each containing two convolutional layers (kernel size 3) with filter sizes progressively increasing as 32, 64, and 128. Each block integrates skip connections and optional 1×1 convolutions to align tensor dimensions.

After the residual layers, GAP is applied, followed by a dense layer with 128 units. The final classification layer has 9 units with softmax activation. The model contains 135,081 parameters (527.66 KB), of which 134,057 are trainable and 1,024 are non-

trainable. The detailed architecture of the ResNet-ID model is presented in Table 4.6.

TABLE 4.6 – Residual Network (ResNet-ID) architecture

Layer	**Type**	**Configuration**	**Output**
1	Input	$(30, 5)$	$(30, 5)$
2	Conv1D + BN + ReLU	64 filters, kernel size 3, padding='same'	$(30, 64)$
3	Residual Block 1	2×Conv1D (32 filters, $k = 3$) + skip connection	$(30, 32)$
4	Residual Block 2	2×Conv1D (64 filters, $k = 3$) + skip connection	$(30, 64)$
5	Residual Block 3	2×Conv1D (128 filters, $k = 3$) + skip connection	$(30, 128)$
6	GAP	–	(128)
7	Dense	128 units (ReLU)	(128)
8	Dense (Softmax)	9 units (appliance classes)	(9)

Temporal Convolutional Network. The TCN model is built using causal and dilated convolutions to efficiently model long-range dependencies in time-series data, following principles outlined by (Pelletier et al., 2018). It consists of three residual blocks, each composed of two 1D convolutional layers (64 filters, kernel size 3) with increasing dilation rates of 1, 2, and 4, respectively. Each convolutional layer is followed by batch normalization and dropout (rate = 0.3). Causal padding ensures that the output at each time step depends only on the current and previous inputs, preserving the temporal structure required for sequence modeling.

The model integrates skip connections in each residual block to preserve gradient flow, using 1×1 convolutions when needed to match dimensions. After the final residual block, global average pooling (GAP) is applied to reduce the temporal dimension, followed by a dense layer with 9 units and softmax activation for multi-class classification. The total parameter count is 73,609 (287.54 KB), with 72,841 trainable and 768 non-trainable parameters. The architecture of the TCN model is detailed in Table 4.7.

TABLE 4.7 – Temporal Convolutional Network (TCN) architecture

Layer	**Type**	**Configuration**	**Output**
1	Input	$(30, 5)$	$(30, 5)$
2	Residual Block 1	2x Conv1D (64 filters, k=3, dilation=1), BN, Dropout	$(30, 64)$
3	Residual Block 2	2x Conv1D (64 filters, k=3, dilation=2), BN, Dropout	$(30, 64)$
4	Residual Block 3	2x Conv1D (64 filters, k=3, dilation=4), BN, Dropout	$(30, 64)$
5	GAP	–	(64)
6	Dense (Softmax)	9 units (appliance classes)	(9)

Squeeze-and-Excitation Time-Series Network. SE-TSNet enhances a conventional 1D CNN architecture with Squeeze-and-Excitation (SE) blocks that dynamically recalibrate channel-wise feature responses. The model consists of three convolutional layers designed to capture varying temporal dependencies : the first layer uses 32 filters (kernel size 3) for short-term features, the second employs 64 filters (kernel size 5) for mid-range patterns, and the third layer utilizes 128 filters (kernel size 7) to extract long-range dependencies.

After each convolutional layer, an SE block is applied. These blocks include global average pooling (GAP) followed by a fully connected squeeze layer and a sigmoid excitation layer, enabling the model to emphasize informative channels while suppressing less relevant ones, as originally proposed by Hu et al. (Hu et al., 2018). Following the final SE block, GAP is applied again, and the output is passed through a dense layer with 9 softmax units for classification. The model has 75,077 trainable parameters (293.27 KB). The detailed architecture is provided in Table 4.8.

TABLE 4.8 – Squeeze-and-Excitation Time-Series Network (SE-TSNet) architecture

Layer	**Type**	**Configuration**	**Output**
1	Input	$(30, 5)$	$(30, 5)$
2	Conv1D	32 filters, kernel size 3, padding='same'	$(30, 32)$
3	SE Block	GAP $\rightarrow$ FC(4) $\rightarrow$ Sigmoid $\rightarrow$ Scale	$(30, 32)$
4	Conv1D	64 filters, kernel size 5, padding='same'	$(30, 64)$
5	SE Block	GAP $\rightarrow$ FC(4) $\rightarrow$ Sigmoid $\rightarrow$ Scale	$(30, 64)$
6	Conv1D	128 filters, kernel size 7, padding='same'	$(30, 128)$
7	SE Block	GAP $\rightarrow$ FC(4) $\rightarrow$ Sigmoid $\rightarrow$ Scale	$(30, 128)$
8	GAP	–	(128)
9	Dense (Softmax)	9 units (appliance classes)	(9)

Simple Recurrent Neural Network. The SimpleRNN model employs recurrent neural units to capture sequential dependencies within time-series data, a widely used approach in early time-series classification architectures (Smirnov and Nguifo, 2018). It consists of two stacked SimpleRNN layers : the first layer contains 64 units and returns the entire sequence, while the second layer has 32 units and outputs only the final hidden state. Both layers use `tanh` activation and incorporate L2 regularization ($\lambda = 0.001$) to mitigate overfitting risks.

Each recurrent layer is followed by batch normalization to stabilize training, and a dropout layer (rate = 0.3) is applied for additional regularization. Finally, a dense

layer with 9 softmax-activated units generates the classification output corresponding to the appliance classes. The model has a total of 8,265 parameters (32.29 KB), with 8,073 trainable and 192 non-trainable parameters. The full architecture is presented in Table 4.9.

TABLE 4.9 – Simple Recurrent Neural Network (SimpleRNN) architecture

Layer	**Type**	**Configuration**	**Output**
1	Input	$(30, 5)$	$(30, 5)$
2	SimpleRNN	64 units, return_sequences=True, `tanh`, L2($\lambda = 0.001$)	$(30, 64)$
3	BatchNorm + Dropout	Dropout rate = 0.3	$(30, 64)$
4	SimpleRNN	32 units, return_sequences=False, `tanh`, L2($\lambda = 0.001$)	(32)
5	BatchNorm + Dropout	Dropout rate = 0.3	(32)
6	Dense (Softmax)	9 units (appliance classes)	(9)

4.3.4 Model Training

The training process for all deep learning models follows a structured and systematic methodology aimed at maximizing classification accuracy while ensuring robust generalization. Each model is trained using a supervised learning paradigm, where the input consists of 30 time steps per sample, with 5 electrical features recorded at each step. The corresponding output is a one-hot encoded vector indicating the active appliance during that time window.

During training, the **Adam optimizer** is employed with its default learning rate of 0.001. Adam is selected for its adaptive learning rate capabilities and efficient convergence in deep neural networks. The optimization objective is defined by the **categorical cross-entropy loss function**, which is widely used in multi-class classification problems due to its ability to measure the divergence between the predicted probability distribution and the true class label.

To prevent overfitting and enhance the model's generalization capability, several regularization techniques are integrated into the training loop :

— **EarlyStopping** is applied with a patience of 5 epochs. If the validation loss does not improve for five consecutive epochs, training is halted to prevent overfitting and unnecessary computation.

— **ReduceLROnPlateau** is utilized to dynamically reduce the learning rate. When the validation loss plateaus for three epochs, the learning rate is halved, allowing

for more fine-grained weight updates in subsequent training stages.

— **ModelCheckpoint** is activated to store the best-performing model based on validation accuracy. This ensures that the most optimal version of the model is preserved, even if later epochs degrade performance.

The training is executed for a maximum of 50 epochs ; however, in practice, the process often terminates earlier due to early stopping. A **batch size of 32** is used, which provides a balance between computational efficiency and gradient estimation stability.

This training strategy is consistently applied across all model architectures evaluated in this study, ensuring fair comparison and reproducible performance metrics.

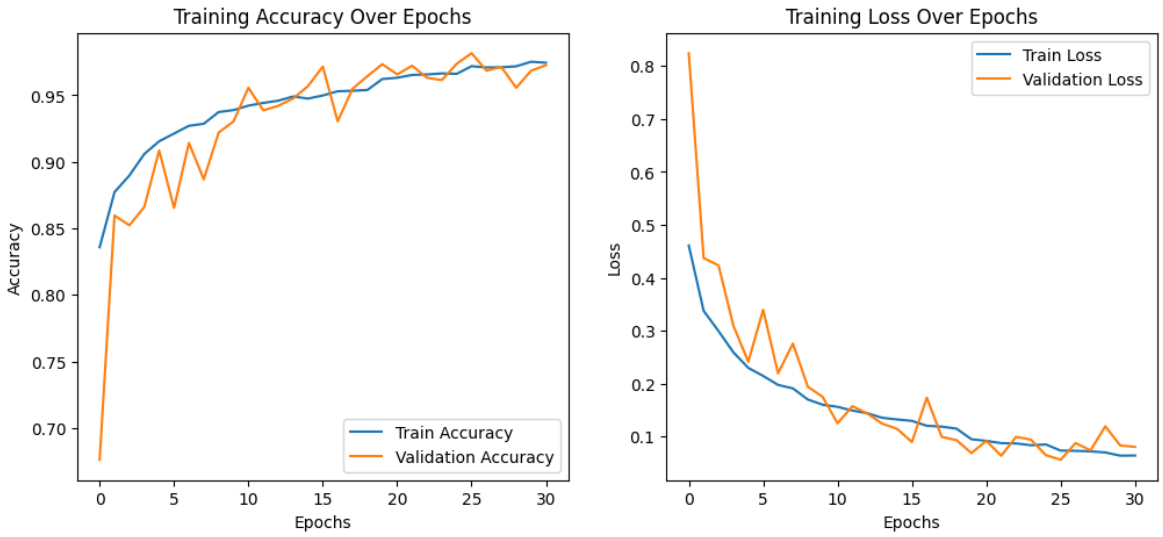


FIGURE 4.3 – FCN model training and validation accuracy/loss over epochs.

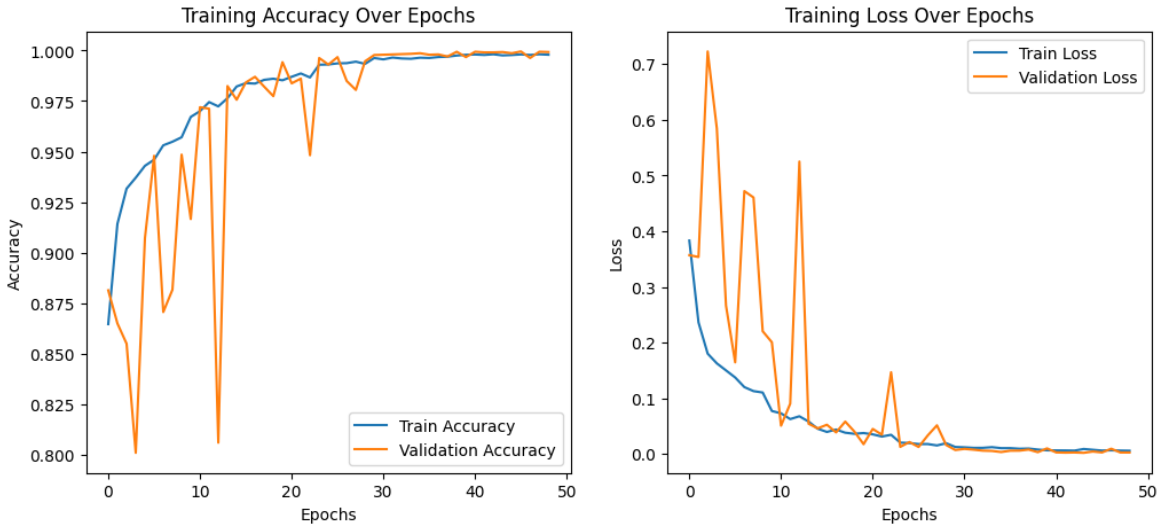


FIGURE 4.4 – ResNet-ID model training and validation accuracy/loss over epochs.

