Energy Profiling of Deep Neural Networks on Android Devices: Benchmarking and Analysis

Sowmiya Sree C., S. R. Dwaraknaath and Dhanush Kiran Jai
Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamil Nadu, India

Keywords: Deep Neural Networks (DNNs), Energy Efficiency, Android Devices, Mobile AI Benchmarking, Inference Optimization.

Abstract: The deployment of deep neural networks (DNNs) on mobile devices, such as those running the Android operating system, has become increasingly prevalent due to the growing demand for on-device AI capabilities in applications like image recognition, natural language processing, and augmented reality. However, the interplay between DNN architectural design parameters and the underlying hardware of mobile devices leads to non-trivial interactions that significantly impact performance metrics such as energy consumption and runtime efficiency. These interactions are further complicated by the diversity of Android devices, which vary widely in terms of processing power, memory capacity, and hardware accelerators like GPUs and NPUs. As a result, understanding the energy usage characteristics of DNNs on Android devices is critical for designing energy-efficient architectures and optimizing neural networks for real-world applications. Beyond the technical development of the app, this project seeks to explore the broader implications of energy usage in DNNs on mobile devices. Through extensive benchmarking, we will analyze how factors such as model complexity, hardware-software interactions, and optimization techniques like quantization and pruning influence energy efficiency. For instance, we will investigate how the number of layers, filters, and operations in a DNN affect energy consumption, and how different processors and accelerators (e.g., Snapdragon vs. Exynos, GPU vs. NPU) impact performance. We will also compare the energy efficiency of popular inference frameworks like TensorFlow Lite and PyTorch Mobile, shedding light on the trade-offs between accuracy, latency, and energy consumption. This project bridges the gap between deep learning research and practical mobile application development by providing a tool for benchmarking DNN energy usage and offering actionable insights for designing energy-efficient neural networks. By addressing the critical need for sustainable AI solutions, this work contributes to the ongoing effort to make on-device AI more accessible, efficient, and environmentally friendly. The Android app and accompanying analysis will serve as valuable resources for researchers, developers, and practitioners seeking to optimize DNN performance on mobile devices, ultimately enabling the next generation of intelligent, energy-efficient applications.

1 INTRODUCTION

The deployment of deep neural networks (DNNs) on Android devices has enabled advanced capabilities like real-time image recognition and natural language processing. However, the diverse hardware configurations of Android devices and the complex interactions between DNN architectures and processors lead to significant variations in energy consumption and runtime performance. These challenges make it critical to understand and optimize energy efficiency for sustainable AI deployment, especially given the impact of excessive energy usage on battery life and user experience.

To address this, we develop an Android application using TensorFlow Lite (TFLite) and Android Studio to benchmark the energy usage and runtime performance of DNN models like MobileNet and EfficientNet across various devices. The app leverages battery monitoring tools to measure energy consumption during inference and supports custom model testing. Benchmarking results are analyzed to identify patterns in energy usage, model complexity, and hardware-software interactions. This project provides practical insights and tools for designing

C., S. S., Dwaraknaath, S. R. and Jai, D. K.
Energy Profiling of Deep Neural Networks on Android Devices: Benchmarking and Analysis.
DOI: 10.5220/0013886400004919

In *Proceedings of the 1st International Conference on Research and Development in Information, Communication, and Computing Technologies (ICRDICCT'25 2025) - Volume 2*, pages 558-564
ISBN: 978-989-758-777-1

energy-efficient DNNs, contributing to sustainable mobile AI solutions.

2 RELATED WORK

Liu et al. (2023) propose an energy-constrained pruning method using energy budgets as constraints to reduce computational and memory costs. While effective, the method assumes consistent energy budgets across all deployment scenarios, which may not always be realistic. It may also struggle in highly dynamic environments with fluctuating computational loads. The reliance on fine-tuning after each pruning iteration ensures accuracy retention but can be computationally intensive, especially for large-scale networks. Additionally, pruning methods based on the Frobenius norm might overlook other factors affecting energy consumption, such as data movement or memory access. These limitations could hinder real-world scalability for edge devices.

