Chapter 4: Imagimob

After completing this chapter, you will understand how to use the machine learning solution from Imagimob, an Infineon Technologies company, to develop and train ML models for use with Infineon MCUs. You will work with solutions using inputs from motion and audio sensors.

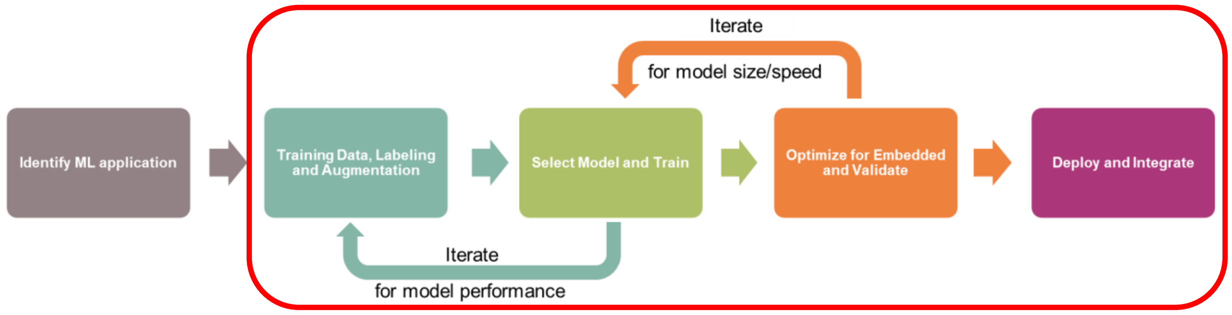
Table of contents

Document conventions

Convention	Usage	Example
`Courier New`	Displays code and text commands	`CY_ISR_PROTO(MyISR);` `make build`
Italics	Displays file names and paths	*sourcefile.hex*
[**bracketed, bold**]	Displays keyboard commands in procedures	[**Enter**] or [**Ctrl**] [**C**]
Menu > Selection	Represents menu paths	**File > New Project > Clone**
Bold	Displays GUI commands, menu paths and selections, and icon names in procedures	Click the **Debugger** icon, and then click **Next**.

4.1 Overview

In this chapter we will be using tools from Infineon's fully-owned subsidiary Imagimob (https://imagimob.com/). Their tools help facilitate and automate the steps of capturing raw data from a sensor, labeling the data, generating, and training a machine learning model, validating the model on the target hardware, and then optimizing it and validating the optimized model on the target hardware.



Imagimob can output C files that can be directly deployed and integrated on the MCU. However, it can also output a Keras *.h5* model that can be used by the ModusToolbox™ ML configurator for additional optimization such as quantization.

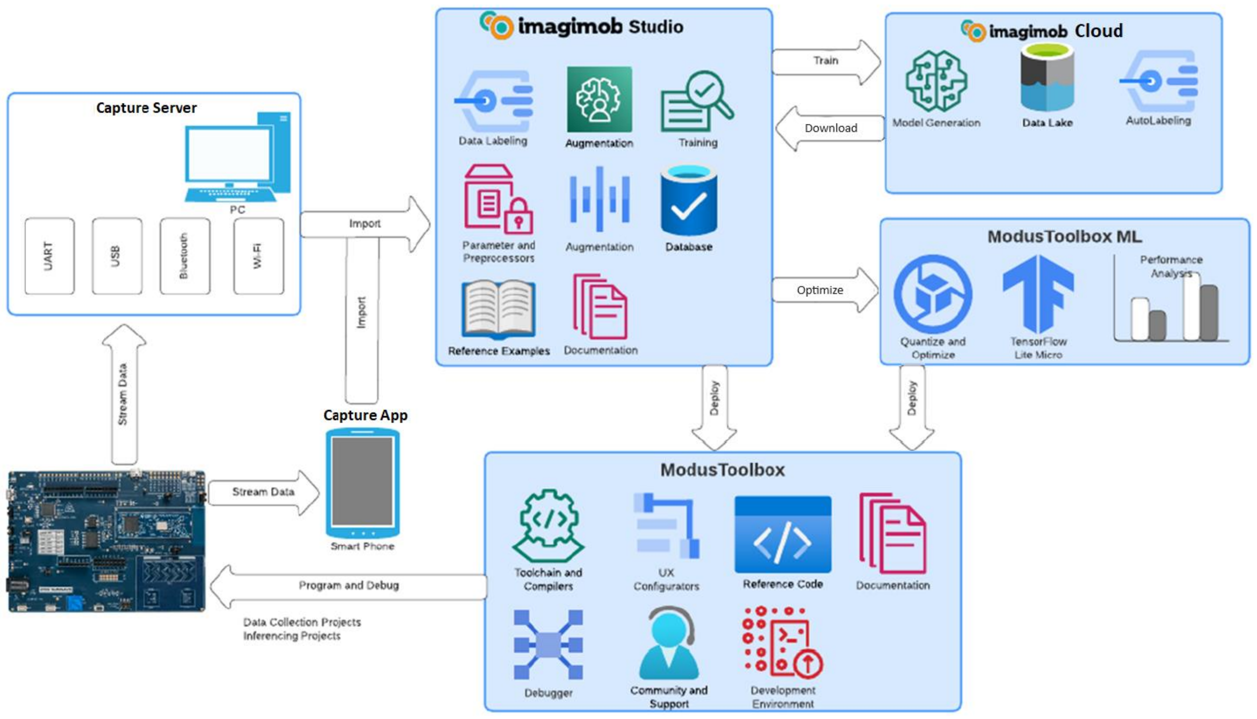
The Imagimob AI is designed to provide solutions to a wide range of problems including:

- **Predictive maintenance -** Recognize machine state, detect machine anomalies and act in milliseconds
- **Audio applications -** Classify sound events, spot keywords, and recognize your sound environment
- **Gesture recognition -** Detect hand gestures using low-power radars, capacitive touch sensors or accelerometers
- **Signal classification -** Recognize repeatable signal patterns from any sensor
- **Fall detection -** Fall detection using IMUs or a single accelerometer
- **Material detection -** Real time material detection using low-power radars

The Imagimob AI software consists of the Imagimob Cloud, Imagimob Studio, the Capture App, and the Capture Server.

Data from the PSoC™ 6 and its attached sensor(s) is captured and relayed to either the Capture App or the Capture Server. These tools associate the collected data with a label file and an optional video showing exactly what was done. When the dataset is ready, Imagimob Studio is used to collect and manage data files and define expectations of the model. When the model is defined, the data is all uploaded to Imagimob Cloud which generates different models and trains them. Results are then presented as downloadable options via Imagimob Studio again.

The models and code generated by Imagimob Studio are fully reusable on any device. The ModusToolbox™ ML Configurator can optionally be used to optimize the model generated by Imagimob Studio for optimal performance on the PSoC™ 6 hardware.



4.2 Workflow

Now we'll go into more detail about the workflow in Imagimob and the skills required at each step.

4.2.1 Collect and annotate data

As you learned earlier, getting a good dataset is one of the largest factors in developing a successful machine learning solution. With Imagimob, the Capture App or Capture Server can be used to quickly collect data directly from your sensor(s) of choice and label it. As you will see, the videos captured along with the data make it easy to tell exactly what was happening. This facilitates creating precise labels for each point in time and helps with future reviews of the data.

A code example (Imagimob Machine Learning Data Collection) provided in ModusToolbox™ allows you to program the PSoC™ 6 device to capture data from audio or motion sensors without requiring you to write any firmware. Options in the *Makefile* allow you to specify the method used to send the data – Bluetooth® LE, USB, UART and Wi-Fi are supported.

Data collection from additional sensors such as pressure and radar sensors will be added in the future.

The Python based Capture Server desktop tool and the Android based Capture App can be used interchangeably. Both tools serve the same purpose and the choice of which to use ultimately comes down to which makes the data collector’s job easier. In some cases, it may even make sense to collect some data with one tool and other data with the other tool. The Capture Server can support receiving data over Bluetooth® LE, USB, UART, or Wi-Fi connections and uses an attached webcam to capture video. The Capture App supports receiving data over Bluetooth® LE or Wi-Fi and uses the phone’s built-in camera. Either way, once data is collected, it must be transferred to a PC for use with Imagimob Studio.