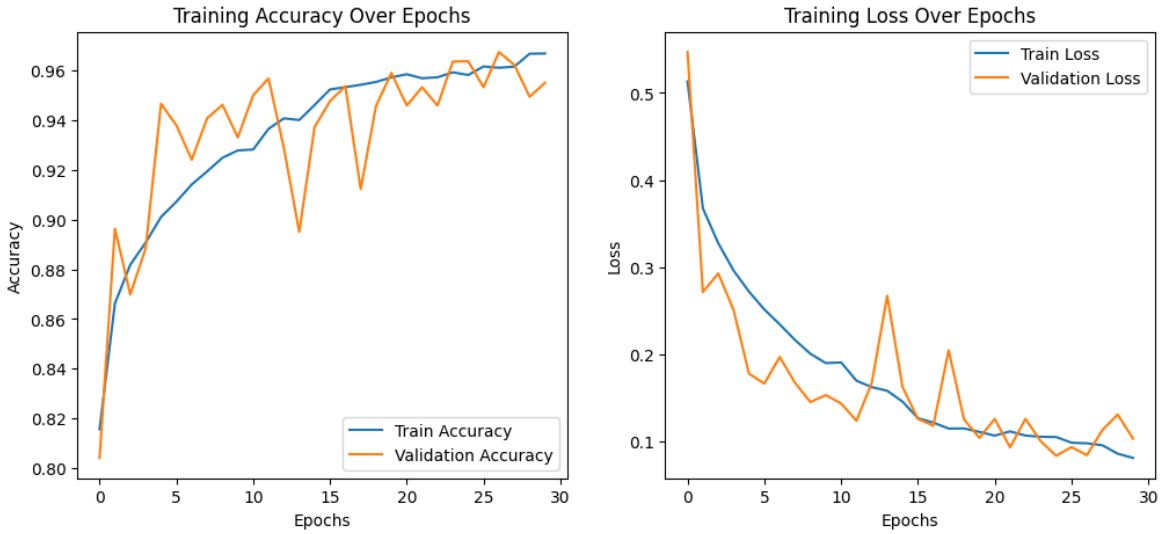


FIGURE 4.5 – Time-CNN model training and validation accuracy/loss over epochs.

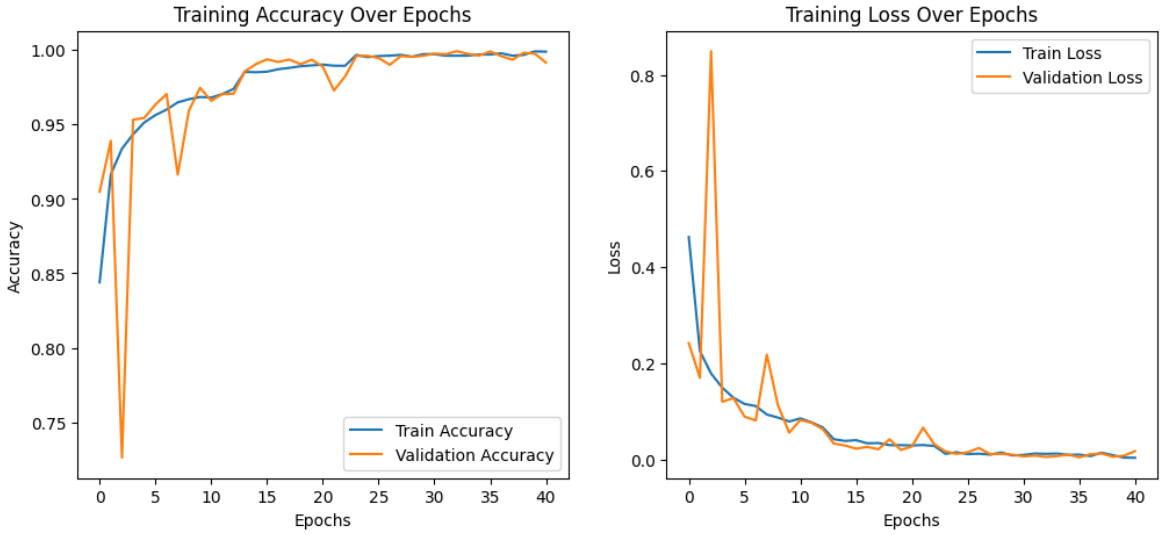


FIGURE 4.6 – SE-TSNet model training and validation accuracy/loss over epochs.

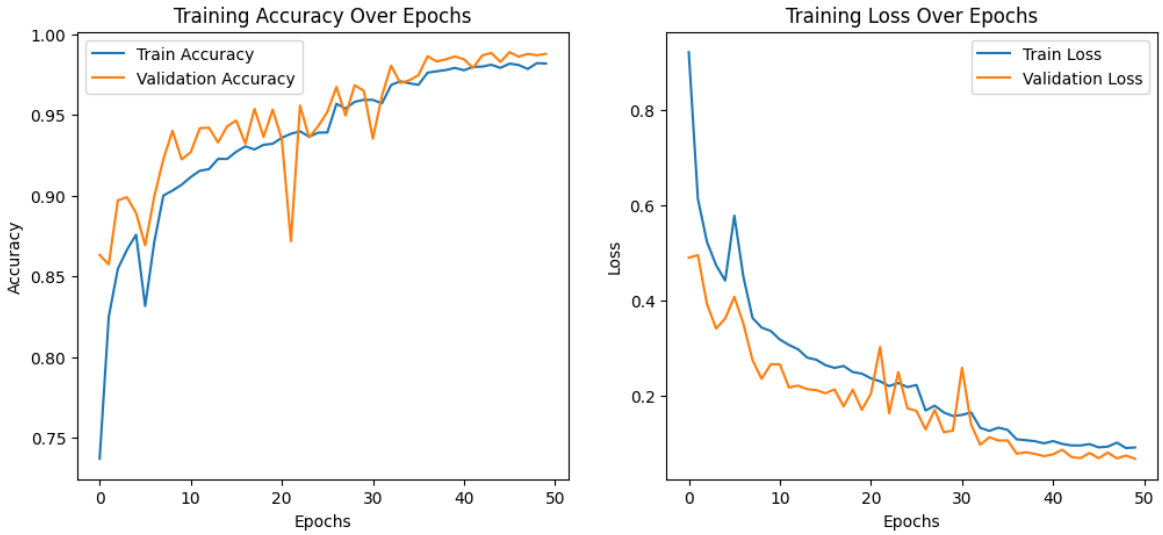


FIGURE 4.7 – SimpleRNN model training and validation accuracy/loss over epochs.

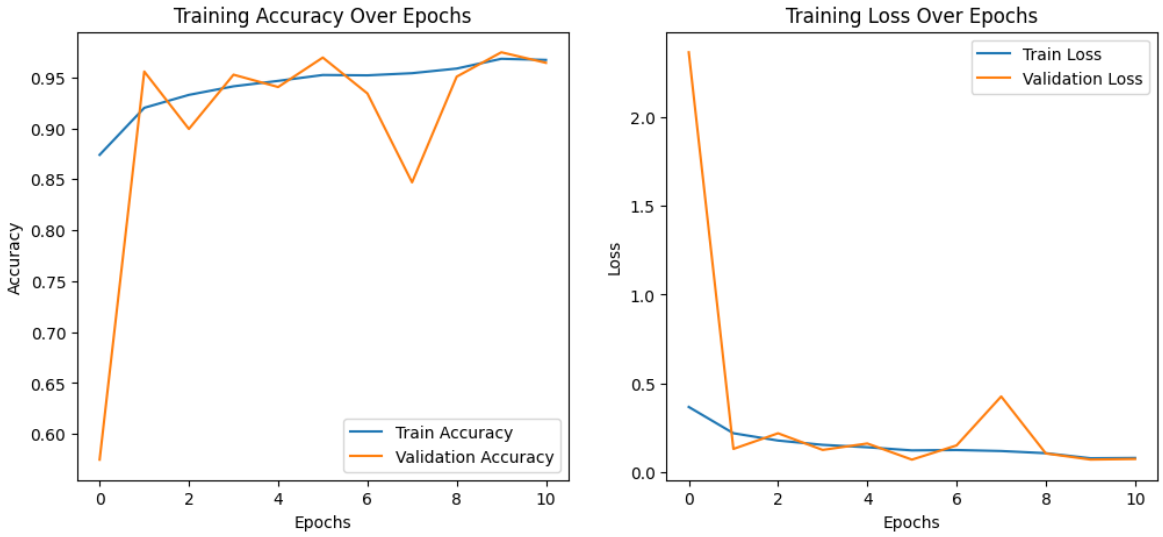


FIGURE 4.8 – TCN model training and validation accuracy/loss over epochs.

The training and validation performance of all six deep learning models used in this study are illustrated through the evolution of classification accuracy and loss over epochs. These are shown in Figures 4.3 to 4.8.

Among the tested models, ResNet-ID and FCN demonstrate fast and stable convergence with minimal overfitting, as shown by the close alignment between training and validation curves. Time-CNN and SE-TSNet also perform well, although minor fluctuations in validation loss are observed, particularly in the early epochs. The SimpleRNN model exhibits greater instability, suggesting challenges related to vanishing gradients

or sequential dependency learning. The TCN model, however, provides competitive performance with smooth loss minimization and increasing validation accuracy, highlighting its suitability for time-series classification on short windows.

These plots support the conclusion that advanced architectural features—such as residual connections, temporal convolutions, and attention mechanisms—improve training dynamics and generalization in energy-based device classification tasks.

4.3.5 Model Optimization

To ensure compatibility with memory-constrained edge devices and support efficient real-time inference, a uniform model optimization strategy was applied across all trained architectures. The goal of this process was to minimize memory footprint, reduce computational overhead, and enable seamless deployment in TinyML environments.

The optimization pipeline involved the use of **TensorFlow Lite** post-training quantization techniques. Specifically, two quantization methods were employed :

— **Integer-only quantization :** All weights and activations were converted from 32-bit floating-point precision to 8-bit integers (INT8), which significantly reduces model size and allows for fast, low-power inference on embedded hardware.

— **Dynamic range quantization :** In this approach, model weights are stored in reduced precision (e.g., INT8), while activations are dynamically quantized at runtime. This method allows more flexibility during inference by adapting activation ranges based on the input distribution.

To support accurate quantization, a representative dataset drawn from real test data was used for calibration. This ensures that the quantized model maintains high fidelity to the original floating-point model during inference.

