User's Guide

Out-of-the-Box Edge AI Demo on TI MCUs With Hardware Acceleration

ABSTRACT

Edge AI brings intelligence directly to embedded devices, enabling low-latency, secure, and reliable decision-making without reliance on cloud connectivity. In industrial applications, this translates to faster response times, improved data privacy, and greater system autonomy. This guide introduces Texas Instruments' new Edge AI hardware and tools, and goes through an out-of-the-box demo of a smart signal classifier. The demo showcases how our tools can be used to quickly build a simple application that classifies between sine, square, and sawtooth waveforms.

Table of Contents

List of Figures

Trademarks

C2000™ is a trademark of Texas Instruments.
All trademarks are the property of their respective owners.

1 Introduction

Texas Instruments offers a broad portfolio of MCUs and processors, ranging from real-time microcontrollers to high-performance application processors and mmWave sensing devices. Whether you're building low-power edge nodes or advanced AI-enabled systems, TI provides scalable solutions to meet a wide range of design needs.

To explore our full Edge AI portfolio and discover how TI enables intelligence at the edge—from development tools to deployment—visit our Edge AI Technology page, where you can find the complete story of Edge AI at TI.

This document focuses on our Edge AI Microcontrollers and explores a complete out-of-the-box demo based on those devices. MCUs often fall short when executing complex AI models, especially under constrained power and latency budgets. To meet the demanding performance and efficiency requirements of industrial use cases, dedicated hardware acceleration for machine learning (ML) workloads is essential. TI's latest Edge AI hardware platforms are equipped with specialized accelerators optimized for running neural networks efficiently, reducing inference times and system power consumption.

Complementing the hardware, TI introduces Edge AI Studio—a fully integrated, graphical development environment that streamlines the entire ML pipeline. From data collection and labeling, to model training, optimization, and deployment onto supported TI edge devices, Edge AI Studio simplifies each step through a no-code, GUI-based interface. This empowers both AI experts and embedded developers to build robust edge intelligence solutions quickly and efficiently.

2 TMS320F28P55x

The TMS320F28P550SJ is the first in the C2000™ family to integrate a hardware Neural Processing Unit (NPU), positioning it as cutting-edge platform for Edge AI in real-time control systems. Built on the robust C28x DSP core, this MCU blends deterministic control, advanced signal processing, and on-chip machine learning inference, enabling smarter, more autonomous embedded applications in power, motion, and industrial systems. The integrated NPU is designed to accelerate inference of quantized neural networks, dramatically reducing CPU overhead and enabling real-time AI use cases such as predictive maintenance, motor fault detection, sensor classification, and fault detection. While the TMS320F28P550SJ includes a dedicated NPU for acceleration, it is important to note that all Edge AI models can still be compiled and run entirely in software on the C2000 MCU, without requiring any dedicated NPU hardware.

The following resources are available for customers to start evaluating arc fault detection, system fault and motor fault detection:

- Machine learning Arc Fault Detection user guide
- Orderable hardware for DC arc detection in solar applications based on artificial intelligence (AI)
- Optimizing system fault detection in real-time control systems with edge AI-enabled MCUs
- Motor Fault Detection using Embedded AI models

3 Edge AI Studio

Edge AI Studio is a collection of graphical and command line tools designed to accelerate edge AI development on TI processors and microcontrollers. Whether developing a proof of concept using a model from the TI Model Zoo or leveraging your own model, Edge AI Studio provides the tooling you need:

- Model Composer is a fully integrated solution for collecting and annotating data, training and compiling models for deployment on a live development platform. Model Composer hosts a variety of example solutions complete with demonstration data sets to enable testing the toolchain without importing any of your own data. Model composer also supports Bring-Your-Own-Data (BYOD), enabling the re-training of models from the TI Model Zoo with custom data to improve accuracy and performance. Model Composer is available as a cloud-based application for vision-based tasks and as both a cloud and desktop application for real-time control tasks.
- Model Analyzer is a free online service for vision applications that allows for the evaluation of accelerated deep learning inference on remotely accessed development boards. With Model analyzer it only takes minutes to login, deploy a model and obtain a variety of performance benchmarks.

- Model Maker is a command-line end-to-end model development tool. Model Maker includes features for dataset handling, model training and compilation. Expert users can use these tools to leverage their own models.

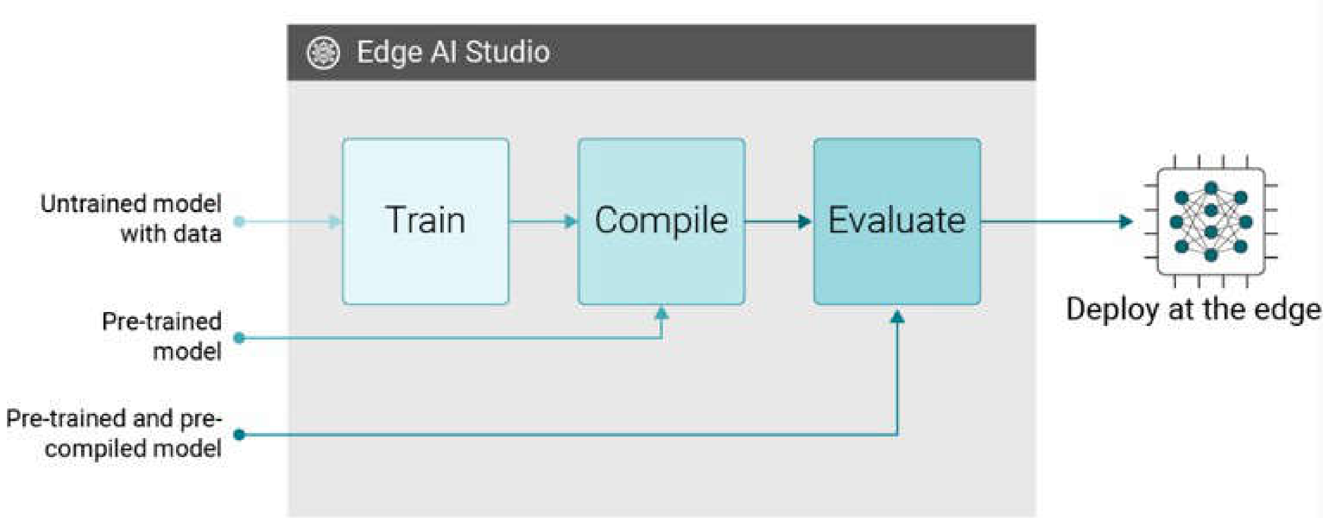

Figure 3-1. Edge AI Studio

4 Out-of-the-Box Demo (Smart Signal Classifier)

This demonstration is meant to be a simple introduction into TI's edge AI toolchain and products. Our software is meant as a starting place and engineering tool not as production ready examples. This example is a generic time series “Hello World” type of demo that is meant to introduce the workflow of data collection, training, compilation, and execution of an AI model. The entire workflow of this demo was created using Model Composer, a GUI-based tool that is part of the Edge AI Studio toolset.

