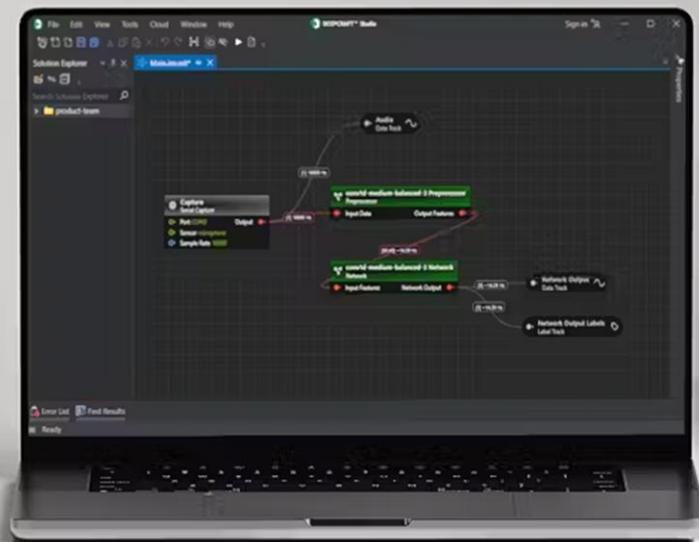




DEEPCRAFT™ Studio Hands-On Training Manual

Version 1.4 – July 2026

Hands-On Guide to DEEPCRAFT™ Studio, Infineon's complete end-to-end Edge AI Development Suite.

This tutorial will guide users through the full pipeline of building a machine learning model and deploying it on Infineon MCUs.

Gioele.Mombelli@infineon.com
Lukas.wirkestrand@imagimob.com
Yogesh.shankar2@infineon.com

Table of Contents

1. Introduction	2
1.1 - Machine Learning Algorithms in DEEPCRAFT™ Studio	3
1.1.1 - Supervised Machine Learning Algorithms: Regression and Classification	4
1.1.2 - Computer Vision Algorithms: Object Detection	5
1.2 - PSOC™ Edge E84 Artificial Intelligence Kit	5
1.3 - Goal of the Training	6
2. Hands On – Time-series Data	7
2.0 - Theory and Terms in DEEPCRAFT Studio™	7
2.1 - Exercise 1: Flashing the Data Collection Firmware to PSOC™ Edge E84 AI Kit	7
2.2 - Exercise 2: Downloading the MovementTypeDetection Studio Accelerator	10
2.3 - Exercise 3: Creating a Graph UX Data Collection Project	12
2.4 - Exercise 4: Real-Time Data Collection and Labeling	16
2.5 - Exercise 5: Adding Data to the Project and Splitting for Training/Testing/Validation	19
2.6 - Exercise 6: Design the Data Preprocessor (Solved Exercise)	22
2.7 - Exercise 7: Generating and Training Neural Networks	24
2.8 - Exercise 8: Real-Time Model Evaluation using Graph UX	31
2.9 - Exercise 9: Deploying the Model on PSOC™ Edge AI Kit	33
3. Hands On – Object Detection	39
3.1 - Exercise 1: Creating an Object Detection Data Collection Project	39
3.2 - Exercise 2: Collecting Data with the Data Collection Object Detection project	41
3.3 - Exercise 3: Creating a Rock, Paper, Scissors Studio Accelerator	42
3.4 - Exercise 4: Adding Data to the Rock, Paper, Scissors Detection Project	44
3.5 - Exercise 5: Defining the Model (Solved Exercise)	46
3.6 - Exercise 6: Selecting Data Augmentation (Solved Exercise)	48
3.7 - Exercise 7: Training the Model and Evaluating the Test Results	49
3.8 - Exercise 8: Evaluating the Model in Real-Time	53
3.9 - Exercise 9: Deploying the Model on PSOC™ Edge	56
4. Final Considerations and Additional Documentation	64

DEEPCRAFT™ Studio Hands-On Training Manual

Version 1.4 – July 2026

1. Introduction

DEEPCRAFT™ Studio is a **comprehensive solution for developing AI and machine learning applications on edge devices**. This end-to-end development platform for machine learning covers the entire machine learning workflow, from collecting and annotating high-quality data, managing, analyzing and processing data to building, evaluating, and selecting the best models and finally deploying the models on the target edge device.

This tutorial will provide a hands-on guide to DEEPCRAFT™ Studio, teaching the basics by improving on a pre-existing machine learning project for classification. These projects are called DEEPCRAFT™ Studio Accelerators and cover a specific use-case - providing data, preprocessing and a defined model which can be built upon and improved. All DEEPCRAFT™ Studio Accelerators are open source, and the projects can be downloaded either via the DEEPCRAFT™ Studio software or from the github via the [DEEPCRAFT™ AI Hub](#). DEEPCRAFT™ AI Hub is a centralized place for Infineon customers to find DEEPCRAFT™ software for any Edge AI needs: platform, tools, solutions, and models all in one place.

For those wishing to leverage a model that is already trained there is another product, the DEEPCRAFT™ Model Converter which this manual does not cover. You can read about its technical details and how to use it [here](#).

This manual does not go into detail about machine learning theory or basics, instead serving to present how to practically apply such knowledge to create machine learning models optimized to run on microcontrollers. Here are some recommended references to start, but there is also plenty available online:

- [Machine Learning Crash Course](#)
Online class developed by Google
- [Deep Learning with Python 2nd Edition by François Chollet](#)
E-book with lots of examples that cover deep learning using Python, TensorFlow and Keras
- [Machine Learning Mastery](#)
Website of guides and tutorials that cover and define many ML concepts

1.1 Machine Learning Algorithms in DEEPCRAFT™ Studio

This section introduces the machine learning algorithms supported in DEEPCRAFT™ Studio for building machine learning (ML) models to solve various business problems.

Before diving into machine learning, here is a glossary of common terms.

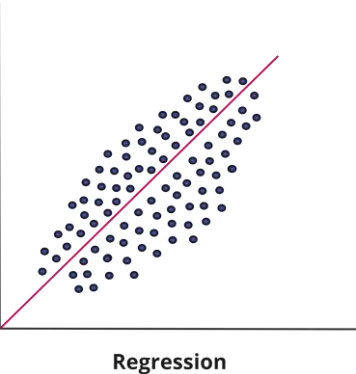
Term	Definition
Back-propagation	An algorithm used during training to test for errors working back from output nodes to input nodes.
Batch or Mini-Batch	The number of samples to send through the model during training before updating the model's weights.
Binary classification	A task in which outputs can only be one of two categories.
Classes	The set of possible labels for a classification machine learning model.
Epoch	One training pass of the entire training data set. If the data is split into batches, an epoch is complete once all batches have been put through the model.
Features	Independent variables that act like inputs into the system.
Gradient descent	An optimization algorithm used during training to minimize loss.
Generalization	The model's ability to provide good results on previously unseen data.
Inference	The process of sending data into a machine learning model to calculate an output. The "Inference Engine" is the system component (in our case, a software library) that applies rules to the model to produce a prediction.
Label	This is attached to a sample to indicate what the target is for the given sample.
Layer	A component of a machine learning model that performs some transformation on the data. Deep learning models have three or more layers, but often many more.
Learning rate	A measure of how much the weights are changed between iterations to attempt to find the minimum loss point for the model.
Loss	The difference between the expected output from a model and the actual output from the model. This is the quantity that should be minimized during training.
Multi-class classification	A task in which outputs are categories from a list of possible outputs (>2).
Multi-label classification	A task in which a single sample can have more than one label from the list of possible outputs.
Neural Network (NN)	A specific type of machine learning model that uses interconnected nodes in a layered structure that resembles neurons in the human brain.
Prediction, output or result	The output from a machine learning model.
Rules or weights	The parameters that are used to influence how a model determines its predictions. These are adjusted during training.
Sample, input or data	An input to a machine learning model.
Scalar regression	A task in which the output is a continuous value rather than a discrete item from a list.
Target	The expected or correct output from a machine learning model.
Training	The process of adjusting model weights to optimize how well the model predicts the correct output for any given input.
Vector regression	A task in which the output is a set of continuous values.

1.1.1 Supervised machine learning algorithms: Regression and Classification

Regression Algorithms

Regression algorithms are designed to understand the correlations between dependent and independent variables. The dependent variables are those we aim to predict, while the independent variables are those we use for prediction. **Regression algorithms are used to predict continuous numerical outputs.** You can evaluate the regression models using metrics like Mean Squared Error (MSE), R2-Score, and Mean Absolute Percentage Error (MAPE), along with graphical plots such as Quantile - Quantile and Histogram of residuals.

Studio supports regression tasks.

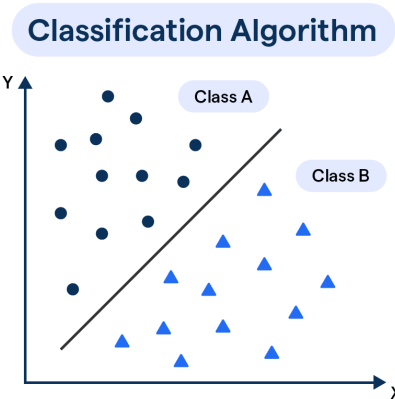


Examples of regression problems
Estimate the temperature of a motor
Estimate the level of liquid in a tank
Estimate the power consumed by a system

Classification Algorithms

Classification algorithms are used to predict categorical or discrete values. **Classification involves predicting a label or category.** These algorithms classify the data into one or more labels. A binary classifier deals with two classes or categories, while a multi-class classification algorithm handles more than two classes. You can evaluate the classification models using metrics, like confusion matrix, recall, F1-Score, accuracy.

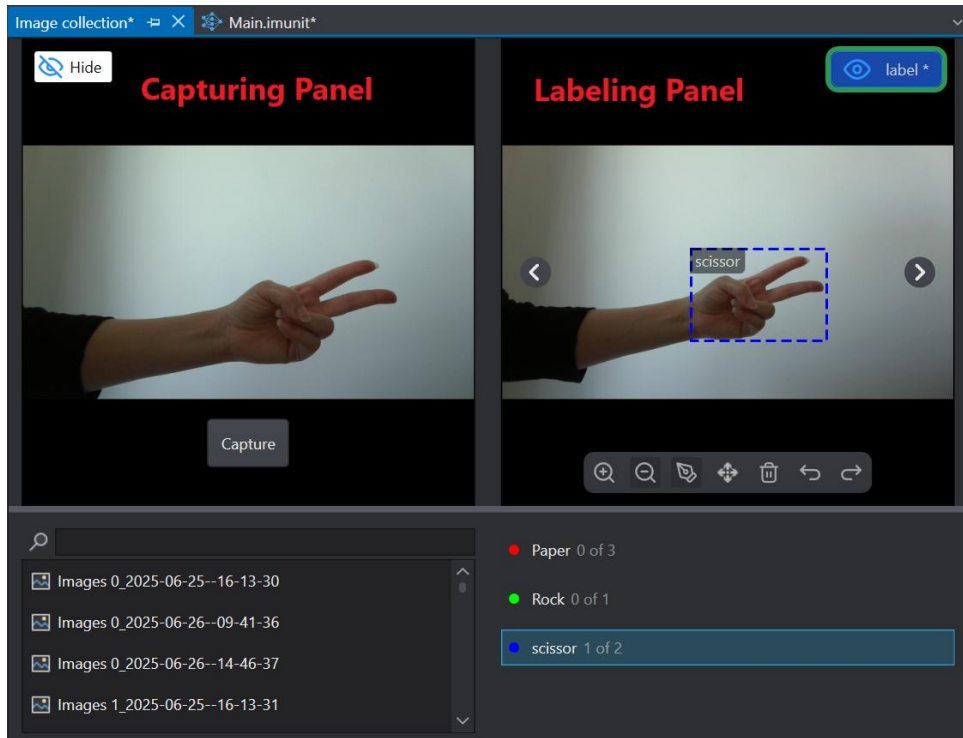
Studio supports classification tasks.



Examples of classification problems
Recognizing different sounds with a microphone
Detect gestures performed in front of a radar
Detect different movement patterns with an IMU

1.1.2 Computer Vision Algorithms: Object Detection

DEEPCRAFT™ Studio supports the end-to-end development workflow for building computer vision models optimized for edge devices. Due to the built-in Ultralytics YOLO model training pipeline, there is no need to be a domain expert in order to build vision-based Edge AI models. The inherent size needed to process images has long prevented models running on the edge. With the advent of powerful new microcontrollers (MCUs) containing neural processing units (NPUs) this is changing, and DEEPCRAFT™ Studio now streamlines the process of making vision models.



More on <https://developer.imagimob.com/deepcraft-studio/getting-started/machine-learning-algorithms#semi-supervised-machine-learning-algorithms-computer-vision>

1.2 PSOC™ Edge E84 Artificial Intelligence Kit

Target hardware for this tutorial is the PSOC™ Edge E84 Artificial Intelligence Kit.

The PSOC™ Edge E84 AI Kit is a low-cost kit with multiple sensors designed for rapid prototyping of AI-powered applications. It features the PSOC™ Edge E84 series microcontroller (MCU) and a multitude of on-board multimedia, machine learning and connectivity features. The evaluation kit carries a PSOC™ Edge E84 MCU, with 512-Mbit QSPI flash, 128-Mbit Octal RAM, and a AIROC™ CYW55513IUBGT based Wi-Fi & Bluetooth® combo Murata Type 2FY (LBEE5HY2FY) connectivity module. This kit features an on-board programmer/debugger (KitProg3) SWD debug header, a MIPI-DSI connector, speaker interface, USB host and device interfaces, IO expansion headers, IMU sensor, magnetometer, barometric pressure sensor, radar and, analog and digital microphones for audio interfaces, and an image sensor.



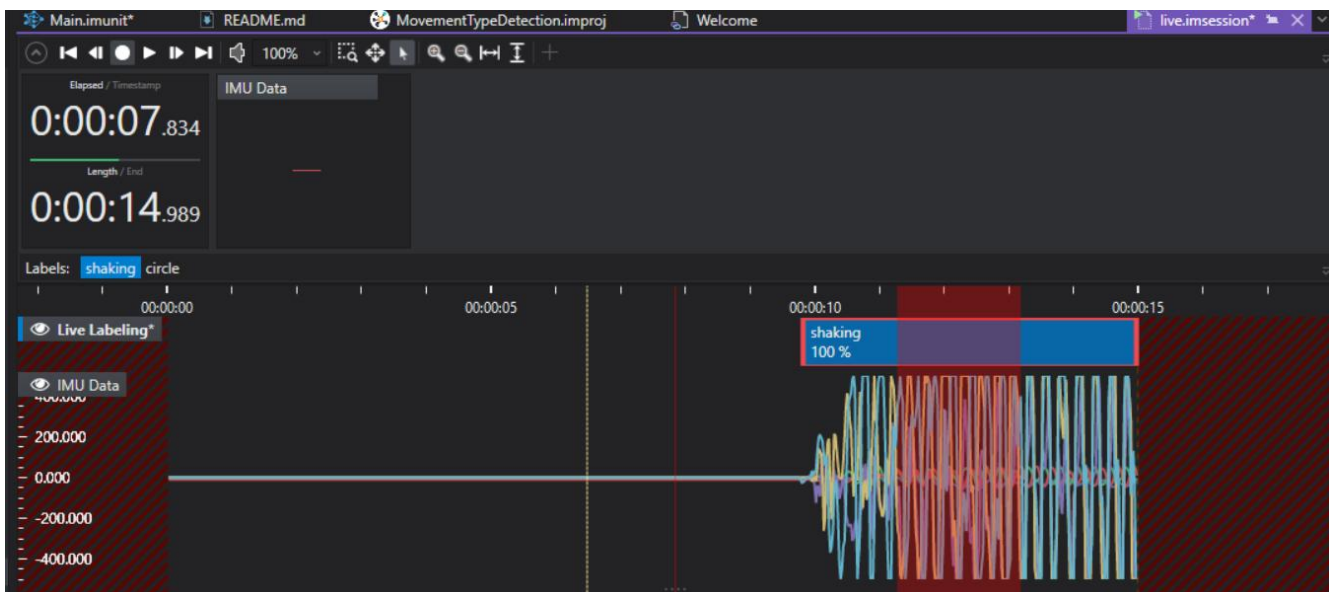
1.3 Goals for the Training

This training guides you through building a neural-network-based machine learning application using Infineon hardware and software. Creating an application entirely from scratch would require extensive data collection that isn't feasible within the available time. Instead, **you will extend an existing classification project by adding a new class of signals.**

You will collect and label data, train the neural network, test it, and deploy the model on-device to evaluate its performance on real hardware.

For this purpose, we'll use the **Studio Accelerator** project **MovementTypeDetection**. This project distinguishes among three movement types (circle, shaking, and stationary) using the PSOC™ Edge AI Evaluation Kit's 6-axis IMU (accelerometer and gyroscope). Note that in this project, stationary segments are not explicitly labeled. As provided, the model performs well on clear movement types but can misclassify edge cases where shaking and circular motions look similar and the ground truth is ambiguous.

Your goal is to add an additional movement (suggestion: rotating the kit around its cable axis) by collecting and labeling data, retraining the neural network, and verifying that the model can reliably detect the new class.



Chapter 3 will focus on another use-case, training computer vision models for deployment on Infineon MCUs. It is recommended to have completed Chapter 2 before proceeding to Chapter 3.

2. Hands On – Time-series Data

Pre-requisites:

- DEEPCRAFT™ Studio version ≥ 5.8
- ModusToolbox™ version ≥ 3.6
- PSOC™ Edge E84 AI Evaluation Kit or PSOC™ 6 AI Evaluation Kit
- USB-C Cable

Make sure you are connected to the network and are logged in to your Imagimob account in DEEPCRAFT™ Studio.

2.0 - Theory and Terms in DEEPCRAFT™ Studio

DEEPCRAFT™ Studio utilizes a graph-based interface to provide machine learning creation without coding. Graph UX is an intuitive **interface to visualize the end-to-end machine learning workflow as graphs**. This graph machine learning framework is designed to provide a clear understanding of the modeling workflow from building to evaluating machine learning models.

Use the **Graph UX library** to create a machine learning workflow by **dragging and dropping** machine learning components on the **canvas**.

State-of-the-art machine learning is all about neural networks, which are easily represented as graphs. This analogy works throughout the machine learning development process, from data collection, model building, model training, all the way to model evaluation and deployment on device. By representing the data, the processing, the modeling, the training and evaluation runs in the same coherent view, with the same basic building blocks and concepts, everything is easier to work with.

What can I do using Graph UX?

Graph UX offers the following functionalities:

- Real-time data collection and labeling
- Real-time model evaluation
- Code generation

Note: The first release of Graph UX covers only data collection, model evaluation, and code generation. In future releases, Graph UX will support the entire end-to-end machine learning workflow, including data management, data cleaning, augmentation, and model training.

2.1 - Exercise 1: Flashing the Data Collection Firmware to PSOC™ Edge E84 AI Kit

The goal of this exercise is to flash the PSOC™ Edge E84 AI kit with streaming firmware that streams sensor data from the USB port into DEEPCRAFT™ Studio, enabling rapid data collection and model testing. The streaming firmware is designed to collect data from all the sensors integrated into the kit, including the microphone, accelerometer, gyroscope, magnetometer, pressure-temperature, humidity-temperature and radar.

There are three ways to flash the Streaming Firmware onto the kit:

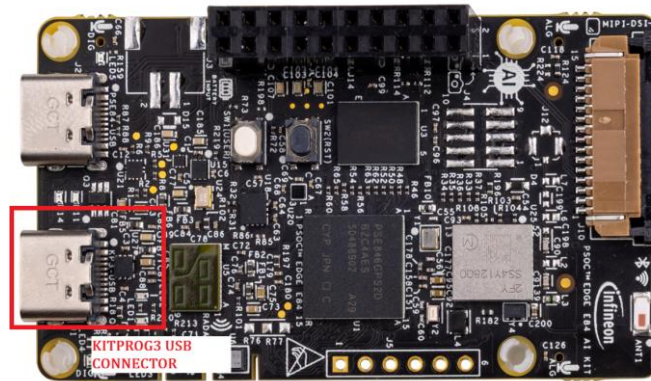
- (Recommended) Using Web-based OOB webpage, outlined below
- Using ModusToolbox™ Programmer, instructions [here](#)
- Using the ModusToolbox™ 'PSOC Edge Machine Learning DEEPCRAFT Data Collection' Code Example, available in the software, instructions [here](#)

This exercise will guide you through the Web-based OOB example but if that for some reason doesn't work, try the other approaches.

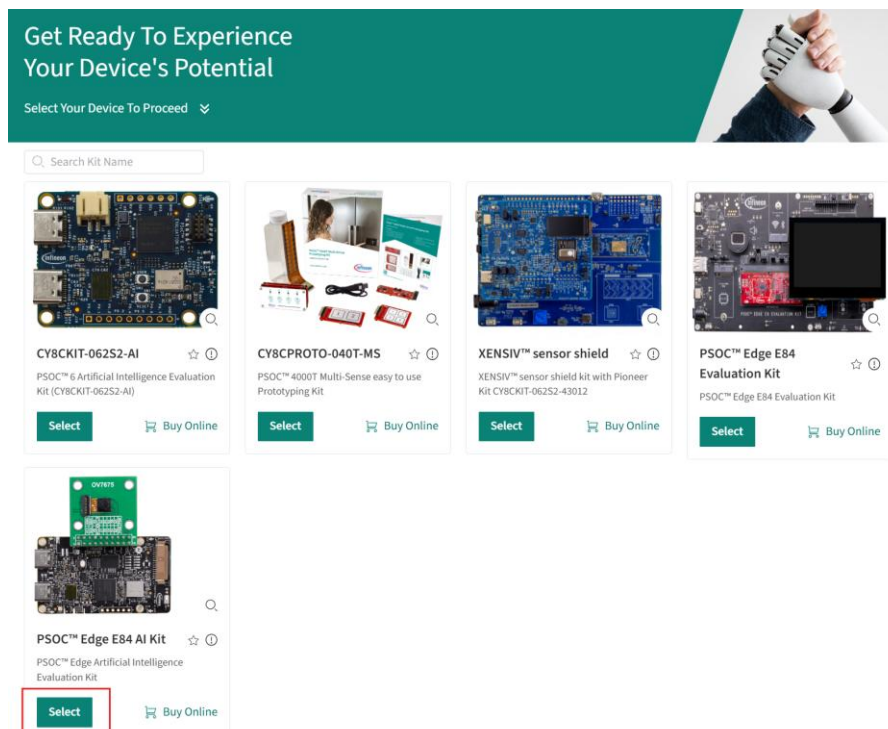
Flashing the Streaming Firmware using OOB Web Page

To flash the streaming firmware onto the kit, follow the pre-requisites and steps outlined below:

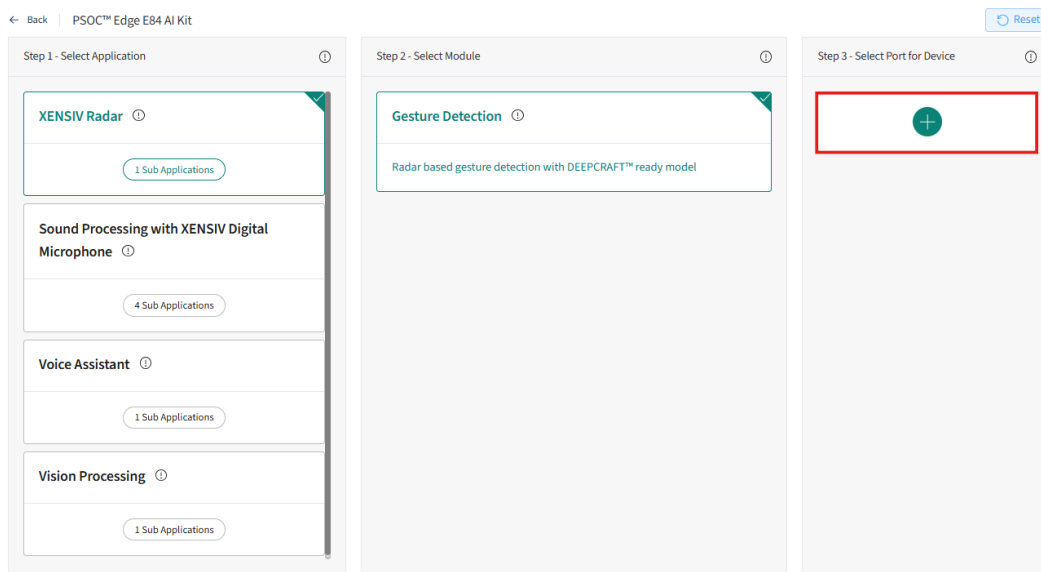
1. Connect the **KitProg3 USB connector (J1)** port on the board to the PC using the USB cable. The green LED (LED1) will start blinking.



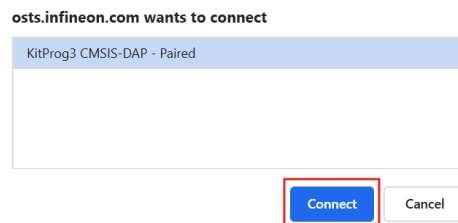
2. Navigate to <https://osts.infineon.com/devkit>
The landing page displays a list of available kits.
3. Locate the PSoC™ Edge E84 AI Kit and click **Select**. The PSoC™ Edge E84 AI Kit screen appears.



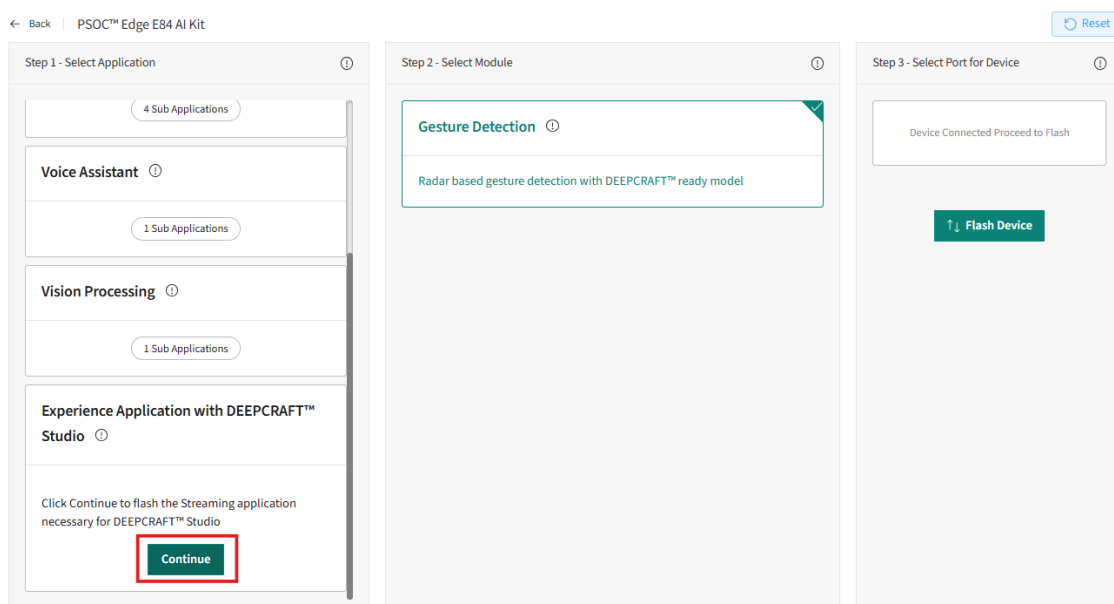
4. In **Select application** and **Select Module**, select the first application and the use-case respectively (it does not matter which one you choose in this step):



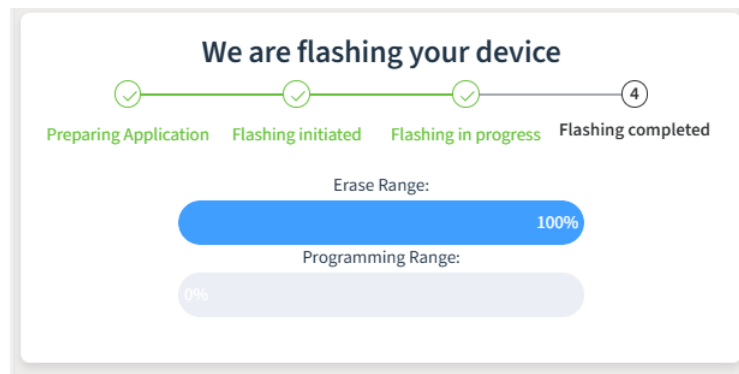
5. In **Select Port for Device**, click the Add icon to initiate the connection. In the dialog, select the device type **KitProg3 CMSIS-DAP** and click **Connect**. The device is ready for flashing.



6. Go back to the **Select application** tab, navigate to **Experience Application with DEEPCRAFT™ Studio** and click **Continue** to program the streaming firmware onto the board.



7. The flashing process is initiated. The following screen appears.

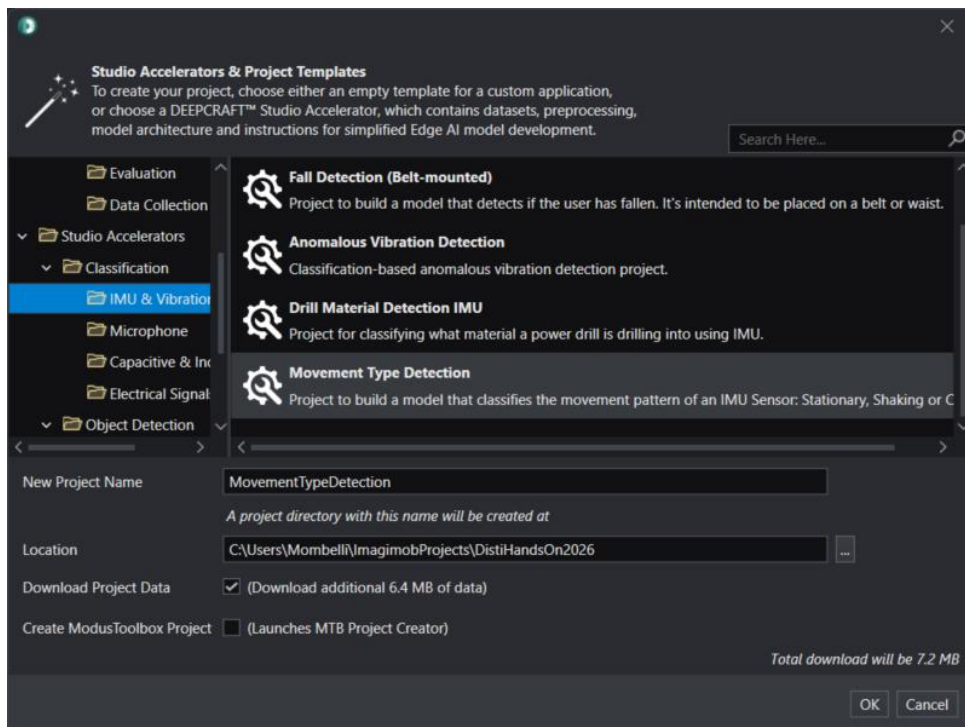


After the kit is programmed successfully, you are ready to proceed with the next exercise.