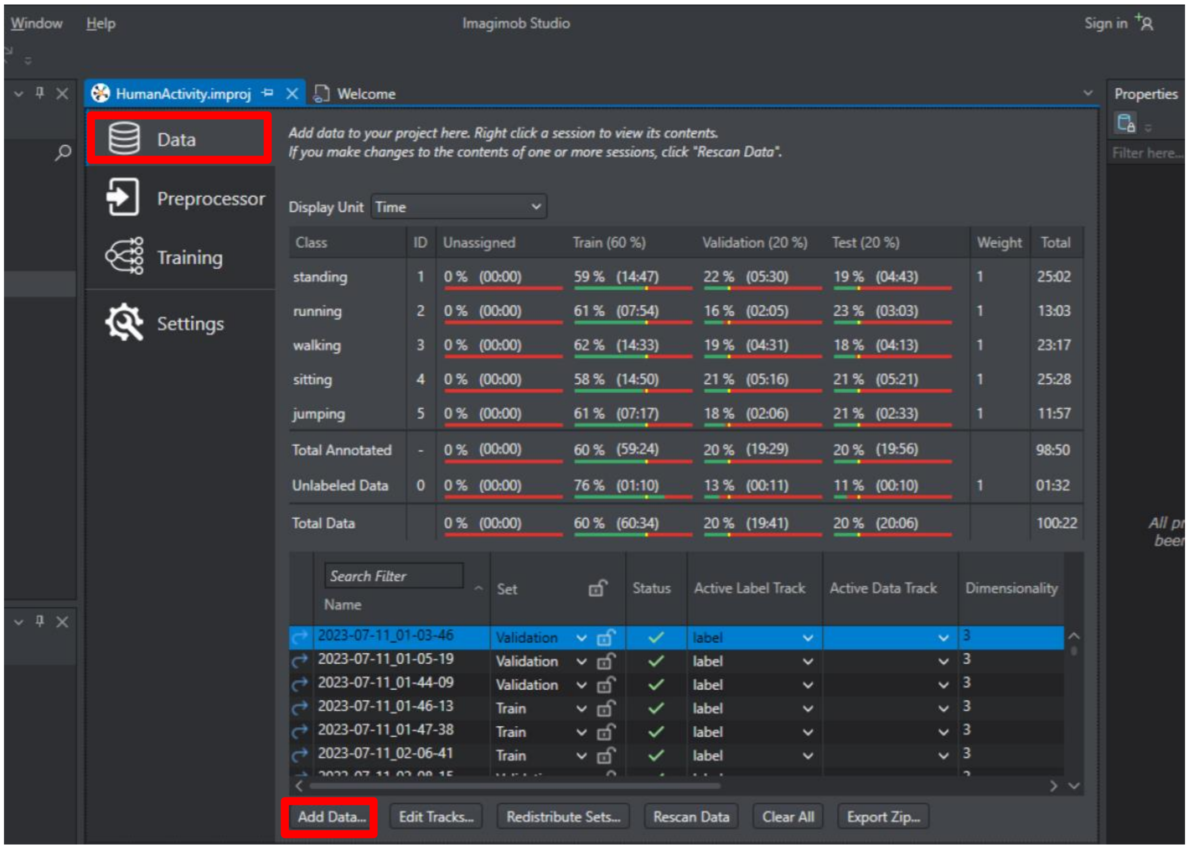
The person(s) responsible for capturing data will need some expertise in the end application. That is, they will need to understand how the sensor(s) will be used in the real world and what the valid range of inputs looks like to ensure that the dataset contains sufficient information to create a good ML model.

Once you have finished collecting data, it must be added to Imagimob Studio, and you can move on to managing and analyzing the data.

Note: *In the first release, only UART is supported for capturing data.*

4.2.2 Manage, analyze and process data

Once data has been collected, it needs to be added to a project within Imagimob Studio. This is done by opening the Project file (*.improj*), selecting the **Data** tab and then clicking the **Add Data** button at the bottom of the screen. The application will automatically detect the label file and video if they are present, and assumes other files are data sets.



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".