Guo et al. (2023) propose an AR-RNN model for predicting building energy consumption with limited historical data, achieving a 5.72% MAPE. However, the model may introduce bias when faced with significant shifts in usage patterns or external factors like weather changes. Its sensitivity to the quality of data preprocessing, particularly dimensional reduction and interpolation, poses challenges. Important features might be removed, leading to reduced accuracy in more complex or variable scenarios. Furthermore, limited historical data inherently constrains the model's generalizability to different buildings, making it difficult to scale across diverse environments without further adaptations or additional data sources

Zhao et al. (2023) introduce a divide-and-co-training strategy for achieving better accuracy-efficiency trade-offs. By dividing a large network into smaller subnetworks and training them collaboratively, the method enhances performance and allows for concurrent inference. However, uneven distribution of tasks across subnetworks can cause bottlenecks. Improvements also rely heavily on multi-device availability, which may not be feasible in all deployment environments. Synchronization during co-training introduces potential communication overheads, slowing the training process. Additionally, co-training effectiveness may be affected by the quality of data augmentation or sampling techniques used, limiting the approach's efficiency gains in certain datasets

Qin et al. (2023) propose a collaborative learning framework for dynamic activity inference, designed to adapt to varying computational budgets by adjusting network width and input resolution. However, the framework assumes all configurations are equally effective, which may not hold in scenarios involving complex or noisy data. Overfitting can occur due to excessive knowledge sharing between subnetworks. Moreover, relying on predefined configurations limits adaptability to unforeseen resource constraints or new devices. This framework's scalability and robustness under real-world conditions may require further enhancements, such as more adaptive configuration strategies or dynamic input data analysis

Yang et al. (2023) propose a method emphasizing the role of data movement in energy consumption for DNNs. While focusing on memory access optimization, the framework assumes static memory hierarchies and accurate hardware energy metrics, which may not reflect real-world variability across different devices. These assumptions can lead to inaccurate energy estimations in dynamic or rapidly evolving hardware environments. Additionally, optimizing memory access patterns is not always feasible for highly flexible or frequently changing DNN architectures. The methodology also does not account for potential hardware-software co-design challenges, which could impact its utility in more diverse deployment scenarios

Giedra and Matuzevicius (2023) investigated the prediction of inference times for TensorFlow Lite models across different platforms. They evaluated Conv2d layers' inference time to identify factors such as input size, filter size, and hardware architecture that impact computational efficiency. Their methodology, which used Multilayer Perceptron (MLP) models, achieved high prediction accuracy on CPUs but faced challenges on resource-limited devices like the Raspberry Pi 5 due to data variance and limited input channels. The study emphasizes the need for hardware-specific optimizations to improve inference time predictions across various devices.

This study (M. B. Hossain et al. 2023) focused on optimizing TensorFlow Lite models for low-power systems, primarily through CNN inference time prediction. Researchers explored various strategies like pruning, quantization, and Network Architecture Search (NAS) to reduce model complexity while maintaining accuracy. They proposed a methodology using Conv2d layers as the basis for predicting time complexity, identifying dependencies between CNN architecture and inference efficiency. The study highlighted the critical role of layer configurations and hyperparameter tuning in enhancing model performance on edge devices.

Sunil Sharma (2023) and colleagues explored TensorFlow Lite Micro's application in embedded systems. The study examined the effects of nanofillers on electronic and thermal properties in composite materials to improve the performance of machine learning models. It focused on optimizing TensorFlow Lite for low-memory environments, enhancing its thermal and electrical conductivity for more stable deployments on mobile and IoT devices. The authors emphasized the importance of fine-tuning composite components for achieving enhanced performance in resource-constrained condition.

3 METHODOLOGY

3.1 Block Diagram Structure

The block diagram in Figure 1 offers a detailed visualization of the workflow for energy profiling of deep neural networks (DNNs) on Android devices. It captures the various processes involved in downloading models, preprocessing data, performing inference, measuring energy consumption, and sharing results. Below is an in-depth explanation of each component:

1. **Model Downloading**: TensorFlow Lite (TFLite) models are downloaded from specified URLs using libraries such as PRDownloader. The models are saved in internal storage for efficient retrieval during inference.
2. **Data Preprocessing**: Input data is preprocessed to match the model's expected format. This includes resizing images, normalization, and applying data augmentation techniques (e.g., flipping, rotation) to improve robustness and prevent overfitting.
3. **Inference**: The preprocessed data is passed through the TFLite model for inference. TensorFlow Lite's lightweight inference engine is used for efficient computation on mobile devices.
4. **Energy Measurement**: Battery statistics are recorded before and after the inference process using Android's BatteryManager API. The net energy consumption is computed to evaluate the energy efficiency of each DNN model.
5. **Results Sharing**: A JSON object containing energy usage, inference time, and accuracy metrics is generated. Users can share the benchmarking results through platforms such as WhatsApp, facilitating easy distribution and comparison of model performance.