2.2 - Exercise 2: Downloading the MovementTypeDetection Studio Accelerator

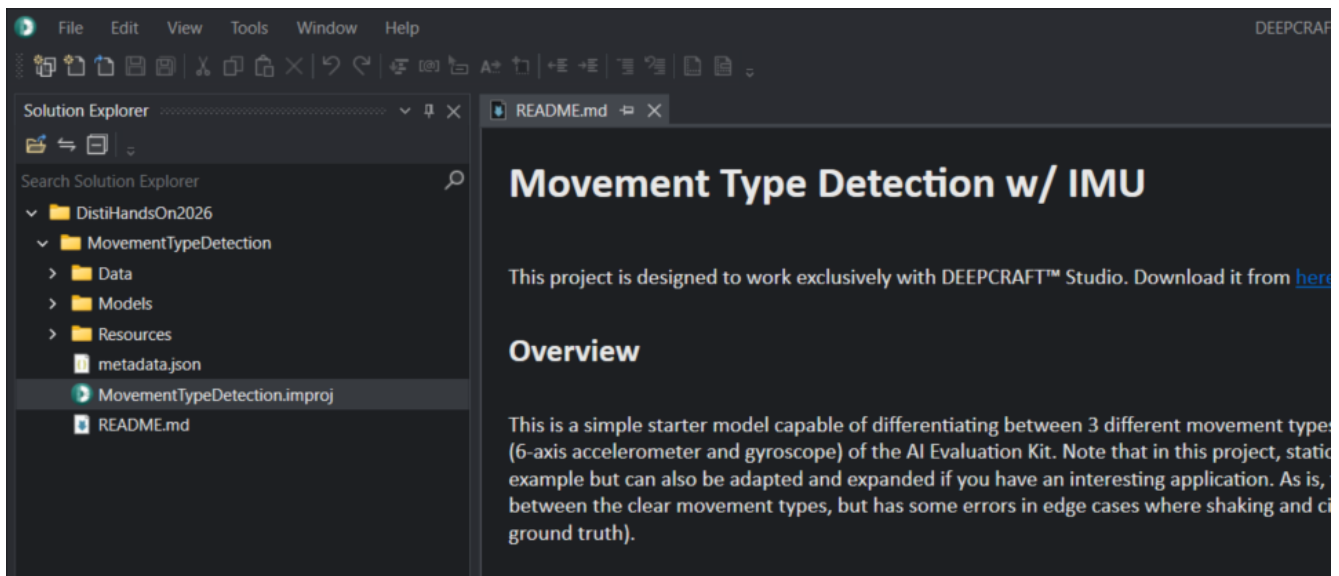
For this specific training, the DEEPCRAFT™ Studio Accelerator **MovementTypeDetection** will be used.

1. Open DEEPCRAFT™ Studio.
2. Click **New Project** in the welcome screen. The welcome screen appears when you open DEEPCRAFT™ Studio for the first time.
OR
Go to **File** and select **New Project**. The New Project window appears.
3. Expand **Studio Accelerators**, and depending on the machine learning problem you want to solve, expand either **Classification** or **Object Detection**. Select the IMU & Vibration sensor family, browse the available projects, and choose MovementTypeDetection as your starting point by clicking on it.

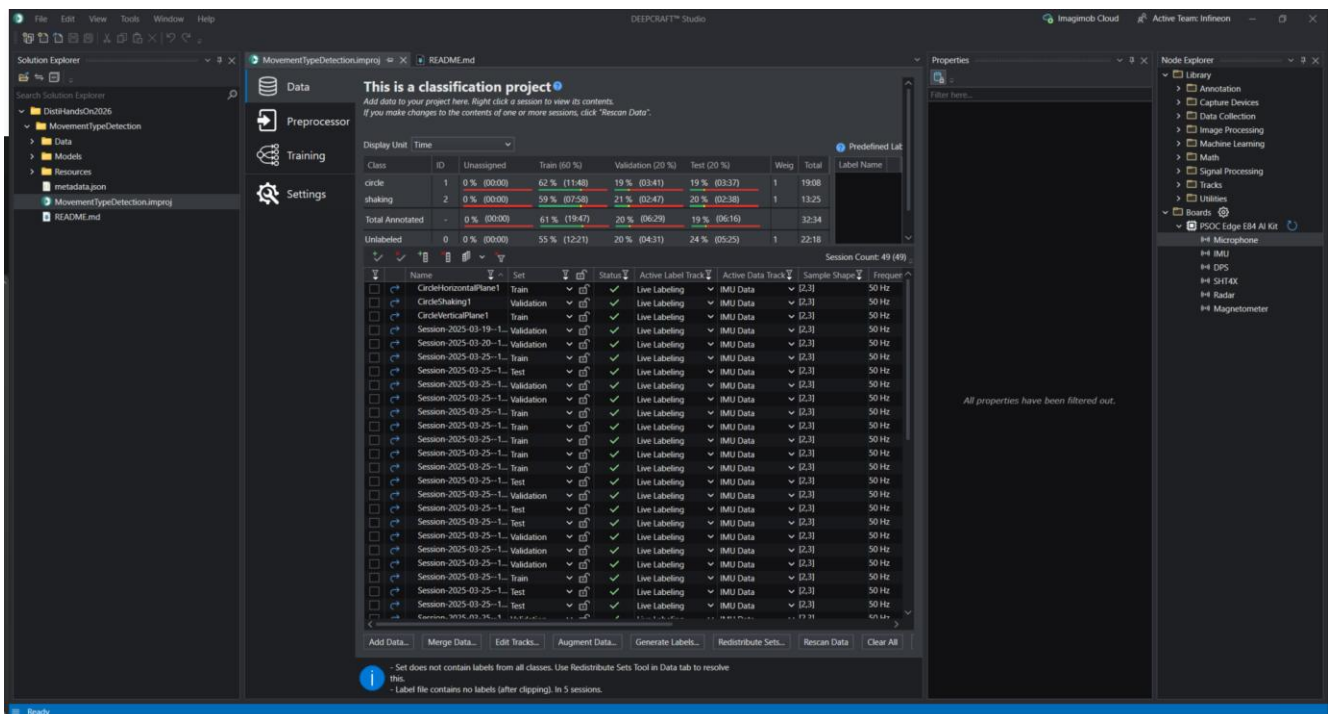


4. In **New Project Name**, edit the project name if needed. By default, the default project name is displayed.

5. In **Location**, specify the location where you want to create the workspace and the project directory.
6. Select **Download project data checkbox** to download the project data to the workspace selected earlier. Leave Create ModusToolbox™ Project unticked. This will be done later in a separate step.
7. Click OK to download the data and create the Studio Accelerator project.
8. The MovementTypeDetection project folder will appear in the Solution Explorer tab of DEEPCRAFT™ Studio.



9. Double click on the **MovementTypeDetection.improj** file to open the DEEPCRAFT™ Studio project. The main DEEPCRAFT™ Studio Project window will open.



Take some time to explore the different tabs (Data, Preprocessor, Training) and get familiar with the way DEEPCRAFT™ Studio manages the project.

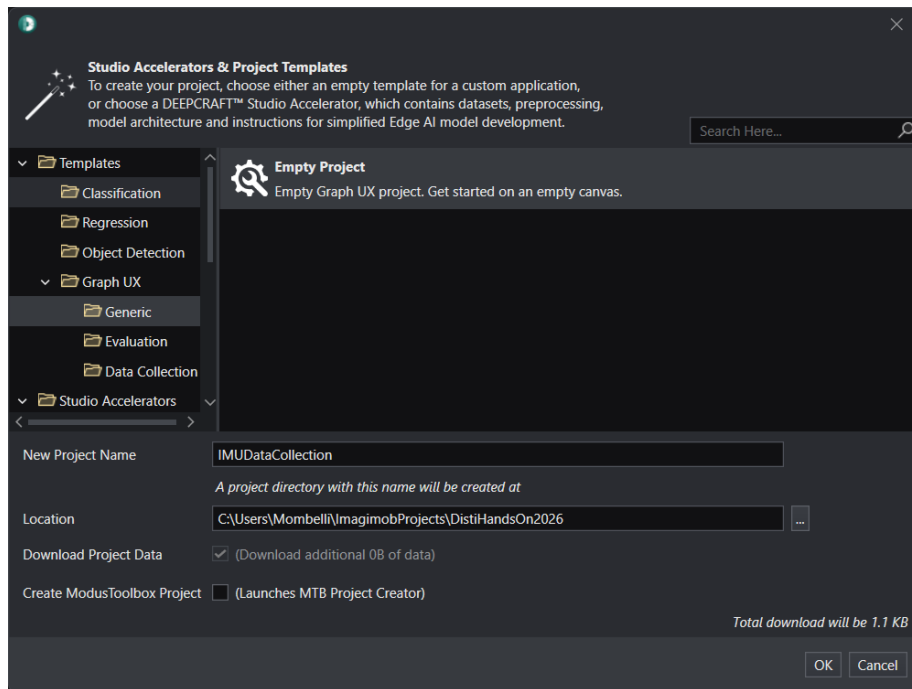
2.3 - Exercise 3: Creating a Graph UX Data Collection Project

This exercise will guide you through the procedure of downloading an empty Graph UX project and configuring it to **collect data from the PSoC™ Edge AI Evaluation Kit's IMU**.

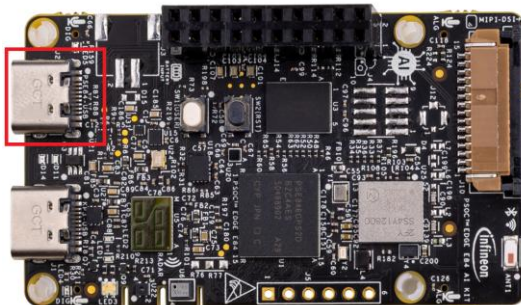
Step 1: Create an empty Graph UX Project

To create a Graph UX project for data collection, follow these steps:

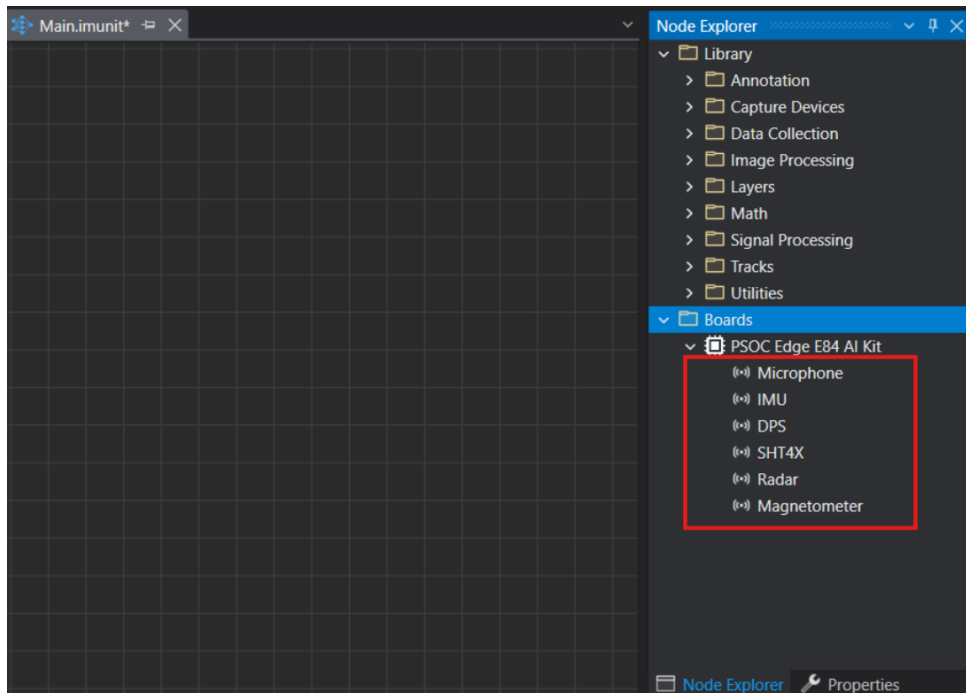
1. In DEEPCRAFT™ Studio, navigate to **File> New Project**. The New Project window appears.
2. Expand **Templates> Graph UX> Generic** and to select **Empty project**.
3. In **New Project Name**, modify the name of the project to **IMUDataCollection**.
4. In **Location**, specify the location where you want to save the project directory.
5. Click **Ok** to create the project. The **Main.imunit** file will be automatically opened.



Step 2: Connect PSoC™ Edge E84 AI Kit to PC/laptop



Connect the PSOC™ Edge E84 AI Kit to the laptop or PC through USB connector (J2) using a Type-C USB cable. After connecting the kit, navigate to **Node Explorer Window> Boards** to check if the board is connected properly. The PSOC™ Edge E84 AI Kit, along with all the sensors is displayed under Boards:



Take some time to explore the different Nodes available.

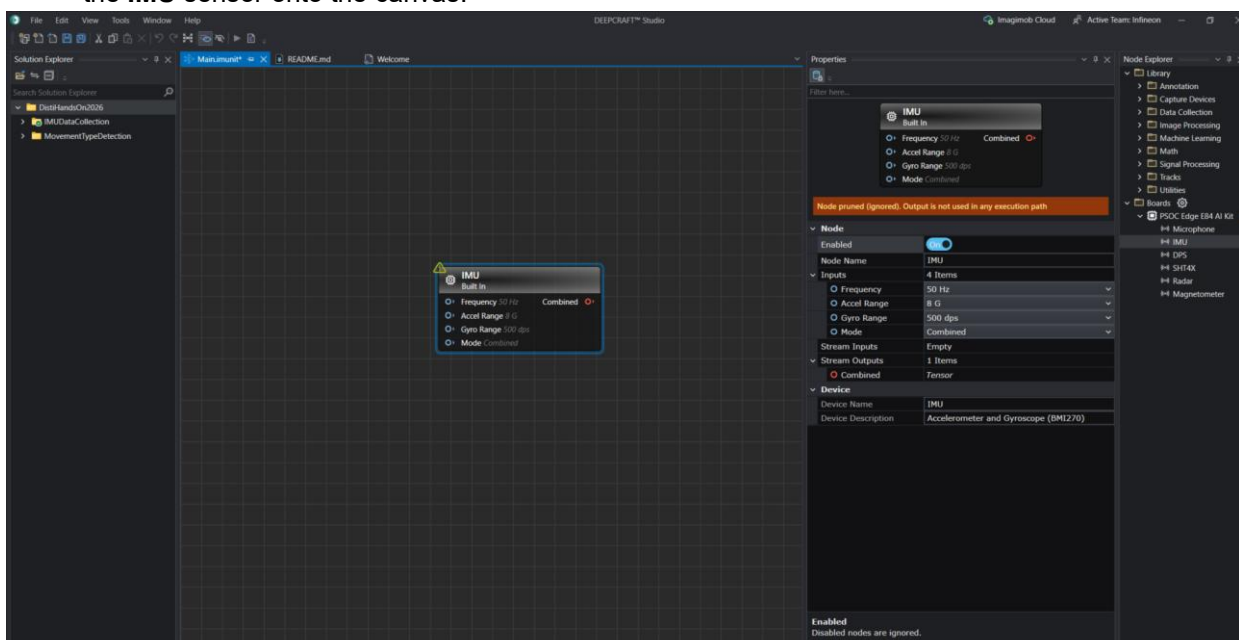
Step 3: Create Data Collection and Data Labeling Graph

To create a simple data-collection and labeling graph, add a **sensor** to stream data, a **data track** to visualize and label the stream, and a predefined labels node to define the labels.

1. Select the Sensor node to stream data

After connecting the kit to the PC and creating a Graph UX project, let's set up the IMU Sensor unit for data collection and Predefined unit for data labeling on the canvas.

1. If not already open, expand **IMUDataCollection** directory and double-click the **Main.imunit** to open the canvas.
2. Navigate to the **Node Explorer** window, expand **Boards> PSOC™ Edge E84 AI Kit** and drag and drop the **IMU** sensor onto the canvas:



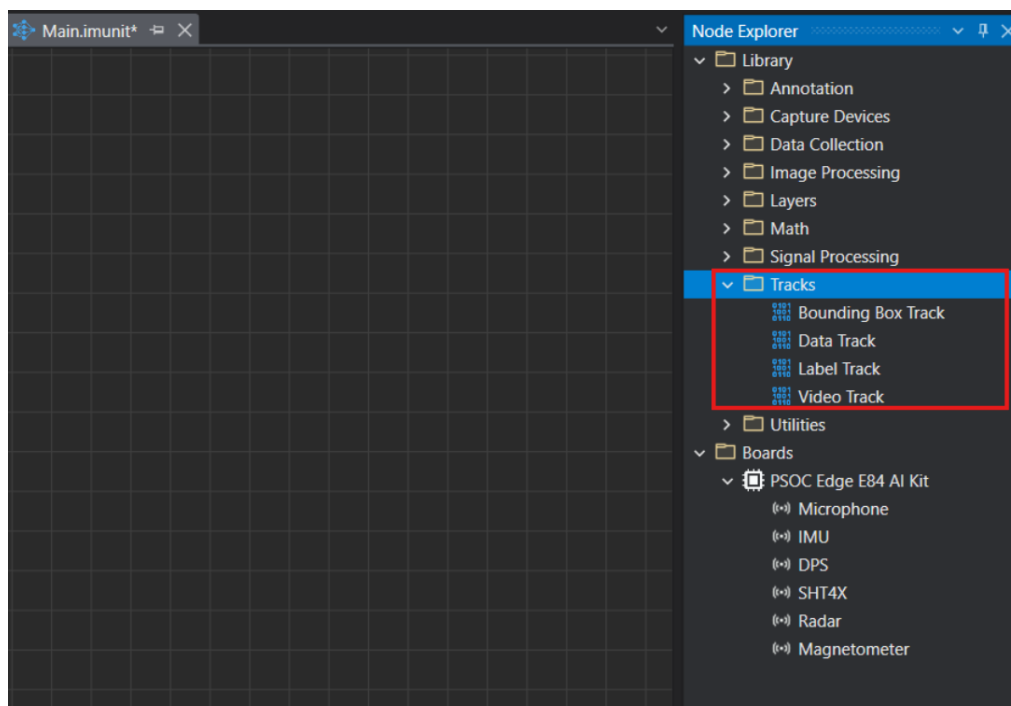
By clicking on each node in the canvas, the Properties window will be populated with information about the configuration of the node:

Sensor Node	Properties
Node	Enabled: Turn the radio button ON or OFF to enable or disable the node respectively. Node Name: Enter the name you want to assign to the node.
Inputs	Frequency : Select the sampling frequency at which you want to collect data in Studio - 50 Hz, 100 Hz, 200 Hz, 400 Hz. Accel Range: Select the min/max gravity range from the list - 2G, 4G, 8G, 16G. Gyro Range: Select the Angular rate measurement range from the list - 125 dps, 250 dps, 500 dps, 1000 dps, 2000 dps. Mode: Select the mode of data collection from the following: 1. combined: collect both accelerometer and gyroscope in one data track 2. Accel only: collect data from accelerometer only 3. Gyro only: collect data from gyroscope only 4. Split: collect data from both accel and gyro in different data tracks
Device	Device Name: Displays the name of the sensor. Device Description: Displays the description of the sensor.

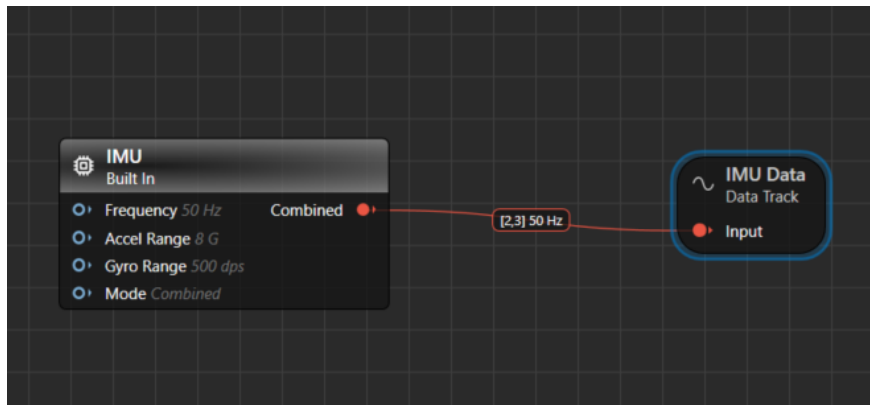
2: Set up the Visualization Tracks

The Visualization Nodes help in visualizing the data collected from the sensor nodes as tracks in the session screen.

Only the data streams connected to a Data Track Node can be stored, so it's important to connect a Data Track Node to each signal that must be collected.



1. Navigate to the **Node Explorer** window, expand **Library> Tracks** and drag and drop the **Data Track** Visualization unit onto the canvas.
2. Click on the red icon in the **IMU** sensor node and drag over to the red icon in the **Data Track** node. This creates a connection between the two nodes.



The Graph UX project is now set correctly to collect data from the IMU and store it in a Data Track file. There are several types of Visualization Nodes available:

- **Data Track:** for visualizing and/or saving the sensor data collected from the board.
- **Label Track:** for visualizing the predictions of a Neural Network for classification.
- **Bounding Box Track:** for visualizing the output of an Object Detection model (bounding boxes)
- **Video Track:** for visualizing video data collected from the Local Camera node. (Applicable only if you set the Local Camera node to collect video data.)

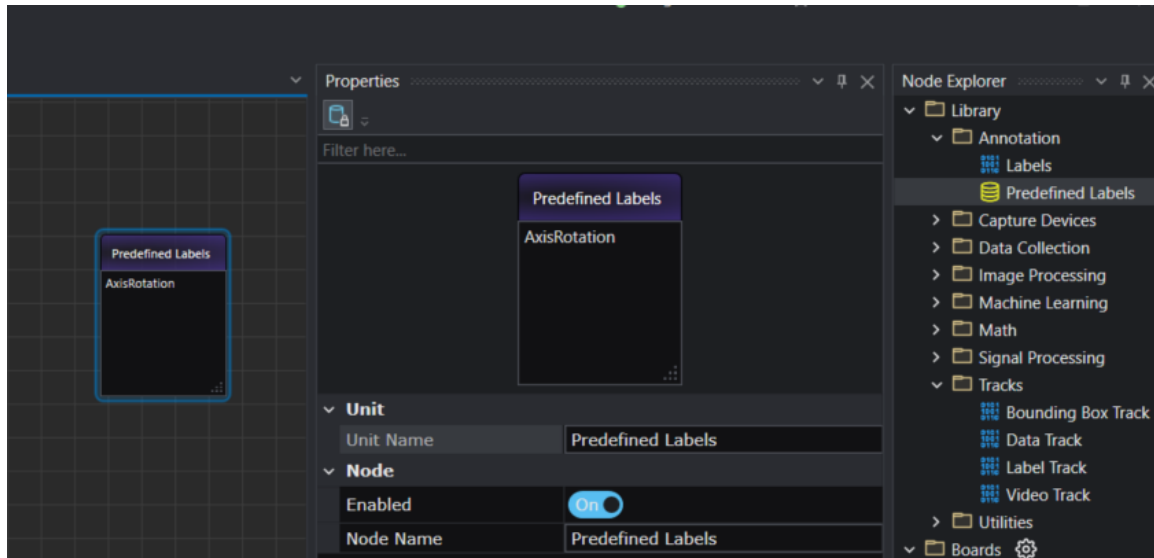
3: Setup the Predefined Labels

When collecting data for a Classification project, you should know in advance the different classes that data can belong to.

DEEPCRAFT™ Studio allows you to define the class names in the Graph UX interface with a specialized node. You will be then able to utilize the class names as labels and assign them to portions of the collected data (this operation can be done either in real-time during data collection or after the data is collected).

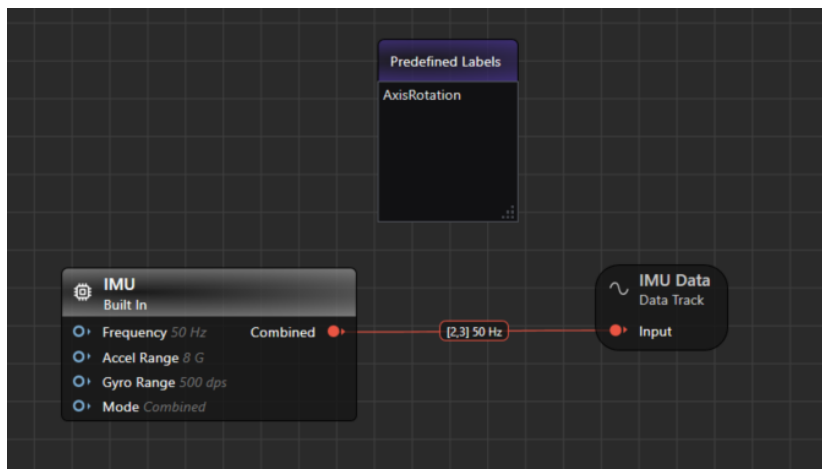
1. Navigate to the **Node Explorer** window, expand **Library> Annotation** and drag and drop the **Predefined Labels** unit onto the canvas. The **Predefined Labels** unit displays the default classes MyLabel1 and MyLabel2.
2. In the node itself, replace **MyLabel1** and **MyLabel2** with a new label related to the new gesture to be recognized. To perform this operation, simply click on the body of the **Predefined Labels** node and **delete with the Backspace button** the labels.

3. Type the label you want to add. Each label must lie on a new line:



Suggestion: use “AxisRotation” as new label, since it’s an easy and simple gesture that will require a small dataset to be recognized effectively.

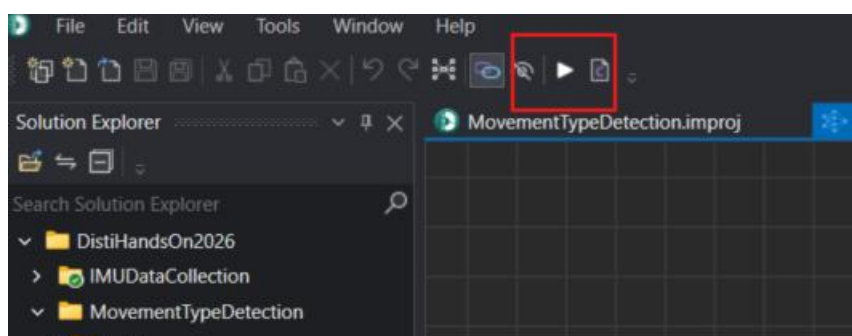
Your final Graph UX project should look like this:



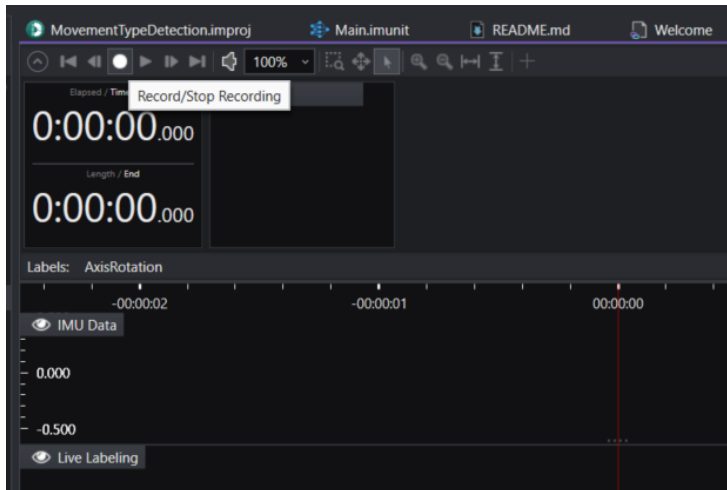
2.4 - Exercise 4: Real-Time Data Collection and Labeling

It is now time to collect data that will be used for training the model. The scope of this data is to provide **examples of a new gesture** for the model to learn from.

1. Navigate to the toolbar and click the **Start** button to open the session file (live.imsession). An empty session file opens displaying the pre-defined classes in the Labels bar:



- Click the **Record** button to start capturing the real-time data.



- When data collection starts, **hold the kit from the cable and rotate it along the cable axis**. Vary speed, angles and smoothness to increase data variety. Data should look like this:



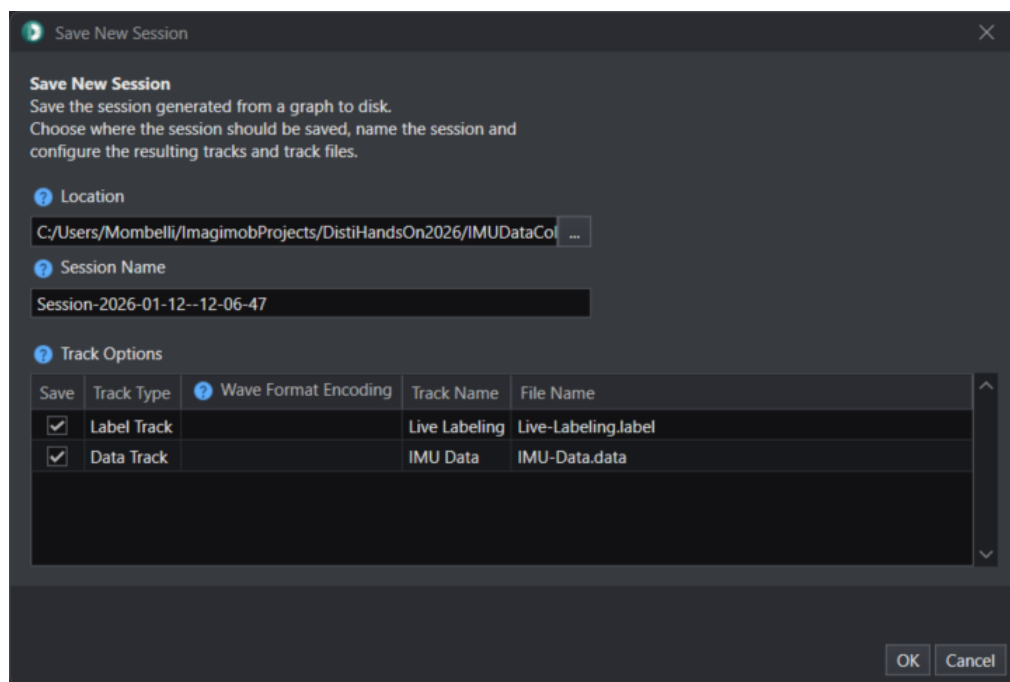
Try to collect a couple of minutes of meaningful data (rotation data) per session.

- Click the **Stop** button to stop collecting the data.
- Now it's time to label the data. Press the **AxisRotation** button in the session to enable the label.
- Highlight the rotation-related portions of data in the IMU Data track with the mouse pointer. Studio will assign the label to this portion of data, and it will be visible in the **Live Labeling** track as a blue rectangle:



The data is now labelled. You can repeat point 6 to label different portions of data in the same session.

7. Select **File > Save** to save the session file, data track and label track. The Save New Session window appears.
8. In **Location**, click the three dots and select the desired location to save the files.
Suggestion: use a folder in the current project.
9. In **Session Name**, enter the name of the session file.
10. Under **Track Options**, tick the tracks you want to save to disk. Save both .label and .data tracks:



11. Click **OK** to save the files.
12. **Repeat steps 1-11 to collect several samples.**
Suggestion: collect at least 10 samples, each with ~1min of labelled data.

You are now ready to use the collected data to train a Neural Network able to recognize the new AxisRotation movement.

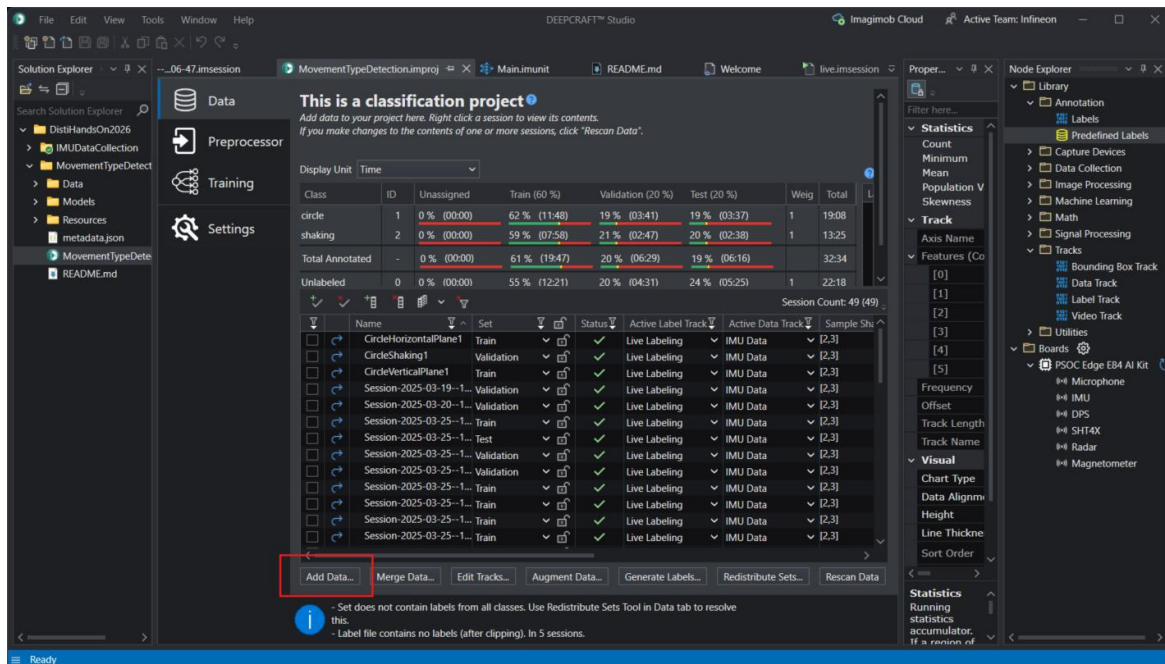
2.5 - Exercise 5: Adding data to the project and splitting for training/testing/validation

This exercise will guide you through the process of adding the data you collected to the **MovementTypeDetection** project data.