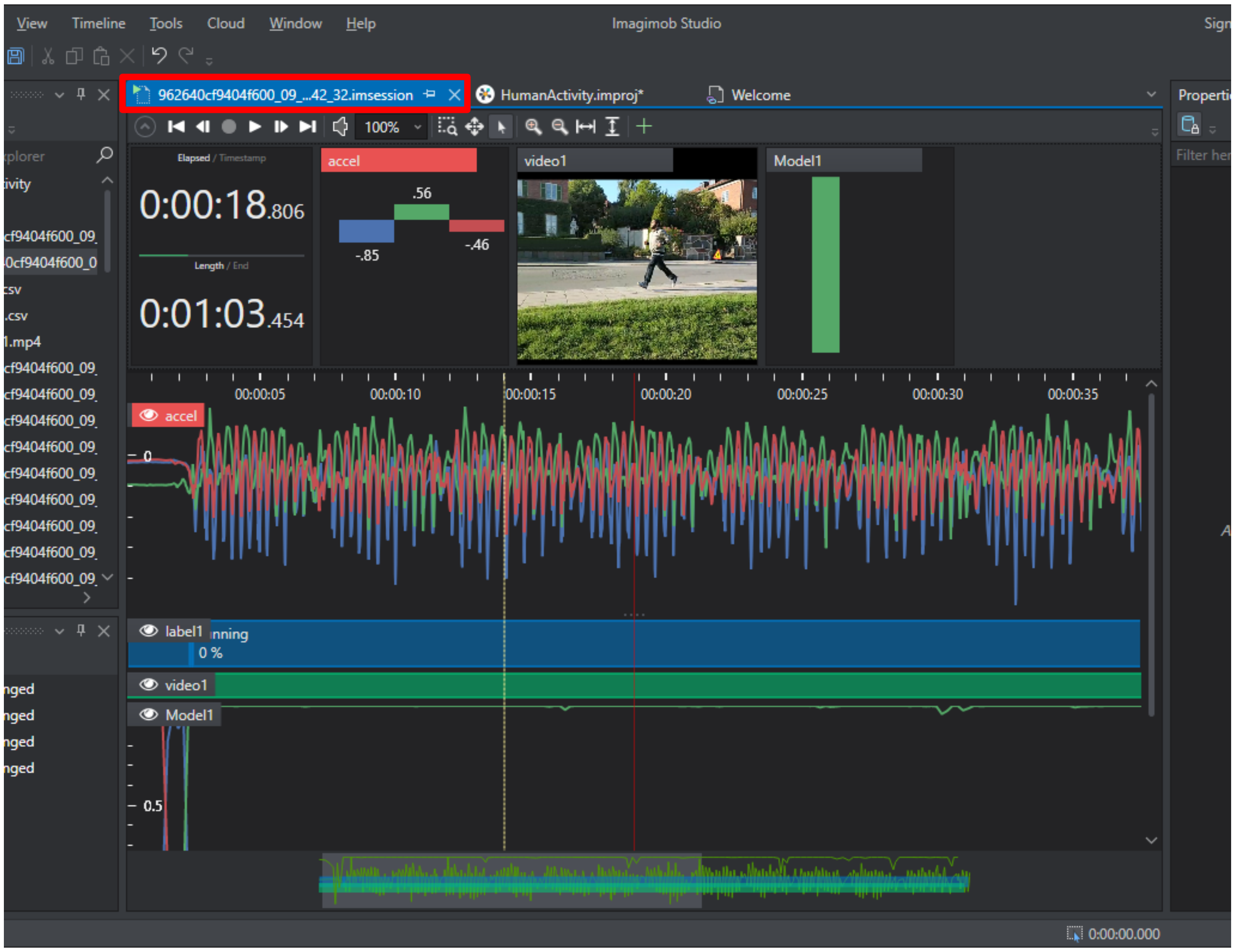
Display Unit: Time

Class	ID	Unassigned	Train (60 %)	Validation (20 %)	Test (20 %)	Weight	Total
standing	1	0 % (00:00)	59 % (14:47)	22 % (05:30)	19 % (04:43)	1	25:02
running	2	0 % (00:00)	61 % (07:54)	16 % (02:05)	23 % (03:03)	1	13:03
walking	3	0 % (00:00)	62 % (14:33)	19 % (04:31)	18 % (04:13)	1	23:17
sitting	4	0 % (00:00)	58 % (14:50)	21 % (05:16)	21 % (05:21)	1	25:28
jumping	5	0 % (00:00)	61 % (07:17)	18 % (02:06)	21 % (02:33)	1	11:57
Total Annotated	-	0 % (00:00)	60 % (59:24)	20 % (19:29)	20 % (19:56)		98:50
Unlabeled Data	0	0 % (00:00)	76 % (01:10)	13 % (00:11)	11 % (00:10)	1	01:32
Total Data		0 % (00:00)	60 % (60:34)	20 % (19:41)	20 % (20:06)		100:22

Name	Set	Status	Active Label Track	Active Data Track	Dimensionality
2023-07-11_01-03-46	Validation	✓	label		3
2023-07-11_01-05-19	Validation	✓	label		3
2023-07-11_01-44-09	Validation	✓	label		3
2023-07-11_01-46-13	Train	✓	label		3
2023-07-11_01-47-38	Train	✓	label		3
2023-07-11_02-06-41	Train	✓	label		3

Each data set (captured data, label, and video) is considered its own session and gets an associated *.imsession* file associated with it. These session files can be opened to review the data and ensure the video, data, and labels are all aligned with each other.