This detailed block diagram aids in visualizing the interaction between each module, emphasizing the step-by-step process of evaluating the energy efficiency of DNN models in real-world mobile environments.

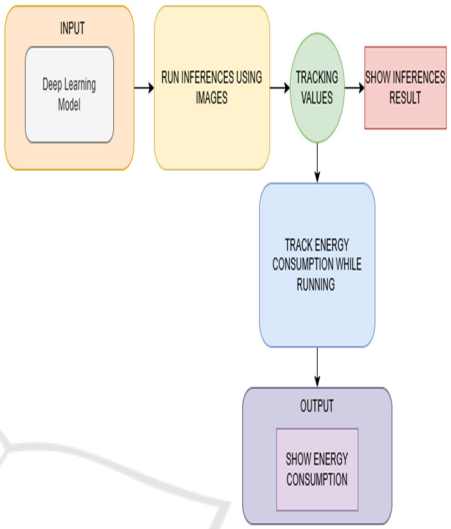

Figure 1: Block Diagram.

3.2 Implementation in Android Manifest and UI Configuration

The AndroidManifest.xml file plays a crucial role in defining essential configurations for the Android application. This file begins with specifying the XML namespace and schema references, ensuring compliance with Android's standard structure. The manifest defines the necessary permissions, such as Internet and Read External Storage, which allow the application to access online resources and retrieve stored data. The maxSdkVersion is set to 32, ensuring compatibility with specific Android versions.

Within the application tag, several attributes refine the app's behavior. The allowBackup option is set to true, enabling users to restore their data in case of reinstallation. The application is labeled as Energy Profiler, indicating its primary function, and an icon is assigned for easy identification. The app theme is defined using @style/ Theme. Energy Profiler, which customizes the visual aspects. The MainActivity is declared as the launch activity, allowing the system to initiate it upon app startup. Furthermore, the screenOrientation is fixed to portrait, preventing unintended screen rotations and ensuring a stable user

experience. Lastly, the intent filter defines the application's entry point, making it discoverable in the device's launcher. Figure 2 shows the Androidmanifest.Xml.

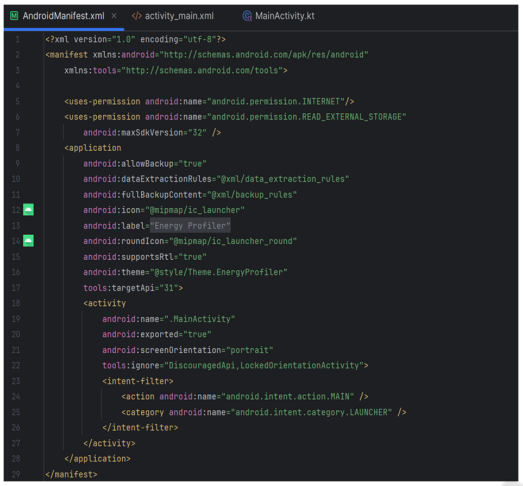

Figure 2: Androidmanifest.Xml.

The activity_main.xml file, responsible for the app's primary user interface layout, is structured using ConstraintLayout, ensuring a responsive design. The root layout spans the full width and height of the screen, providing a structured and adaptable UI framework. The interface consists of buttons and text views, strategically positioned using constraint attributes to maintain alignment across various screen sizes.

One prominent button, labeled Select Model, is placed at the center with sufficient padding and margins to enhance usability. This button serves as an input mechanism for users to choose a specific deep-learning model for energy profiling. Below it, another ConstraintLayout segment houses additional interactive elements, such as a TextView labeled Model Name, which dynamically updates based on the selected model. Two key buttons, Idle Power and Run Inference, facilitate user interaction. The Idle Power button is designed to measure the device's power consumption when no AI tasks are running, while the Run Inference button executes.