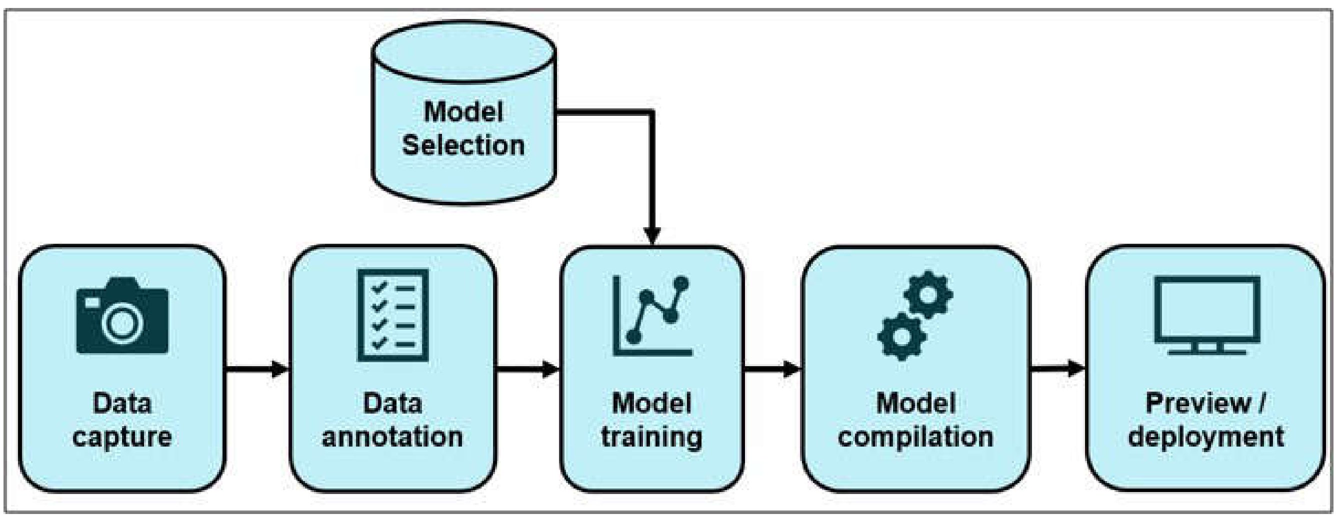

Figure 4-1. Model Composer Workflow

4.1 Dataset

This demo includes a prepared example dataset of three different periodic signals at varying amplitudes and noise levels: sine wave, square wave, and sawtooth.