If there is a discrepancy with the label it can be fixed within the tool. If the data and video are out of sync, you may need to recapture the data.

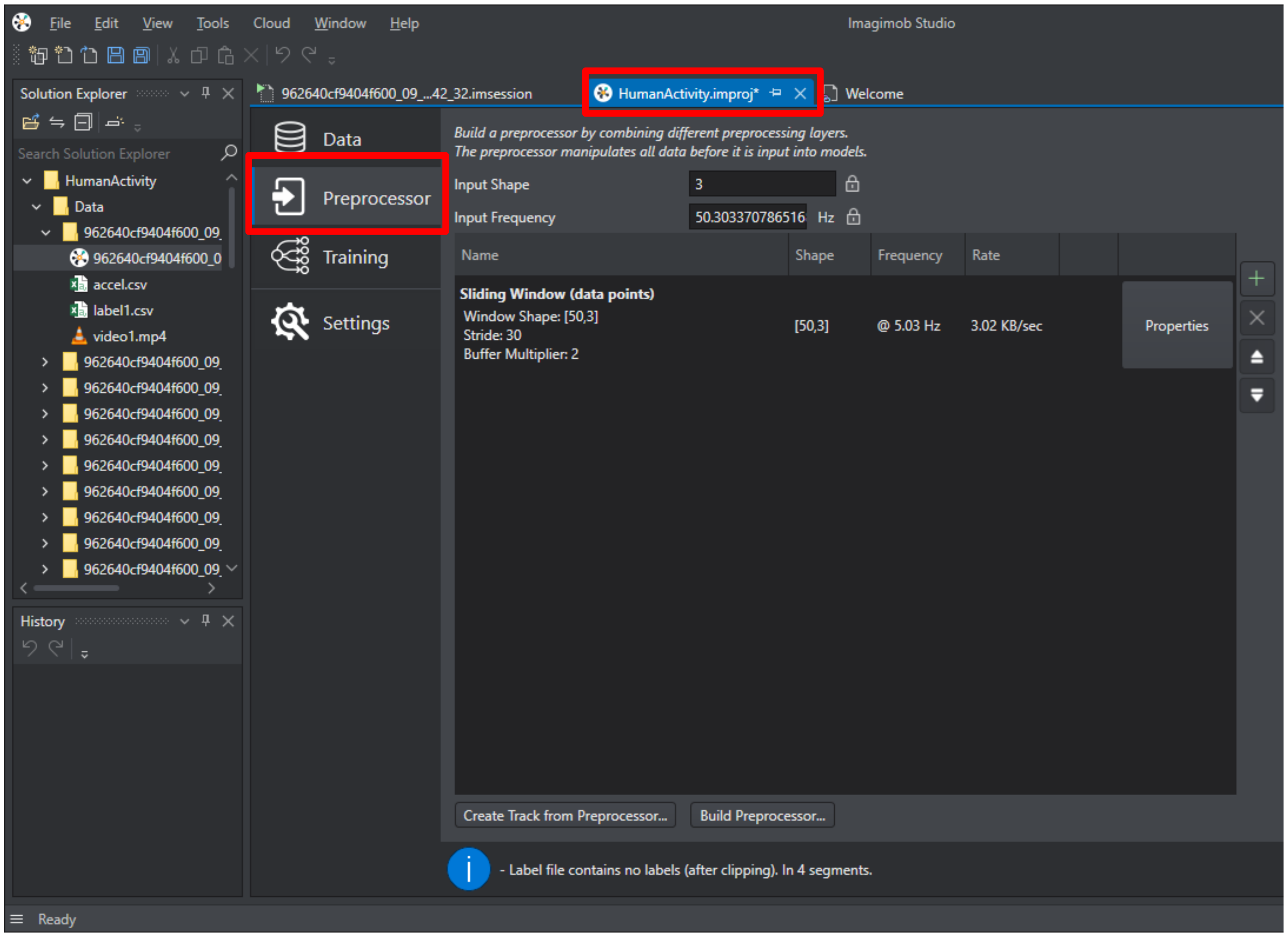


When the data all looks good, and you feel you have a sufficiently robust data set to start trying out models, return to the project configuration and **Rescan** for changes to the data. Finally, **Redistribute sets** to ensure there is some data for each label available to the training, validation, and testing phases. The tool will automatically assign items to provide a 60:20:20 balance. This can be adjusted if desired.