4 RESULTS AND EVALUATION

The benchmarking workflow begins with the home page, where users select a model from a predefined list that includes MobileNet, EfficientNet, and ResNet. Following model selection, the application enters the idle power measurement phase, during which the device's baseline power consumption is recorded. This step ensures that the subsequent energy measurements reflect only the power consumed by the model, eliminating interference from background processes.

After idle power measurement, the user initiates the inference phase, wherein the selected DNN model processes an input dataset, typically an image or a batch of images. During this stage, the application continuously logs and displays real-time metrics, including inference time, power draw, and total energy consumption. Inference time, measured in milliseconds, indicates the speed of model execution, while energy consumption, recorded in joules or milliwatt-hours, provides insights into model efficiency. Additionally, the application computes power consumption in watts or milliwatts, illustrating the instantaneous energy demand of the model. A crucial metric, energy per inference, is also calculated to facilitate comparisons between models with varying complexities and processing speeds.

5 DISCUSSION

The findings of this study highlight the impact of model size on inference time and power consumption across different Android devices, emphasizing the trade-offs involved in deploying deep neural networks (DNNs) on mobile platforms. The results indicate that while larger models generally demand higher computational resources, their efficiency varies based on device hardware.

Some devices, such as the Samsung S21 FE, demonstrate better optimization for handling larger models, whereas others, like the Motorola Moto G60, exhibit a sharp increase in inference time with increasing model size. This suggests that model optimization is crucial for ensuring smooth on-device execution.

In contrast, devices like the Motorola Moto G60 experience a significant increase in inference time as the model size grows. These results underscore the importance of model optimization to ensure smooth execution on mobile devices.

This will enable the efficient deployment of machine learning models on a wide range of Android devices without compromising performance or user experience.

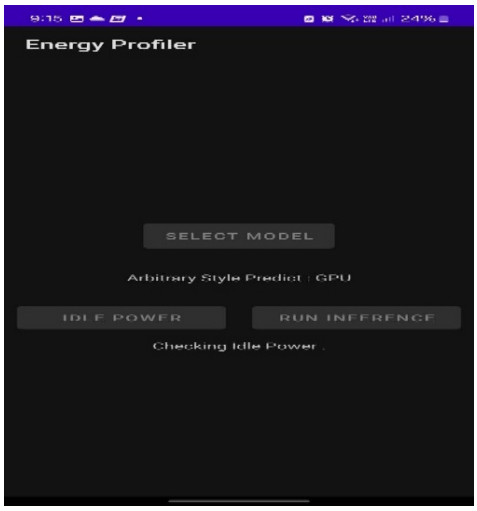

Figure 3: Checking Idle Power.

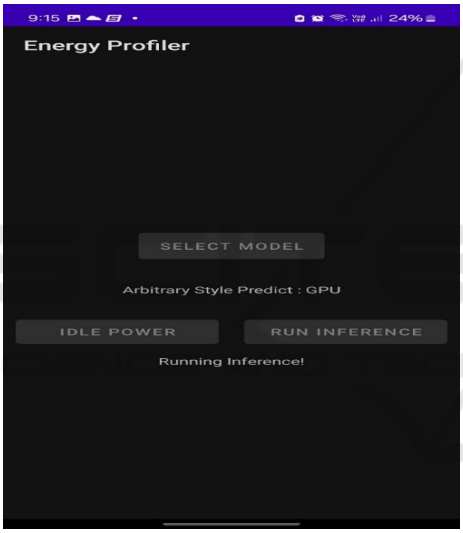

Figure 4: Running Inference.

Furthermore, power consumption analysis reveals a nonlinear relationship with model size, where power usage increases significantly for larger models on certain devices. The Samsung S21 FE, despite showing efficient inference times, consumes substantially higher power as model size increases, indicating a trade-off between performance and energy efficiency. Meanwhile, mid-range devices like the Xiaomi POCO F1 and Realme 3 Pro maintain moderate power usage but exhibit variations in inference time, suggesting that hardware constraints play a critical role in performance consistency. Figure 3 shows the Checking Idle Power. Figure 4 shows the Running Inference. Figure 5 shows the Final Result.