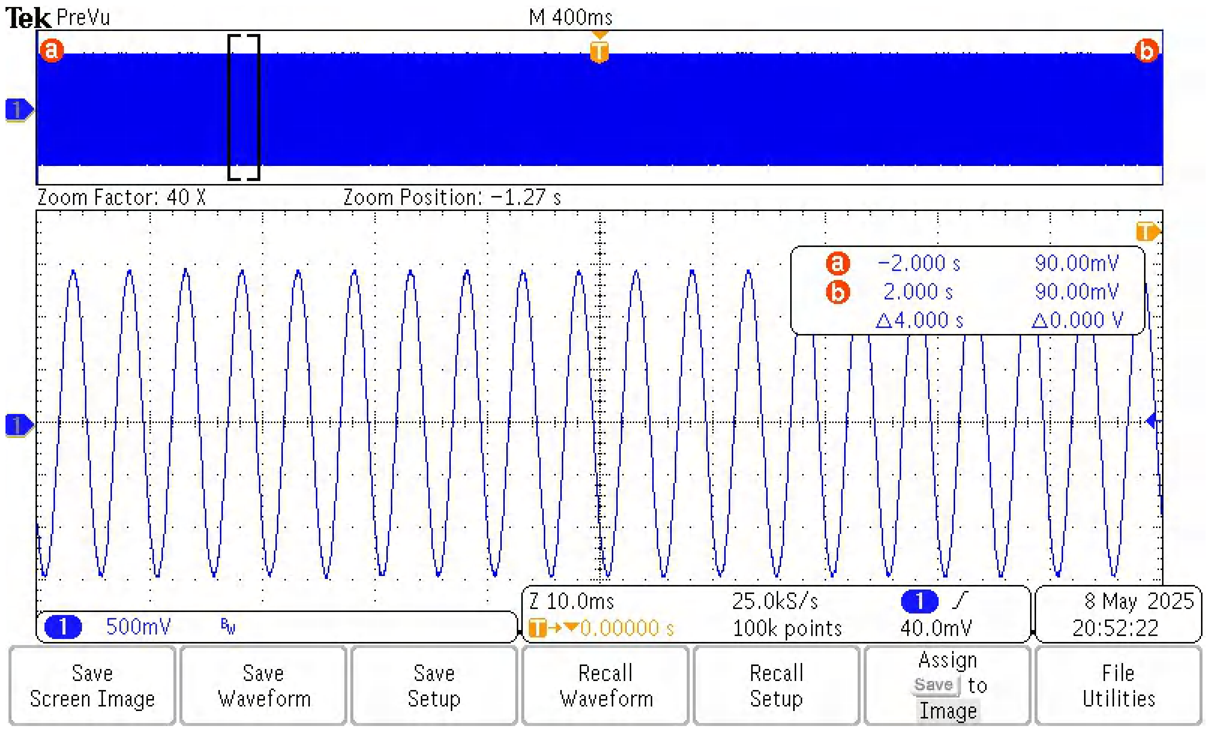

Figure 4-2. Sine Wave

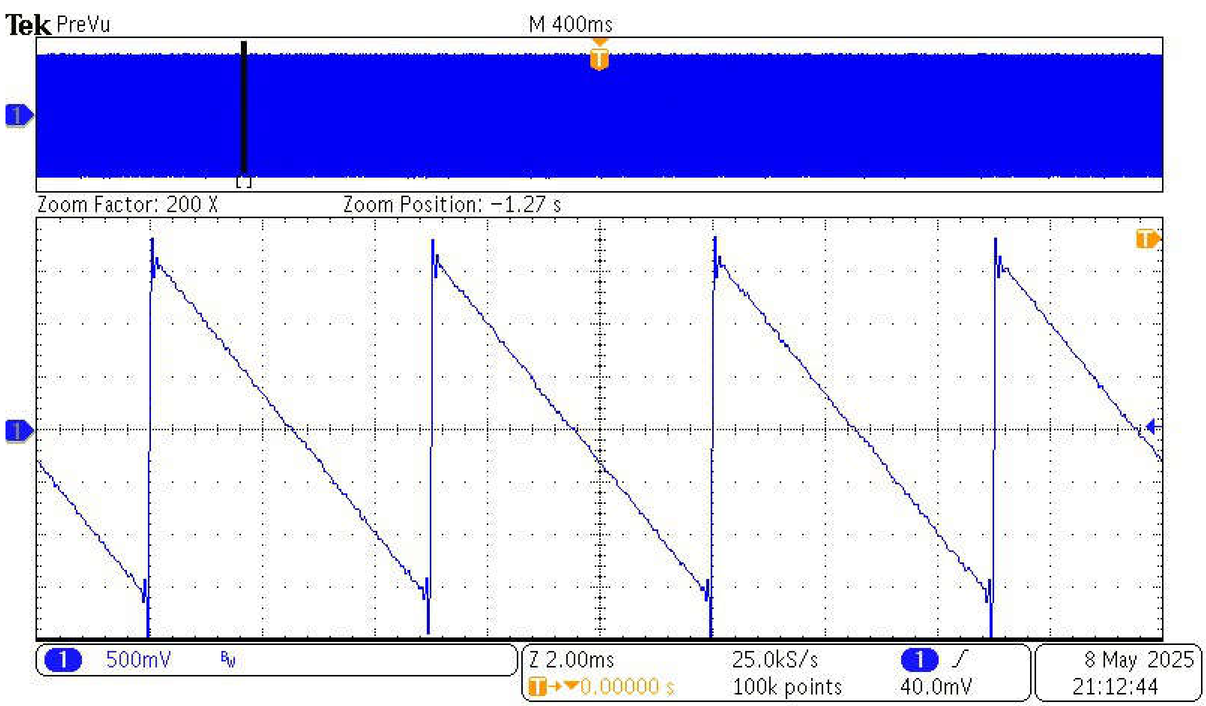

Figure 4-3. Saw Wave

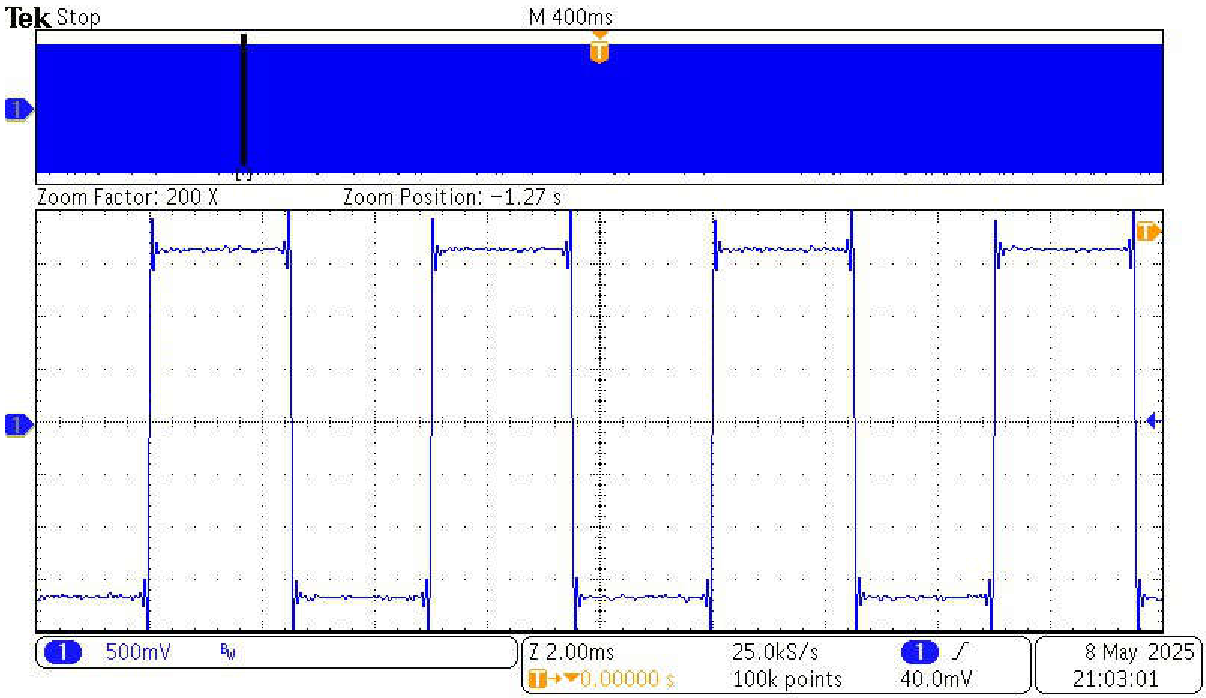

Figure 4-4. Square Wave

4.1.1 Methods for Data Collection

A large dataset is required for successful machine learning training. A recommended minimum starting place is thousands of quality examples for each classification class. The quality and diversity of the data is critical. TI offers reference designs with automatic class labeling, and software to analyze dataset quality to assist engineers in evaluating edge AI applications.

High-quality data:

- Sample rate: Sampled well above the Nyquist frequency of the signals of interest
- Steady state: Labeled frames can't be transitioning between labeled classes highest resolution available

Data collection: The final AI model can be trained on data that is representative of the final circuit design and real-world application. Lab collected data can be used as a starting place to tune the full signal chain including the model and analog circuit design. In this example we collected three different classes of periodic signal data: sine, square and sawtooth. They are saved into three separate folders

4.1.1.1 Collection Process

Ideally the data is collected through the analog front end and sensor used in the final application. *MCU Control Center Tool Developer's Guide* is available on F28P55 processors to setup a PC to LaunchPad connection for data streaming. The example found at C:\ti\c2000\C2000Ware_5_05_00_00\driverlib\f28p55x\examples\transfer\transfer_ex3_customlog_export_sci_logArrays can be used to stream an array of sensor data into a automatically generated GUI on the host PC. Then the data can be exported to CSV.

4.1.2 Data Formats

A dataset must be configured into a dataset folder containing two directories named classes and annotations. The classes directory contains a folder for each classification filled with CSV files of data in each column. Multi-channel classes require multiple data columns. The annotations folder contains three text files instances_test_list.txt, instances_train_list.txt, and instances_val_list.txt. These files are used by model composer to split the dataset into training, testing and validation sets. It is recommended that 50% of the files are in the training list, 30% in the validation, and 20% in the testing list. A visualization of the architecture is shown in Figure 4-5.