Following quantization, the models were converted to the `.tflite` format for deployment. The resulting TensorFlow Lite models demonstrated significant reductions in size—often exceeding 80% compression—without meaningful degradation in classification performance.

This optimization process enables high-speed, energy-efficient inference on edge platforms such as the ESP32 microcontroller. It forms a critical step in the deployment

pipeline by bridging the gap between complex deep learning architectures and real-world embedded applications.

4.3.6 Edge Inference

To evaluate the real-world applicability of the proposed models, fully quantized TensorFlow Lite versions were deployed to an **ESP32-WROOM-32** microcontroller—an energy-efficient embedded device with limited memory and processing capacity. The models were converted into `.h` C++ header files using the TensorFlow Lite for Microcontrollers (TFLM) framework, enabling direct compilation and execution on-device.

Upon deployment, the ESP32 initializes the model at startup and performs real-time inference using the onboard TFLite interpreter without relying on cloud or external computation. Input sequences are preprocessed on-device to match the quantization format before prediction.

The ESP32's balance of cost, power efficiency, and wireless capabilities makes it well suited for edge AI tasks such as appliance recognition. Its hardware specifications relevant to TinyML deployment are summarized in Table 4.10 ; additional details can be found in the official datasheet (Systems, 2022).

TABLE 4.10 – ESP32 Microcontroller Specifications Relevant to Edge Deployment

Feature	**Specification**
Microcontroller	Tensilica Xtensa LX6 Dual-core
Clock Speed	Up to 240 MHz
Flash Memory	4 MB (external SPI)
SRAM	520 KB
Wireless	Wi-Fi 802.11 b/g/n, Bluetooth v4.2
USB/UART	UART, SPI, I2C, I2S supported
Power Supply	3.3 V
Quantization Compatibility	Supports int8 models via TFLM

This edge inference strategy provides several critical benefits :

— **Low-latency execution :** Predictions are generated in milliseconds, allowing for immediate response to incoming data.

— **Reduced power consumption :** The use of integer-only quantized models significantly decreases computational cost, aligning with the constraints of battery-

powered systems.

— **Enhanced data privacy :** Since all computations are performed locally, no raw data is transmitted externally, ensuring complete control over sensitive energy usage information.

In energy-constrained edge applications, minimizing power consumption is critical to ensure long-term operation and system sustainability. The ESP32 microcontroller offers multiple power modes and supports low-energy communication protocols, making it a strong candidate for real-time inference tasks on the edge. In this work, Bluetooth Low Energy (BLE) is selected as the communication protocol for transmitting classification results due to its significantly lower energy requirements compared to Wi-Fi. To quantify the system's efficiency, we estimate the total energy consumption under a typical operating scenario in which the device performs one inference per minute, transmits the result via BLE, and enters deep sleep mode between operations.

The power consumption values for each mode are based on ESP32 datasheets :

— **Active + BLE transmission :** 20 mA at 3.3 V $\Rightarrow$ 66 mW
— **Deep Sleep :** 0.15 mA at 3.3 V $\Rightarrow$ 0.495 mW

There are 1440 inferences per day (one per minute), resulting in :

Active duration per day = 1440 seconds, Sleep duration per day = 86400−1440 = 84960 seconds

The energy consumed in each state is :

$$E_{\text{active}} = 66 \text{ mW} \times \frac{1440}{3600} = 26.4 \text{ mWh}$$

$$E_{\text{sleep}} = 0.495 \text{ mW} \times \frac{84960}{3600} \approx 11.72 \text{ mWh}$$

Total daily energy consumption :

$$E_{\text{daily}} = E_{\text{active}} + E_{\text{sleep}} = 26.4 + 11.72 = 38.12 \text{ mWh}$$

Using this daily value, we can compute long-term consumption :

$$E_{\text{weekly}} = 7 \times 38.12 = 266.84 \text{ mWh}$$

$$E_{\text{monthly}} \approx 30 \times 38.12 = 1143.6 \text{ mWh} = 1.14 \text{ Wh}$$

$$E_{\text{yearly}} = 365 \times 38.12 = 13913.8 \text{ mWh} = 13.91 \text{ Wh}$$

These results show that even with continuous daily operation, the system consumes less than 14 Wh per year, making it highly suitable for long-term deployment in energy-constrained environments.

Besides that, the successful deployment and operation of the models on ESP32 validate the feasibility of the proposed TinyML-based classification system for real-time appliance recognition in constrained edge environments.

5 COMPUTATIONAL STUDIES

5.1 Performance of Standard Models

The performance of standard models was evaluated using the test dataset prior to post-training quantization. Evaluation metrics included accuracy, precision, recall, and F1-score, each computed using weighted averages to account for class imbalance in the dataset. Additionally, the total model size was recorded to assess the feasibility of deployment on resource-constrained edge devices. The results are summarized in Table 5.1.

TABLE 5.1 – Test Performance of Standard Models

Model	**Accuracy (%)**	**Precision**	**Recall**	**F1-Score**	**Model Size (MB)**
FCN	98.55	98.55	98.55	98.53	3.16
TimeCNN	96.52	96.58	96.52	96.42	3.21
ResNet-ID	99.95	99.95	99.95	99.95	1.68
TCN	97.08	97.22	97.08	97.10	0.97
SE-TSNet	99.88	99.88	99.88	99.88	0.94
SimpleRNN	98.60	98.60	98.60	98.56	0.14

Among all models, **ResNet-ID** achieved the highest classification performance, reaching 99.95% in all core metrics. Its use of residual connections enables superior feature extraction while maintaining a moderate model size (1.68 MB), making it a strong candidate for deployment on higher-capacity edge platforms.

The presence of residual connections appears to enhance feature discriminability, leading to its outstanding performance. The ResNet-ID model produced nearly perfect

classification across all classes as illustrated in Figure 5.1

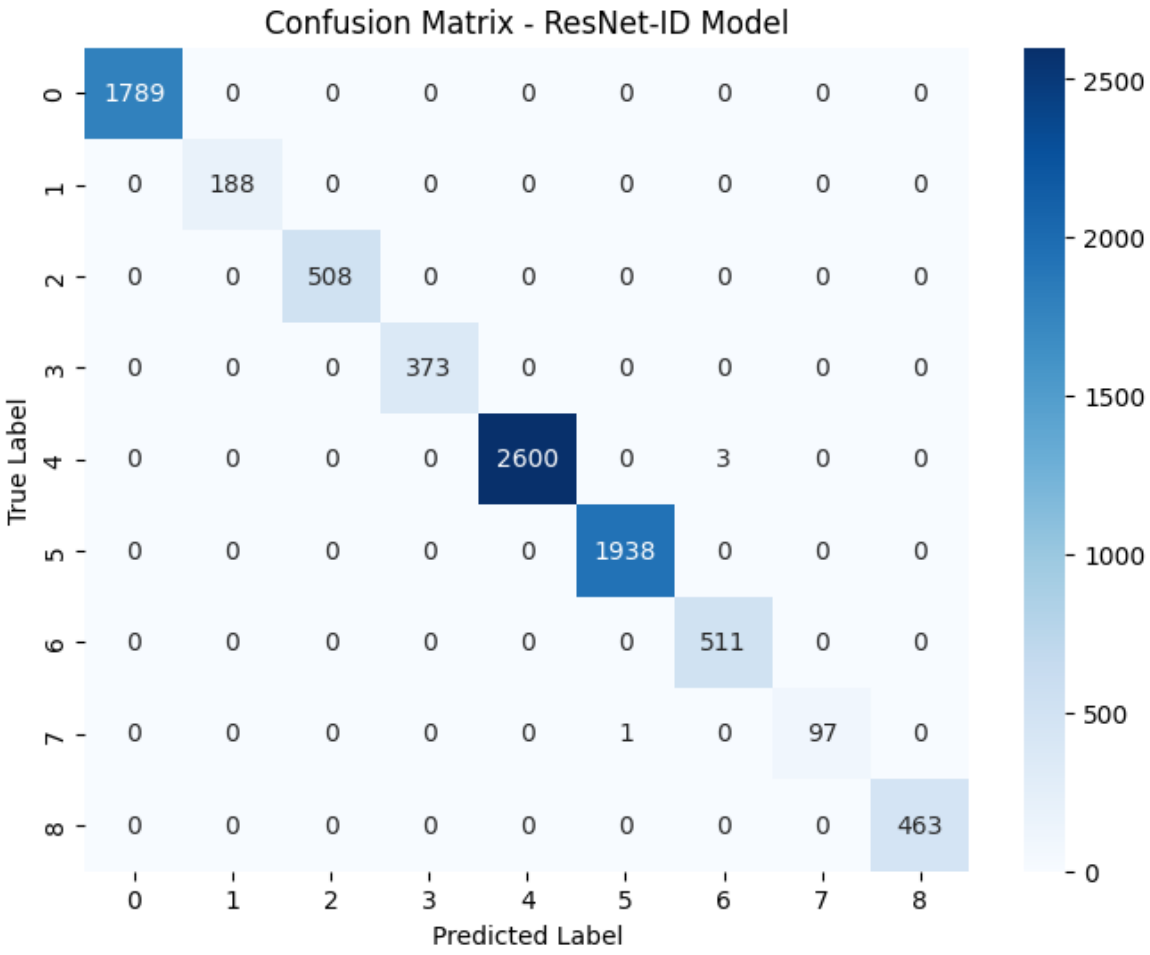


FIGURE 5.1 – Confusion matrix for ResNet-ID model.

SE-TSNet also performed exceptionally well, with a test accuracy of 99.88% and an F1-score of 99.88. It leverages squeeze-and-excitation blocks to dynamically recalibrate channel-wise features, leading to robust generalization with a compact model size of only 0.94 MB—the smallest among convolution-based architectures.

SE-TSNet achieves highly accurate classification with minimal misclassification, effectively distinguishing between appliance classes even with similar temporal behaviors

(see Figure 5.2).