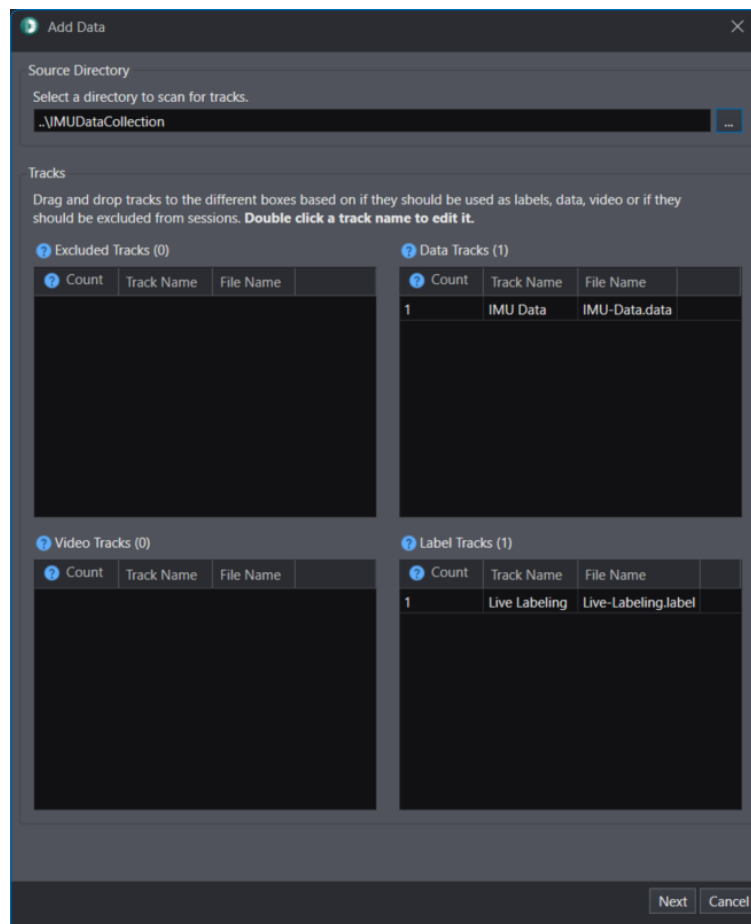
Step 1: Adding data to the project

To add the data to a project, follow the steps:

1. Navigate to **Workspace** and expand the project directory in which you want to add the data.
2. Open the MovementTypeDetection project file (*.improj) present in the project directory.



3. In **Data** tab, click the **Add Data** button at the bottom of the project file. The Add Data window appears.
4. In **Source Directory**, click the three dots and select the folder containing the data you collected in the previous exercise. The files are automatically sorted into the appropriate boxes based on their type:
 - Files with extensions .data, .csv and .wav are placed in the **Data Tracks** box
 - Files with extensions .mp4 are placed in the **Video Tracks** box
 - Files with extensions .label and .data are placed in the **Label Tracks** box
 - Drag and drop the file that you do not want to import in the **Excluded Tracks** box
5. If needed, you can drag and drop the tracks into the appropriate boxes based on whether they should be used as data or label and click **Next**. The summary window displays the details related to the new or existing and total sessions that will be generated after the import. It also displays any missing tracks.



6. Click Ok to import the data into the project.
7. After you have imported the data into the Studio, they will be listed in the Data tab as “unassigned” data sessions, meaning that each sample is not assigned to one between training, testing and validation sets. You must distribute the data into different datasets by following the next section.

Step 2: Distributing data in training/testing/validation sets

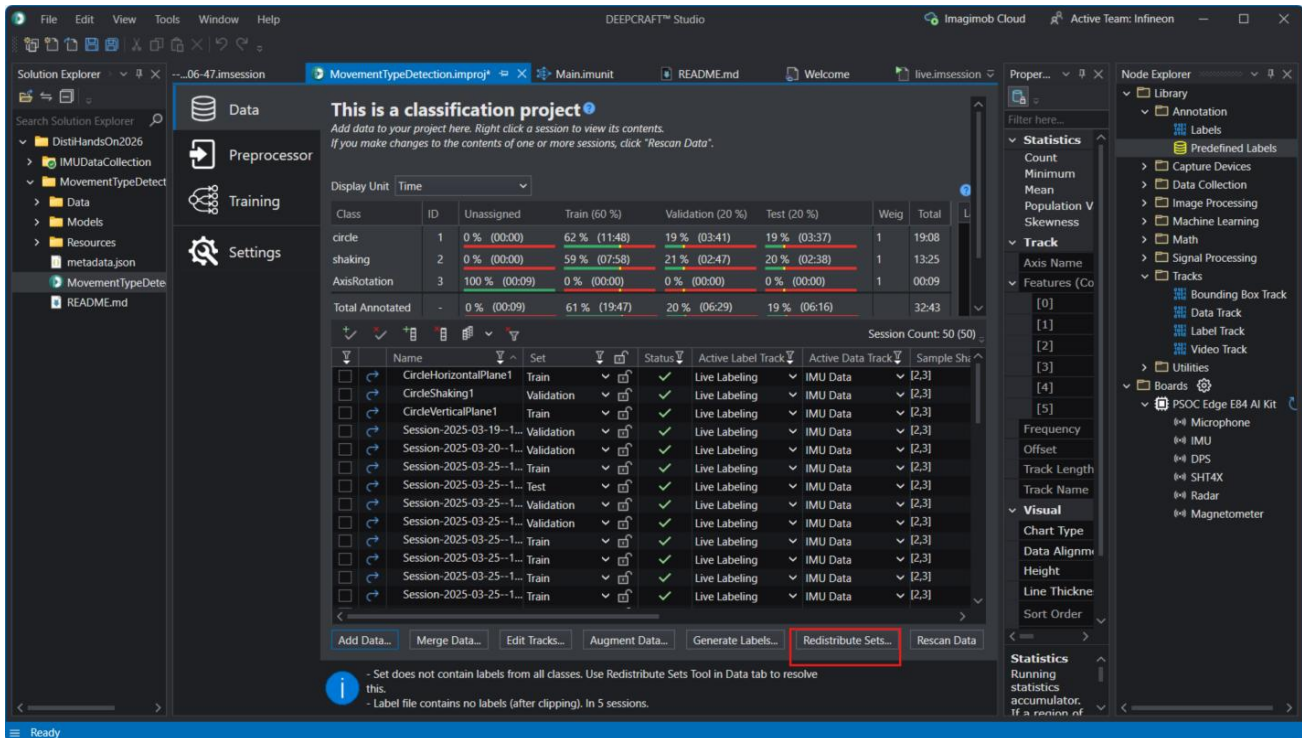
After you add data to the project, you need to distribute the data into different datasets. You can split the data into the following three sets:

Training Set	A subset of labeled data used to train a machine learning model, where the model learns to make predictions based on the input-output pairs provided.
Validation Set	A separate subset of labeled data used to evaluate and fine-tune the trained model's performance during the development phase.
Testing Set	A holdout subset of labeled data, not used during training or validation, that is used to assess the final, trained model's performance and generalizability to unseen data, providing an unbiased estimate of its accuracy and reliability.

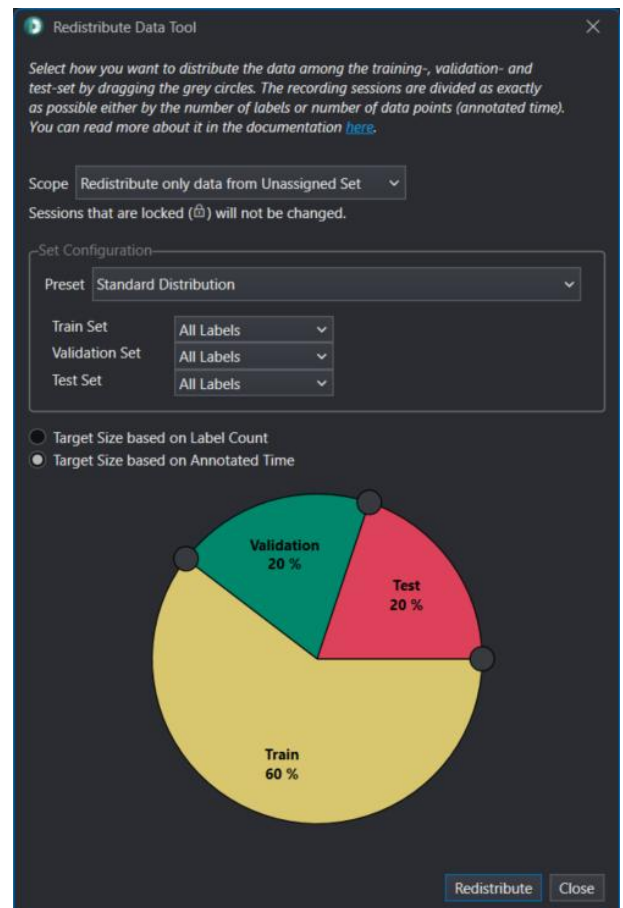
Suggestion: It is recommended to keep the training set significantly bigger than the validation and test sets. Some standard splits are (train/validation/test) 60/20/20 or 80/10/10. The more data you have collected, the smaller you can make your validation and test set target size.

Studio offers an automatic tool for distributing data. To distribute the data into the three sets, follow the steps:

1. Click on **Redistribute Sets** button in the Data tab of the project. The redistribute data tool window appears.



2. Set **Scope** as **Redistribute data from Unassigned sets** to only redistribute the new data.
3. Use the graphical tool to distribute the data into different datasets. The target size can either be based on Label Count or Annotated Time. Keep it as **Annotated Time** and select a proportion 60/20/20:
4. Select **Redistribute** and the data in the selected scope will be distributed among the three data sets.



2.6 - Exercise 6: Design the Data Preprocessor

(Solved exercise)

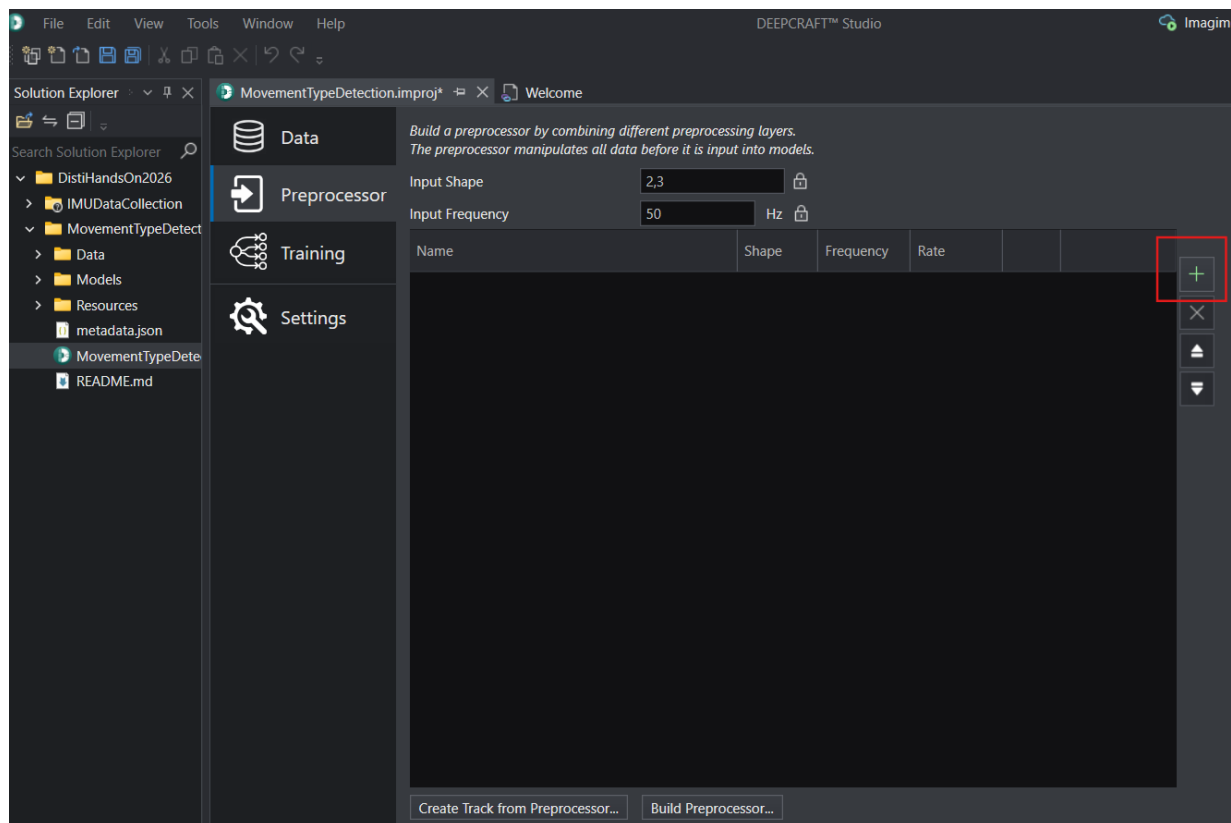
Data Preprocessing is the process of cleaning, transforming, and preparing raw data into a suitable format for machine learning model training. Data preprocessing can range from simple windowing of time-series data to complex transformations, according to the task.

Since this tutorial is based on a Studio Accelerator Project, the Data Preprocessor is already set up.

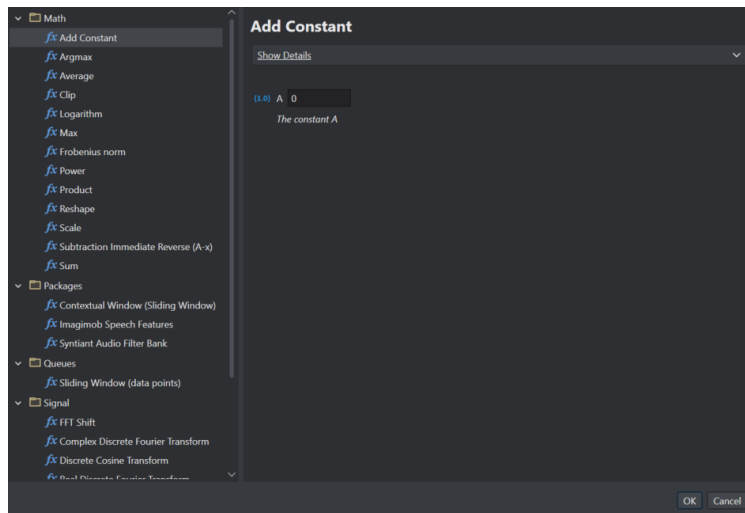
Configuring Preprocessor

To configure the preprocessor, follow the steps:

1. If not already open, navigate to your project directory and double-click the project file (.improj). The project file opens in a new tab.
2. Click **Preprocessor** tab on the left pane.
3. In **Input Shape**, the shape or dimensionality of the input data is provided to the machine learning model. This parameter is computed automatically by Studio. For example, an accelerometer sensor generates data in the format [x,y,z] so the shape of the input data is 3.
4. In **Input frequency**, the frequency of the input data is provided to the machine learning model. This parameter is computed automatically by Studio.
5. Click **+** (Add new layer) to add preprocessing layer to the machine learning model.

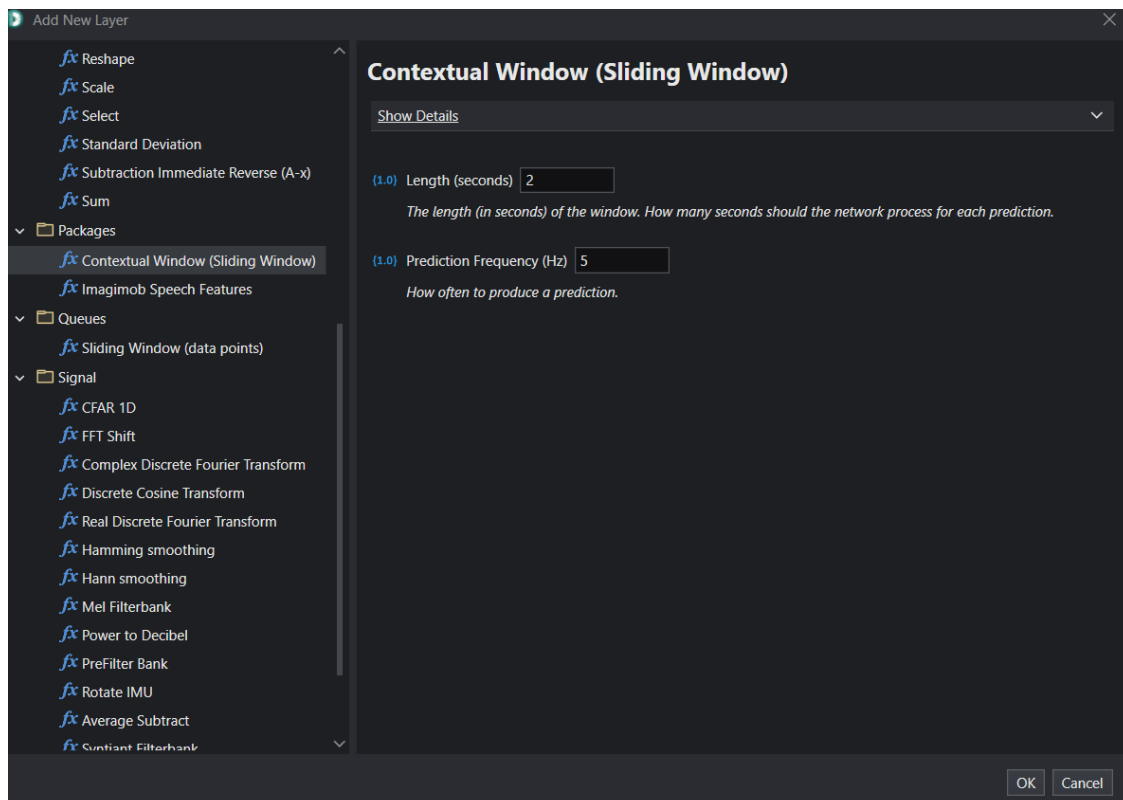


6. The Add New Layer window appears. It allows you to select a single preprocessing layer or group of preprocessing layers bind as a package applicable to your project. You can add multiple preprocessing layers or packages to the model.



Let's see how to add a preprocessing package **Contextual Window (sliding window)**. A contextual window layer groups several sensor readings together into a window and lets the AI model classify these readings together. It is the standard way to extract fixed-length data points from a continuous data stream to be used as input for ML classifiers.

7. Under **Packages**, select **Contextual Window (sliding window)**.
8. In **Length**, enter the length of the sliding window in seconds.
9. In **Prediction frequency**, specify the frequency at which windows should be extracted from the input signal. This parameter will affect how many samples you “move” the window forward.



10. Click **OK**.
11. The Contextual window layer is added as one of the preprocessing layers. Similarly, you can add multiple preprocessing layers as per your requirement.

DEEPCRAFT™ Studio allows users to view and evaluate the output of the preprocessing by generating a dedicated track in the timeline, called “preprocessor track”. Studio also allows users to export the preprocessor as Python code.

Please refer to DEEPCRAFT™ Studio documentation for information and step-by-step instructions.

2.7 - Exercise 7: Generating and Training Neural Networks

This exercise will cover various methods to **generate machine learning models**, how to **initiate and monitor the training** job, and the steps to **download the model** files from the Imagimob Cloud. Setup and instructions for running the training locally or on a company cloud can be found [here](#).

Step 1: Generating model

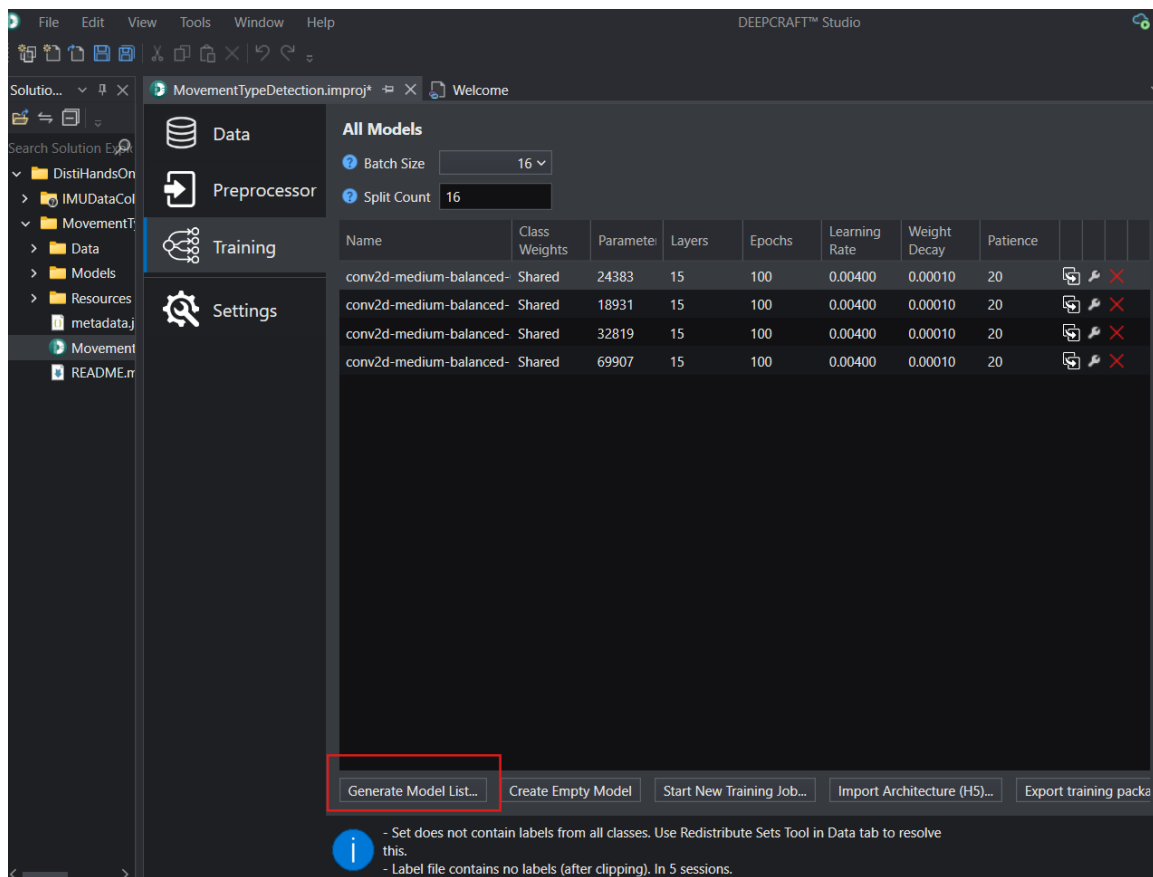
After you collect the data and design the preprocessor, the data is passed to the model for training. Before the model can be trained, you need to generate the model and define the layers of the model. DEEPCRAFT™ Studio allows generation of multiple different models to be trained and compared to find the best fitting model.

You can generate the model using any of the following ways in Studio:

- **Model Wizard:** You can generate multiple model architecture using the Model Wizard. The Auto ML tab in Wizard can be configured for different model families, classifiers, sizes, and learning rates. The Wizard generates models based on the configuration, speeding up the model development phase by prioritizing the main features of a model. The generated models have different layers and layer configurations, and produce slightly different results. The models generated by the Auto ML Wizard can be changed layer by layer, allowing more advanced users to fine-tune the models.
- **Empty model:** You can create a custom model from scratch, or clone one of the Auto ML models as a starting point for a new model. For each model in the list, you can change the training settings such as the number of epochs to be run and the learning rate used by entering the desired values. You can click the Model properties icon to set additional properties. If you select a model, you can view the different layers of that model. You can change the order of the layers, properties of each layer, and add or remove layers before training.
- **Import architecture:** You can import an existing Keras .h5 model in different modes into DEEPCRAFT™ Studio to generate a model.

The exercise will focus on generating models via the Model Wizard. To generate models, follow these steps:

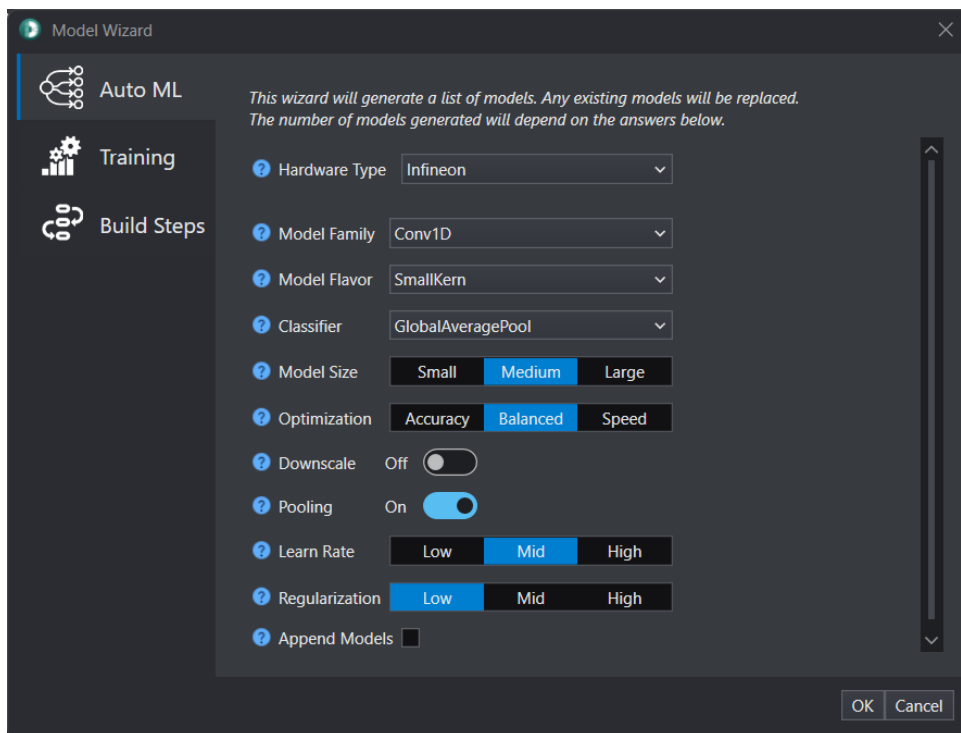
1. If not already open, navigate to your project directory and double-click the project file (.improj). The project file opens in a new tab.
2. Click the **Training** tab on the left pane.
3. Click **Generate Model List**. The Model Wizard window appears.



4. In **Auto ML** on the left pane, you are free to configure the following parameters:

- In **Hardware**, select the architecture that is applicable to your target device or solution.
- In **Model family**, select the model family as per your requirement:
 - **Conv1D** - Convolution-1D are lightweight models effective for time series data.
 - **Conv1DLSTM** - Convolution-1D and LSTM 1D-convolution are high accuracy models effective for one dimensional and two dimensional data.
 - **Conv2D** - Convolution-2D models are effective for two dimensional data, such as audio spectrograms.
- In **Model flavor**, select the model flavor as per your requirement:
 - **SmallKern** - Small kernels capture small patterns, increasing the model speed.
 - **LargeKern** - Large kernels capture large patterns, increasing the performance but slowing model speed.
- In **Classifier**, select the classifier type at the end of the model - GlobalAverage Pool, Hybrid and Dense.
- In **Model size**, select the model size as per your requirement.
- In **Optimization**, select the Optimization level of the model for accuracy and speed.
- In **Downscale**, keep it disabled. Downscaling increases the speed of the model by reducing the input size – which usually lowers model accuracy as a consequence.
- In **Pooling**, enable or disable the radio button as per your requirement. When enabled, this parameter reduces the dimensionality after convolutional layers. In most cases, pooling will increase the speed and reduce the memory consumption without affecting the model accuracy.
- In **Learn rate**, select the training speed of the model as low, mild and high. Setting the learn rate as high will speed up the training, however, this may cause suboptimal training results.
- In **Regularization**, select the desired option as per your requirement. This parameter reduces over-fitting to the training data.

- In **Append models**, enable the check box to keep old models and append new models to the list. This parameter is applicable when all models have the same class count.



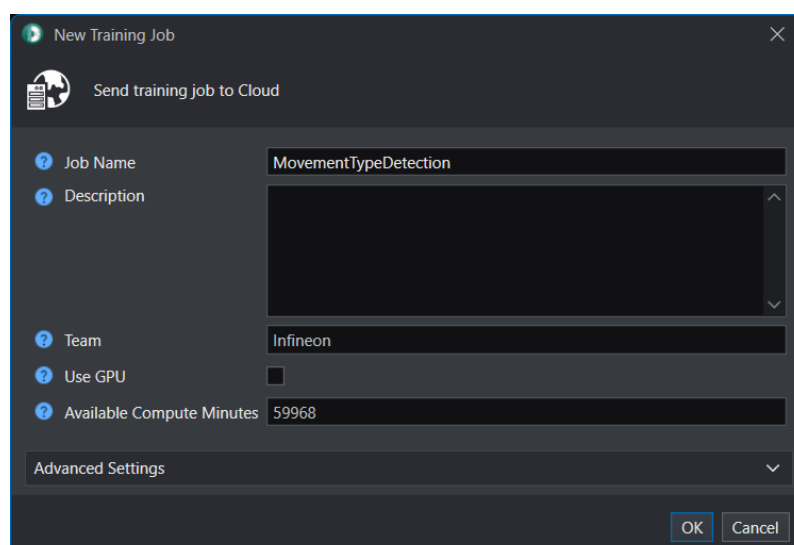
5. Click **Ok** to generate a list of models.

Step 2: Starting and Tracking Training Jobs

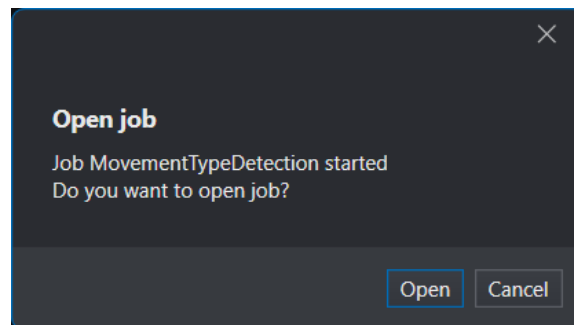
Once the model is generated, you can initiate and monitor the training job either by sending the training package to the **Imagimob Cloud** or running training on the **Local Machine**. Since local training takes some [setup](#) we will use the cloud. At this stage, you have the option to use either GPUs or the default CPUs for training. After starting the job, all data and model information will be uploaded to the Imagimob Cloud for the actual training.