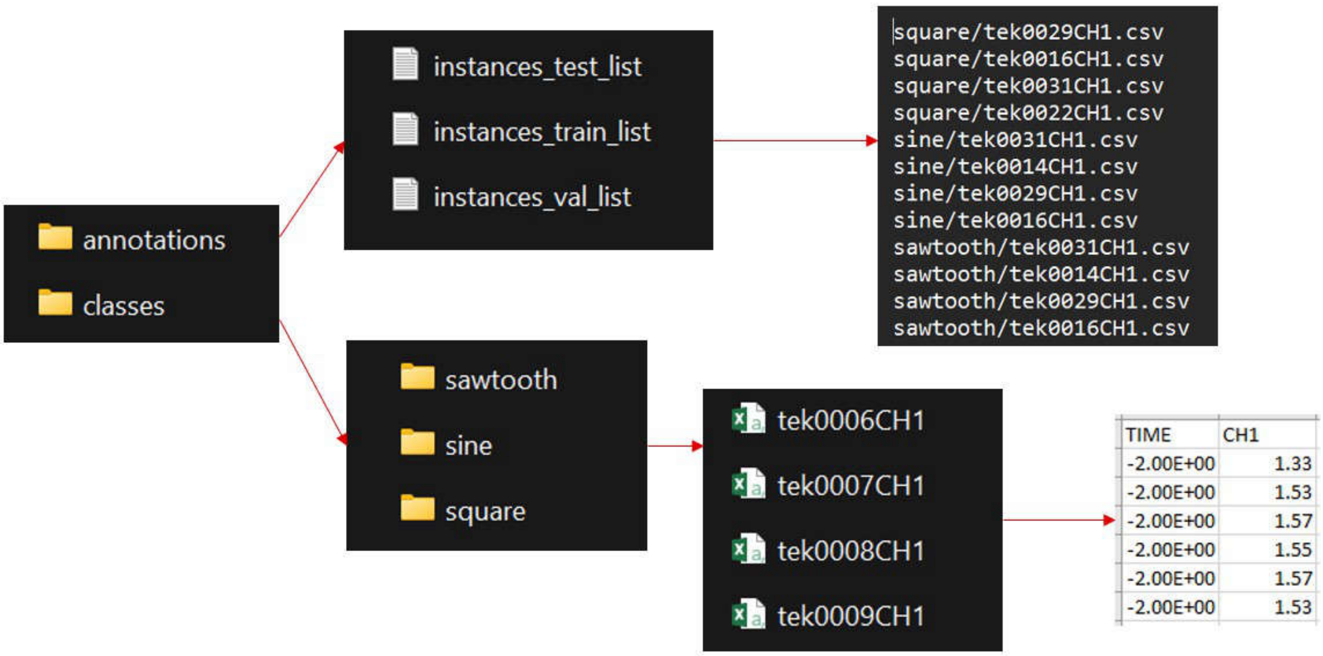

Figure 4-5. Folder Path

4.2 Model Training

This section describes the procedures to upload data, train, and compile models through Edge-AI Model Composer. A command line tool through TinyML Model Maker is available for customers to fine-tune model parameters and modify the models.

This demo features four different sizes of generic time series neural network models that can be used for a wide range of time series classification tasks.

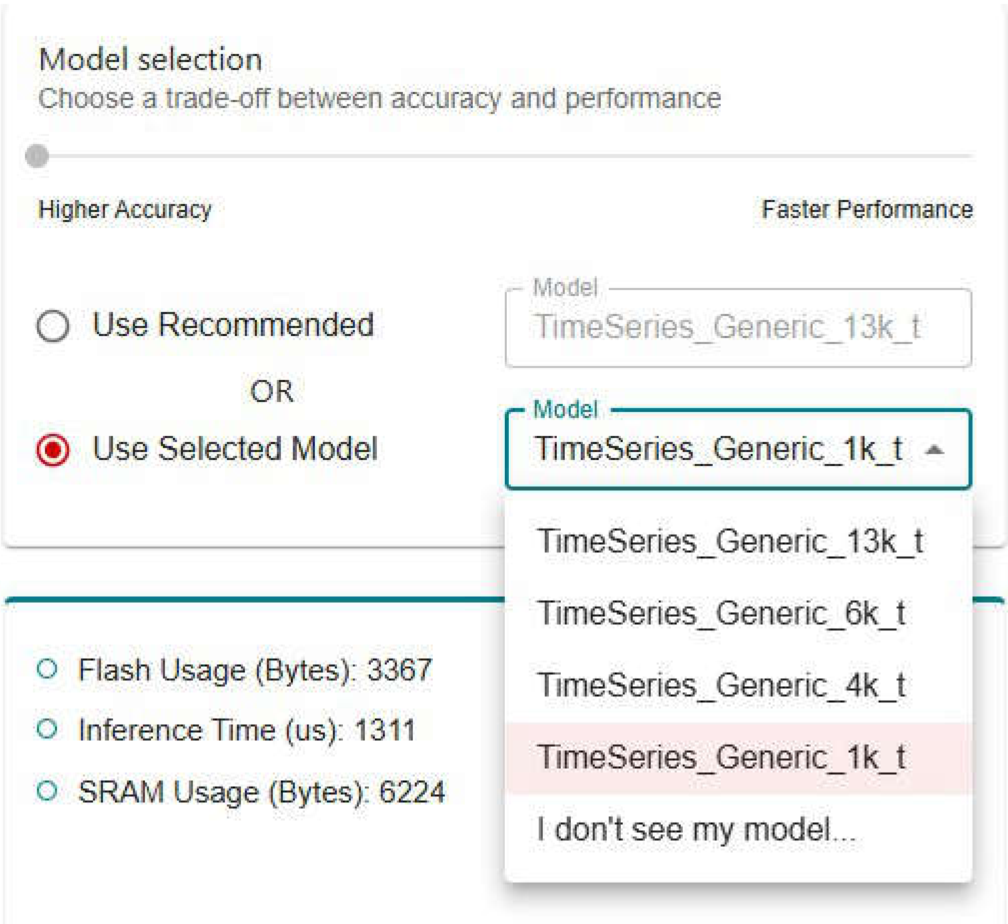

Figure 4-6. Different Model Sizes

Steps to Use Model Composer:

1. Login to https://dev.ti.com/modelcomposer/
2. Click “Example Project”, under Task select “Generic Time Series”, under Tools select “MCU Analytics Backend v1.0.1", under Sample Dataset select “hello_world_example_dsg”. Give the project a name and then click “New Project”.