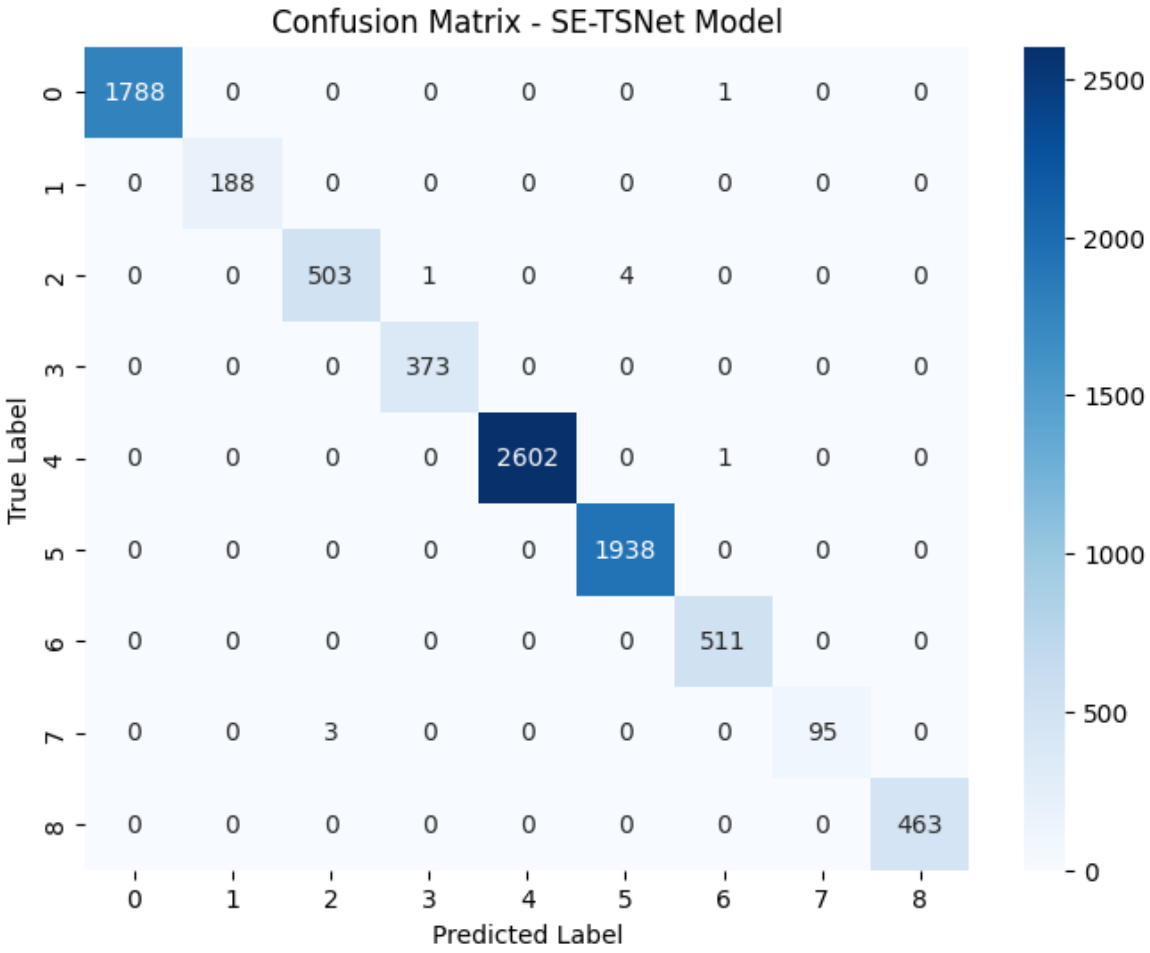


FIGURE 5.2 – Confusion matrix for SE-TSNet model.

FCN and **SimpleRNN** delivered comparable accuracy levels of 98.55% and 98.60%, respectively. However, their architectural differences are notable. FCN relies on fully convolutional layers and has a model size of 3.16 MB, whereas SimpleRNN, with its recurrent structure, achieves similar accuracy with the smallest model footprint (0.14 MB). While lightweight, SimpleRNN may struggle to capture long-range dependencies due to its sequential nature.

The FCN model achieved strong classification accuracy across all classes, with very

limited confusion between similar devices (see Figure 5.3).

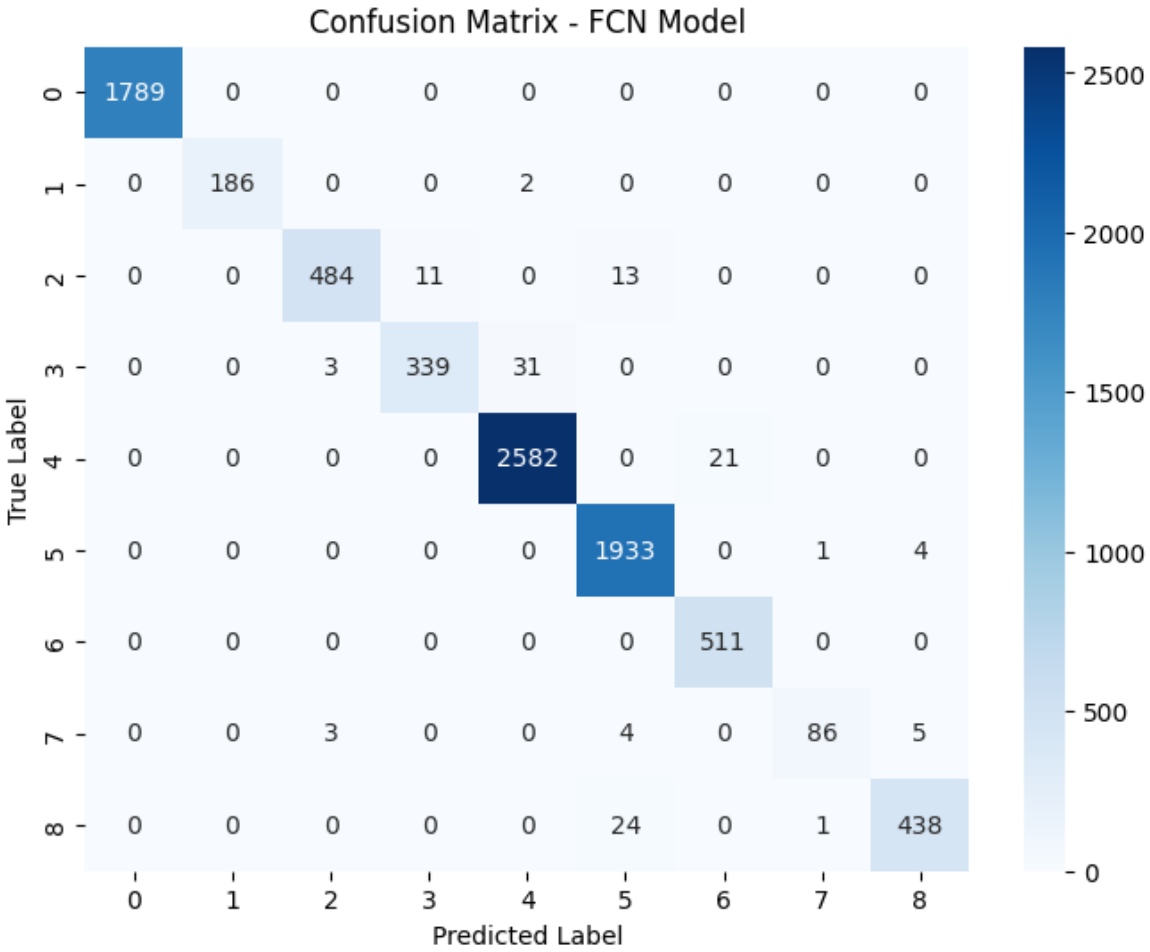


FIGURE 5.3 – Confusion matrix for FCN model.

The SimpleRNN model performed well overall but showed slight confusion between

appliance types with overlapping activation patterns (see Figure 5.4).

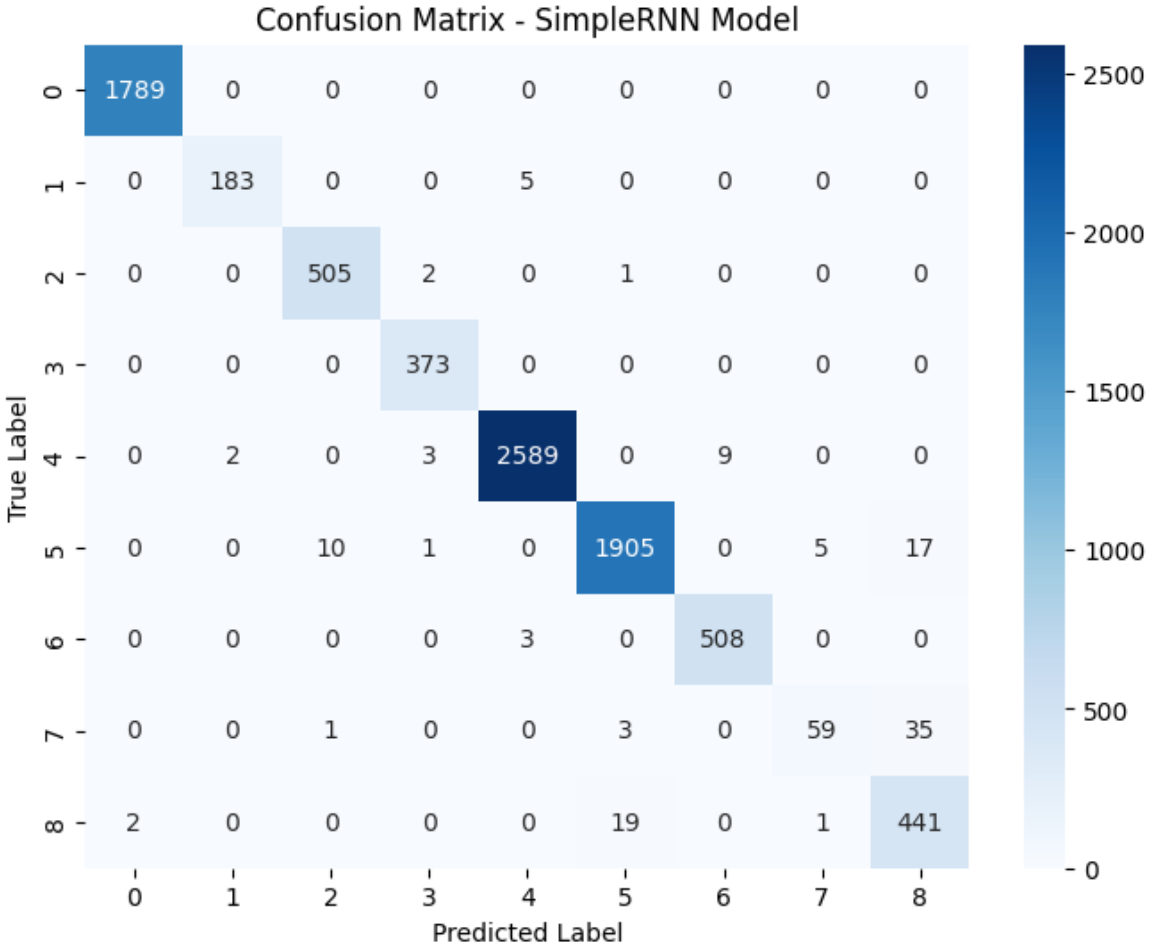


FIGURE 5.4 – Confusion matrix for SimpleRNN model.

TimeCNN demonstrated slightly lower performance (96.52%), potentially due to its fixed kernel sizes limiting flexibility in capturing multi-scale temporal features. Despite this, its precision and recall remained competitive, indicating reliable generalization across appliance categories.

TimeCNN experienced moderate confusion particularly in classes with similar periodic

behaviors, which may be attributed to limitations in kernel diversity (see Figure 5.5).