4.2.3 Build and train models

Once the dataset is complete, it's time to create and train a model. For these steps you will continue to use Imagimob Studio but will also rely on the Imagimob Cloud for the computationally intense training.

The first step is to define any pre-processing operations necessary. This can reshape the data before it is passed into the model and is thus important to do first so the input layer can be created with the right shape. Some preprocessing operations that might be considered are to create a Sliding Window (e.g. for repetitive motion data), FFT Shift, or some type of smoothing function (e.g. for jittery motion data).



Build a preprocessor by combining different preprocessing layers.
The preprocessor manipulates all data before it is input into models.

Input Shape: 3
Input Frequency: 50.303370786516 Hz

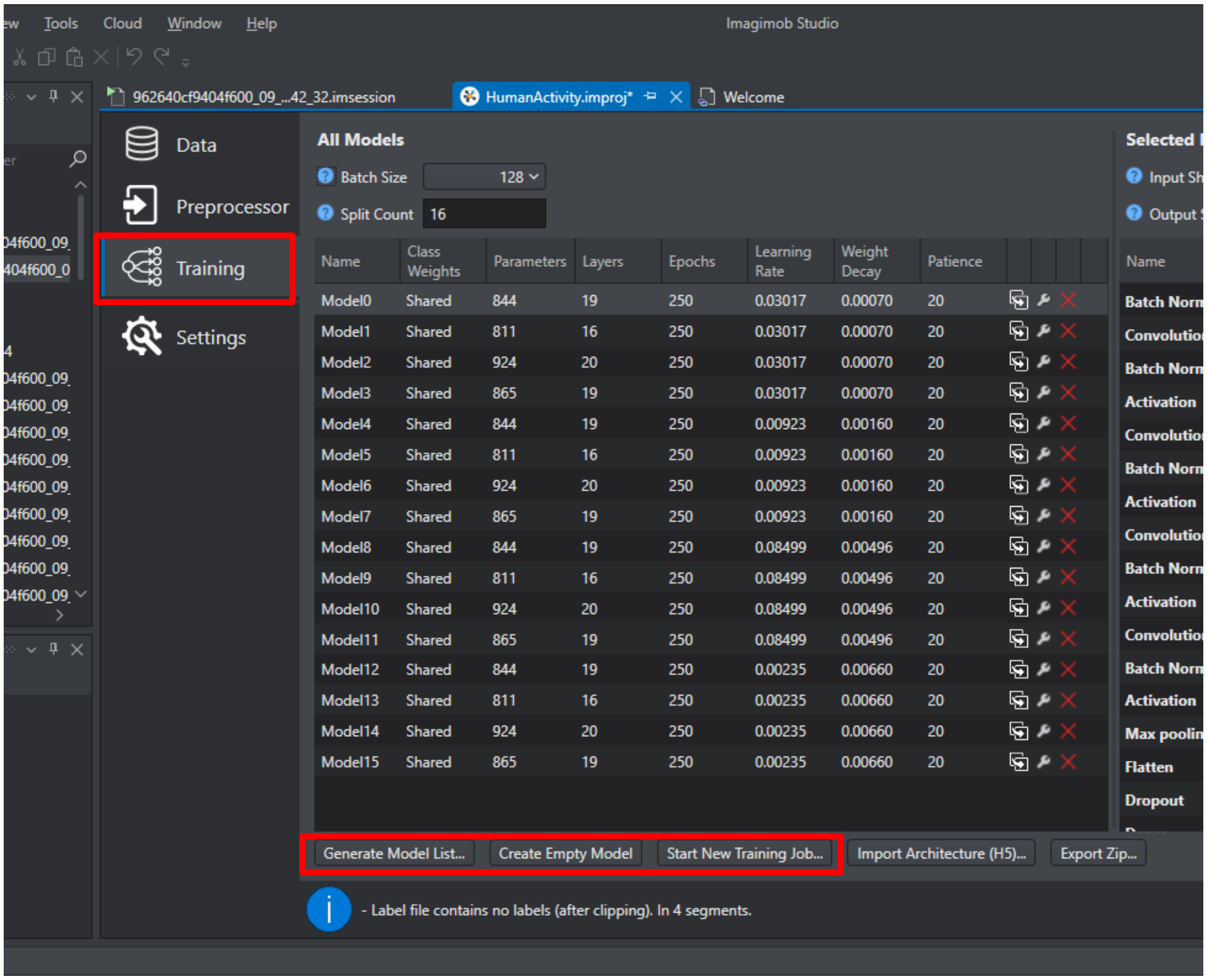
Name	Shape	Frequency	Rate
Sliding Window (data points) Window Shape: [50,3] Stride: 30 Buffer Multiplier: 2	[50,3]	@ 5.03 Hz	3.02 KB/sec

- Label file contains no labels (after clipping). In 4 segments.