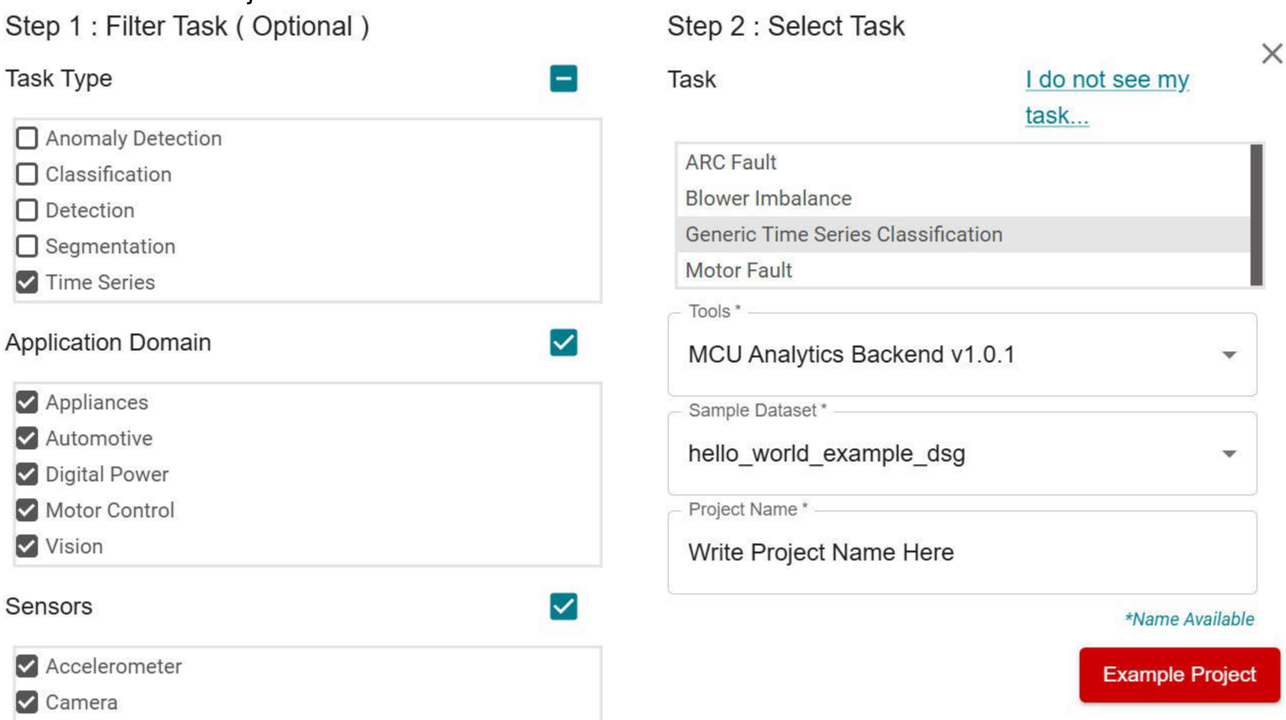

Figure 4-7. New Example Screenshot

3. The “Capture” tab is used for data visualization, labeling, and importing. The tabs at the top of the page can be used to iterate through the project. Selecting a file on the left menu brings up a visualization of that data file. Options for rendering are shown on the right side. Additional data can be imported using the “Import Data” button. This button can also be used to import your own dataset when creating a new generic time series project.

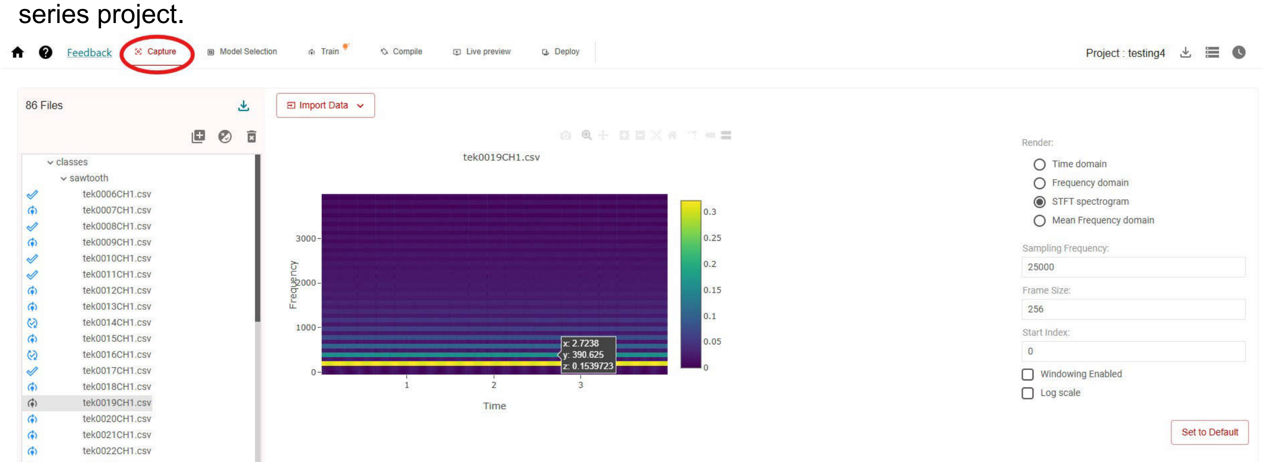

Figure 4-8. Capture Tab

4. Go to the model selection tab by using the link on the top of the page and select target device and model. The slider can be used to select a recommended device or the user can manually select a device. For this example, it is recommend using the F28P55 that uses TI's neural processing unit (NPU).