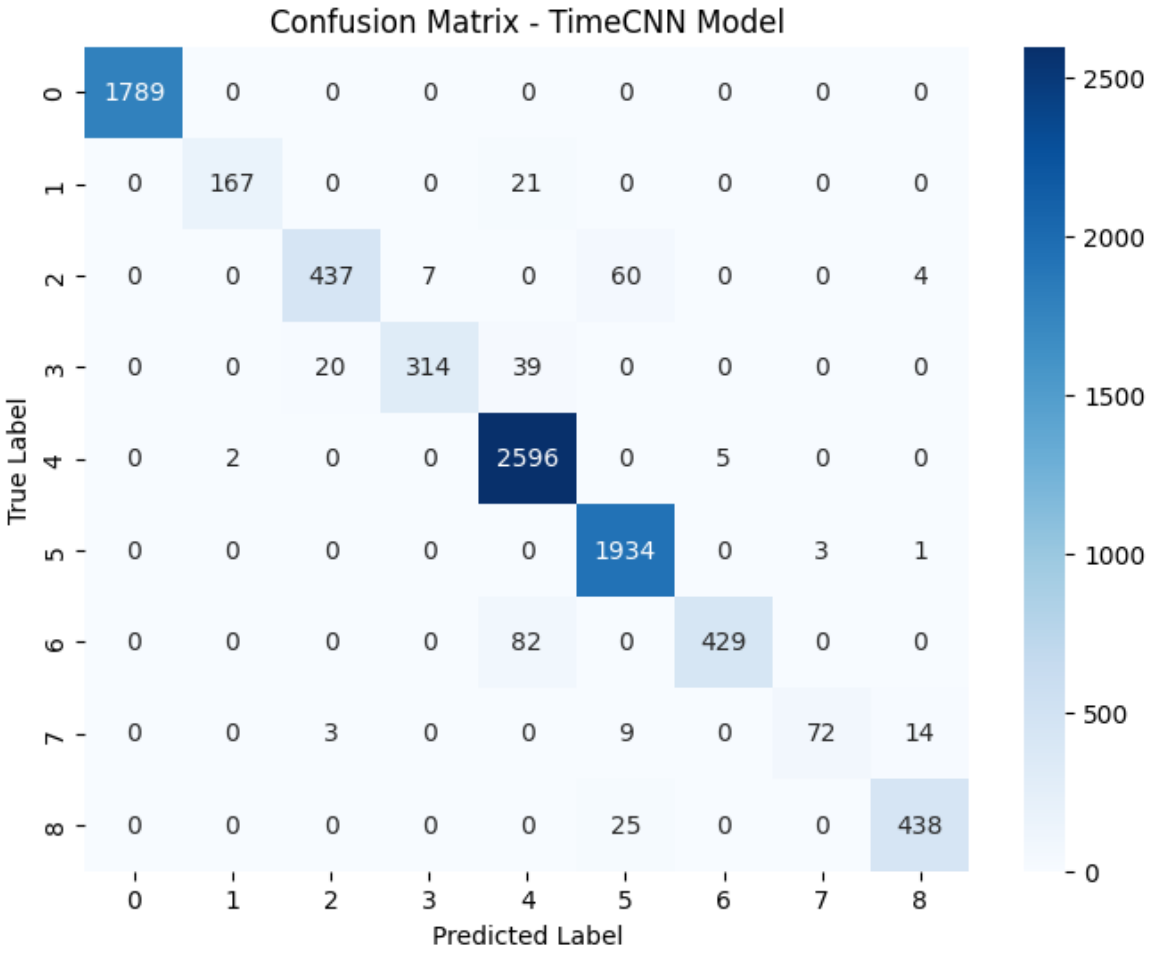


FIGURE 5.5 – Confusion matrix for TimeCNN model.

TCN achieved a test accuracy of 97.08% with a model size of only 0.97 MB, validating the effectiveness of dilated convolutions for long-range dependency modeling. Its balance between accuracy and compactness makes it a viable choice for size-critical edge deployments.

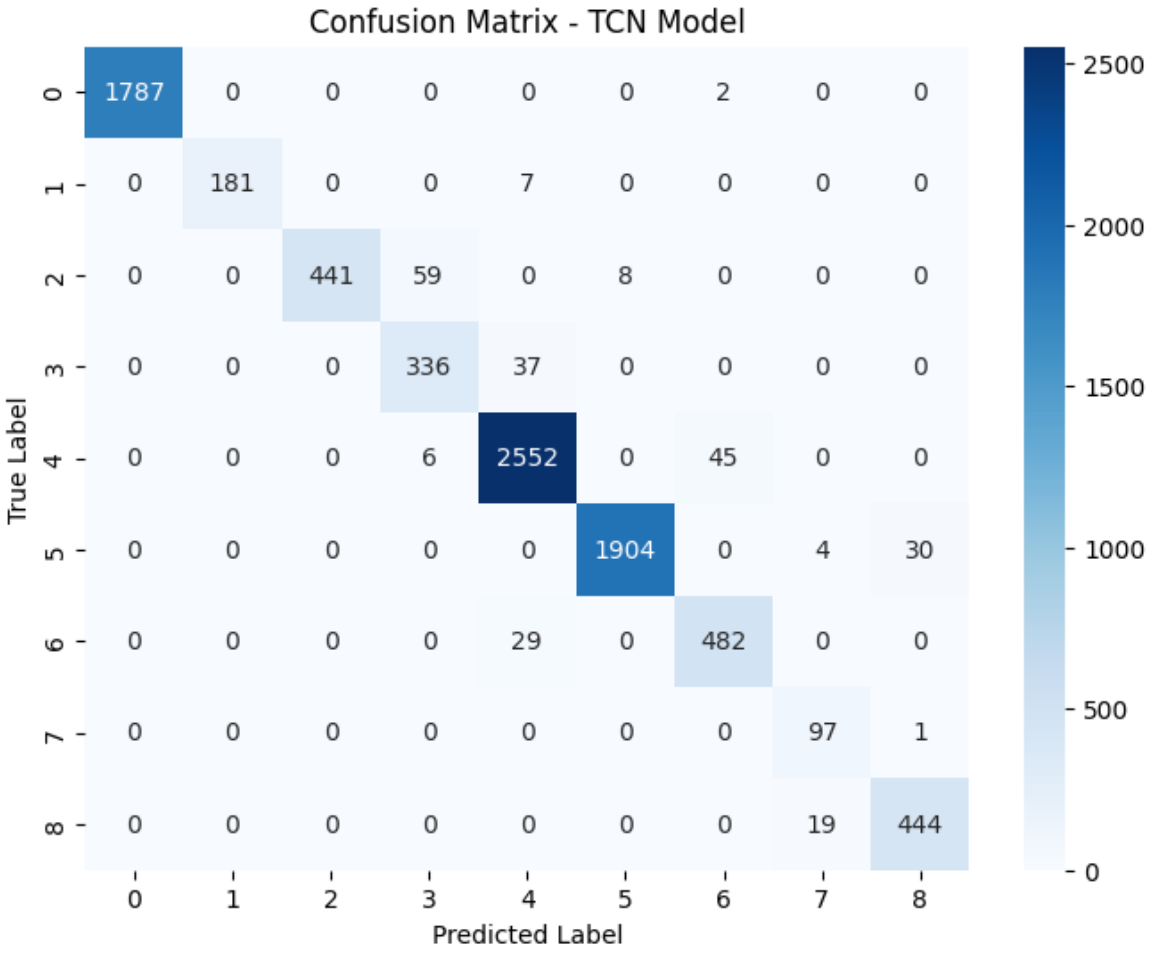


FIGURE 5.6 – Confusion matrix for TCN model.

The TCN model, as seen in Figure 5.6, showed consistent predictions but struggled slightly with classes that require fine-grained temporal resolution.

These results highlight trade-offs between model complexity, predictive performance, and hardware compatibility, guiding selection based on deployment constraints and target use cases.

5.2 Performance of Quantized Models

After applying post-training quantization, all models were re-evaluated on the test dataset to assess their classification performance in a memory-constrained environment. In this phase, core evaluation metrics such as accuracy, precision, recall, and F1-score were recalculated. In addition, the post-quantization model sizes were recorded to mea-

sure the memory reduction achieved. The results are summarized in Table 5.2.

TABLE 5.2 – Test Performance of Quantized Models

Model	Accuracy (%)	Precision	Recall	F1-Score	Model Size (KB)
FCN	98.39	98.41	98.39	98.38	282.84
TimeCNN	96.29	96.40	96.29	96.19	285.51
ResNet-ID	94.75	94.90	94.75	94.70	177.59
TCN	87.71	89.81	87.71	88.05	107.87
SE-TSNet	99.32	99.38	99.32	99.33	98.38
SimpleRNN	97.65	97.63	97.65	97.63	192.98

SE-TSNet preserved its classification reliability after quantization, maintaining minimal confusion across appliance classes (see Figure 5.7).

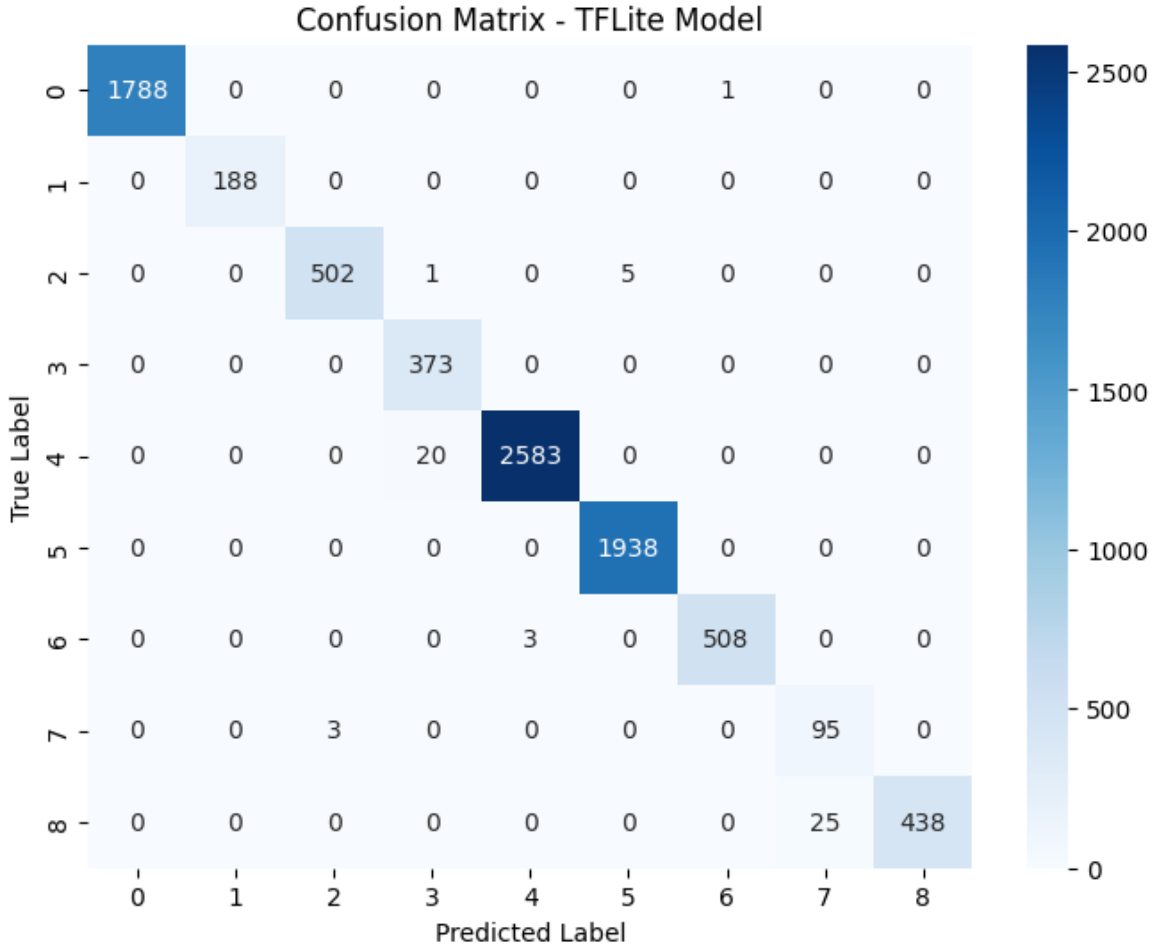


FIGURE 5.7 – Confusion matrix for quantized SE-TSNet model.