To start the training job, follow the steps below:

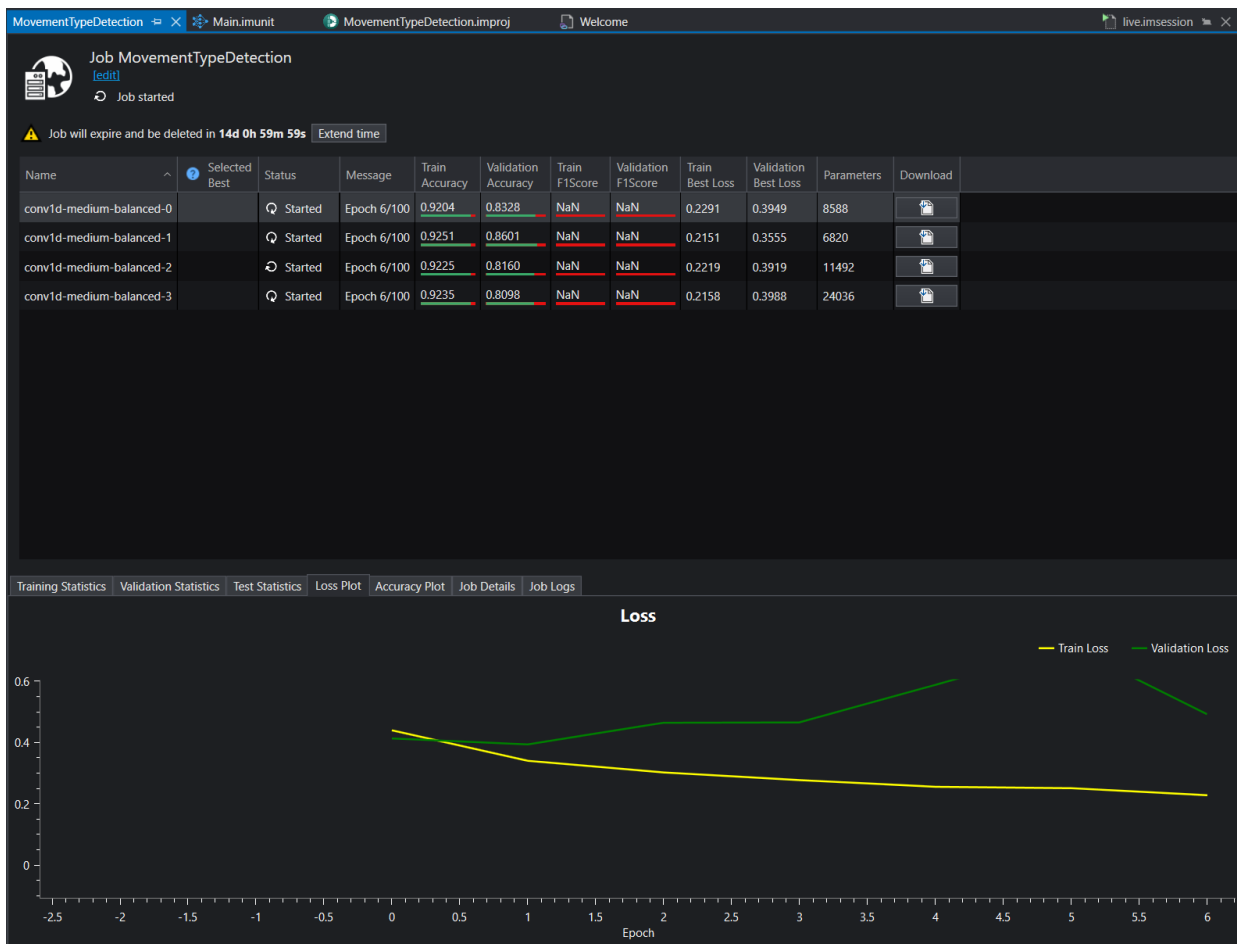
1. Click **Start New Training Job**. The **New Training Job** window appears.



1. In **Job Name** and **Description**, enter the name of the job and a description respectively.
2. In **Use GPU**, enable the GPU powered training.
3. In **Available Compute Minutes**, your total compute minutes available are displayed.
4. Click **OK** to begin the training. After data is uploaded, a popup window appears indicating that the job has been started.



5. Click **Open** to view the training progress in a new tab.



You can now wait for your models to finish the training.

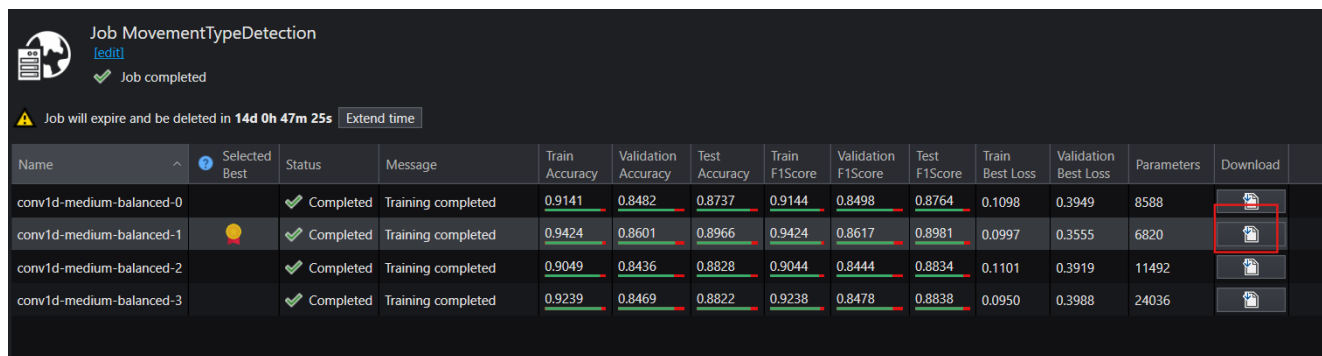
At the end of the training, take some time to explore the Statistics and model evaluation metrics.

Step 3: Downloading Model Files

After the model is trained in the Infineon Cloud, the next step is to download the trained model files. You can also import the model predictions as tracks into the sessions containing the original data. This provides a much more detailed view for evaluating the performance of the model.

To download the model files, follow these steps:

1. Scroll right to the **Download** column and click the **download icon** to download the model files for the related model. DEEPCRAFT™ Studio automatically assigns the “Selected Best” badge to the model with the **best validation loss** (i.e. the lowest)



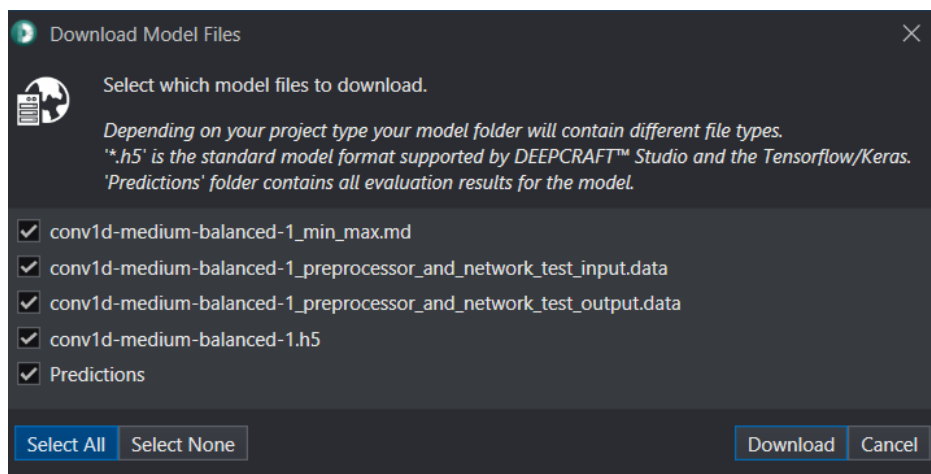
Job MovementTypeDetection
[Edit](#)
Job completed

Job will expire and be deleted in 14d 0h 47m 25s [Extend time](#)

Name	Selected Best	Status	Message	Train Accuracy	Validation Accuracy	Test Accuracy	Train F1Score	Validation F1Score	Test F1Score	Train Best Loss	Validation Best Loss	Parameters	Download
conv1d-medium-balanced-0		Completed	Training completed	0.9141	0.8482	0.8737	0.9144	0.8498	0.8764	0.1098	0.3949	8588	
conv1d-medium-balanced-1		Completed	Training completed	0.9424	0.8601	0.8966	0.9424	0.8617	0.8981	0.0997	0.3555	6820	
conv1d-medium-balanced-2		Completed	Training completed	0.9049	0.8436	0.8828	0.9044	0.8444	0.8834	0.1101	0.3919	11492	
conv1d-medium-balanced-3		Completed	Training completed	0.9239	0.8469	0.8822	0.9238	0.8478	0.8838	0.0950	0.3988	24036	

2. The Download model files window appears with the option to download:
 - trained model (.h5 file)
 - model predictions for evaluating the trained model
 - test input and output datasets to be used in the ‘Generate Data Compare Test’Select the all the files and click **Download**.

3. Save the model in the “Model” folder of the DEEPCRAFT™ Studio MovementTypeDetection folder



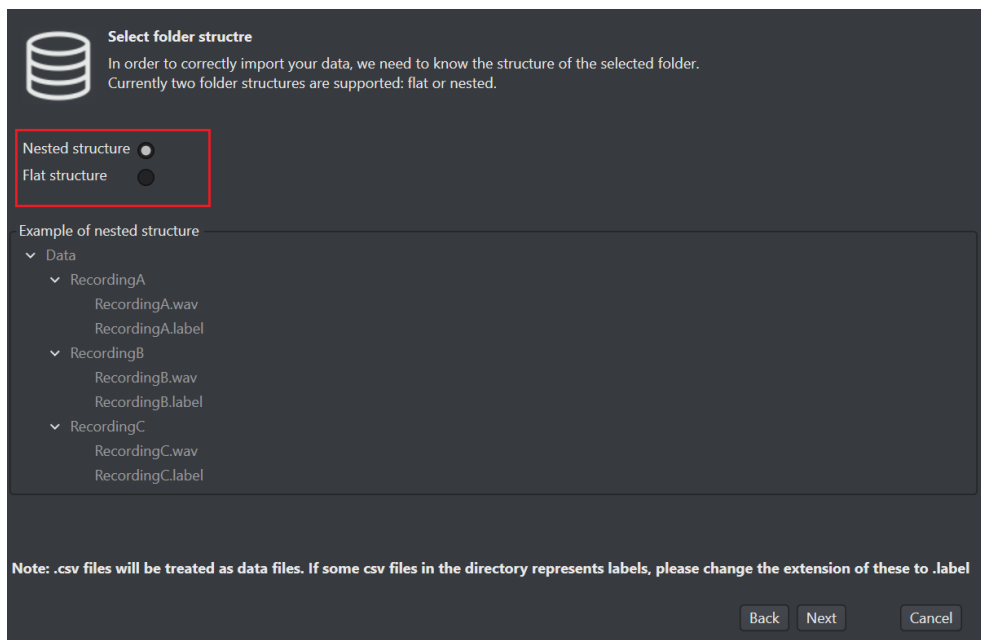
Step 4 (optional): Importing predictions into your project

DEEPCRAFT™ Studio allows a side-by-side view of the performance of a model on data by merging the model predictions as tracks into the sessions containing the original data.

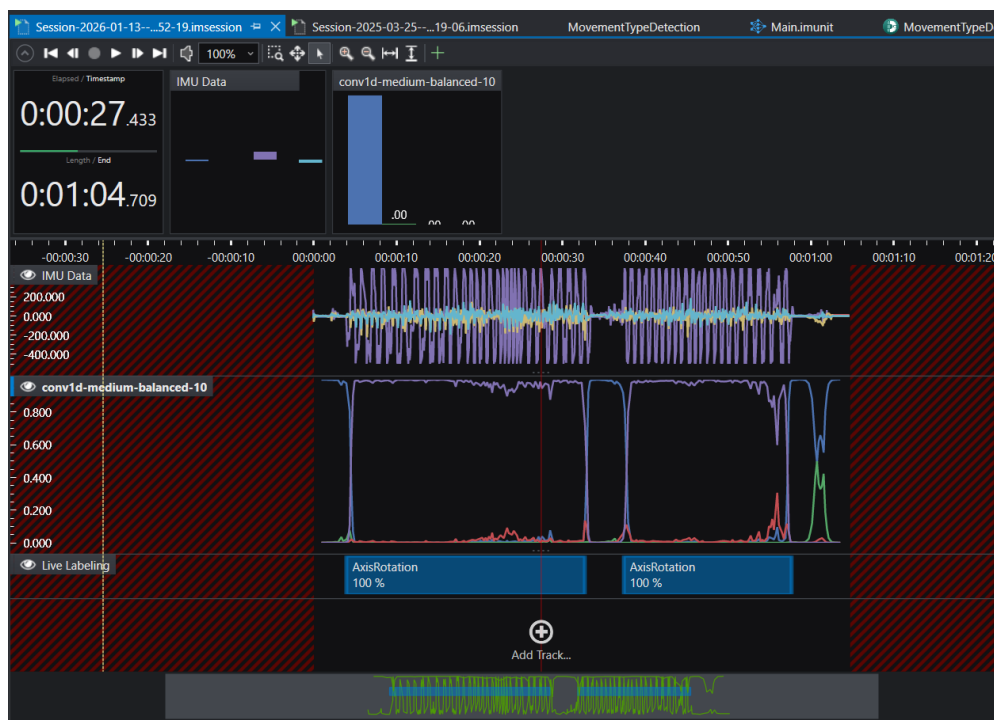
To import predictions, follow these steps:

1. Navigate to the project directory and open the project (.improj) file.
2. In the **Data** tab, click the **Merge** button and browse to select the **Predictions** folder, which is inside the model folder you downloaded before.

3. Select the **Nested structure** radio button and click **Next**. Predictions from the selected folder will be merged as tracks into the corresponding local sessions.



4. In the next window, select or deselect what you want to import and click **OK** to complete the merge. Take care to check that the column related to your new model is completely selected.
5. You can now visualize the output of the neural network side-by-side with the input data by double clicking on data sessions in the Data tab.



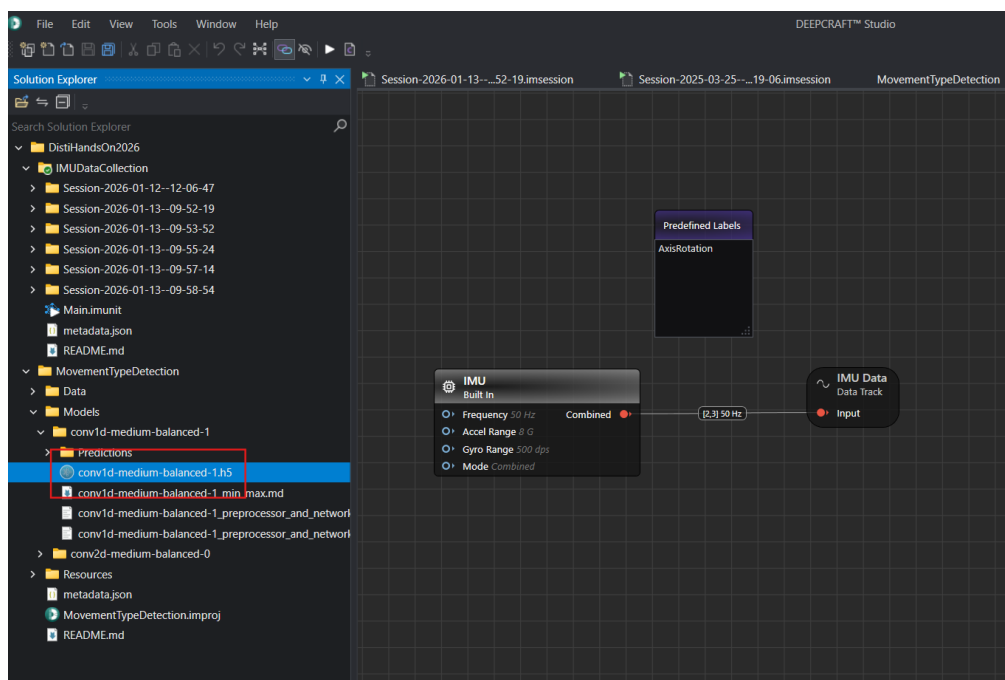
2.8 - Exercise 8: Real-Time Model Evaluation using Graph UX

Model evaluation is a crucial step in building a machine learning model. It helps in analyzing the model performance by using multiple statistics and metrics. However, a model performing ideally in the evaluation phase may not perform as expected in real world. To address this, the model should be thoroughly evaluated in real-time before deploying. Graph UX supports real-time model evaluation functionality which helps in analyzing and monitoring the model predictions before deploying a model to production. It also ensures that the model generates accurate predictions on real-time data.

This exercise will leverage the Graph UX project we used for data collection to evaluate the model in real-time.

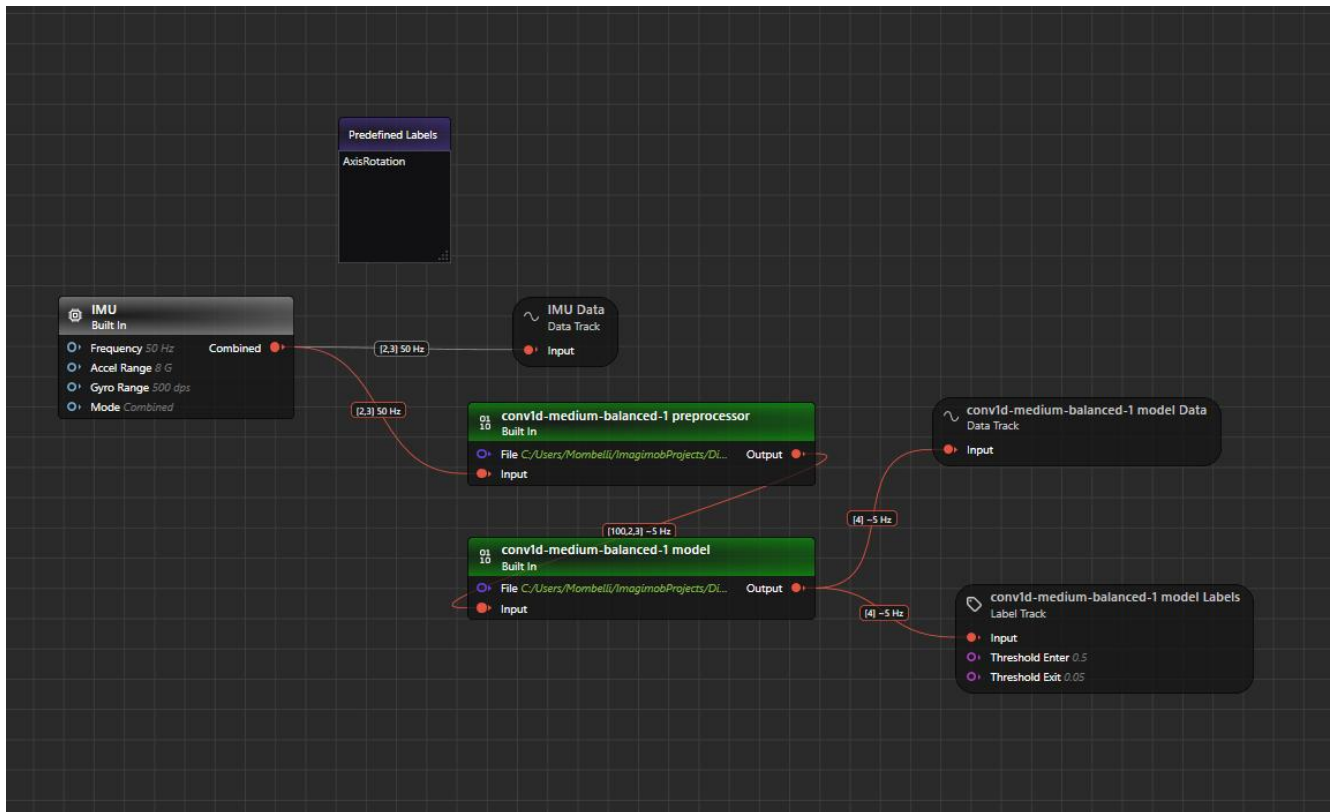
To create an evaluation graph for evaluating a model, follow these steps:

1. Navigate to the IMUDataCollection folder from the Solution Explorer and open the Main.imunit file to open the Graph UX Project.
2. Expand the Models folder of the **MovementTypeDetection** project and locate the **.h5 file of your downloaded model**.



3. **Drag and drop** the .h5 file in the canvas. The model will appear as a double node including the preprocessor and the neural network itself, already connected.
4. Click on the red icon in the **IMU** sensor node and drag over to the red Input icon in the Preprocessor node. This creates a connection between the two nodes.
5. Add two tracks: a Data Track and a Label Track.
6. Connect the output of the Model node to both tracks. They will be needed to visualize the raw model output and the predictions.

Your final graph should look similar to this:



- Click on the Start button on the toolbar and run the graph. Click on record to start recording data from the IMU sensor. When you perform the AxisRotation gesture, you will see the neural network prediction change accordingly:

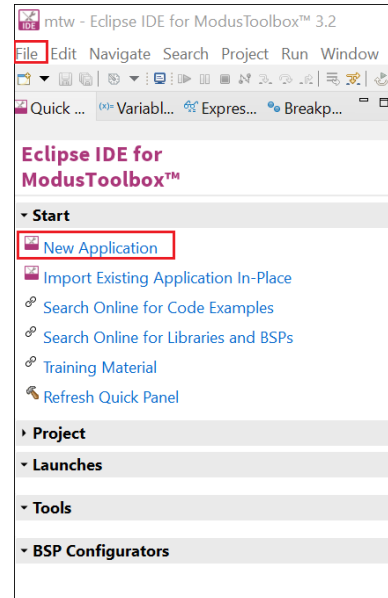


2.9 - Exercise 9: Deploying the model on PSoC™ Edge AI Kit

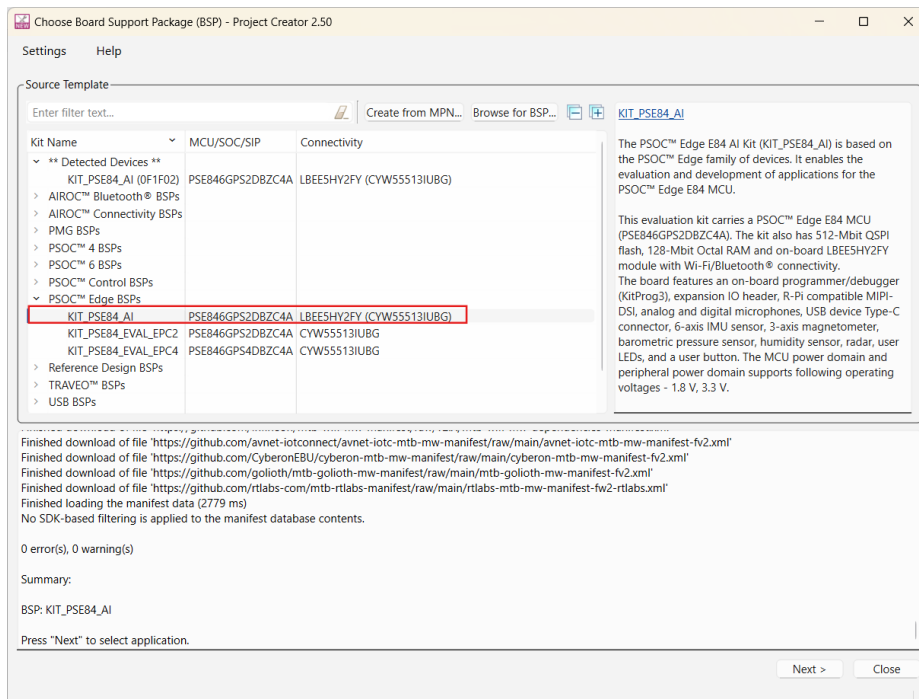
This last task will guide you through deploying the trained Neural Network on PSoC™ Edge. To do so, we will use a code example provided within ModusToolbox™. You need to create the required machine learning project for your board within ModusToolbox™, convert your neural network in C code and then replace the model.c and model.h files in the project with the **model.c** and **model.h** files for your model following by building the project. Follow the step-by-step instructions to learn the deployment process.

Step 1: Create the ModusToolbox™ Project

1. Open **ModusToolbox™ > Eclipse IDE for ModusToolbox™** from the **ModusToolbox™ Dashboard** software. The Eclipse IDE for ModusToolbox™ window appears.
2. Browse and select the workspace directory for your project and click **Launch** to open the ModusToolbox™ workspace.
Suggestion: Use a workspace with a short name (ex: mtb_ai) and store it close to your C:\ root folder.
3. Select **New Application** from the Quick Panel or navigate to **File> New> Modus Toolbox™ Application** to open the Project Creator Tool.

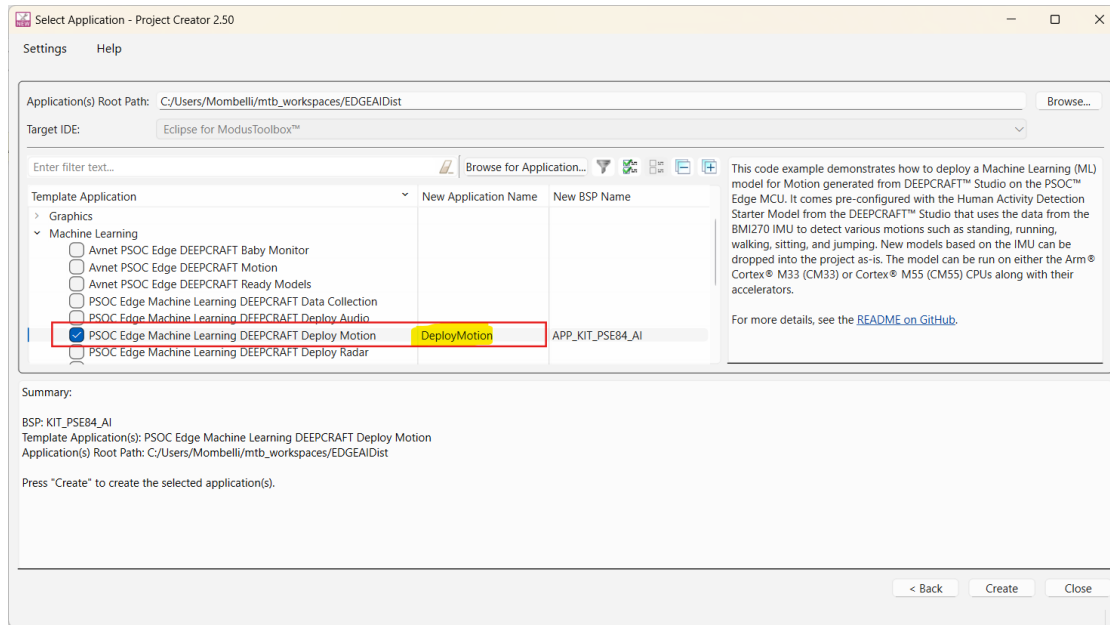


4. Expand the Kit Name PSoC™ Edge BSPs from the list and select the KIT_PSE84_AI as BSP for your board and click **Next**. The Select Application window appears.



5. In **Template Application**, expand **Machine Learning** and select the **PSoC_Edge_Machine_Learning_DEEPCRAFT_Deploy_Motion** code example.

Rename it with a shorter name (ex: DeployMotion) and click **Create**:



Wait for the project creation to be completed. You are now ready to go back to DEEPCRAFT™ Studio for the next step.

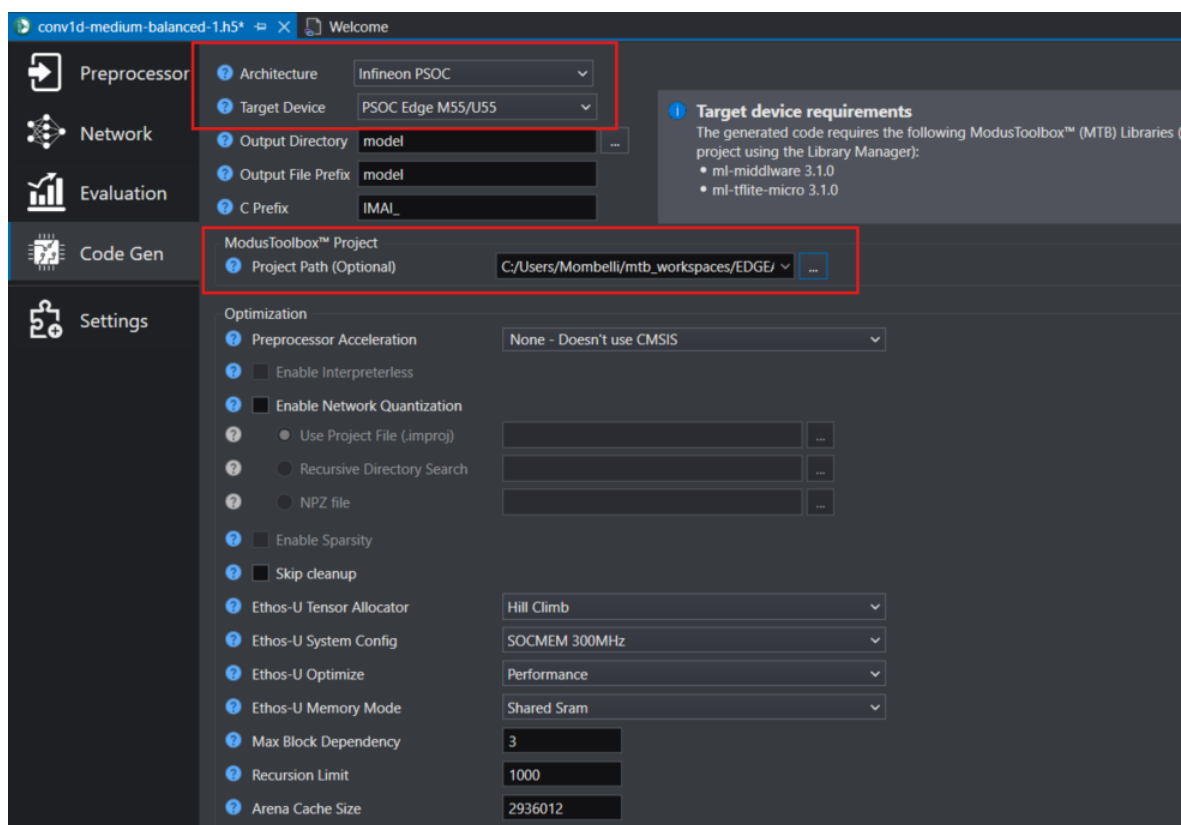
Step 2: Converting the neural network and flashing the code

To bridge DEEPCRAFT™ Studio and ModusToolbox™, we provide a bidirectional integration that supports an end-to-end workflow for PSoC™ Edge boards.

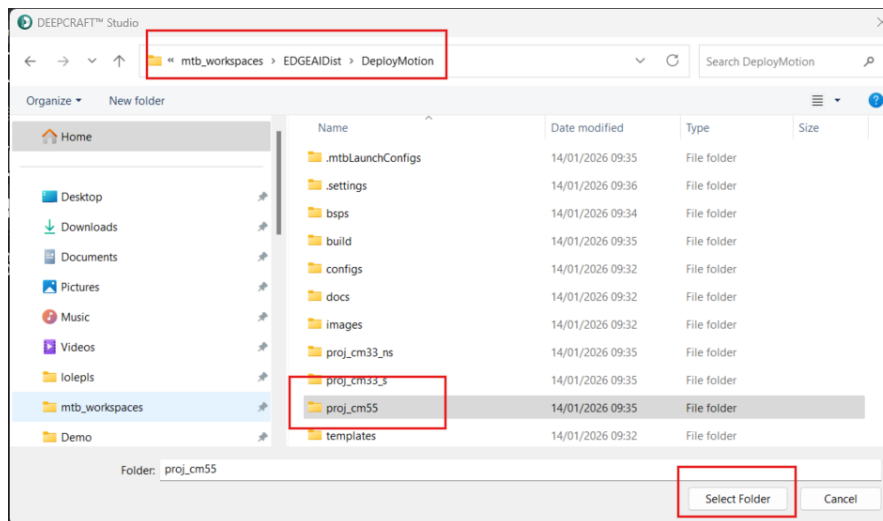
After you train a model with Studio, you can directly target a ModusToolbox™ Project to export the converted neural network.