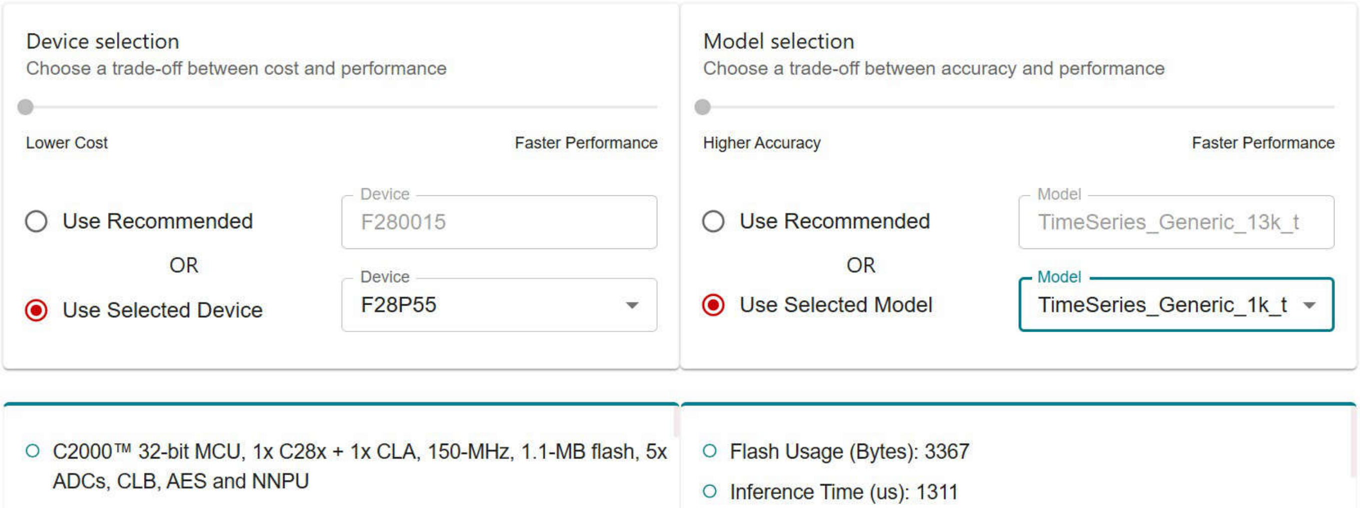

Figure 4-9. Model Selection Tab

5. Models included are all generic time series models. In example, the *1k_t* is the smallest model, for more simple classification tasks. The 13k_t is the largest for more complex tasks.
6. The flash and inference time estimates are shown below the model selection option. These can be used to determine the right model and device for your application.
7. Navigate to the training page using the links at the top and select your pre-processing options.

4.2.1 Preprocessing Options

1. Preprocessing generic time series data generally enables neural networks to more accurately classify signals. This process is referred to as feature extraction. Since the goal of the FFT and other operations are to isolate features unique to specific signals of interest. In this simple example you can see by the below figures how the sawtooth wave has a unique STFT spectrogram vs the STFT of the sine wave. More complex signals, non-periodic signals (such as the vacuum and arc fault) have a unique time varying frequency spectrogram. Concatenating frames of FFT data allows the unique time varying frequency signature of a signal to be more easily identified by a neural network.