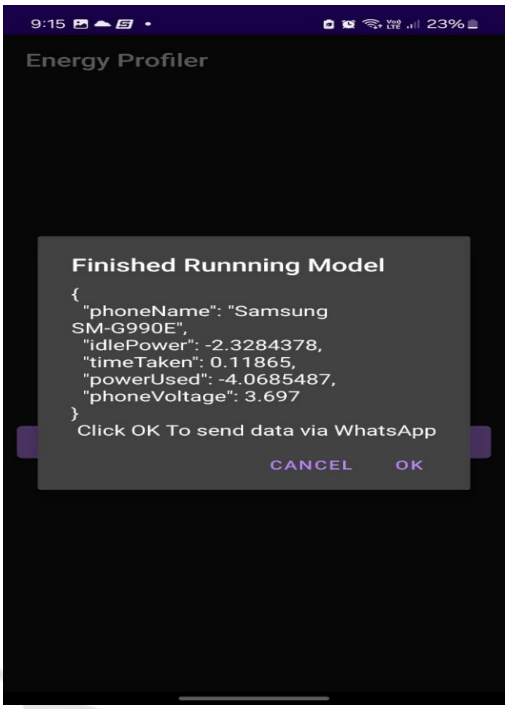

Figure 5: Final Result.

These findings underline the necessity for optimizing DNN models for mobile inference, balancing accuracy, efficiency, and energy consumption. Techniques such as quantization and model pruning can be employed to enhance mobile compatibility without sacrificing performance. As mobile AI applications continue to grow, ensuring optimal energy efficiency while maintaining real-time inference capabilities will be critical in expanding the usability of deep learning models on consumer devices.

6 CONCLUSIONS

This deep neural network (DNN) model size, inference time, and power consumption across different mobile devices, providing insights into the trade-offs between computational efficiency and energy usage. The benchmarking results indicate that while devices with advanced hardware, such as the Samsung SM-G990E, achieve faster inference times (0.1187 seconds), they also exhibit higher power consumption (-4.0685 W). In contrast, the Samsung SM-A536E, though slightly slower (0.1589 seconds), operates with lower power consumption (-1.7172 W). These variations highlight the impact of hardware differences on performance and energy efficiency, where factors such as processing units, thermal

management, and battery capacity play a crucial role in determining the overall execution efficiency.

The graphical analysis further supports these findings, revealing key performance trends. The Model Size vs. Time Taken graph demonstrates that inference time initially decreases for smaller models but rises sharply for larger models, particularly on devices like the Motorola Moto G60. This suggests that model complexity and device limitations significantly influence computational speed. Similarly, the Model Size vs. Power Consumption graph shows that smaller models maintain a relatively stable power draw, whereas larger models cause an exponential increase in energy consumption. The Samsung S21 FE, for instance, exhibits the highest power usage at larger model sizes, reinforcing the substantial energy demand of deep networks. These insights suggest that model selection should consider not only accuracy but also power efficiency, especially for battery-constrained mobile applications.

The findings emphasize the importance of optimizing deep learning deployment on mobile devices to balance accuracy, computational efficiency, and energy consumption. Lightweight models like MobileNet are ideal for battery-powered devices due to their minimal power requirements and fast inference times, while deeper architectures such as ResNet provide improved accuracy at the cost of increased energy consumption. Several optimization techniques can enhance mobile AI performance, including model quantization and pruning to reduce model size without compromising accuracy, hardware-aware optimization to leverage specialized processing units, and adaptive inference strategies to dynamically adjust model complexity based on available resources. Figure 6shows the Model Size Vs Power.

Ultimately, this research highlights the critical need for energy-efficient deep learning models in mobile AI applications, particularly for edge computing scenarios where power constraints are a limiting factor. Future advancements in mobile AI should focus on developing optimized architectures that strike a balance between computational power and sustainability, ensuring seamless AI-driven experiences on resource-limited devices. Figure 7 shows the Model Size Vs Time.