FCN retained a high degree of precision and generalization, with only slight confusion

in certain classes (see Figure 5.8).

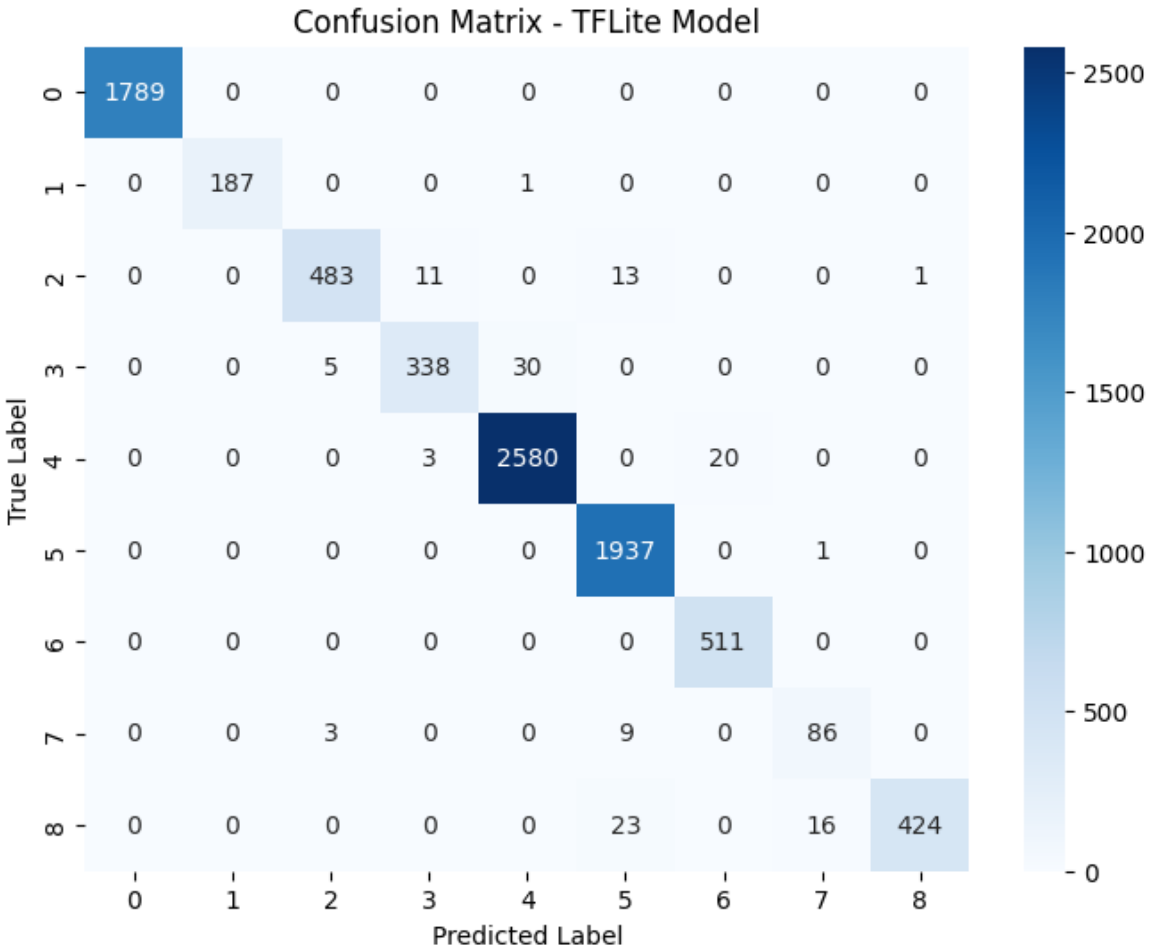


FIGURE 5.8 – Confusion matrix for quantized FCN model.

TimeCNN exhibited a modest increase in misclassification among certain temporally

overlapping classes after quantization (see Figure 5.9).

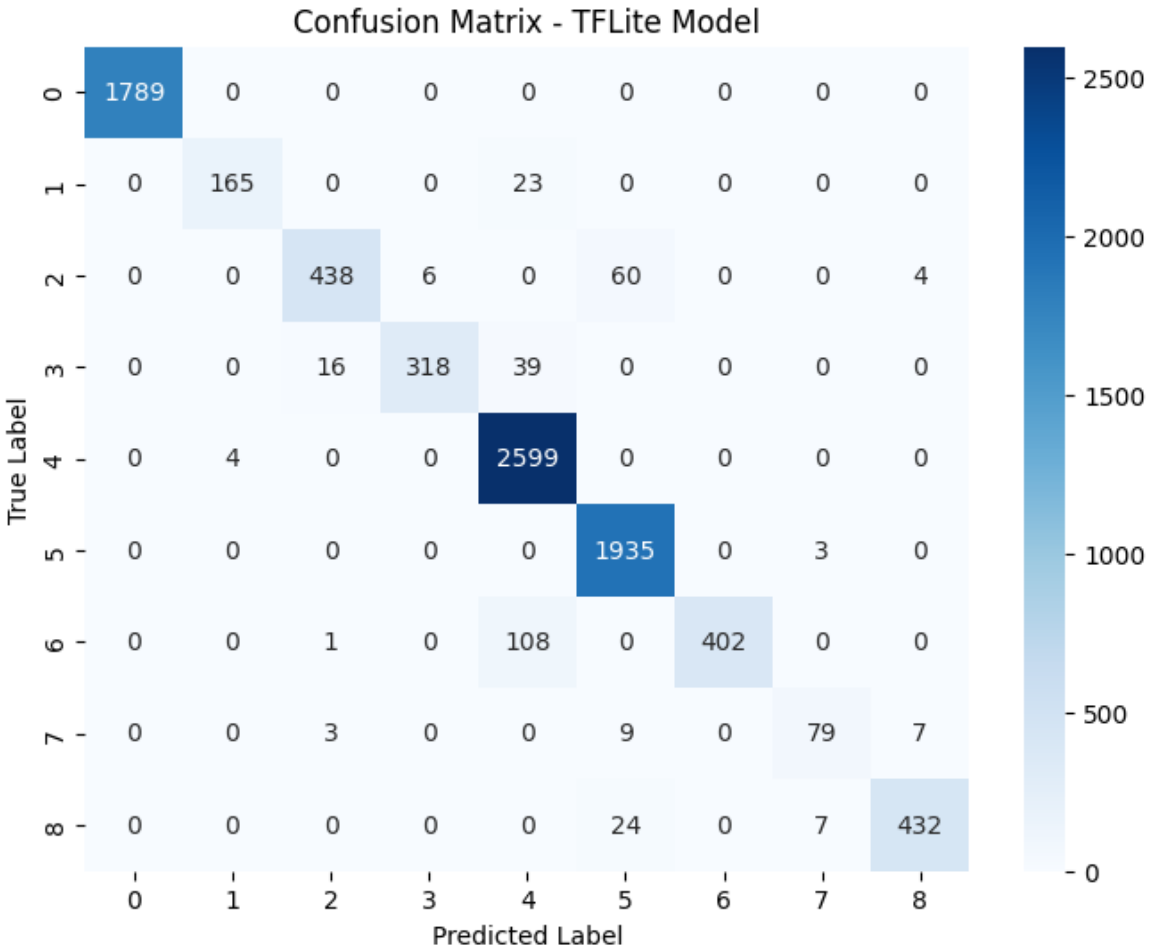


FIGURE 5.9 – Confusion matrix for quantized TimeCNN model.

SimpleRNN preserved its lightweight and effective classification capability even in quan-

tized form (see Figure 5.10).

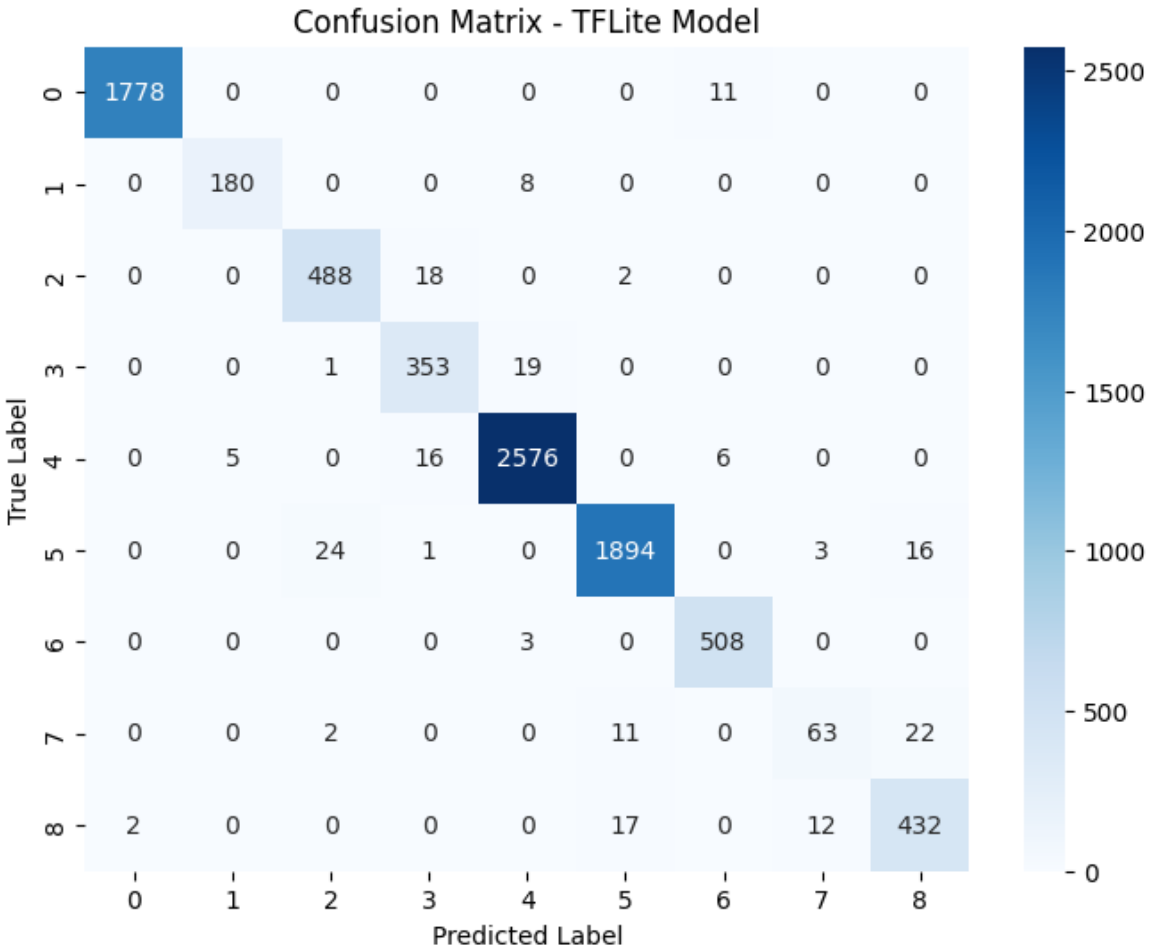


FIGURE 5.10 – Confusion matrix for quantized SimpleRNN model.