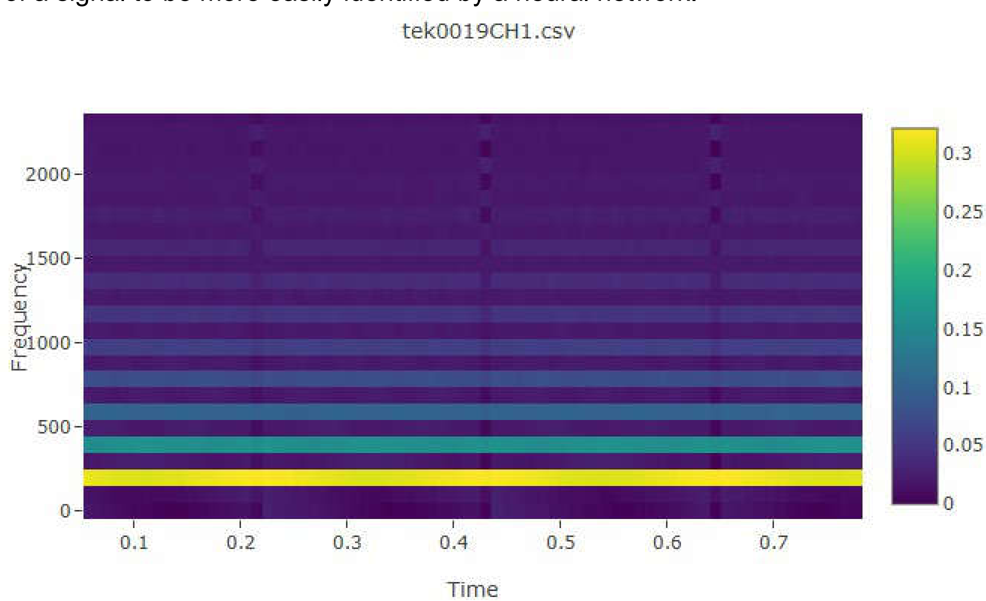

Figure 4-10. Saw Wave STFT

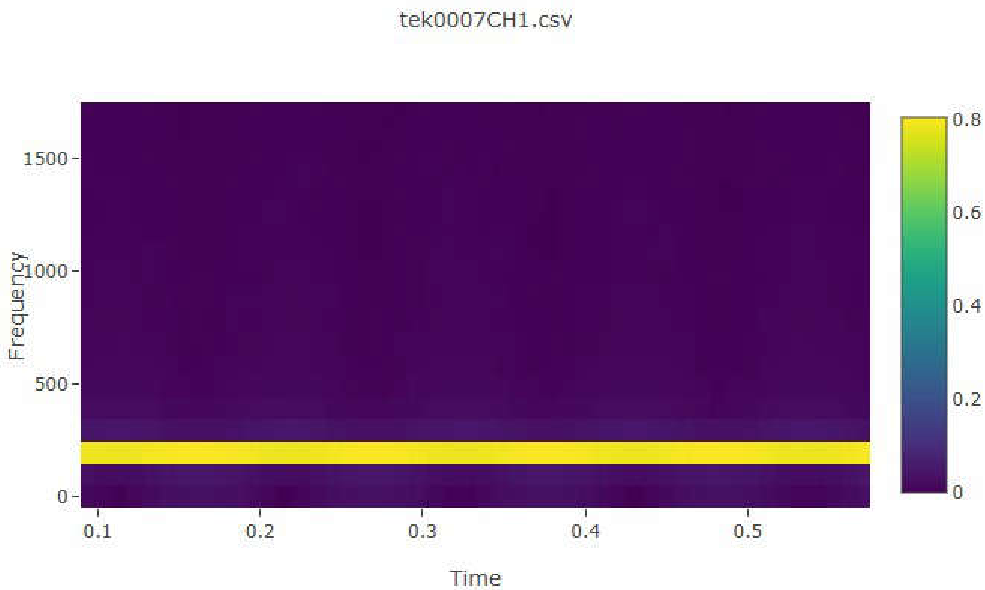

Figure 4-11. Sine Wave STFT

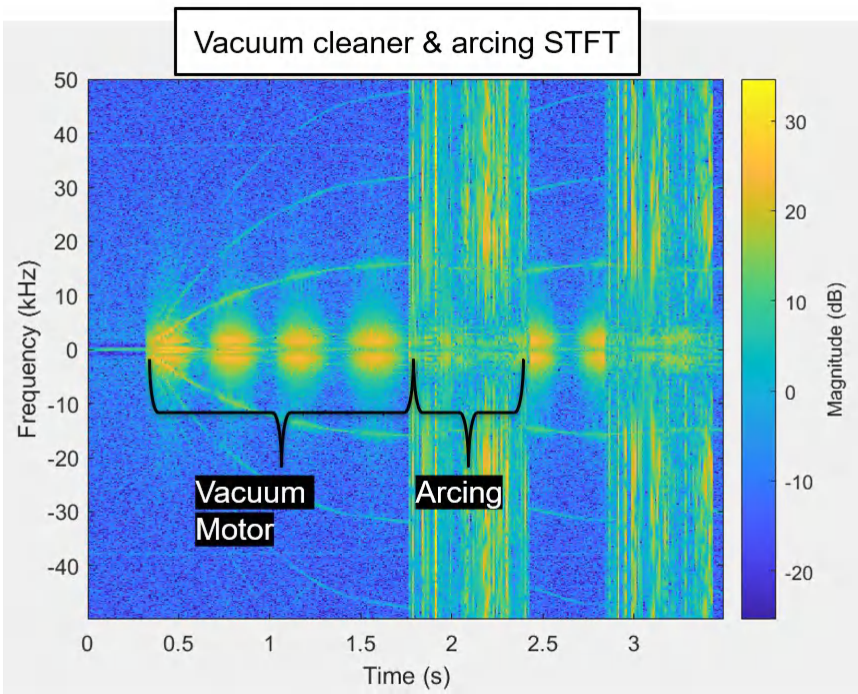

Figure 4-12. Vacuum Cleaner and Arcing STFT

2. In “Train” tab, configure the pre-processing parameters before training. There are nine preset configurations and a custom option. All the parameter fields are updated when a preset is selected. User can further adjust the parameters to improve performance. The following drop-down options are given:
   a. Preprocessing preset: Choose from a variety of preset configurations.
   b. Transform: This is the transformation applied to the raw time series data.
      - Fast Fourier Transform: (FFT) is used to extract frequency information. FFT sizes supported are all factors of 2. The output size is equal to *frame size/2*. If this option is chosen, *frames to concatenate* must be 1.
      - FFT BIN: (takes the FFT output and combines it into a number of bins equal to the *feature size per frame* parameter)
      - RAW (Raw time series data)

   c. Frame Size: (number of input samples)
   d. Feature size per frame: (number of bins for each frame of data, more bins give higher accuracy at the cost of a larger model)
   e. Frames to Concatenate: Concatenates the bins of past input frames in a buffer. This allows the model to detect time varying changes across the frequency spectrum. More frames give the model more context at the cost of a larger model.
   f. Channels: Number of sensor channels

3. Once the desired options are selected click train. The selected model goes through the training and validation data. The results are displayed after training is complete.
4. The training results displays once the model optimization is complete.

Figure 4-13. Training Results

5. A confusion matrix is also generated from the test dataset to identify which classes the neural network can confuse. In this simple example case, 100% of the data files were correctly identified.

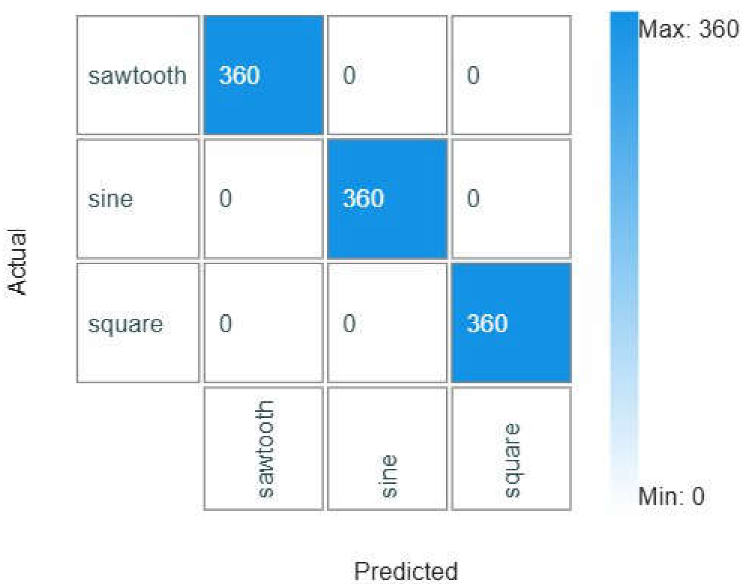

Figure 4-14. Confusion Matrix

6. TI uses quantization aware training that first trains the model in full precision, then quantizes the model parameters and re-trains the model. This approach maintains a high accuracy and significantly reduced model size.

4.3 Deploying to TI's Hardware

4.3.1 TVM Compiler

This tool is utilized in the backend of the software to generate optimized executable code based on the processor you selected. If using a device with TI neural processing unit (NPU) the generated function call automatically utilizes the accelerator in parallel to the main MCU core. This option can be disabled by choosing the “forced soft NPU preset” under compilation parameters. For more information about the compiler and "bring your own model" feature, see https://software-dl.ti.com/mctools/nnc/mcu/users_guide/. On the compile tab verify that the active model and device are properly selected, then press “Start compiling”. The compiled model can be exported with the “export compiled model” button.