After applying any preprocessor transformations to the data, it is ready to be fed into the training process. For this, switch to the **Training** tab.

By clicking **Generate Model List**, the tool can automatically create a set of models. These models will have different layers and layer configurations and thus will produce slightly different results. You are always free to create a custom model by clicking **Create Empty Model**.

All models can be trained and evaluated to see what produces the best results. The training is started by clicking the **Start New Training Job**. This will upload all the data and model information to the Imagimob Cloud to do the actual training. As jobs finish, you can review the performance.



All Models

Batch Size 128

Split Count 16

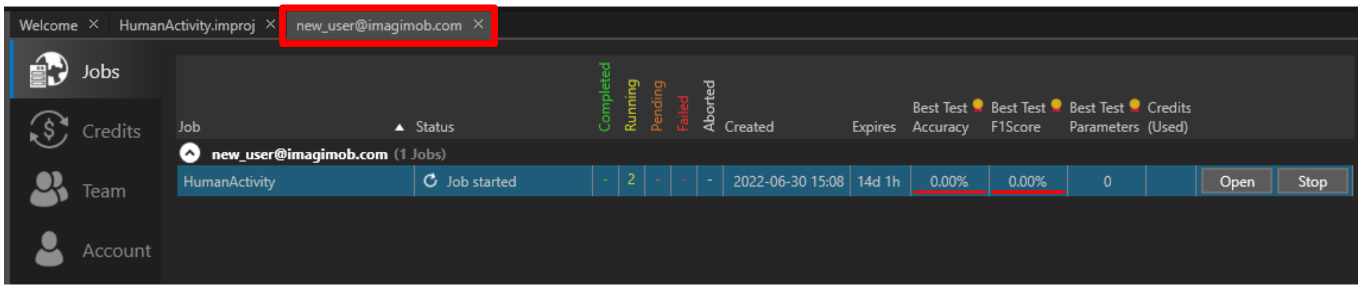
Name	Class Weights	Parameters	Layers	Epochs	Learning Rate	Weight Decay	Patience
Model0	Shared	844	19	250	0.03017	0.00070	20
Model1	Shared	811	16	250	0.03017	0.00070	20
Model2	Shared	924	20	250	0.03017	0.00070	20
Model3	Shared	865	19	250	0.03017	0.00070	20
Model4	Shared	844	19	250	0.00923	0.00160	20
Model5	Shared	811	16	250	0.00923	0.00160	20
Model6	Shared	924	20	250	0.00923	0.00160	20
Model7	Shared	865	19	250	0.00923	0.00160	20
Model8	Shared	844	19	250	0.08499	0.00496	20
Model9	Shared	811	16	250	0.08499	0.00496	20
Model10	Shared	924	20	250	0.08499	0.00496	20
Model11	Shared	865	19	250	0.08499	0.00496	20
Model12	Shared	844	19	250	0.00235	0.00660	20
Model13	Shared	811	16	250	0.00235	0.00660	20
Model14	Shared	924	20	250	0.00235	0.00660	20
Model15	Shared	865	19	250	0.00235	0.00660	20

- Label file contains no labels (after clipping). In 4 segments.

If you want more control of a model, you can add/remove/edit any of the layers within the model. This allows complete control of the pipeline to potentially get even better results.