To generate the code, follow these steps:

1. Navigate to the **MovementTypeDetection** project directory, locate and open the trained model file (*.h5) by **double clicking** on it.
2. Click the **Code Gen** tab on the left pane and configure the following parameters:



- In **Architecture**, select **Infineon PSOC** as the architecture type.
- In **Target Device**, select PSOC Edge M55/U55.
- In **Project Path**, select the path to the root directory of the ModusToolbox™ project we created in previous step, where you want to save the generated code. Take care to select the **proj_cm55** folder inside the DeployMotion project:



3. Click the Generate Code button and wait for the conversion process to complete.

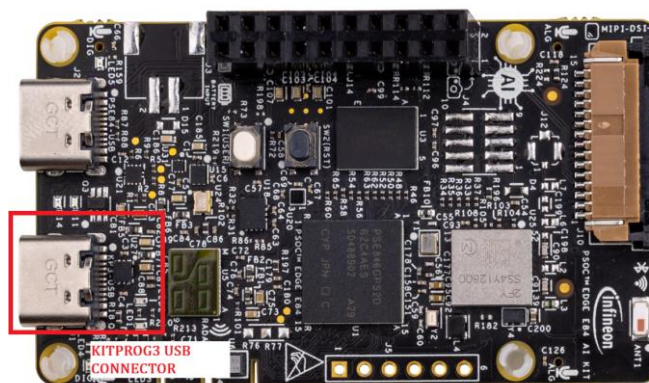
This will export your neural network as-is (in floating point precision) in a format suitable for the execution on Infineon MCUs.

Step 3: Flash and Run the project

The last step involves using ModusToolbox™ to flash and run the project.

Open ModusToolbox™ and follow these steps:

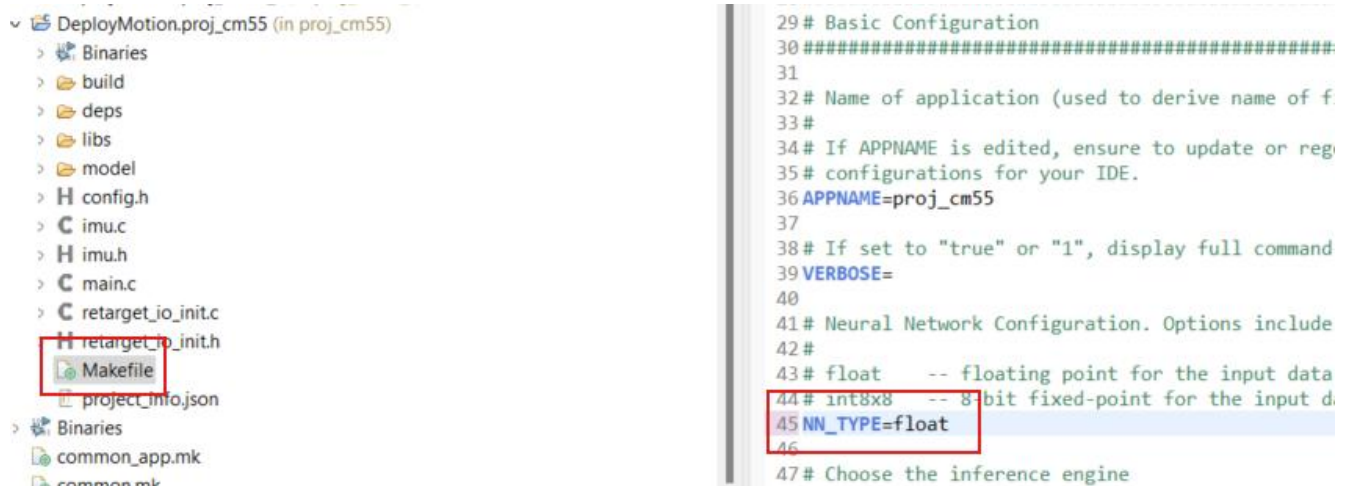
1. Connect the **KitProg3 USB connector (J1)** port on the board to the PC using the USB cable.



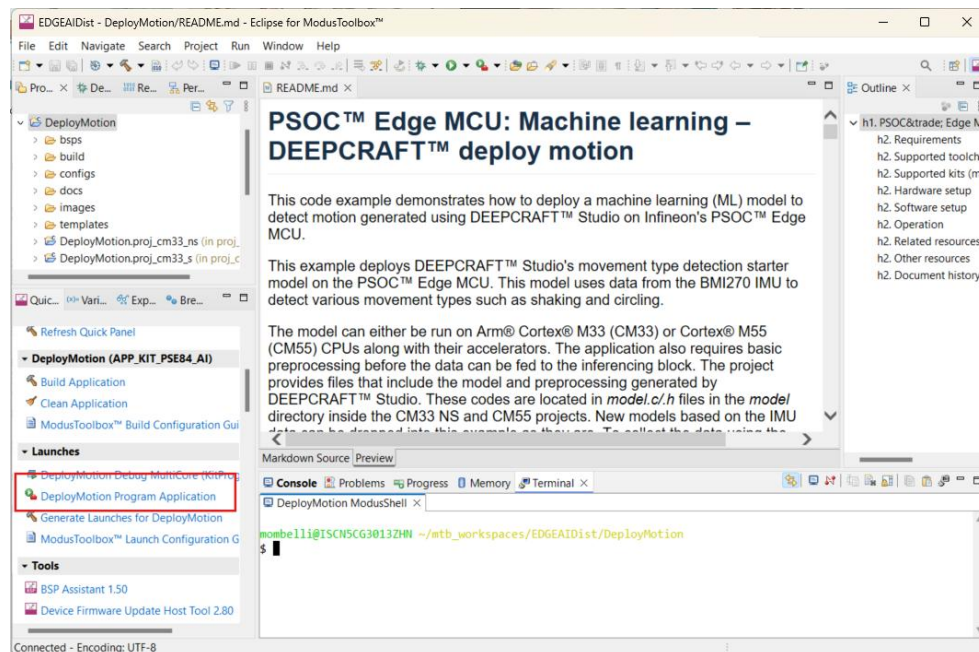
2. Locate the **Makefile** in the *proj_cm55 folder of your workspace and open it. Locate the **NN_TYPE** variable and set it to **float**. Save the Makefile.

This variable is needed to configure the inference engine according to the quantization level of the neural network exported via DEEPCRAFT™ Studio. Floating point models will be executed by M55 core only, without acceleration.

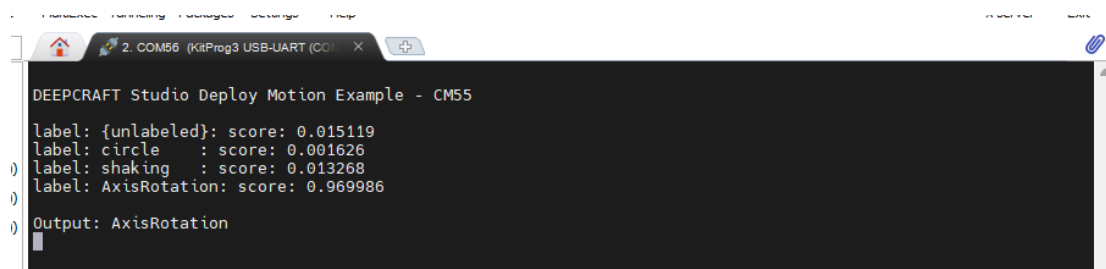
Refer to the next Step to see how to deploy a quantized model on U55 accelerator.



3. In **Quick Panel> Launches**, click the program. The code is deployed on the board.



4. Open a Serial Terminal window, connect to the PSOC™ Edge COM Port with BaudRate=115200 and perform the gestures. The output of the neural network is displayed in real time as confidence values:

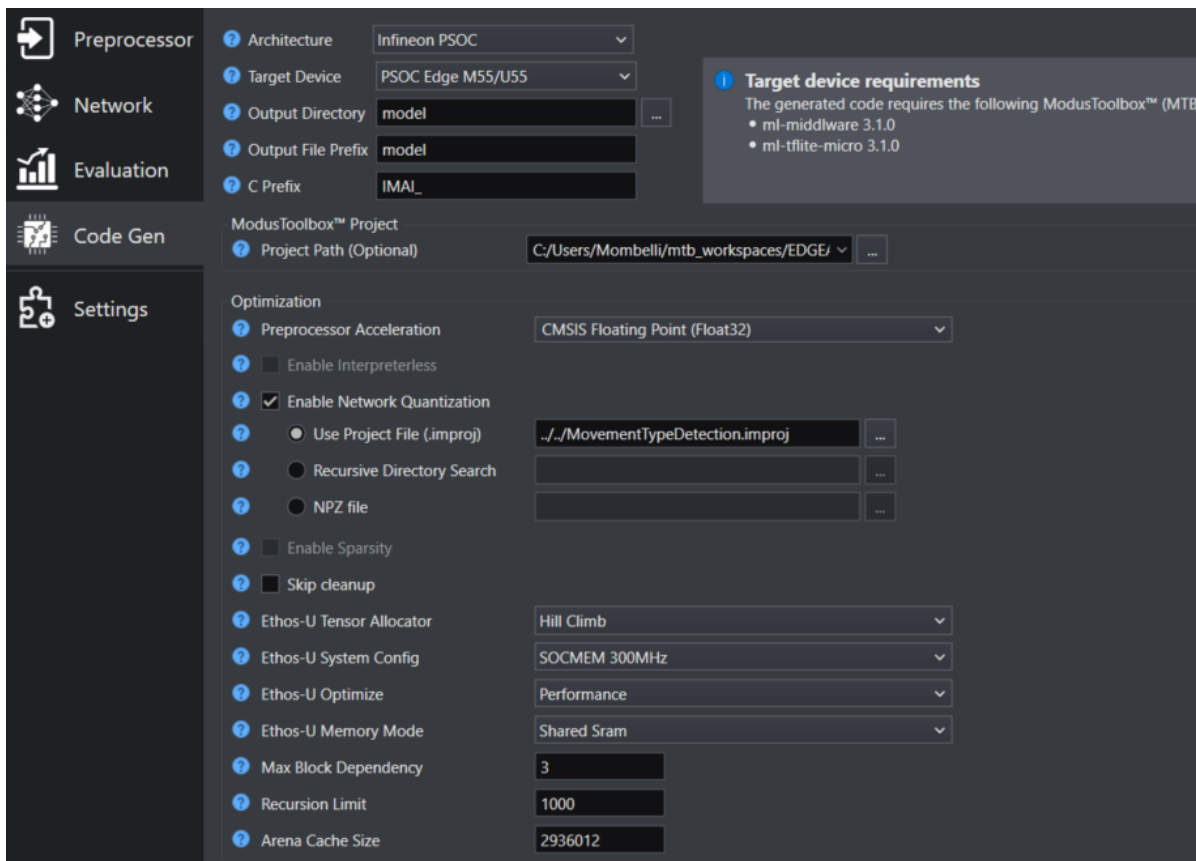


Optional Step: Running the model on PSoC™ Edge U55 Neural Network Accelerator

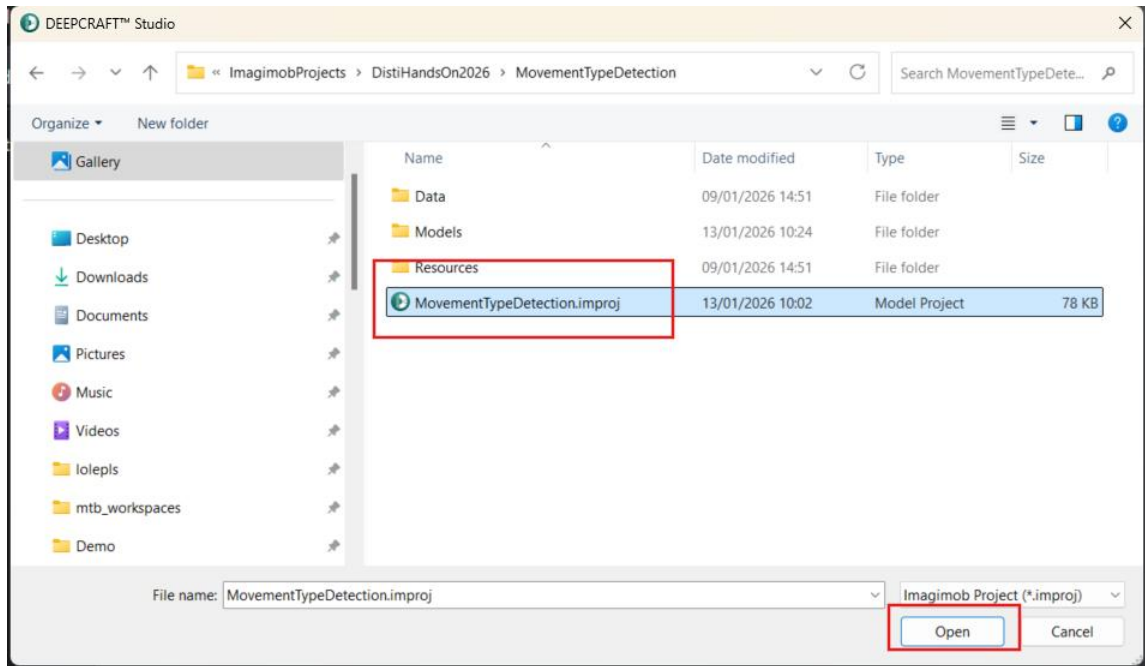
To target the PSoC™ Edge Neural Network accelerator, it is mandatory to export the neural network in **quantized** format. The procedure is exactly the same as the previous step: only some settings of the export window will change.

To run the model leveraging the accelerator, follow the steps below.

1. Navigate to MovementTypeDetection project directory, locate and open the trained model file (*.h5) by **double clicking** on it.
2. Click the **Code Gen** tab on the left pane and configure the following parameters:



- In **Architecture**, select **Infineon PSoC** as the architecture type.
- In **Target Device**, select **PSoC Edge M55/U55**.
- In **Project Path**, select the path to the root directory of the ModusToolbox™ project we created in previous step, where you want to save the generated code. Take care to select the **proj_cm55** folder inside the DeployMotion project (see previous exercise).
- In **Preprocessor Acceleration**, set **CMSIS Floating Point (Float32)**.
- Tick **Enable Network Quantization** to quantize the model.
- Make sure to enable the radio button **Use Project File (.improj)** and select the .improj file that you used to train the model (**MovementTypeDetection.improj**):



3. Click the Generate Code button and wait for the conversion process to be completed.
4. Repeat “Step 3: Flash and Run the project” as per previous instructions, but this time **take care to set the NN_TYPE variable in the makefile to “int8x8”**.

The firmware detects the quantization level of the neural network during compilation thanks to the Makefile settings, and loads the library to run the model on the U55 accelerator.

3. Hands On – Object Detection

This chapter includes exercises that walk through the process of creating a camera-based object detection model in DEEPCRAFT™ Studio and deploying it on the PSOC™ Edge E84 AI Evaluation Kit. It is recommended to have completed Chapter 2

Pre-requisites:

- DEEPCRAFT™ Studio version >= 5.8
- ModusToolbox™ version >= 3.6
- PSOC™ Edge E84 AI Evaluation Kit (PSOC™ Edge E84 EVK can also be used)
- USB-C Cable
- Waveshare 4.3" Raspberry Pi 800x480 display
- [OV7675 0.3MP DVP Camera module](#) or [Logitech c920 HD PRO Webcam](#) + USB-C -> USB dock

Make sure you are connected to the network and are logged in to your DEEPCRAFT™ Studio account.

Object Detection models in DEEPCRAFT™ Studio leverage Ultralytics YOLO models for training the specific model you require. These are free of charge for DEEPCRAFT™ Studio users, even for commercialization. License information can be found [here](#). This means that instead of training a model completely from scratch, state-of-the-art models are used as a starting point. You can then choose to retrain the entire model, or to fine-tune only the final layers of the model. The exact model as baseline contains 80 classes, you can read more about it [here](#). YOLO models don't use the common confidence value threshold of 0.5 to create a classification, instead they have a very low threshold and provide an abundance of bounding boxes which are intended to be filtered.

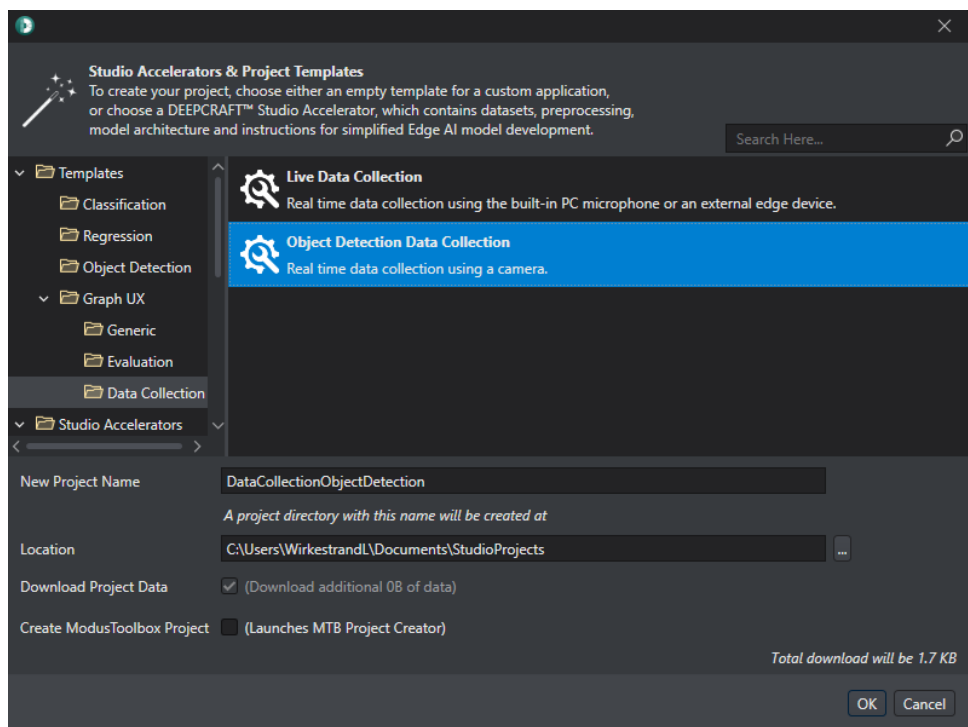
This section will cover the entire process of retraining a YOLO model, leveraging a DEEPCRAFT™ Studio Accelerator including: Data Collection, Model Definition, Model Training, Postprocessing (filtering), Model Evaluation, Code Generation and Model Deployment.

3.1 - Exercise 1: Creating an Object Detection Data Collection Project

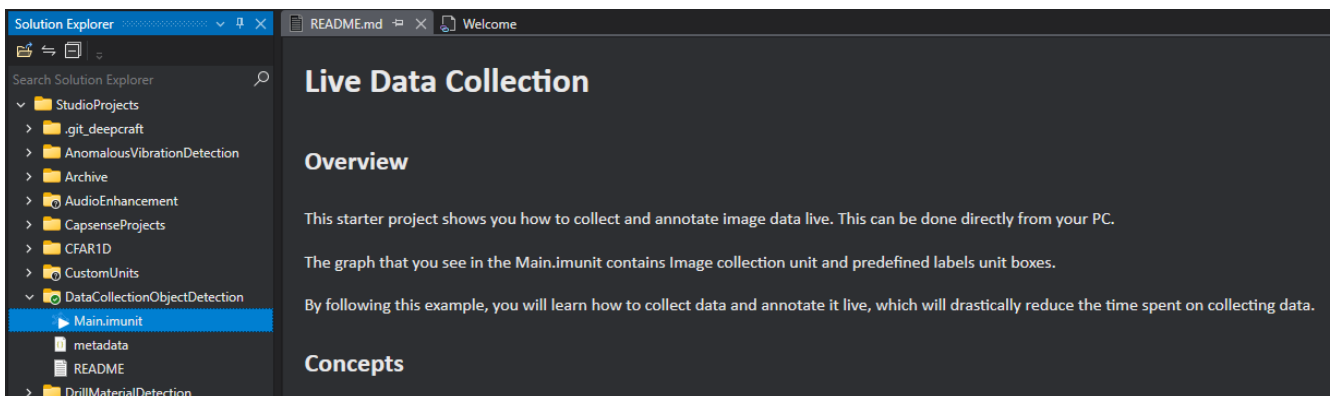
The goal of this exercise is to create the project needed for collecting the object detection data that will be used in the project. For this step any camera connected to the computer will be usable, either a built-in laptop or external USB camera will work, or even a mobile phone camera via a third-party like Camo Studio. Best machine learning practice is to collect data with a camera similar to the one which will be used in the final product.

For this specific training, the Object Detection Data Collection project will be used to collect data.

1. Open DEEPCRAFT™ Studio.
2. Click **New Project** in the welcome screen. The welcome screen appears when you open Studio for the first time.
OR
Go to **File** and select **New Project**. The New Project window appears.
3. Expand **Graph UX** and then **Data Collection**. Select Object Detection Data Collection.

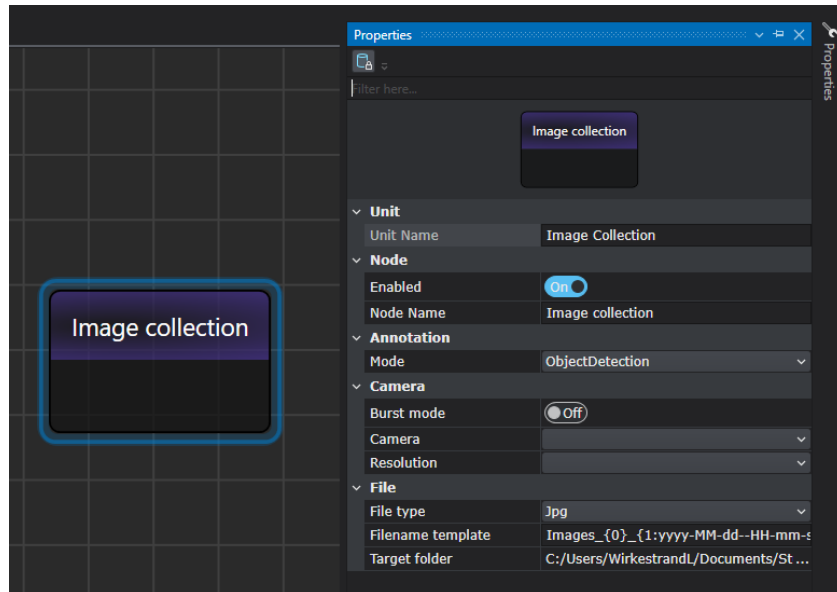


4. In **New Project Name**, edit the project name, if needed. By default, the default project name is displayed.
5. In **Location**, specify the location where you want to create the workspace and the project directory.
6. Click OK to download the data and create the Studio Accelerator project.
7. The **DataCollectionObjectDetection** project folder will appear in the Solution Explorer tab of Studio.



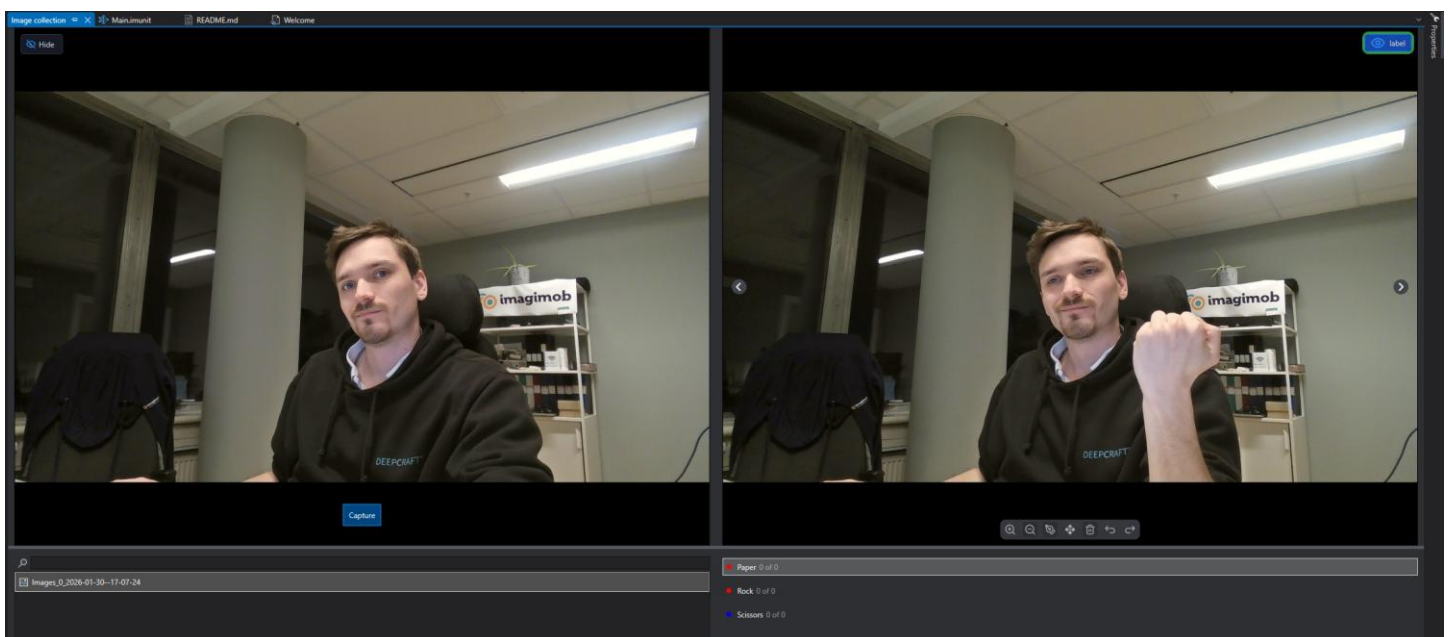
8. Double click on the **main.imunit** file to open the DEEPCRAFT™ Studio project. The main DEEPCRAFT™ Studio Project window will open.

3.2 - Exercise 2: Collecting Data with the Data Collection Object Detection project



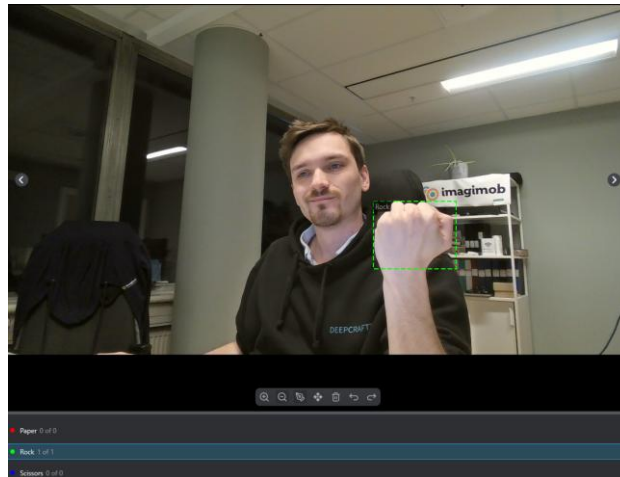
Now you have the project which is used to collect and label the image data. It consists of only two nodes: *Image Collection* and *Predefined Labels*. Whatever labels you add to the Predefined Labels node will be the ones utilized for labeling in the next steps. The Image collection node is where you change the settings of the data collection, do this by left clicking on it and using the *Properties* tab.

1. Click on the Predefined Labels node and remove the default labels. Replace them with Rock, Paper and Scissors. **Note: These labels are case sensitive to work with future steps.**
2. Click on the Image Collection node and open the *Properties* tab.
3. Select the Camera you want to use. Best practice is to use the same camera as the one you intend to use when deployed.
4. Select a resolution. DEEPCRAFT™ Studio will automatically resize images of different sizes in the training step, once again best practice is to be as similar as possible to the final use-case.
5. Select *Target Folder* as DataCollectionObjectDetection folder, or wherever you wish to save the data.
6. Double click on the Image collection node, to open the Image Collection tab.



The Image Collection tab contains two panes. The leftmost one has the current view of the camera. This can be hidden and showed via a toggle in the top left. Whenever the *Capture* button in the bottom is pressed, the image is saved and displayed in the right pane. Underneath the *Capture* button is a list of all captured images which can be navigated between using the arrow keys. Underneath the captured image on the right are the labels from the Predefined Labels node.

1. Capture an image where you're posing with either Rock, Paper or Scissors.
2. Select the appropriate label from the bottom right list and draw a bounding box around the shape by left clicking and dragging around the object.



3. Repeat steps 1-2 a couple more times with different shapes. Take the opportunity to play around. You can go back and edit the Predefined Labels node if you wish – however make sure you don't accidentally import any images taken with other labels into the rest of the exercises. Images can be deleted from the list directly or through the Solution Explorer.
4. **[Optional]** Open the Properties pane again. This is accessible from the Image Collection tab too, without needing to go back to the main.imunit. Here there is an option for enabling *Burst mode* which will change the capture settings to take one photo per second, instead of whenever the Capture button is pressed. This is useful for quickly capturing a large dataset. Pressing the button again will stop the capturing.
5. Once you have captured and labeled some images, save and close the Image Collection tab.

DEEPCRAFT™ Studio organizes each captured image within a nested folder structure. Each folder includes the image, a session file (.imsession), and a label file (.labelxml) containing bounding box information. If the image is not labeled, the folder will not contain a label file.

TIPS: To read more details, go to: <https://developer.imagimob.com/deepcraft-studio/data-preparation/data-collection/collect-data-without-kit/collect-image-data-using-graph-ux#tools-for-labeling-the-images>

3.3 - Exercise 3: Creating a Rock, Paper, Scissors Studio Accelerator

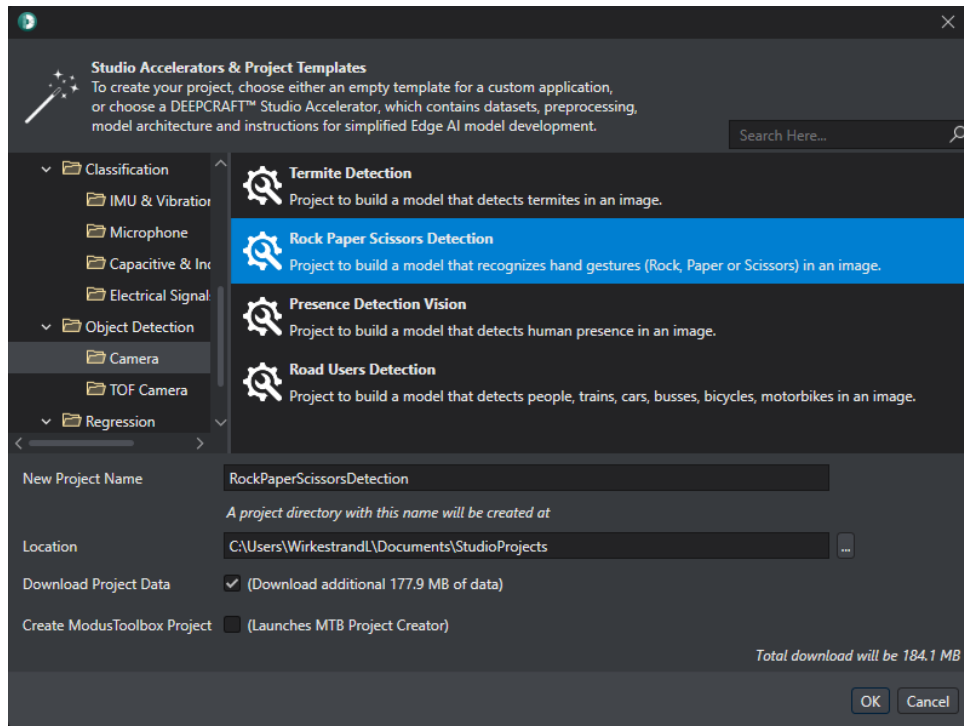
Similarly to Chapter 2, we will create a DEEPCRAFT™ Studio Accelerator. To learn what that is, refer back to Chapter 2.2.

1. Click **New Project** in the welcome screen. The welcome screen appears when you open DEEPCRAFT™ Studio for the first time.

OR

Go to **File** and select **New Project**. The New Project window appears.

