1. Introduction and Topic Statement

The advent of Artificial Intelligence has opened up a plethora of new opportunities. One sector that has been looked at with increased scrutiny and uncertainty is the Energy Sector. As the need for GPUs and other AI-specific hardware increases, new energy plants, datacenters and technology hubs are being built at record paces as corporations rush to provide consumers with the best possible models.

At what point is it enough?

TinyML offers a unique solution: deploying AI models on low-power microcontrollers. These MCUs run at not even a fraction of the energy that CPUs or GPUs run on. Of course, most models have to be modified, through compression, pruning and/or quantization and are run inference-only. This new development is called Edge AI and the upside is tremendous as further research strongly indicates the future of IoT is reliant on the success of TinyML.

The goal of this project is to not only discuss new methods that have been studied, but to implement some of them as well.

2. Technologies and Techniques

I have access to an **STM32F767ZI** Nucleo MCU. I also have 2 additional extension boards for capturing both audio input/output for more advanced models.

This MCU is known for its high performance and is known as one of the more advanced MCUs available on the consumer market. It runs on an ARM Architecture, but most importantly, it offers a Floating Point Unit (FPU) with single precision. This enables the development of small Neural Networks to be deployed on-device.

For this project, I will also be using **TensorLite Micro (TLFM), Edge Impulse and STM32Cube.AI.**

Techniques include:

- Model Quantization (Int8, Float16)
- Pruning and Weight Sparsity
- Feature Extraction (MFCC)
- Performance Benchmarking (RAM usage, latency, Flash Usage)
- Memory Optimization for static arrays
- Energy Profiling using on-board power measurements

3. Data Sources

   Sine Wave Synthetic Dataset. This is traditionally known as the "Hello World" of TinyML examples to validate model correctness and deployment pipelines. Generated programmatically, so no external dataset is required. This can be any ML model we have studied in class.

   ECG5000 Dataset: Classification dataset for a simple 1D CNN on-device. It comprises around 5000 mini-samples that will be quantized later.

   Google Speech Commands Dataset (Subset): Used in more advanced systems to spot audio commands such as yes, no, up or down. This will be done with either a MFCC approach or a Spectogram approach.

   Because of the limited memory and compute resources of microcontrollers, some of these datasets might end up slightly altered or replaced with other datasets if it is found to not be fully feasible. This will be discussed in the report later.

4. Deliverables

   - Technical Report containing an overview of each modern TinyML model explored, comparisons between TFLM, Cube.AI and Edge Impulse deployment workflows, and the overall winner based on benchmarking and ease-of-use.

   - Working TinyML models. At least 2 fully deployed models on the STM32 using one or more of the datasets above. These models will be exported in a common format, with both the Python code from training the model as well as the C code used to run the inference. *(STM32 uses the C Programming Language, so some conversion will be necessary)*

   - Benchmarking Results, including execution time per inference, RAM and Flash footprint and a comparison of both modified vs. unmodified models. "Modified" refers to applying any of the techniques discussed above.

The overall goal of this project is to begin exploring TinyML by building simple, functional models on a small microcontroller and beginning a discussion on the feasibility of future systems.