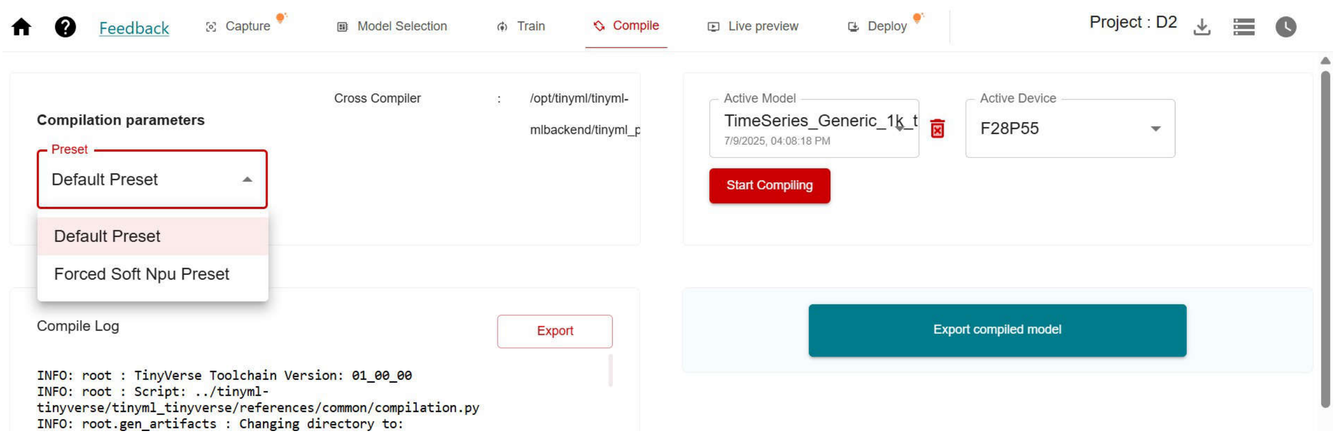

Figure 4-15. Compile Tab

4.3.2 Model Execution

- The Deploy tab provides the option to export the compiled model for deploying onto the target device or exporting a trained model for further analysis by other engineering tools

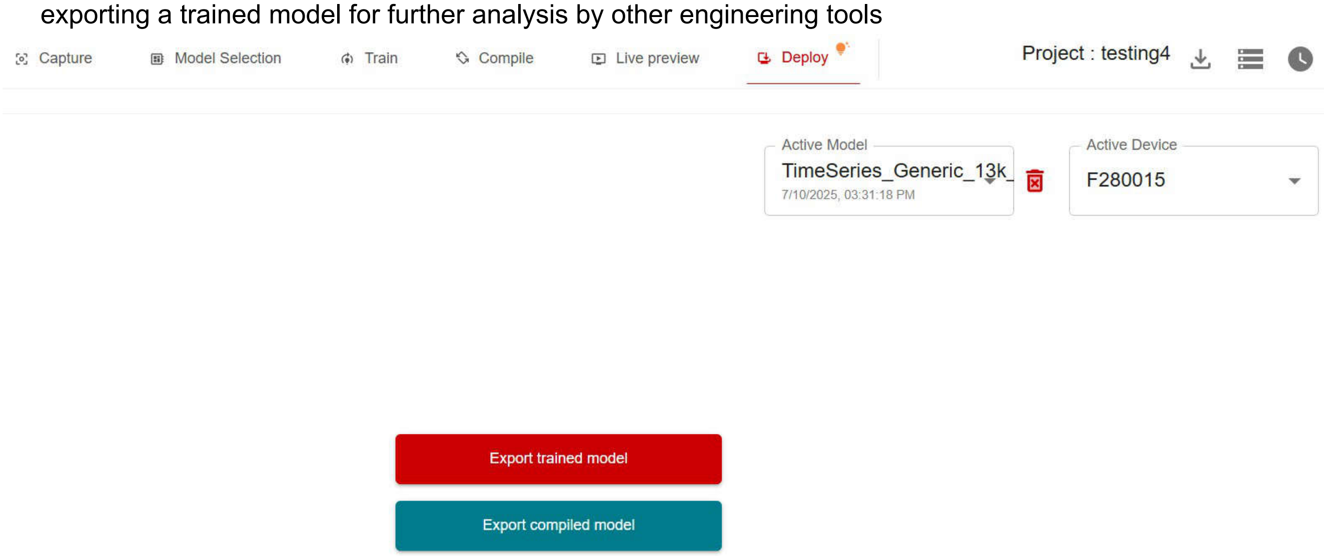

Figure 4-16. Deploy Tab

- The compiled model folder contains all the necessary configuration and code to run the model using the C2000 5.04 or later SDK

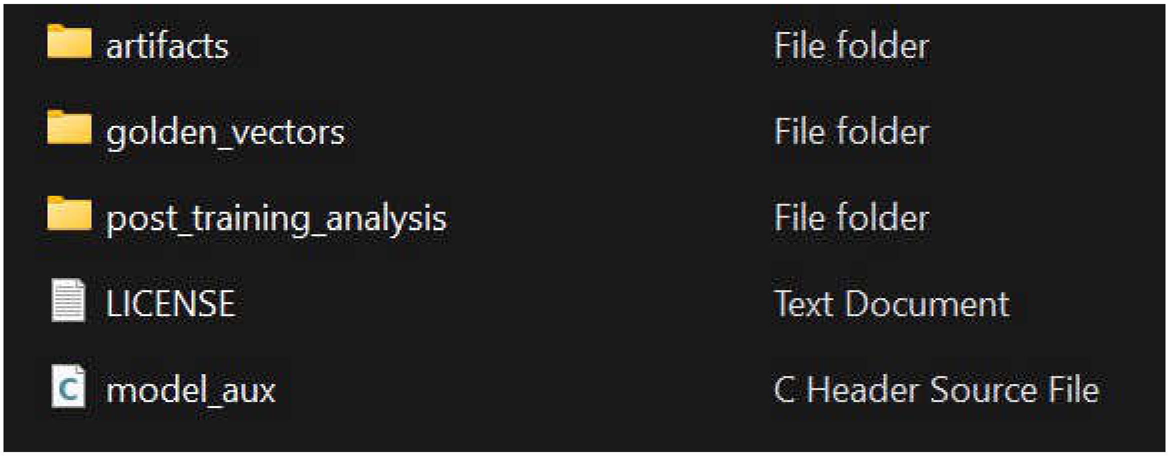

Figure 4-17. Compiled Model Folder

Steps to import trained model into CCS project:

1. Open Code Composer Studio and navigate to **File-> import project ->**.
   a. Select this folder

      \ti\c2000\C2000Ware_5_05_00_00\libraries\ai\feature_extract\c28\examples\ex_motor_fault_dataset_validation_f28p55x
2. Import the necessary files from your compiled model download
   a. Unzip the compiled model
   b. Copy the artifacts folder into the imported project and overwriting the original files.
   c. Copy the test_vector.c and user_input config.h from the golden_vector directory inside the compiled model directory into your project. These overwrite the original test_vector and user_input_config.h file.
3. Next, rebuild the project.
4. Connect a F28P55 LaunchPad or control card. With the device debugger connected, press debug to verify the model feature extraction and neural network

5 Summary

The purpose of this demo is to guide customers through Texas Instruments' Edge AI software and hardware capabilities. It is designed as a hands-on exploration tool to demonstrate key features and the development flow. This demo is not intended to represent a real-world industrial production use case, but rather to showcase the foundational elements that developers can build upon for their own applications.

IMPORTANT NOTICE AND DISCLAIMER