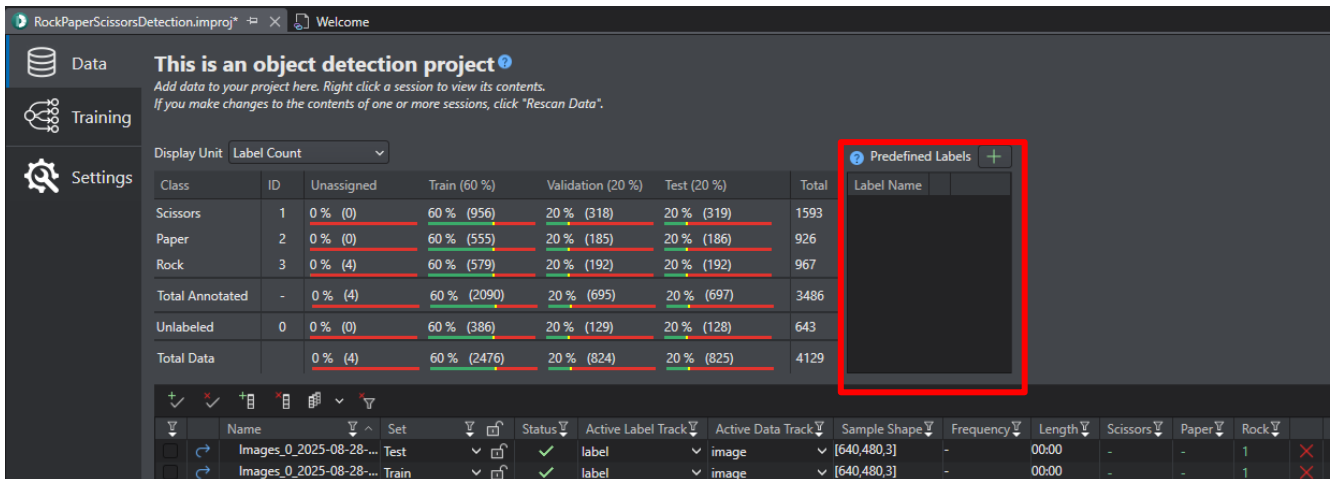
2. Expand **Studio Accelerators** and depending on the machine learning problem you want to solve expand either **Classification** or **Object Detection**. Select **Camera**, browse the available projects, and choose the **Rock Paper Scissors Studio Accelerator** as your starting point by clicking on it.



3. In **New Project Name**, edit the project name, if needed. By default, the default project name is displayed.
4. In **Location**, specify the location where you want to create the workspace and the project directory.
5. Select **Download project data checkbox** to download the project data to the workspace selected earlier. Leave **Create ModusToolbox™ Project** unticked. This will be done later separately.
6. Click **OK** to download the data and create the Studio Accelerator project.
7. The **RockPaperScissorsDetection** project folder will appear in the Solution Explorer tab of DEEPCRAFT™ Studio.
8. Double click on the **RockPaperScissorsDetection.improj** file to open the DEEPCRAFT™ Studio project.
The main DEEPCRAFT™ Studio Project window will open.

(Optional) Add new labels to existing data

The data collected in Exercise 3.2 can be added by following the steps in section 3.4. However, if you wish to modify the existing data in the Studio Accelerator, this can be done in a similar manner as exercise 3.2. Instead of a node, use the **Predefined Labels** section in order to add more labels, before opening the images in the .improj file.



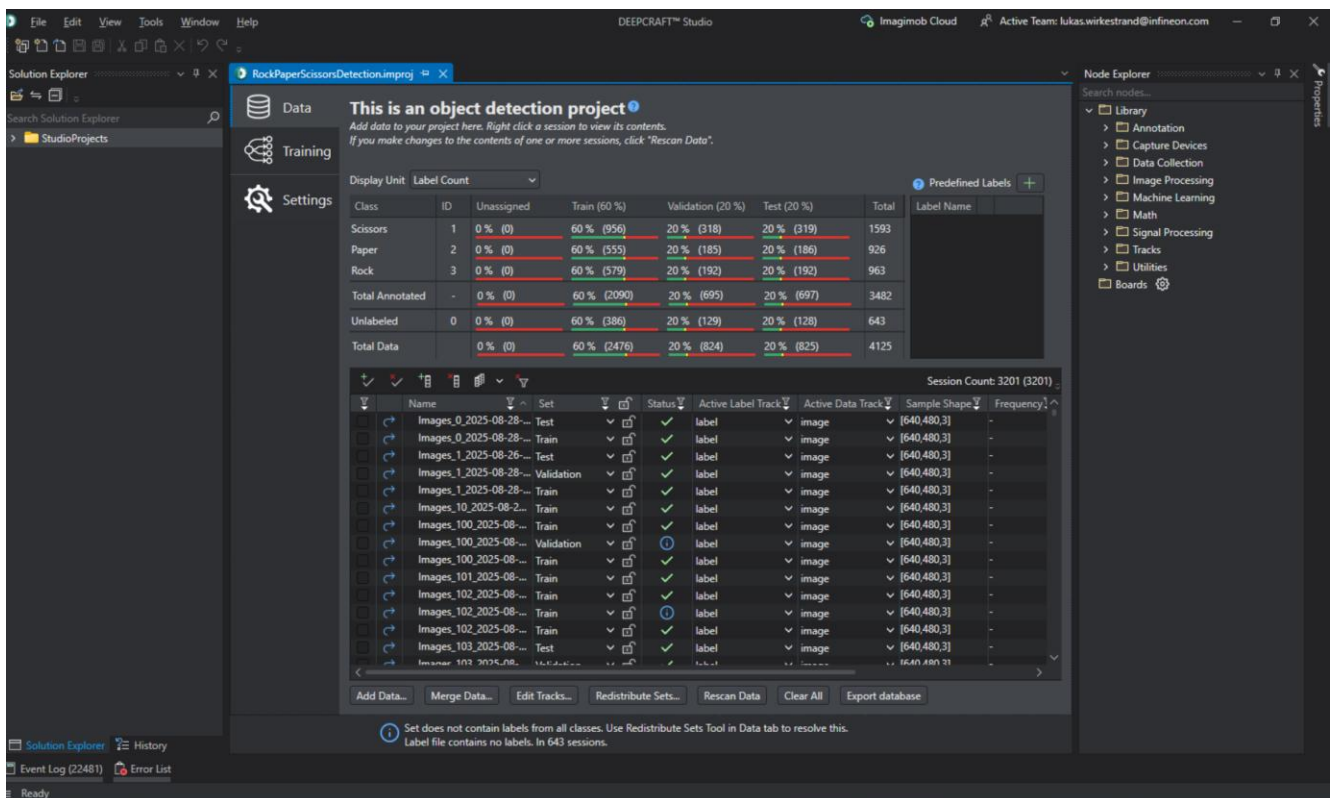
3.4 - Exercise 4: Adding Data to the Rock, Paper, Scissors Detection Project

Similarly to exercise 2, this exercise will guide you through the process of adding the data you collected to the RockPaperScissorsDetection project data.

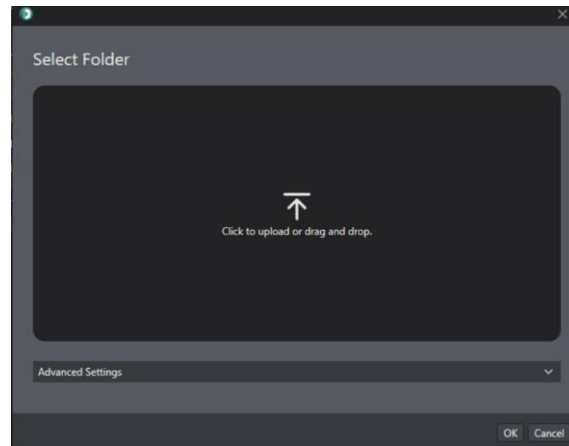
Step 1: Adding data to the project

To add the data to a project, follow the steps:

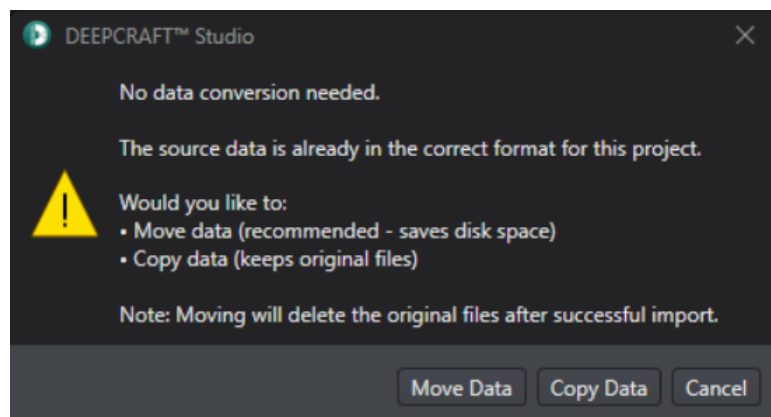
1. Navigate to **Workspace** and expand the project directory in which you want to add the data.
2. Open the RockPaperScissorsDetection project file (*.improj) present in the project directory.



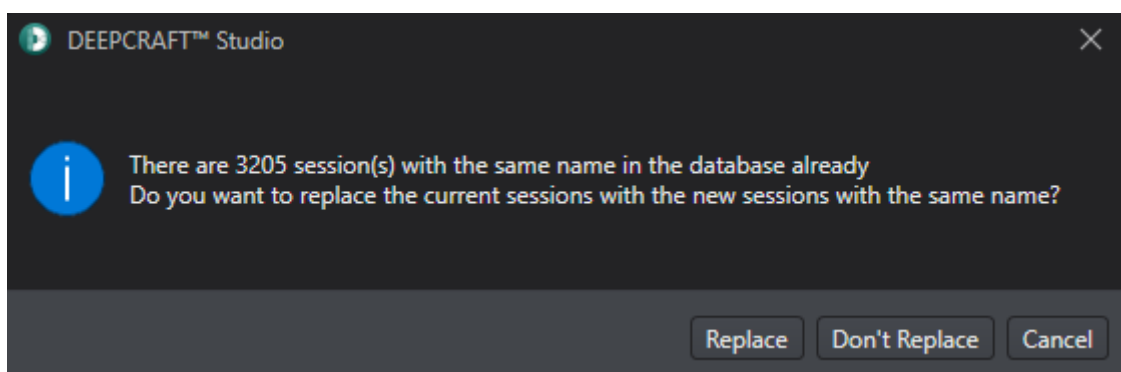
3. In the **Data** tab, click the **Add Data** button at the bottom of the project file. The Add Data window appears.
4. Click the central area and navigate to the folder containing your data, if you followed the process it should be the DataCollectionObjectDetection folder. Click on OK.



5. The **Data Import** window will open and parse your images and labels. If you are adding data not collected from DEEPCRAFT™ Studio it supports the YOLO, COCO, PascalVOC and LabelXml formats. Click Next.
6. If the files are already on your computer, you will get a pop-up asking if you wish to copy the data or move it. Select 'Copy Data'.



7. A check of duplicate sessions is performed, click 'Next'.
8. Choose not to replace sessions with identical names. Now your collected data is added to the project.



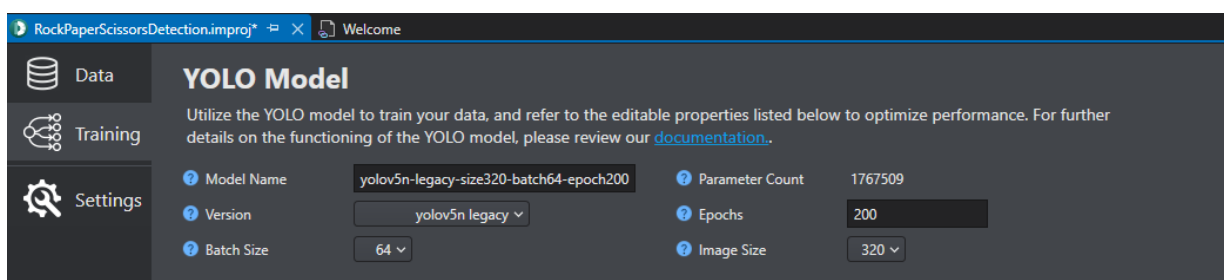
Step 2: Distributing data in training/testing/validation sets

Redistribute sets as in section 2.5. Note that data files of images are displayed identically to time series data in the *.improj Data tab. Double-clicking on any existing image will open a **Sessions** tab to view them.

3.5 - Exercise 5: Defining the Model

(Solved exercise)

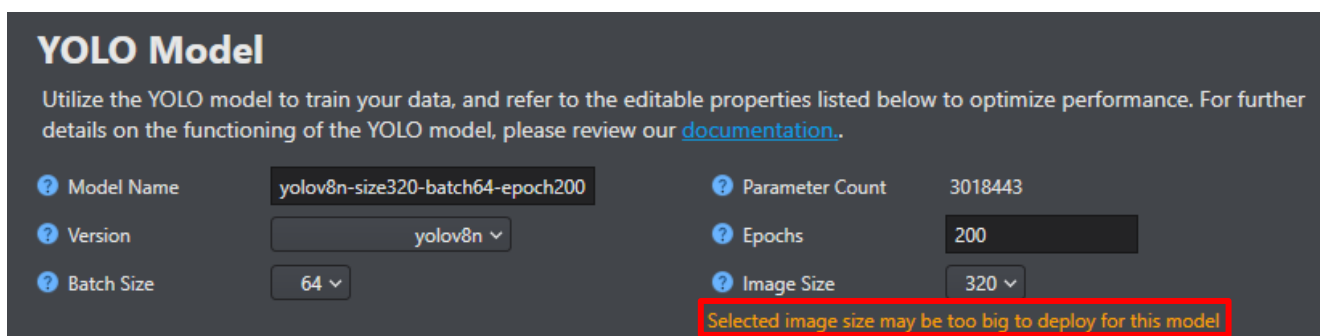
Firstly, as opposed to Chapter 2 there is no preprocessing necessary for images. Instead there will be data augmentation which will be described in the next Exercise. Navigate to the **Training** tab of the .improj file.



Here you select the parameters and hyperparameters of the YOLO model. The most important aspect to keep in mind is the version and image size selected. Newer versions of YOLO require more memory, as do models on larger image sizes. Epochs and Batch Size are the same as in Exercise 2.

Note: You do not have to worry about the size of the images in your dataset, these will automatically be resized and still used in training your model.

For the PSOC™ Edge E84 AI Evaluation Kit, a limit of 320x320 image size is what fits inside the internal memory of the board when using yolov5n legacy. As such, this is the default expectation that will be used later, and what this exercise will use. An adaptive warning will appear when the settings are too large to deploy in the internal memory.



Keep the settings as-is, using yolov5n-legacy and an image size of 320.

We will cover Data Augmentation in Section 3.6. For now, open the Advanced Options:

Advanced Options

Training

Optimizer

SGD

Patience

0

Learning Rate

Confidence Threshold

Initial

0.01

Validation

0.001

Final

0.01

Momentum

0.937

Evaluation

Weight Decay

0.0005

Confidence Threshold

Iou Threshold

0.7

Prediction

0.25

Image Weights

☐

Bring your own model

Model Mode

Evaluation

In the Advanced Options, the following can be configured:

- **Optimizers** are the algorithms that the training loop utilizes to adjust the model's parameters every epoch in order to minimize the loss function. YOLO's loss function is a complex composite loss function, that you can read more about [here](#). If you select 'Auto' the training algorithm will automatically optimize and select an optimizer.
- **Patience** is the number of epochs the model will continue training after no improvement is measured.
- **Learning Rate** is a measure of how much the model updates it's parameters every iteration, moving towards the minimum of the loss function. In Object Detection and Image Classification a Learning Rate is commonly increased every epoch in order to provide the training a way to get out of local minimums of the loss function or escape saddle points. The **Initial** and **Final** values of the learning rate allow increasing, or decreasing **Learning Rates** over time. Different optimizers have different recommended Learning Rates, SGD has 0.01 and ADAM has 0.001. If you select Auto as the Optimizer, Learning Rate is chosen automatically upon starting training.
- **Momentum** incorporates a "velocity" to the training, meaning that results of past epochs are used in later ones. As a result, the training is smoother with less drastic changes based on outliers.
- **Confidence Threshold** sets the limit of how much confidence the YOLO model needs to create a classification. This default value of 0.001 for validation and 0.25 for predictions is very low due to the way YOLO models work; they create a large abundance of bounding boxes which are then intended to be filtered.
- **Weight Decay** is a method to reduce overfitting to the data. It introduces a variable penalty to weights that are large. Larger weight decays encourage simpler models.
- **IOU Threshold** is an Intersection-over-union threshold that determines how many overlapping bounding boxes are combined together. Consider two different use-cases: detection of people in a crowd and detecting cars on a road. The bounding boxes around the people in the crowd should be overlapping and clustered, so we want to use a lower IOU Threshold compared to the cars. Selecting a higher IOU threshold merges more overlapping boxes into singular detections.
- **Model Mode** selects how you wish to train the YOLO model. Selecting 'Evaluation' utilizes the YOLO model's weights as initial values, and then retrain all of them with the uploaded data. 'Fine Tuning' freezes a variable number of layers, meaning they will be identical to the YOLO model. By default 'Fine

Tuning' freezes the backbone of the YOLO model, which consists of the first 10 layers, including the initial convolutions and the 'feature pyramid'. More details can be found in [Ultralytics' documentation](#). Finally, 'Evaluation' doesn't retrain the model at all, but it does adapt the model head to function with the defined classes in your project.

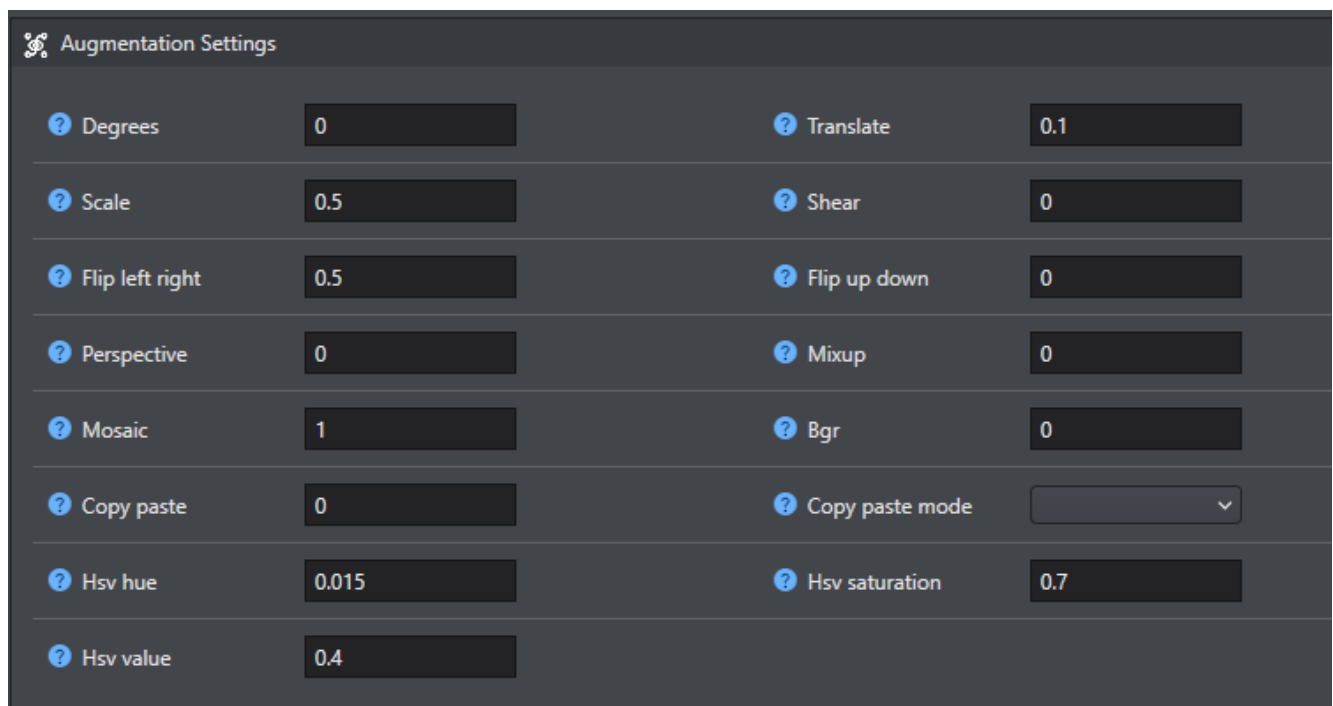
For this exercise, keep all default values.

3.6 - Exercise 6: Selecting Data Augmentation

(Solved exercise)

In the Augmentation Settings tab you can adjust parameters in order to bolster the dataset with augmentations. The reason this is done is twofold; it increases the amount of data that is used to train the model, and it makes the model significantly more robust to various situations and noise. For example, a model trained on images of telephones held in hands might fail when applied on someone holding a phone horizontally whilst taking an image. Applying 'Degrees' data augmentation would take the existing vertical images and **also** add images into the dataset that have been rotated a random number of degrees within the range specified, and the resulting model is likely to perform better on images of horizontally held phones. The data augmentations can also be useful to train models to perform better on noisy images and differently configured cameras by varying the saturation, hues and values of the training data.

A detailed explanation of all the Augmentation Settings below can be found in [Ultralytics' documentation](#). Spend a couple minutes looking through those.



Augmentation Settings			
Degrees	0	Translate	0.1
Scale	0.5	Shear	0
Flip left right	0.5	Flip up down	0
Perspective	0	Mixup	0
Mosaic	1	Bgr	0
Copy paste	0	Copy paste mode	
Hsv hue	0.015	Hsv saturation	0.7
Hsv value	0.4		

Once again, keep the values as the default for this exercise.

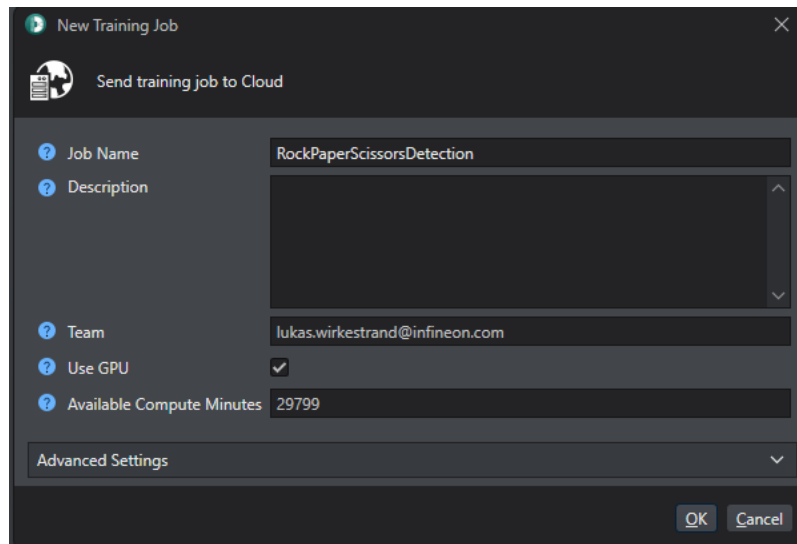
3.7 - Exercise 7: Training the Model and Evaluating the Test Results

This exercise is also similar to Chapter 2, but has some minor differences. The training backend uses the YOLO training loop, but from the development perspective not much changes.

Step 1: Starting and Tracking Training Jobs

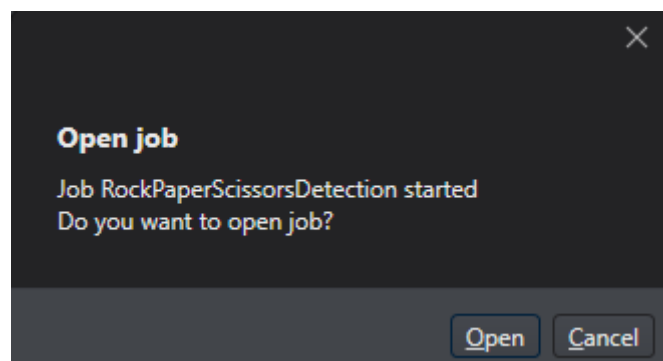
To start the training job, follow the steps below:

1. Click **Start New Training Job**. The **New Training Job** window appears.

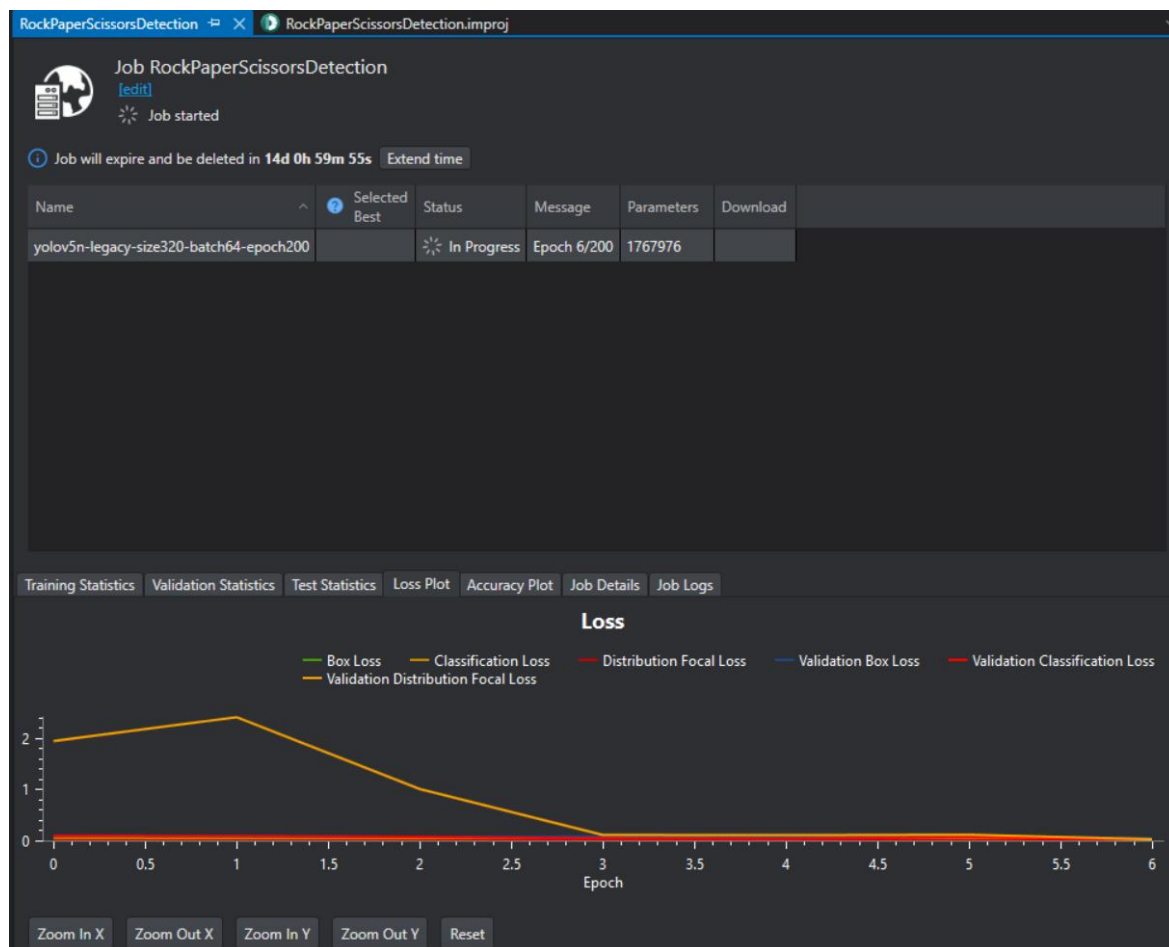


The 'New Training Job' dialog box is shown. It has a title bar with a close button. Below the title bar is a section 'Send training job to Cloud' with a cloud icon. The main area contains several fields: 'Job Name' with the value 'RockPaperScissorsDetection', 'Description' (empty), 'Team' with the value 'lukas.wirkestrand@infineon.com', 'Use GPU' with a checked checkbox, and 'Available Compute Minutes' with the value '29799'. At the bottom, there is an 'Advanced Settings' dropdown menu and 'OK' and 'Cancel' buttons.

2. In **Job Name** and **Description**, enter the name of the job and a description, respectively.
3. In **Use GPU**, enable the GPU powered training.
4. In **Available Compute Minutes**, your total compute minutes available are displayed.
5. Click **OK** to begin the training. After data is uploaded, a popup window appears indicating that the job has been started.

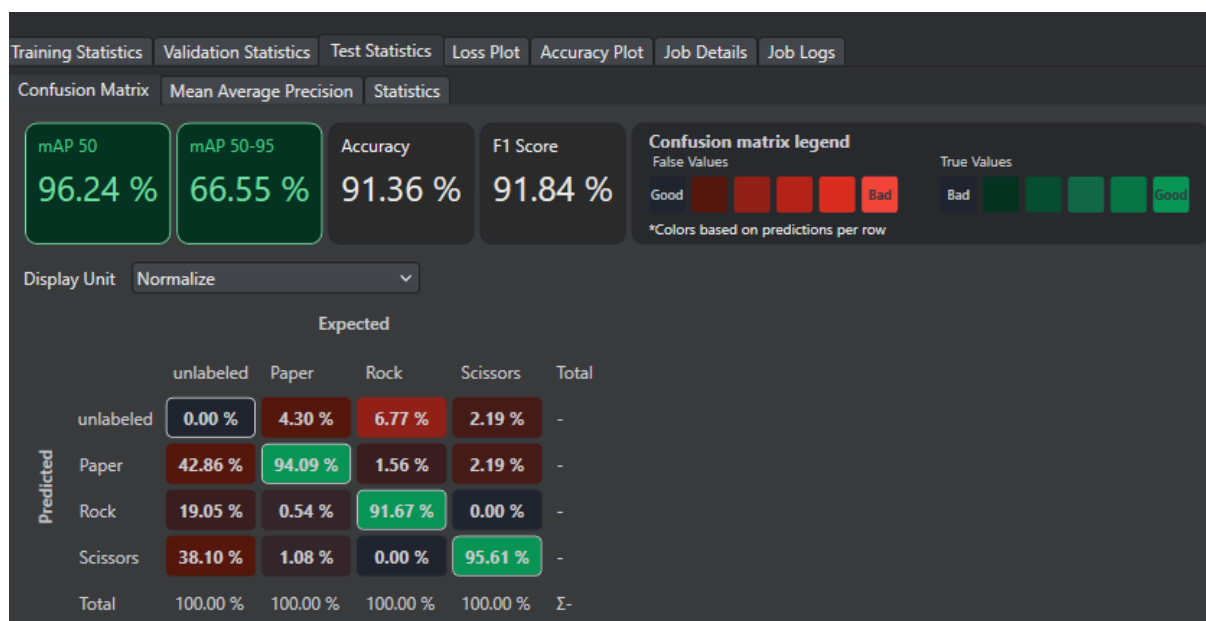


- Click **Open** to view the training progresses in a new tab.



You can now wait for your models to finish training. Be aware that this will take upwards of an hour.

Once the model training is completed, the statistics can be viewed to gauge model performance. Since accuracy is not a great measurement of performance for Object Detection models, it's recommended to use the mAP50-95 (mean Average Precision) which checks how much of a bounding box overlaps with the ground truth. mAP50 requires half the bounding boxes to overlap whilst mAP95 requires 95% of it. These metrics are saved under the downloaded 'Advanced' folder and can also be viewed for 2 weeks in the cloud job, accessed at "Imagimob Cloud" in the top right.



Step 2: Downloading Model Files

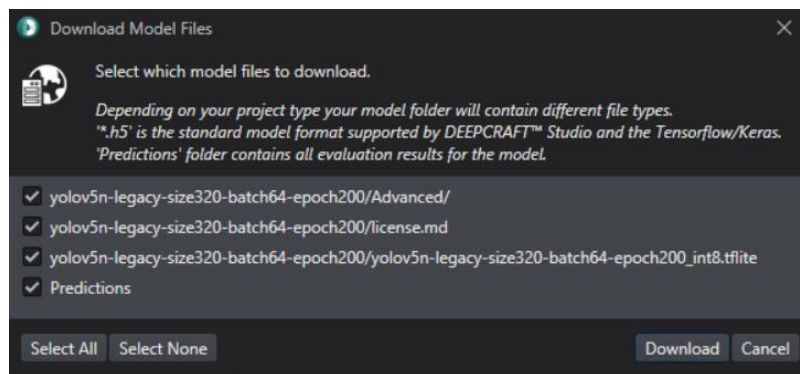
After the model is trained in the Infineon Cloud, the next step is to download the trained model files. This is also similar to Chapter 2, but the types of model files downloaded are different.