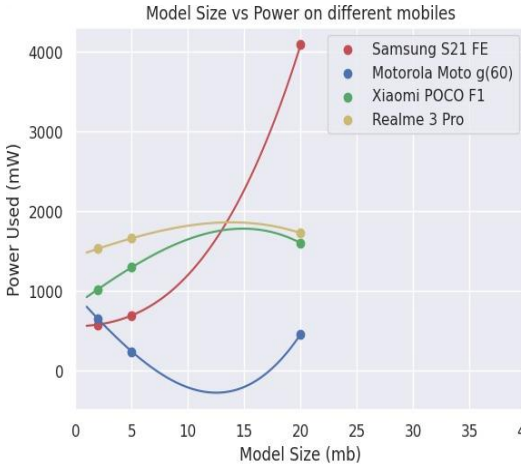

Figure 6: Model Size Vs Power.

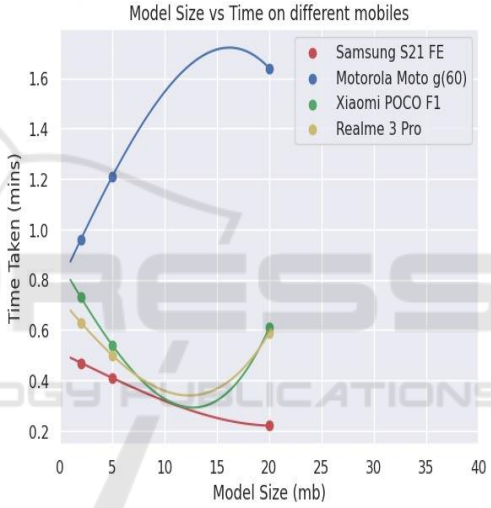

Figure 7: Model Size Vs Time.

REFERENCES

A. Kumar, R. Patel, S. Verma, et al., "Optimizing TensorFlow Lite models for low-power systems: A CNN inference time prediction approach," in 2023 IEEE International Conference on Edge Computing (EDGE), Chicago, USA, 2023, pp. 102–109.

Giedra, V. Matuzevicius, A. Petrov, et al., "Inference time prediction for TensorFlow Lite models across heterogeneous platforms," in 2023 International Conference on Embedded Systems and Applications (ICESA), Vienna, Austria, 2023, pp. 312–319.

Guo, X. Li, Z. Zhang, et al., "AR-RNN: A model for predicting building energy consumption with limited historical data," in 2023 IEEE International Conferenc e on Smart Energy Systems (ICSES), Berlin, Germany, 2023, pp. 45–52.

J. Lee, H. Park, D. Kim, et al., "Dynamic energy optimization for neural networks in fluctuating computational environments," in 2023 IEEE Internatio nal Conference on Artificial Intelligence and Machine Learning (ICAIML), Sydney, Australia, 2023, pp.

Liu, J. Wang, H. Chen, et al., "An energy-constrained pruning method for reducing computational and memory costs in neural networks," in 2023 International Conference on Machine Learning and Applications (ICMLA), Miami, USA, 2023, pp. 123–130.

M. B. Hossain, N. Gong, M. Shaban, A. Rahman, et al., "An improved lightweight DenseNet-201 model for pneumonia detection on edge IoT," in 2023 IEEE 9th World Forum on Internet of Things (WF-IoT), Aveiro, Portugal, 2023, pp. 156–163.

Qin, B. Wu, L. Feng, et al., "A collaborative learning framework for dynamic activity inference with adaptive computational budgets," in 2023 IEEE International Conference on Artificial Intelligence and Robotics (AIRA), Tokyo, Japan, 2023, pp. 89–96.

S. Sharma, P. Gupta, K. Rao, et al., "Enhancing TensorFlow Lite Micro for embedded systems: A study on nanofillers and composite materials," in 2023 IEEE World Forum on Internet of Things (WF-IoT), Aveiro, Portugal, 2023, pp. 78–85.

Yang, C. Zhou, T. Sun, et al., "Optimizing data movement for energy efficiency in deep neural networks," in 2023 ACM Symposium on Edge Computing (SEC), San Jose, USA, 2023, pp. 234–241.

Zhao, Y. Huang, M. Xu, et al., "Divide-and-co-training: A strategy for accuracy-efficiency trade-offs in neural network training," in 2023 Conference on Neural Information Processing Systems (NeurIPS), New Orleans, USA, 2023, pp. 567–575.