ResNet-ID shows more widespread confusion across classes, which supports the obser-

ved accuracy decline(see Figure 5.11).

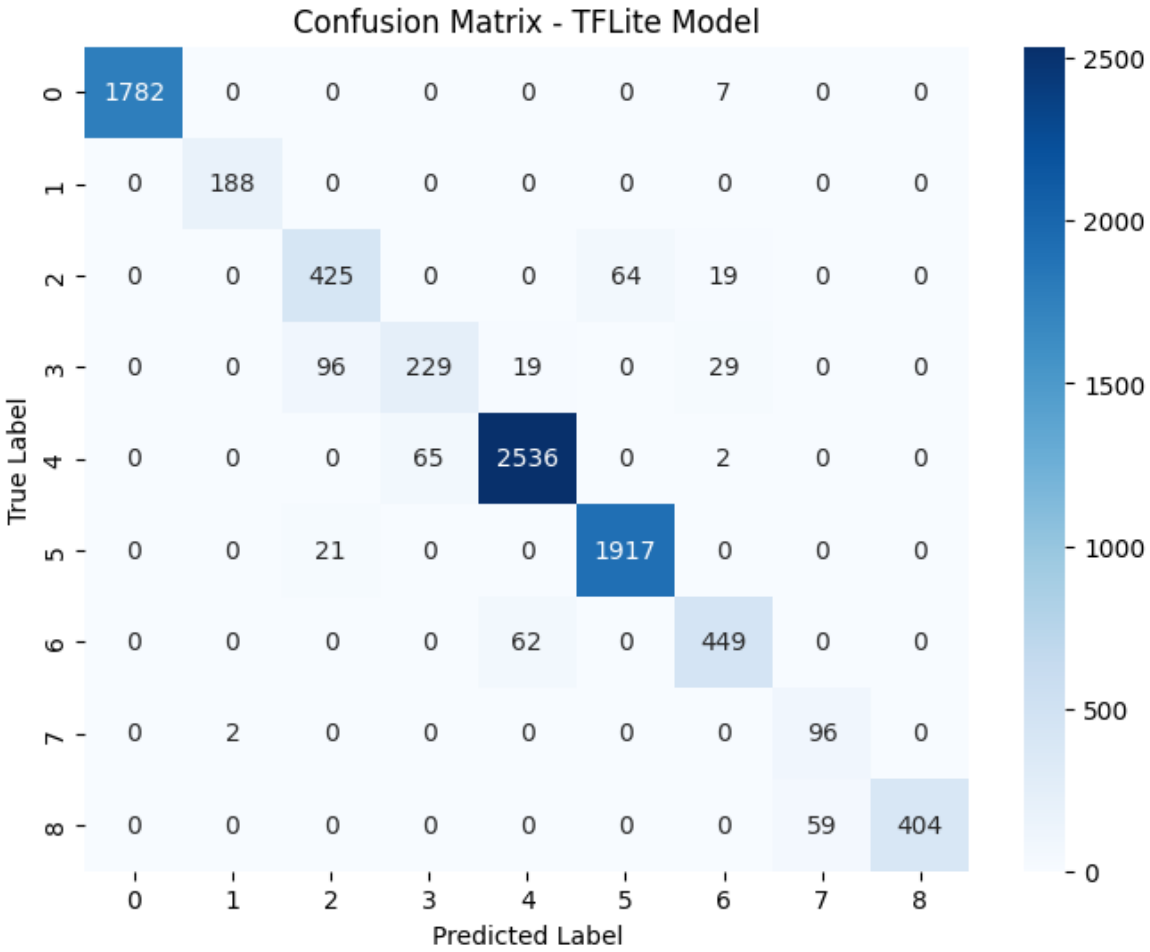


FIGURE 5.11 – Confusion matrix for quantized ResNet-ID model.

The TCN model's confusion matrix reveals increased misclassification in multiple classes, underlining the challenges faced by dilated architectures under quantization (see Fi-

gure 5.12).

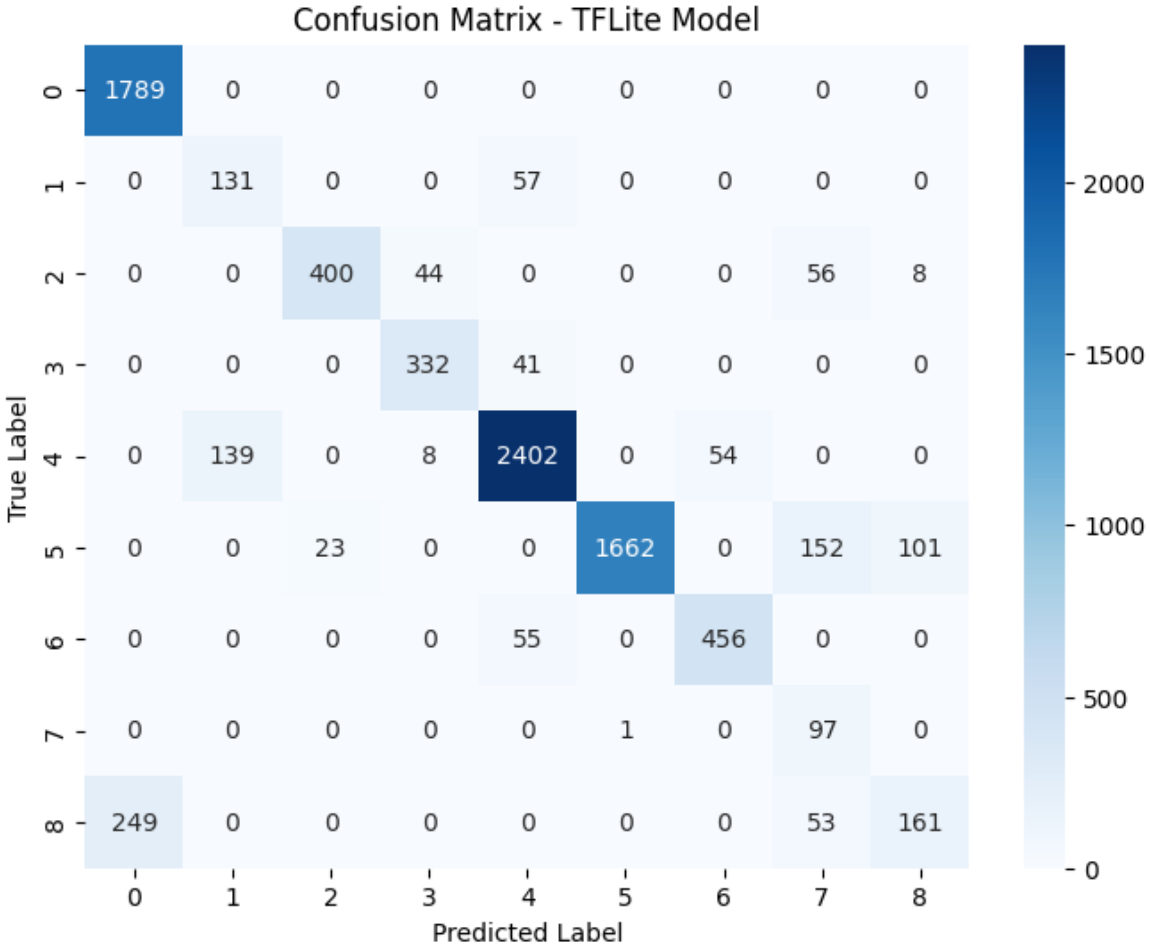


FIGURE 5.12 – Confusion matrix for quantized TCN model.

Overall, the results highlight that **SE-TSNet** and **FCN** offer the best post-quantization accuracy, while **SimpleRNN** and **TCN** demonstrate strong memory efficiency. **ResNet-ID** and **TimeCNN** provide a middle ground between accuracy and model size, although they may require further tuning for optimal edge performance.

The next section provides an empirical evaluation of these quantized models on embedded hardware, focusing on inference latency and memory utilization.

5.3 Performance on the Edge Device

The quantized models were deployed to an ESP32 microcontroller to evaluate their runtime performance and memory efficiency in a resource-constrained environment.

The ESP32 used in this study features a dual-core processor operating at 240 MHz, 355 KB of available heap memory, and 4 MB of flash storage, making it an ideal candidate for TinyML applications. Inference was performed using the TensorFlow Lite for Microcontrollers (TFLite Micro) runtime, which supports integer-only quantized models.

Two key performance indicators were considered :

— **Memory Usage :** The amount of SRAM consumed by the model during inference, including temporary buffer allocations.
— **Inference Time :** The latency required to process one input window and generate a classification output.

Table 5.3 summarizes the performance of each model on the ESP32 device.

TABLE 5.3 – Test Performance on the Edge Device

Model	Memory Usage (KB)	Inference Time (Sec)
FCN	20.65	0.53
TimeCNN	20.17	0.53
ResNet-ID	24.54	0.23
TCN	19.70	0.09
SE-TSNet	13.82	0.14
SimpleRNN	33.18	0.01

Memory Usage. Memory availability is one of the most critical limitations in embedded AI systems. The results indicate that **SE-TSNet** had the lowest memory footprint, consuming only 13.82 KB of RAM. This makes it particularly suitable for deployment on ultra-constrained microcontroller platforms. **TCN** also exhibited strong memory efficiency with 19.70 KB of usage, highlighting the advantage of its lightweight convolutional structure.

In contrast, **SimpleRNN** had the highest memory usage at 33.18 KB, primarily due to the recurrent nature of the architecture and the requirement to maintain hidden states across time steps. **ResNet-ID** consumed 24.54 KB, likely attributed to its use of residual connections. **FCN** and **TimeCNN** demonstrated similar memory footprints (approximately 20 KB), consistent with their architectural similarities.

Inference Speed and Real-Time Processing. Inference time is another crucial

factor for edge deployment, especially in scenarios demanding rapid decision-making. **SimpleRNN** achieved the fastest inference time at 0.01 seconds per sample, showing exceptional suitability for real-time use cases. **TCN** followed with 0.09 seconds, indicating that dilated convolutions can be efficiently executed on microcontrollers.

Among the convolutional models, **SE-TSNet** performed well with a latency of 0.14 seconds, suggesting that the added squeeze-and-excitation blocks did not introduce significant computational overhead. **ResNet-ID** required 0.23 seconds per sample, while both **FCN** and **TimeCNN** exhibited the slowest inference speed, taking 0.53 seconds per prediction.

These results emphasize that while **SimpleRNN** is optimal for ultra-low latency, **SE-TSNet** provides the best overall trade-off between speed, memory efficiency, and classification accuracy, confirming its potential for practical TinyML deployment.

6 CONCLUSION

6.1 Conclusion

This study presented an in-depth evaluation of multiple deep learning architectures for the classification of electrical appliance usage from time-series energy data, with a particular focus on their feasibility for deployment in memory- and power-constrained edge computing environments. By leveraging TinyML principles and deploying quantized models onto ESP32 microcontrollers, this research aimed to bridge the gap between high-performance machine learning and practical, real-world energy monitoring applications.