To download the model files, follow these steps:

1. Scroll right to the **Download** column and click the **download icon** to download the model files.
2. The Download model files window appears with the option to download:
 - /Advanced/ which contains train/test/validation result metrics and a pytorch file version
 - Predictions using the trained model on all uploaded data
 - Int8.tflite which is the quantized int8x8 model file in tensorflow light formatSelect the all the files and click **Download**.

Note: Newer YOLO versions contain additional files, such as the model in .onnx and float32 format.

3. Save the model in the “Models” folder of the DEEPCRAFT™ Studio RockPaperScissorsDetection folder.

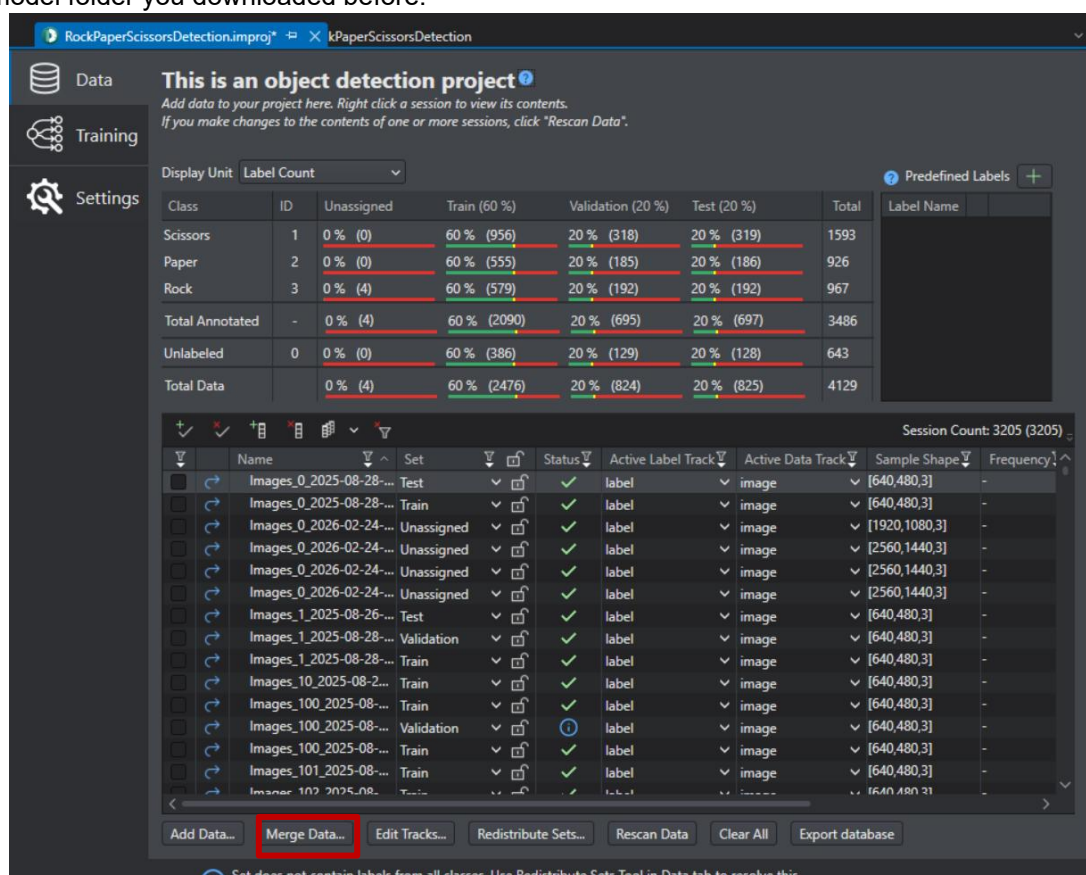


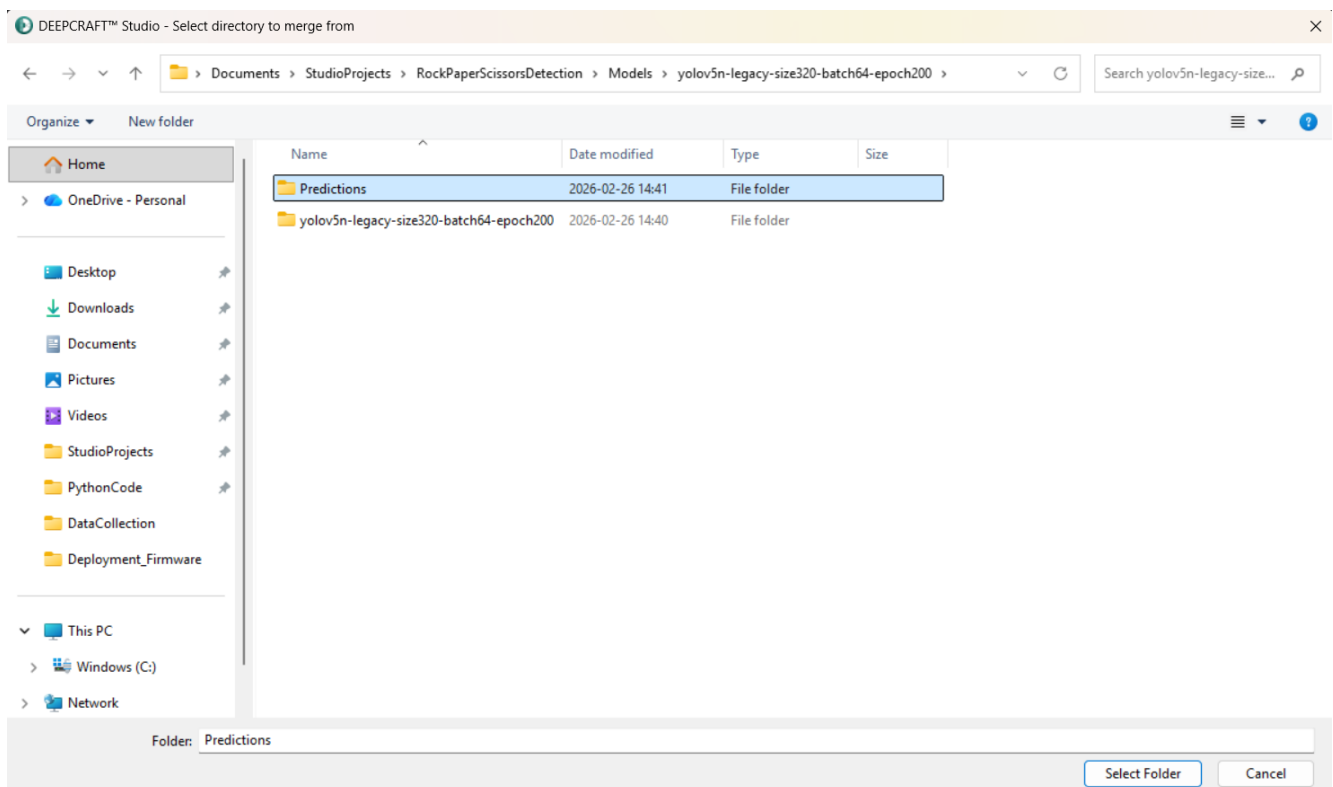
Step 3: Importing predictions into your project

DEEPCRAFT™ Studio allows a side-by-side view of the performance of a model on images by merging the model predictions as tracks into the sessions containing the original data.

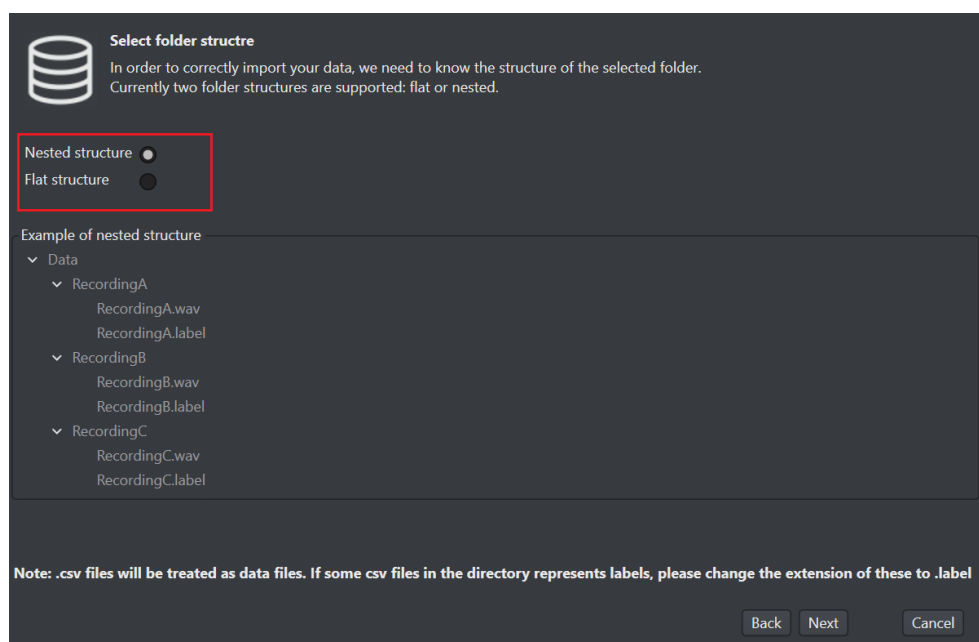
To import predictions, follow these steps:

1. Navigate to the project directory and open the project (.improj) file.
2. In the **Data** tab, click the **Merge** button and browse to select the **Predictions** folder which is inside the model folder you downloaded before.





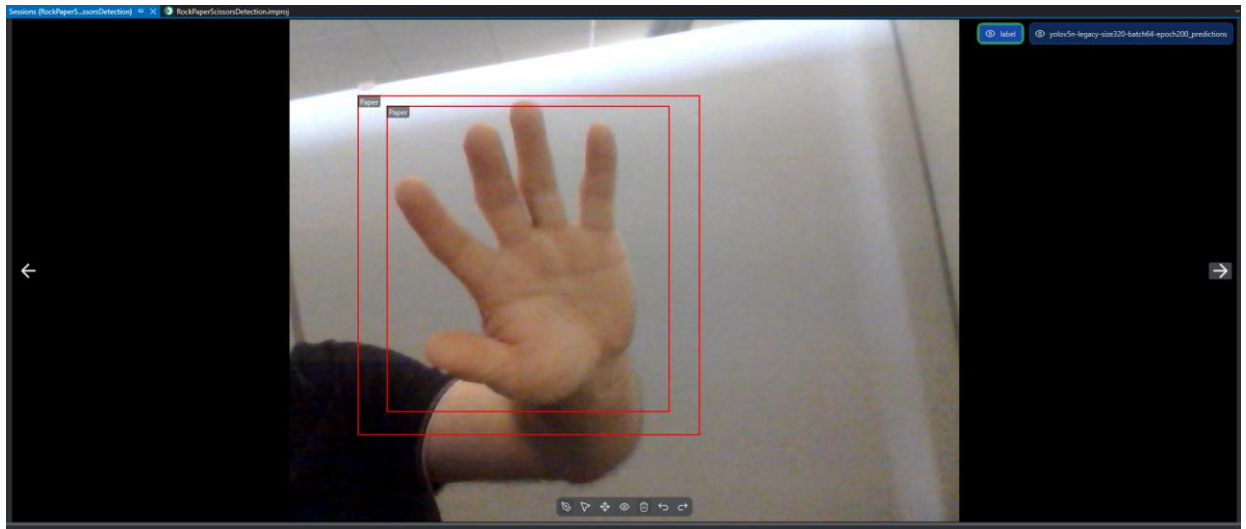
3. Select the **Nested structure** radio button and click **Next**. Predictions from the selected folder will be merged as tracks into the corresponding local sessions.



4. In the next window, select or deselect what you want to import and click **OK** to complete the merge. Take care to check that the column related to your new model is completely selected. This may take a little while to process.
5. You can now visualize the bounding boxes generated by your model side-by-side with the labels for every image in the project. Click the Eye icon in the **label** or **model-predictions** buttons to hide either the original or the predicted label to evaluate properly. This allows for easy comparison to determine if

there are any discrepancies between the label and predictions. If so, the data can be looked at to see if there are anomalies or if the label is possibly incorrect.

Note: Don't use images in the training set to gauge model performance, as that introduces bias.



3.8 - Exercise 8: Evaluating the Model in Real-Time

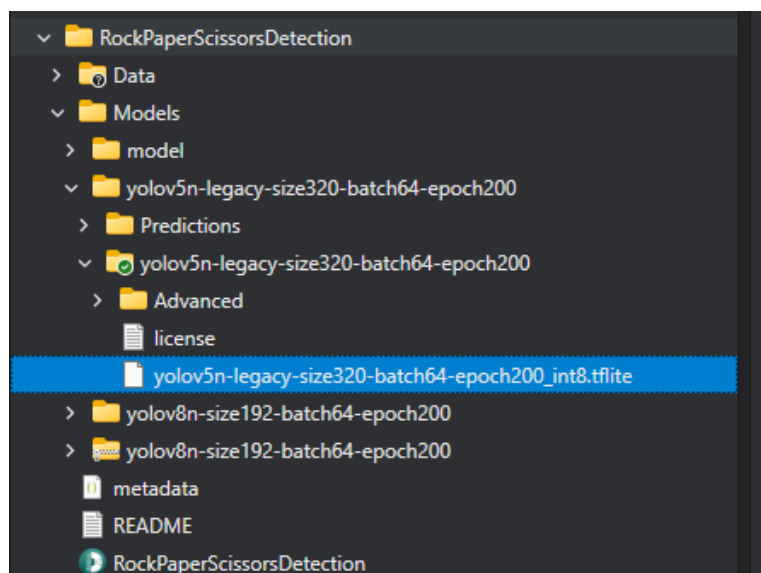
DEEPCRAFT™ Studio provides the following ways to evaluate the Object Detection model:

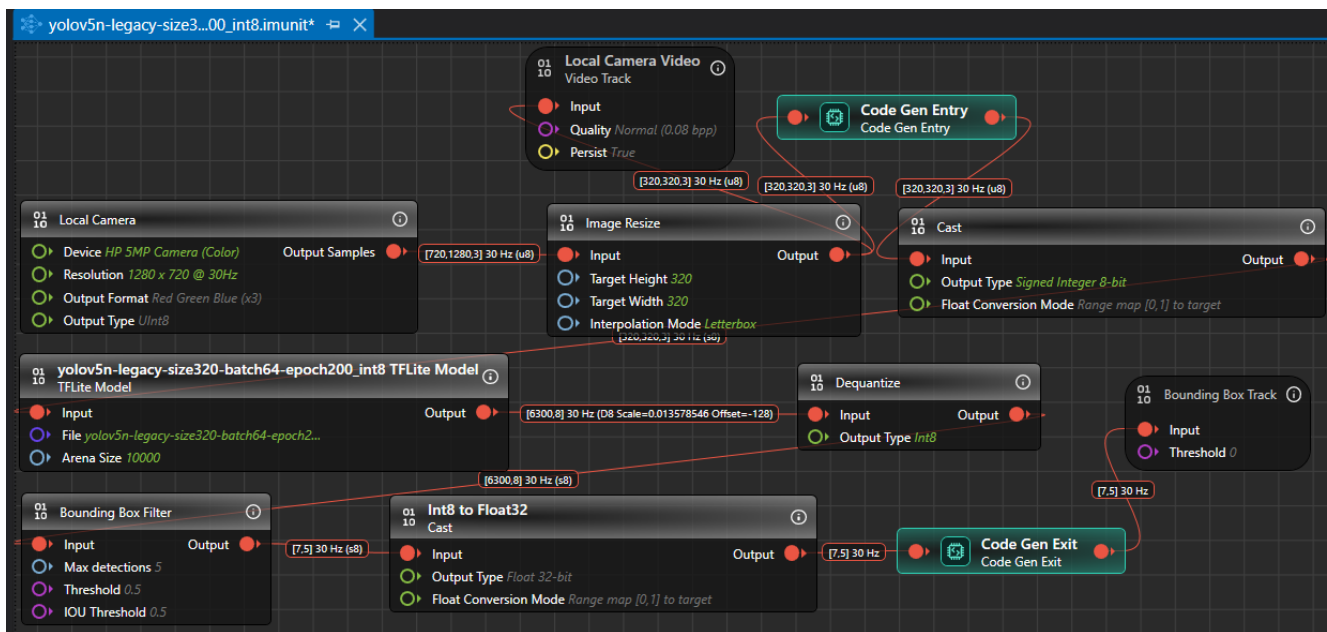
- Evaluating the model using statistics and metrics
- Evaluating the model Predictions with Original Data
- Evaluating the model using Graph UX in real time

The first two points were shown in Exercise 3.7, with the metrics of the model displayed after the training completes and accessible via the Imagimob Cloud or downloaded Advanced folder. The downloaded Predictions are used to view the performance on the uploaded data. However, that leaves the last and most important way to evaluate models: on unseen data in real-time. This is possible with any classes; however, it is obviously a requirement for the objects in question to be at hand. Luckily this use case very much is.

Step 1: Creating and understanding the evaluation project

Navigate to the downloaded .tflite file in the Solution Explorer and double click it. This will automatically generate a Graph UX project (.improj) in the same directory which can be utilized for evaluation and code generation.





The Model Evaluation Object Detection Graph begins with the **Local Camera** Node, which streams live video data at the resolution, channel settings and output type specified in the node. For instance, the output shape from the **Local Camera** Node is [720,1280,3]30 Hz (u8), where 720x1280 represents the video frame dimensions, 3 indicates the RGB color channels, 15 is the frame rate and u8 is the output type in unsigned Integer-8 format. The output from the local camera is input to **Local camera video track** node and **Image Resize** node. The local camera video track node helps to visualize video data collected from the Local Camera node.

The streaming video frames from the local camera node are then passed to the **Image Resize** Node. The **Image Resize** node preserves the aspect ratio by scaling the image to fit within the target size and padding the remaining space with a constant color. This approach avoids cropping, stretching, and blurring, which helps the model predict more accurately and consistently. By default, the **Image Resize** node applies letterbox padding to reach the target input size expected by the model. You can configure the interpolation mode to bilinear or nearest-neighbor, depending on your performance and quality requirements. For YOLO models, use letterbox because it mirrors the preprocessing typically used during training and aligns with the vision code example, thereby improving inference reliability.

Correct resizing is critical for optimal performance; improper resizing can degrade accuracy, stability, and overall model reliability. In this example graph, the model was trained on images of size 320x320, so the **Image Resize** Node is set to resize frames to 320x320 automatically. If you are unsure of the expected image size, you can check it on [Netron](https://netron.io/). By default, the settings for the Image Resize node are defined. The local camera video track node helps to visualize video data collected from the Image resize node. By default, the settings for the local camera node are defined. Click the node if you want to view or edit the settings in the Properties window accordingly.

The light-blue nodes 'Code Gen Entry' and 'Code Gen Exit' do not perform any operations, they are instead used to indicate where the generated embedded code in Exercise 9 will be. Multiple entry and exit nodes can be placed, which will result in multiple inputs and outputs to the generated code. The reason they are placed where they are is to align with the default settings of the Code Examples which are used to deploy the embedded code in Exercise 9. The code example has its own resizing code already and therefore we don't need to include it.

Once resized, the image frames with an input shape of [320,320,3]@15 (u8) are fed to the **Cast Node**, which converts tensor elements from one type to another. The camera provides unsigned 8-bit values in the range 0–255, while the model in the graph expects signed 8-bit values in the range –128 to 127. To preserve the full dynamic range during integer-to-integer conversion, we apply full-range linear mapping so that 0 maps to –128 and 255 maps to 127. This approach avoids clipping or saturation, maintains the original contrast and detail, and ensures the model receives data in the format it was trained to consume. This conversion stage standardizes the input values to the model's expected signed domain, improving numerical consistency across inferences. By using full-range mapping, we retain sensitivity to both dark and bright regions, which supports stable predictions.

The output from the cast node is fed into the model, which detects the classes or objects it was trained on. In this case, the model detects hand gestures such as Rock, Paper and Scissors. The output shape of the model is [6300,8] 30 Hz (D8 scale and offset), where 8 represents the sum of three classes (Rock, Paper, Scissors),

four bounding box coordinates, and one Detected Flag option, to detect the validity of each prediction (1 for a valid detection; 0 for padding) and 6300 represents the prediction results and D8 contains the information about the scale and offset to convert the quantized values to float 32.

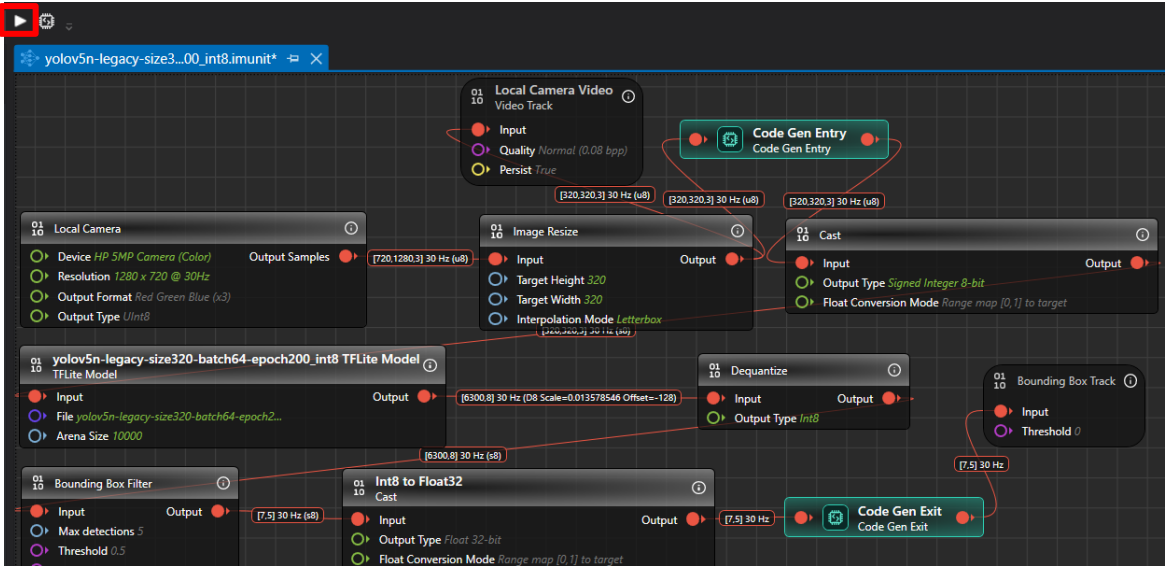
The output from the model is fed to the **Dequantize** node, which converts quantized integers to floating-point values. This conversion provides a more interpretable data type than quantized integers.

This output shape from the Dequantize node becomes the input for the **Bounding Box Filter** Node. The **Bounding Box Filter** Node applies Non-Max Suppression (NMS), a post-processing technique used in object detection algorithms. NMS eliminates redundant overlapping bounding boxes by filtering out those with low confidence scores and selecting only the most confident ones. It sorts the remaining bounding boxes by their confidence scores and iteratively suppresses boxes that overlap significantly with the highest-scoring box.

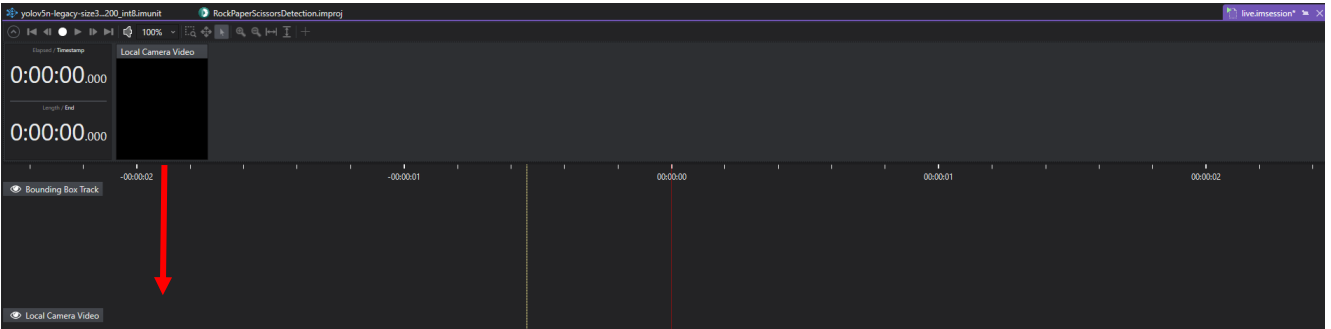
The output from the **Bounding Box Filter** Node is then fed to the **Bounding Box Track** Node, which displays the bounding boxes. The input shape for this node should be [Confidence, Detection], where Confidence includes (Center X, Center Y, Width, Height, ClassN...). These bounding boxes are finally displayed on the **Video Track** Node on the timeline, allowing you to visualize the model in action with live streaming video data and bounding boxes drawn on the detected objects.

Step 2: Running the graph and evaluating the model

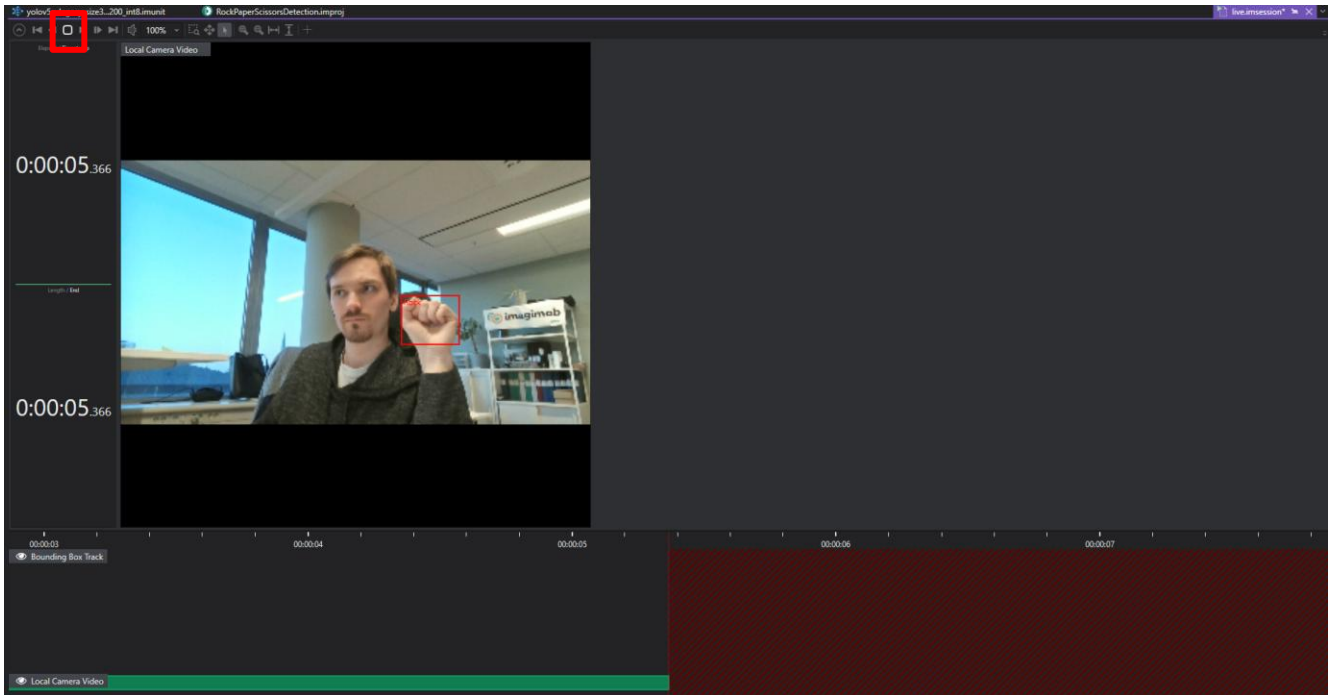
Run the graph by hitting the play button at the top. Wait for the timeline to generate.



This compiles the timeline, and the Local Camera Video will display the captured video, which can also be enlarged as shown below.



Press record to perform the live evaluation.



Spend some time evaluating your model; try different values in the Display Bounding Box node like Threshold and IOU Threshold, and regenerate the .imsession.

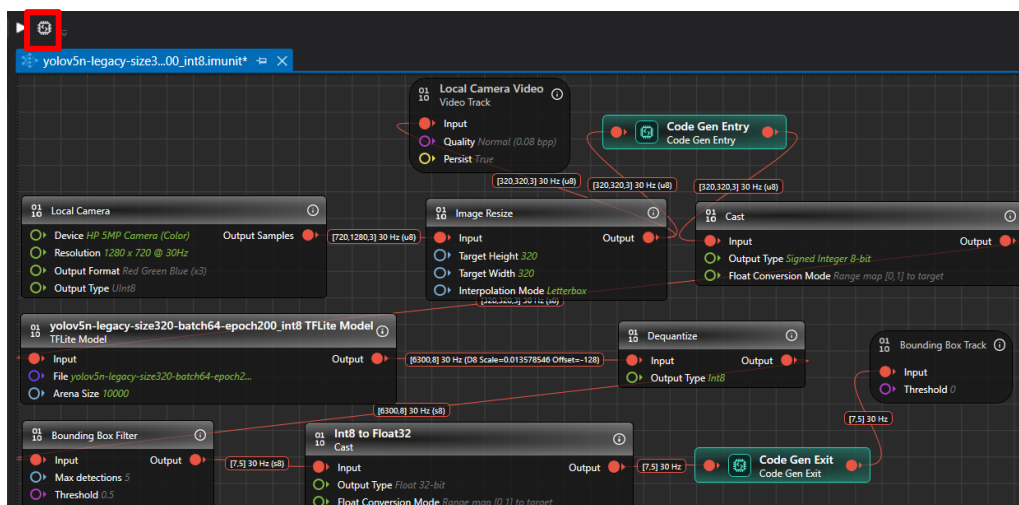
3.9 - Exercise 9: Deploying the Model on PSOC™ Edge

Step 1: Generating the Code

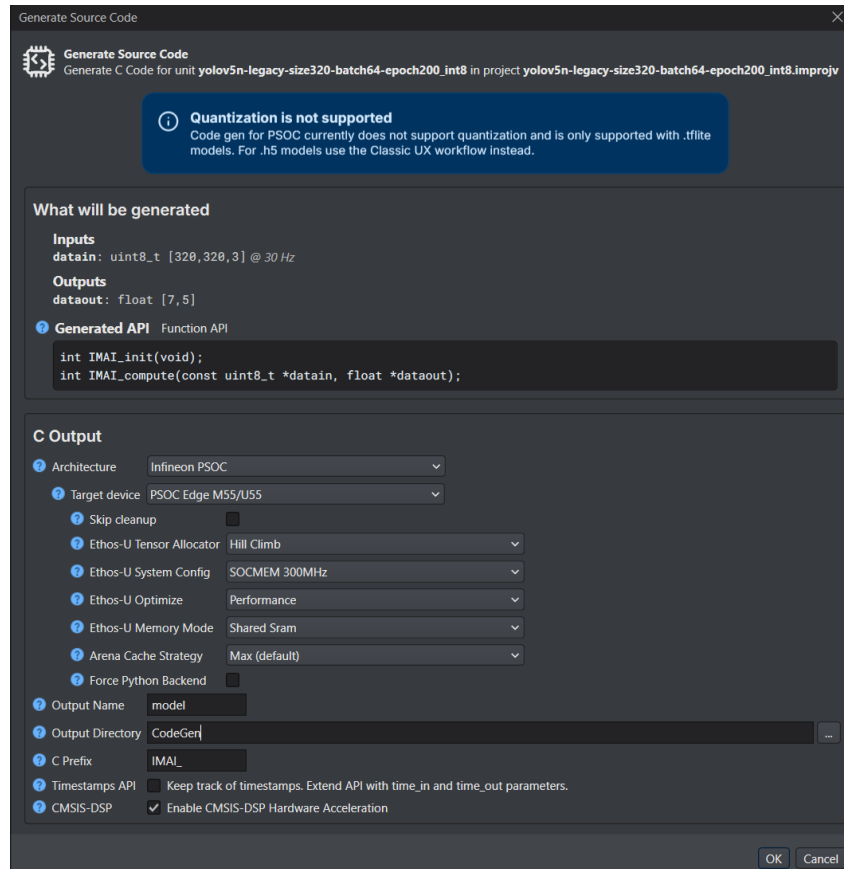
This section provides step-by-step information on generating the code for object detection models trained in DEEPCRAFT™ Studio. After you have evaluated the performance of the model, you can convert the model into optimized C-code that can be deployed on a PSOC™ Edge device. Graph UX facilitates code generation between any two nodes in a graph with a simple click of a button. You can select the starting and ending nodes in the graph to generate the corresponding C code for model deployment.

To generate the code, follow these steps:

1. Navigate to the toolbar and click the **Generate Source Code** button while in the Graph UX project.



2. Configure the following parameters:



- In **Architecture**, select **Infineon PSOC** as the architecture type.
- In **Target Device**, select **PSOC Edge M55/U55**, as that is required to run the Object Detection model.

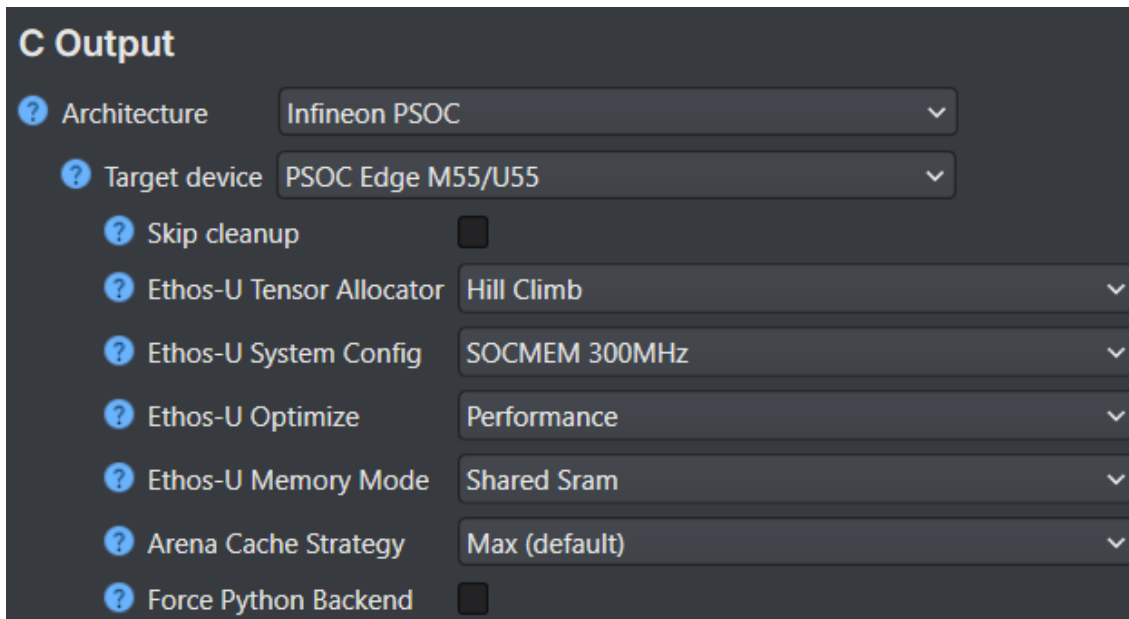
What **Generate Source Code** does is generate the C code for the entire pipeline between the 'Code Gen Entry' and 'Code Gen Exit' nodes. This approach is not only applicable for computer vision but also works for generating C code for a generic Graph UX project. The Generated API

Note: Before generating code for PSOC™ targets, ensure that your model only uses the layers and operators listed as supported in the [Supported layers and operators for PSOC Targets](#) table. If your model contains any layers or operators that are not supported, select the ANSI C99 architecture type. For instructions on generating code using the ANSI C99 architecture, refer to [Code Generation for Other Development Boards](#).

- Select the **Skip Cleanup** checkbox to not delete the temporary files created during code generation. If necessary, you can delete these files manually at a later time. When this checkbox is disabled, the temporary files are deleted automatically.

Advanced optimization for PSOC™ Edge M55+U55

The following optimization settings are part of the Vela compiler and generate optimizations for code targeting the Ethos-U55 NPU. These options reduce scratch memory usage with minimal to no impact on model accuracy.



C Output

- ? Architecture: Infineon PSOC
- ? Target device: PSOC Edge M55/U55
- ? Skip cleanup: ☐
- ? Ethos-U Tensor Allocator: Hill Climb
- ? Ethos-U System Config: SOCMEM 300MHz
- ? Ethos-U Optimize: Performance
- ? Ethos-U Memory Mode: Shared Sram
- ? Arena Cache Strategy: Max (default)
- ? Force Python Backend: ☐

Parameters	Description
Ethos-U Tensor Allocator	Select Linear Alloc, Greedy or Hill Climb to specify the algorithm for allocating non-constant tensors on the NPU and CPU. By default, Hill Climb is selected.
Ethos-U System Config	Select SOCMEM 300MHz or SOCMEM 200MHz as the system configuration of the Ethos-U NPU. Refer to Silicon documentation to know more.
Ethos-U Optimize	Select the optimization strategy based on Size or Performance. The Size strategy ensures minimal RAM usage by avoiding the use of the arena cache memory area size whereas the Performance strategy prioritizes maximal performance by utilizing the specified arena cache memory area size. By default, Performance strategy is selected.
Ethos-U Memory Mode	Select Shared SRAM or SRAM Only as the memory mode. Shared SRAM is shared between Ethos-U and Cortex-M software, whereas SRAM Only is dedicated to Ethos-U. The non-SRAM memory is read-only. By default, Shared SRAM is selected.
Arena Cache Strategy	Select the strategy to be used for determining how much arena cache is allocated for communication between the M55 and U55. Min results in the minimum memory required for the model. Max results in the maximum memory allowed by the target. Custom allows you to enter an exact specified value. Note that the arena cache is not the same as tensor arena. Tensor arena is the total memory needed by the model, not just the M55 -> U55 communication.
Force Python Backend	Changes the backend to the deprecated Vela Python API which was used in older versions of DEEPCRAFT Studio. Activating this also reveals two new options for that API: Max Block Dependency and Recursion Limit.

Max Block Dependency (Force Python Backend)	Set the maximum value that can be used for the block dependency delay between NPU kernel operations. A lower value may result in longer execution time. You may set: 0, 1, 2 or 3 as the value for NPU block dependency. By default, the value of block dependency is set as 3.
Recursion Limit (Force Python Backend)	Set the Python internal limit to depth of recursion. For very large networks, increasing the recursive limit may be necessary due to the recursive graph traversal algorithm. If generation fails with a RecursionError, increase the limit using this option. By default, the recursion limit is set as 1000.

- In **Output Name**, enter the name of the output files. Ensure this is 'model'.
- In **Output Directory**, browse and select the directory where you want to save the generated code. By default **CodeGen** is selected as the default folder.
- In **C Prefix**, enter the prefix to be added at the beginning of the generated file names.
- In **Timestamps API**, enable the checkbox to track corresponding input time for each output prediction. To know about the edge API, refer to [Edge API](#).
- In **CMSIS-DSP**, enable the checkbox to optimize the code using the ARM CMSIS-DSP Library. The code will be hardware accelerated using the CMSIS-DSP library for compatible nodes.
- Click **Ok** to generate the code. The code generation may take some time. The code (model.c and model.h files) is generated in the *Output Directory* defined earlier.

After you have generated the code, refer to final step below to deploy the model, or [Deploy Vision Model on PSOC™ Edge boards](#) to know the deployment steps.

Step 2: Deploying the Model using the Deploy Vision Code Example in ModusToolbox™

This section explains how to deploy code generated for vision models from DEEPCRAFT™ Studio to PSOC™ Edge using ModusToolbox™. To demonstrate the process, we use the [PSOC Edge™ MCU: Machine learning – DEEPCRAFT™ deploy vision](#) Code Example provided in the ModusToolbox™.

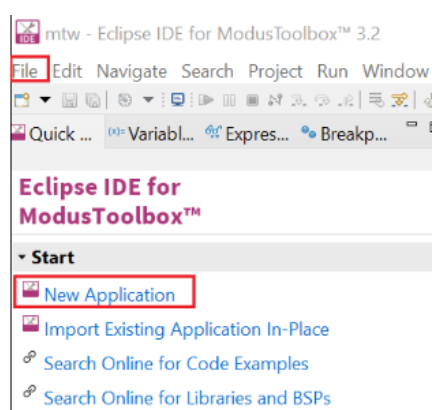
Software pre-requisites:

- ModusToolbox™ Tools Package 3.6.0 or later
- ModusToolbox™ Machine Learning Pack 3.0.0 or later

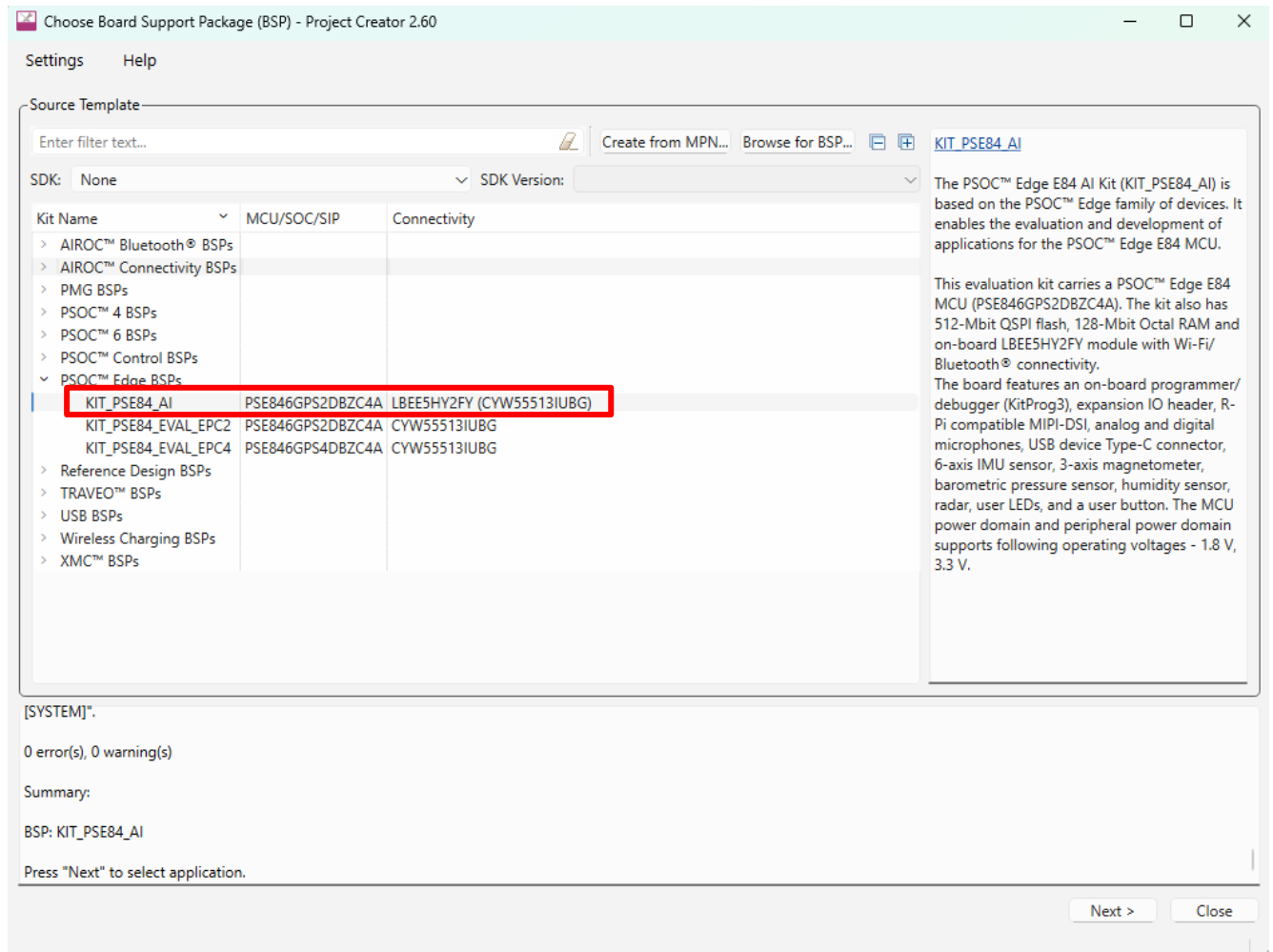
ModusToolbox™ and ModusToolbox™ Machine Learning Pack can be installed through ModusToolbox™ Setup. Download the setup application from [here](#).

Follow the steps to deploy the model onto the board:

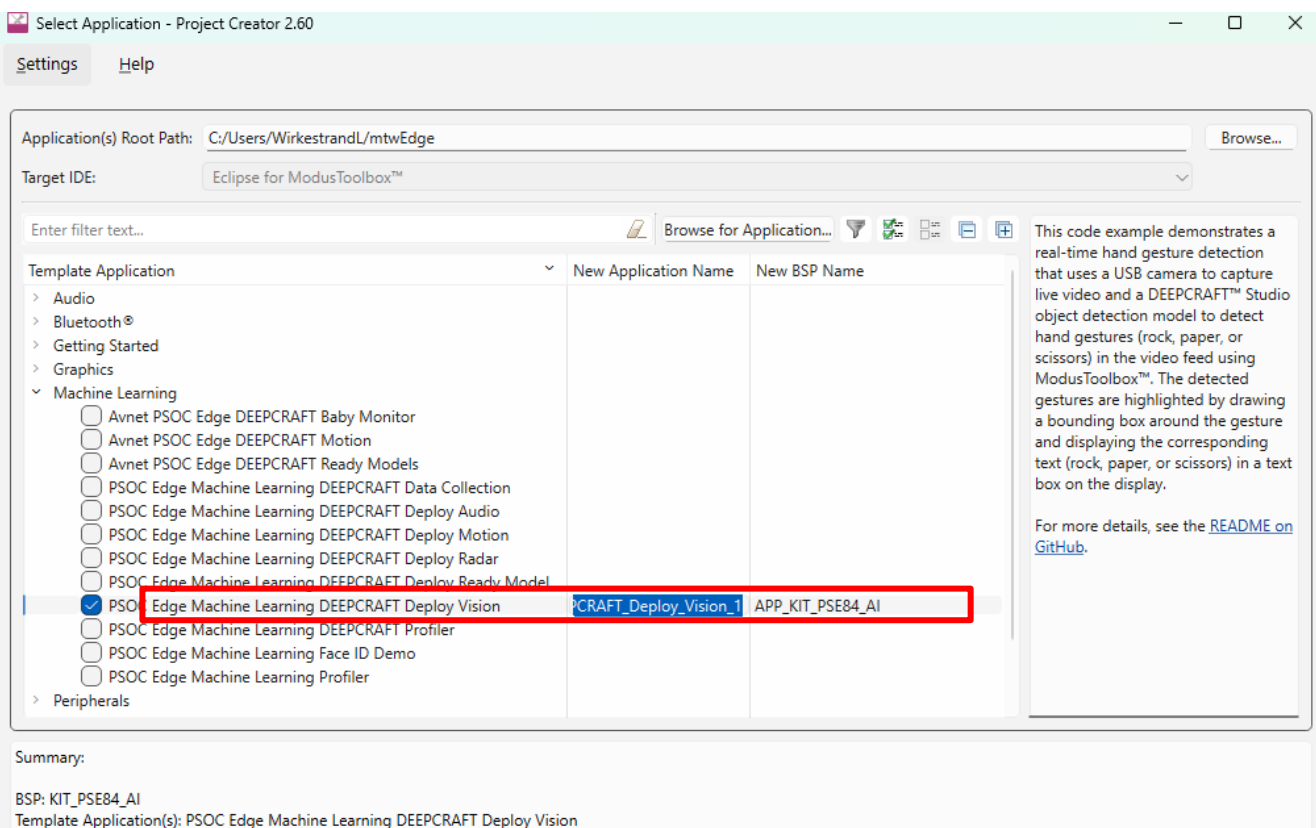
1. Open ModusToolbox™ > Eclipse IDE for ModusToolbox™ from the Windows Start menu. The Eclipse IDE for ModusToolbox™ window appears.
2. Browse and select the workspace directory for your project and click Launch to open the ModusToolbox™ workspace.
3. Select New Application from the Quick Panel or navigate to File> New> Modus Toolbox™ Application to open the Project Creator Tool.



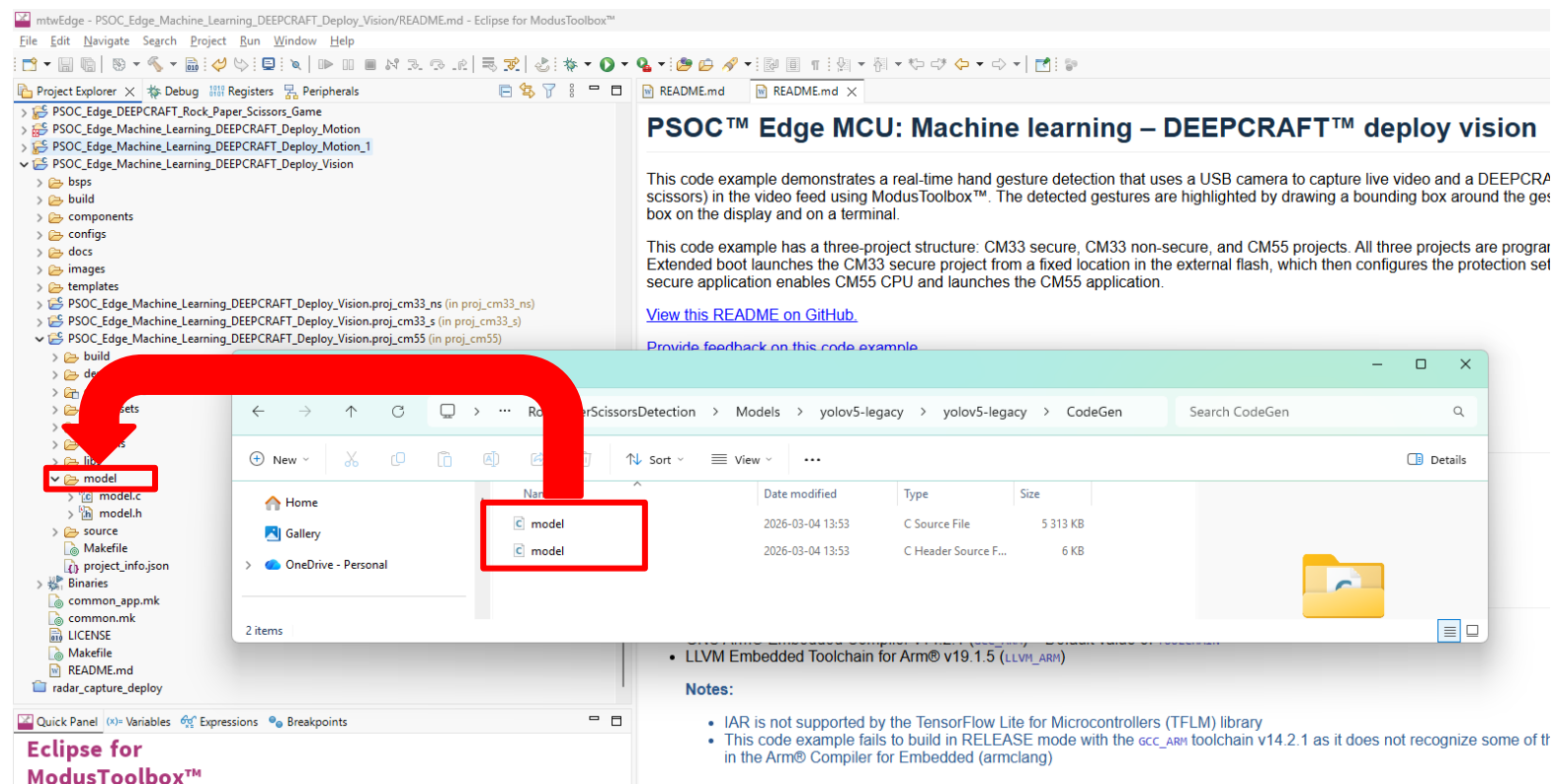
- Expand the Kit Name **PSOC™ Edge BSPs** from the list and select the **KIT_PSE84_AI** as BSP for your board and click **Next**. The Select Application window appears.



- Expand the **Machine Learning** section and select **PSOC Edge Machine Learning DEEPCRAFT Deploy Vision**. Press **Create**. This process will take a few minutes.



- After the project is successfully created, navigate to **proj_cm55> model** directory and substitute the provided example *model.c* and *model.h* files with the actual *model.c* and *model.h* files that you generated for your model in DEEPCRAFT™ Studio. Either copy & paste or drag & drop.



- Open the **source> inference_task.h** file, verify and update the following, if needed:

- the number of classes defined in the **NUM_CLASSES** is exactly same as the classes listed in your DEEPCRAFT™ Studio project. If the number or order of the classes differ, update the classes in the **NUM_CLASSES**. To view the order of the classes for your project, open the project file (.improj) and select the **Data** tab.
- the number of detections defined in **MAX_PREDICTIONS** is same as the **Max Detections** parameter configured in the **Bounding Box Filter** node during code generation in Graph UX. If you changed the Max Detections parameter, update the **MAX_PREDICTIONS** accordingly. By default, the code example supports a maximum of five simultaneous detections.

RockPaperScissorsDetection.improj

Data

Training

Settings

This is an object detection project

Add data to your project here. Right click a session to view its contents.
If you make changes to the contents of one or more sessions, click "Rescan Data".

Display Unit

Label Count

Class

ID

Unassigned

Train (60 %)

Validation (20 %)

Test (20 %)

Total

Scissors

1

0 % (0)

88 % (1337)

8 % (120)

4 % (66)

1523

Paper

2

0 % (0)

86 % (1349)

9 % (139)

5 % (72)

1560

Rock

3

0 % (0)

90 % (1924)

7 % (141)

3 % (65)

2130

Total Annotated

-

0 % (0)

88 % (4610)

8 % (400)

4 % (203)

5213

Unlabeled Data

0

0 % (0)

88 % (2475)

8 % (234)

4 % (116)

2825

Total Data

0 % (0)

88 % (7085)

8 % (634)

4 % (319)

8038

Predefined Labels

Session Count: 7287 (7287)

Name

Set

Status

Active Label Track

Active Data Track

Sample Shape

Frequ

test_10

Test

label

image

[640,640,3]

-

test_100

Test

label

image

[640,640,3]

-

test_101

Test

label

image

[640,640,3]

-

test_102

Test

label

image

[640,640,3]

-

test_103

Test

label

image

[640,640,3]

-

test_104

Test

label

image

[640,640,3]

-

test_105

Test

label

image

[640,640,3]

-

test_106

Test

label

image

[640,640,3]

-

Add Data...

Merge Data...

Edit Tracks...

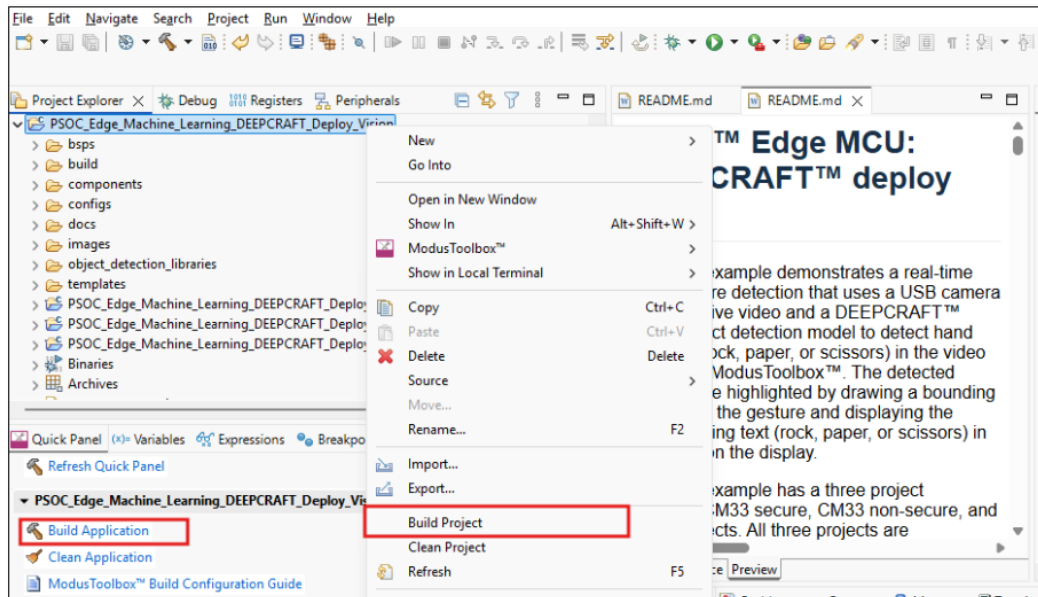
Redistribute Sets...

Rescan Data

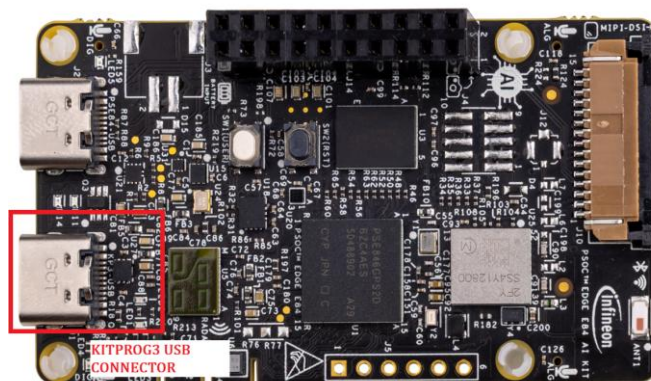
Clear All

Export database

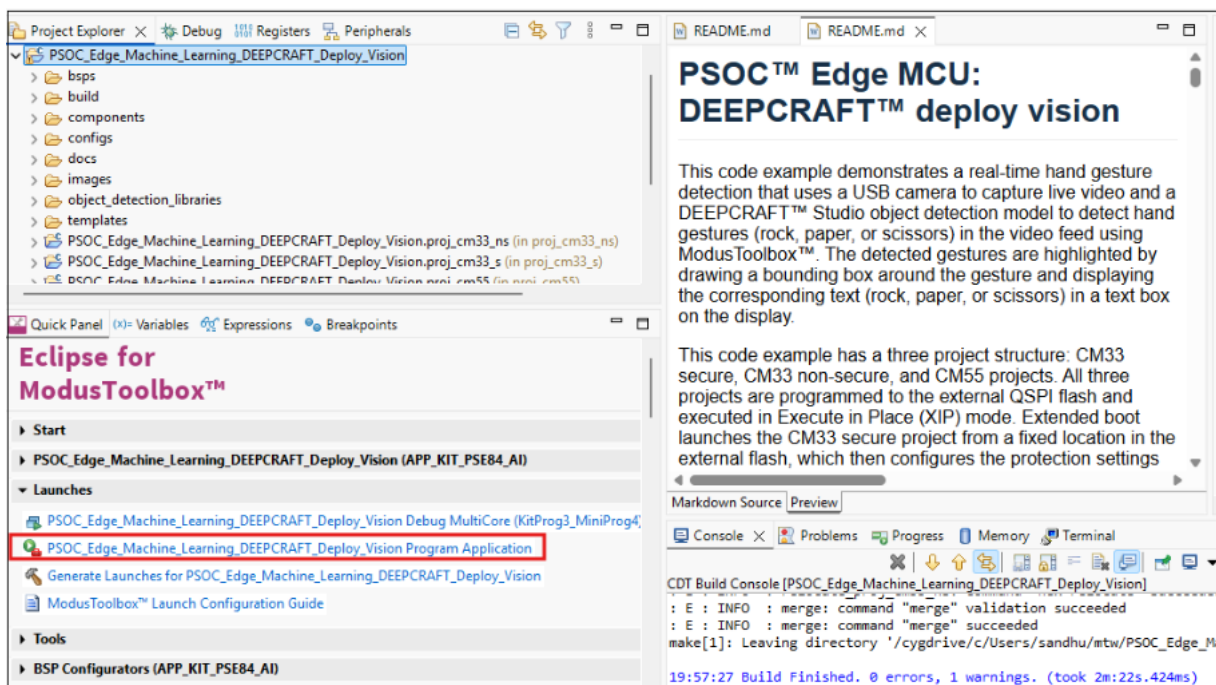
8. Right click the project and select **Build Project** or click **Build Application** in the Quick panel.



9. Connect the KitProg3 USB connector (J1) port on the PSoC™ Edge E84 AI Evaluation Kit to the PC using the USB cable.



10. Connect the camera to the kit. If you are using a USB webcam, connect it to the J2 port of the kit using an adapter dock. If you are using the DVP camera, connect that directly on the board.
11. Connect the Waveshare screen to the board using the MIPI-DSI cable.
12. In Quick Panel> Launches, click program. The code will deploy on the board. Ensure camera and screen are well connected. When the deployment is finished the model results will show in real time.



13. (Optional) Open a Serial Terminal window, connect to the PSOC™ Edge COM Port with BaudRate=115200 and observe the output there too.



4. Final Considerations and additional documentation

This tutorial guided you through DEEPCRAFT™ Studio's complete machine learning workflow. If you want to learn more, please refer to the following resources. The tutorial was based on the official DEEPCRAFT™ Studio documentation:

DEEPCRAFT™ Studio Official Documentation:

- <https://developer.imagimob.com/deepcraft-studio>

Chapter 2 focused on **classification tasks**. If you need to implement a **regression task**, the same workflow applies, but use a .data file as target instead of a .label file. Refer to the following resource to learn more about regression in DEEPCRAFT™ Studio:

Regression Resources:

- Classification vs Regression Data: <https://developer.imagimob.com/deepcraft-studio/getting-started/machine-learning-algorithms#determining-project-type-classification-or-regression>

Computer Vision Resources:

- Real-Time Image Data Collection and Labeling Using an External Camera: <https://developer.imagimob.com/deepcraft-studio/data-preparation/data-collection/collect-data-without-kit/collect-image-data-using-graph-ux>
- Supported Object Detection Dataset Formats: <https://developer.imagimob.com/deepcraft-studio/data-preparation/bring-your-data/bring-your-own-data-object-detection>
- Add Object Detection Data to a Project: <https://developer.imagimob.com/deepcraft-studio/data-preparation/add-project-object-detection>
- Labeling Object Detection Data: <https://developer.imagimob.com/deepcraft-studio/data-preparation/data-labeling/label-image-data>
- Training Object Detection Models: <https://developer.imagimob.com/deepcraft-studio/model-training/training-object-detection>
- Object Detection Model Evaluation (Real-Time): <https://developer.imagimob.com/deepcraft-studio/model-evaluation/evaluating-object-detection-model>
- Code Generation for Computer Vision Models: <https://developer.imagimob.com/deepcraft-studio/code-generation/code-gen-vision-models>
- Deploy Vision Models on PSOC™ Edge Boards: <https://developer.imagimob.com/deepcraft-studio/deployment/deploy-models-supported-boards/deploy-vision-model-PSOC-Edge>

If you need to interface with DEEPCRAFT™ Studio via a board which is not supported (example: non-Infineon kits, custom kits, etc.), you must implement the DEEPCRAFT™ Streaming Protocol:

Custom Board resources:

- Collect Data Using Any Development Board: <https://developer.imagimob.com/deepcraft-studio/data-preparation/data-collection/collect-data-other-boards>
- DEEPCRAFT™ Streaming Protocol Implementation: <https://developer.imagimob.com/deepcraft-studio/getting-started/tensor-streaming-protocol/registering-sensors-using-protocolv2>