A total of six models—**FCN**, **TimeCNN**, **ResNet-ID**, **TCN**, **SE-TSNet**, and **SimpleRNN**—were designed, trained, and evaluated under both standard floating-point and quantized formats. The comprehensive assessment considered not only classical performance metrics such as accuracy, precision, recall, and F1-score, but also operational considerations including inference time and model memory footprint—factors that are essential in embedded systems engineering.

The experimental results consistently demonstrate that **post-training quantization** is a viable technique for reducing model size while maintaining classification performance within acceptable bounds. For instance, **SE-TSNet** achieved superior accuracy (99.32%) even in its quantized form, while also exhibiting the lowest memory consumption among all convolutional architectures tested. This highlights the effectiveness of incorporating *Squeeze-and-Excitation* mechanisms in improving channel-wise attention while keeping the model lightweight.

Similarly, **ResNet-ID**, known for its deep residual learning structure, also performed exceptionally well in terms of classification accuracy, although its quantized variant experienced a slight performance degradation. On the other hand, **SimpleRNN** demonstrated remarkable inference speed, completing predictions in just 0.01 seconds, although this came at the cost of higher RAM usage due to the sequential nature of RNNs and their unrolled time-steps.

FCN and **TimeCNN** preserved strong predictive capabilities but suffered from increased inference latency, which limits their application in scenarios requiring immediate feedback. **TCN**, utilizing dilated causal convolutions, delivered balanced performance across all metrics and emerged as a promising candidate for edge deployment, particularly in systems with moderate latency tolerance.

The real-time, on-device inference capabilities demonstrated through ESP32 deployment validate the central hypothesis of this research : *TinyML can effectively enable scalable, energy-efficient, and private appliance recognition systems in embedded contexts*. The removal of dependence on centralized cloud computation provides significant advantages, including lower bandwidth usage, improved data privacy, reduced latency, and enhanced system autonomy.

Despite these achievements, this study also revealed several challenges intrinsic to edge ML deployment. Trade-offs between model complexity, memory usage, and inference latency persist. The process of balancing these factors requires careful architectural choices, advanced optimization strategies, and hardware-specific adaptations. These challenges point to a rich landscape for continued exploration.

6.2 Future Work

While the results obtained from deploying quantized models on ESP32 microcontrollers are promising, there exist several opportunities for future research to enhance the robustness, scalability, and functionality of TinyML-based energy classification systems.

One compelling direction is the implementation of **federated learning** protocols. In such systems, multiple distributed edge devices collaboratively train a global model while keeping raw data local. This approach not only enhances user privacy but also reduces communication overhead and energy cost associated with data transfers. Furthermore, federated learning would allow the system to adapt dynamically to new devices or changing consumption patterns based on localized experiences, thereby improving model generalization across heterogeneous environments.

Another promising avenue is the exploration of **model compression techniques** such as **pruning**, **quantization-aware training (QAT)**, and **knowledge distillation**. Pruning can reduce the number of non-essential parameters, significantly shrinking

model size without substantial loss in accuracy. Knowledge distillation allows a lightweight model (student) to approximate the behavior of a more complex and accurate model (teacher), striking a practical balance between performance and computational demand. QAT, in contrast to post-training quantization, integrates quantization into the training process itself, which often results in better preserved accuracy for extremely low-bit models.

Additionally, **mixed-precision inference** and **hybrid quantization schemes** could be explored to optimize the numerical representation of different layers based on their sensitivity to quantization. For instance, using 8-bit integers for most layers and keeping sensitive layers in 16-bit floating point can yield improved accuracy with minimal impact on memory usage.

Expanding the hardware evaluation beyond ESP32 devices is another vital direction. Future studies could target microcontrollers with built-in **AI accelerators**, such as the *ARM Cortex-M55 with Ethos-U55 NPU*, or **low-power SoCs** designed for neural inference, like Google's Coral or the Kendryte K210. These platforms may allow for more complex architectures, faster inference, and multi-task learning while still adhering to embedded system constraints.

Furthermore, extending the system's functionality beyond classification is a natural progression. Integrating **anomaly detection mechanisms** can provide early warning for malfunctioning appliances, detect unusual consumption behaviors, or trigger maintenance protocols. This adds a predictive layer to the system, increasing its utility in industrial and residential energy management.

Another long-term vision is to embed this system into larger smart grid ecosystems. The classification outputs could inform **demand-side response strategies**, enable **automated load balancing**, or support **energy consumption forecasting**. This would position TinyML not only as a tool for local intelligence but also as a foundational element of future energy infrastructure.

Finally, from a software engineering standpoint, incorporating features such as **over-the-air (OTA) model updates**, **automated benchmarking pipelines**, and **on-device adaptive learning** could make the solution more robust, maintainable, and adaptable to evolving user needs and hardware capabilities.

Addressing these future directions will contribute significantly to the evolution of Ti-

nyML systems from proof-of-concept prototypes to production-grade, deployable technologies capable of transforming the landscape of real-time energy intelligence and embedded AI.

REFERENCES

Abeykoon, R., Senevirathna, L., Gunawardena, U. S. and Amarasekara, G. (2016). Real-time identification of electrical devices through non-intrusive load monitoring, Vol. 63, pp. 7066–7074.

Agency, I. E. (2024). Global power demand growth accelerates to 4.3% in 2024 ; renewables, gas lead supply gains.
URL: *https ://www.spglobal.com/commodity-insights/en/news-research/latest-news/energy-transition/032425-global-power-demand-growth-accelerates-to-43-in-2024-renewables-gas-lead-supply-gains-iea*

Andrade, L. P., Carvalho, L. M. and Nogueira, D. P. (2021). An unsupervised tinyml approach applied for pavement anomalies detection under the internet of intelligent vehicles, *IEEE Sensors Journal* **21**(22) : 24918–24926.

Bajrami, X., Dika, A. and Raufi, B. (2021). Mqtt protocol in iot systems : A review, *Journal of IoT and Emerging Technologies* **3**(4) : 15–23.

Barker, S., Mishra, A., Irwin, D., Cecchet, E. and Shenoy, P. (2012). Smart* : An open data set and tools for enabling research in sustainable homes, *Proceedings of the Workshop on Data Mining Applications in Sustainability (SustKDD), ACM SIGKDD* .

Chen, Z., Xiao, F., Guo, F. and Yan, J. (2023). Interpretable machine learning for building energy management : A state-of-the-art review, *Advances in Applied Energy* **9** : 100123.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L. and Muller, P.-A. (2019). Deep learning for time series classification : a review, *Data Mining and Knowledge Discovery* **33**(4) : 917–963.

Feng, C., Cui, M. and Dong, Y. (2020). Energy load disaggregation with deep learning : A time-series windowing approach, *Neural Computing and Applications* **32**(15) : 11593–11608.

Fischer, C. (2008). Feedback on household electricity consumption : a tool for saving energy ?, *Energy Efficiency* **1**(1) : 79–104.

Hart, G. W. (1992). Nonintrusive appliance load monitoring, *Proceedings of the IEEE* **80**(12) : 1870–1891.

Hu, J., Shen, L. and Sun, G. (2018). Squeeze-and-excitation networks, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Salt Lake City, USA, pp. 7132–7141.

Klemenjak, C. and Goldsborough, P. (2016). Non-intrusive load monitoring : A review and outlook, *in* H. C. Mayr and M. Pinzger (eds), *Lecture Notes in Informatics (LNI), Proceedings of the Informatik 2016 Conference, Klagenfurt, Austria*, Gesellschaft für Informatik, Bonn, pp. 2199–2205.

Kolter, J. Z. and Jaakkola, T. (2011). Approximate inference in additive factorial hmms with application to energy disaggregation, *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1472–1482.

Lane, N. D., Bhattacharya, S., Mathur, A., Georgiev, P., Forlivesi, C. and Kawsar, F. (2015). Deepx : A software accelerator for low-power deep learning inference on mobile devices, *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*.

Lane, R. O. (2023). Electrical device classification using deep learning, *Research paper*, QinetiQ, Great Malvern, UK.

Liu, Y., Wang, Y. and Ma, J. (2024). Non-intrusive load monitoring in smart grids : A comprehensive review, *Proceedings of the IEEE* .

Mughal, M. A., Mirza, S. R. and Shafiq, O. (2020). Hybrid machine learning approaches for appliance identification using low-frequency smart meter data, *IEEE Transactions on Smart Grid* **11**(2) : 1214–1223.

Mughal, U., Owais, S. M. and Asim, M. (2020). An iot deep learning-based home appliances management and classification system, *IEEE Access* **8** : 24341–24350.

Pelletier, C., Webb, G. I. and Petitjean, F. (2018). Temporal convolutional neural network for the classification of satellite image time series, *arXiv preprint arXiv :1811.10166* .

Razavi, R., Ghiassi-Farrokhfal, Y. and Palensky, P. (2018). Security implications of load disaggregation in smart grids, *IEEE Transactions on Industrial Informatics* **14**(6) : 2669–2679.

Ruzzelli, A. G., Nicolas, C., Schoofs, A. and O'Hare, G. M. P. (2010). Real-time recognition and profiling of appliances through a single electricity sensor, *International Workshop on Agent-Oriented Software Engineering*, pp. 1–10.

Sawada, A., Miyagawa, T., Ebihara, A. F., Yachida, S. and Hosoi, T. (2022). Convolutional neural networks for time-dependent classification of variable-length time series, *arXiv preprint arXiv :2207.03718* .

Smirnov, D. and Nguifo, E. M. (2018). Time series classification with recurrent neural networks, *Proceedings of the 3rd ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, pp. 1–9.

Solatidehkordi, Z., Vahidnia, H. and Sabouri, F. (2023). An iot deep learning-based home appliances management and classification system, *IEEE Internet of Things Journal* **10**(4) : 2378–2389.

Systems, E. (2022). Esp32 series datasheet v3.9, `https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf`. Accessed : May 2025.

Vohra, R., Parashar, S. and Kaur, M. (2023). Parquet as a high-performance storage format for large-scale data systems, *International Journal of Big Data Management* **8**(2) : 123–134. Available at `https://arxiv.org/pdf/2304.05028`.

Wang, Z., Yan, W. and Oates, T. (2017). Time series classification from scratch with deep neural networks : A strong baseline, *International Joint Conference on Neural Networks (IJCNN)*, pp. 1578–1585.

Zeifman, M. and Roth, K. (2011). Nonintrusive appliance load monitoring : Review and outlook, *IEEE Transactions on Consumer Electronics* **57**(1) : 76–84.

Zhang, Y., Zhang, X. and Zhu, Q. (2021). Improving nilm performance through statistical outlier detection and data preprocessing techniques, *Sensors* **21**(9) : 2946.

Zhao, B., Lu, H., Chen, S., Liu, J. and Wu, D. (2017). Convolutional neural networks for time series classification, *Journal of Intelligent & Fuzzy Systems* **33**(6) : 3545–3556.

Zhao, B., Zhang, W., Chen, X., Xu, S. and Li, X. (2018). Learning to recognize electrical appliances via machine learning : Performance evaluation and comparison, *International Conference on Artificial Intelligence and Big Data (ICAIBD)*.