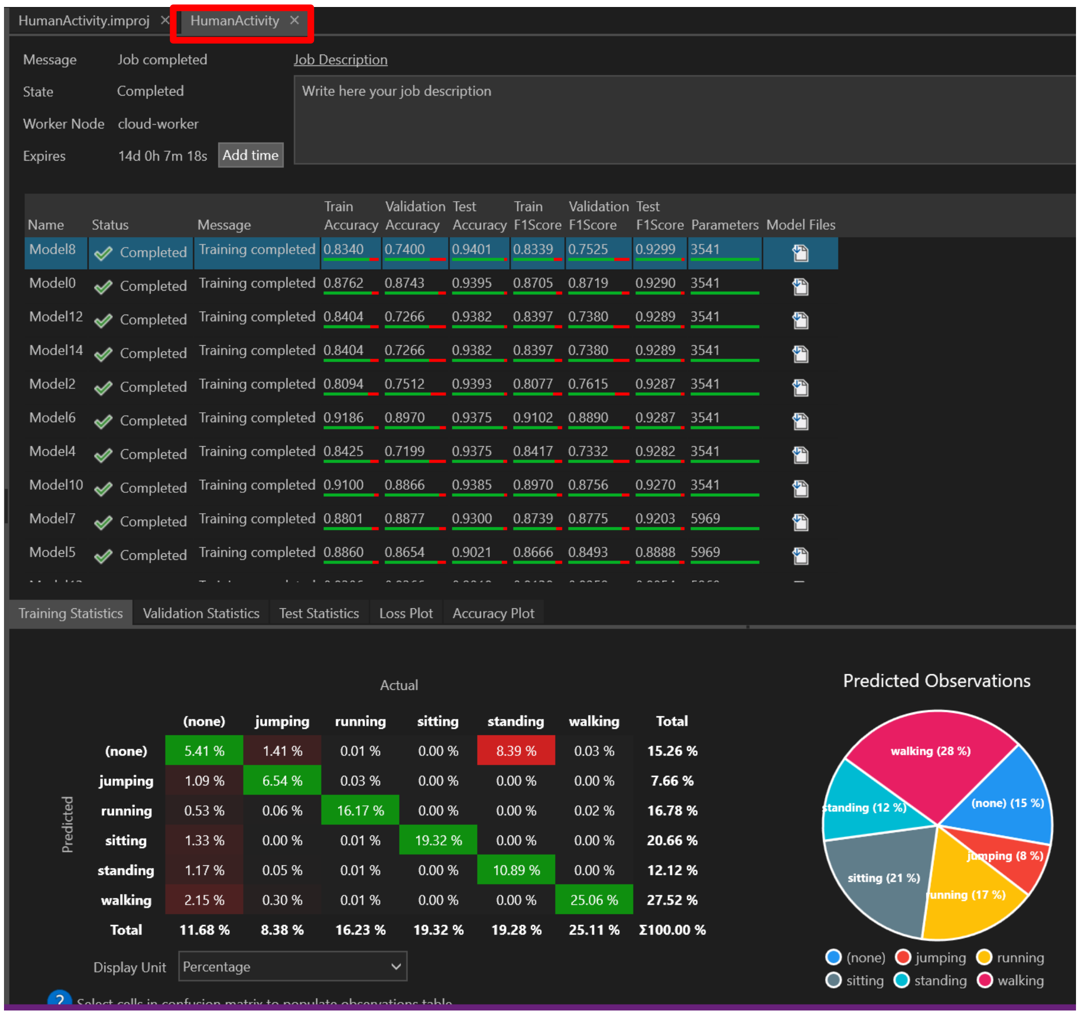
4.2.4 Evaluate and find the best model

The training progress can be monitored by opening the cloud connection. This shows all the projects that you have created and provides a summary of the training status for each project. Clicking on any project will open it to view the result details.



Job	Status	Completed	Running	Pending	Failed	Aborted	Created	Expires	Best Test Accuracy	Best Test F1Score	Best Test Parameters	Credits (Used)
new_user@imagimob.com (1 Jobs)												
HumanActivity	Job started	-	2	-	-	-	2022-06-30 15:08	14d 1h	0.00%	0.00%	0	

After the training jobs have completed running, any model that was included as part of the training set will have performance data available. This includes the accuracy and F1 score for each phase (Train, Validate, Test). It also provides details about the parameters/complexity of the model. Selecting any model will provide a more detailed view showing how it did for each label within a convolution matrix. You can look at charts that show how well the model distinguishes between data points, how well it performs against the test data, what features are used, and what layers are used, etc.



Message	Job completed
State	Completed
Worker Node	cloud-worker
Expires	14d 0h 7m 18s

Job Description

Write here your job description

Name	Status	Message	Train Accuracy	Validation Accuracy	Test Accuracy	Train F1Score	Validation F1Score	Test F1Score	Parameters	Model Files
Model8	✓ Completed	Training completed	0.8340	0.7400	0.9401	0.8339	0.7525	0.9299	3541	
Model0	✓ Completed	Training completed	0.8762	0.8743	0.9395	0.8705	0.8719	0.9290	3541	
Model12	✓ Completed	Training completed	0.8404	0.7266	0.9382	0.8397	0.7380	0.9289	3541	
Model14	✓ Completed	Training completed	0.8404	0.7266	0.9382	0.8397	0.7380	0.9289	3541	
Model2	✓ Completed	Training completed	0.8094	0.7512	0.9393	0.8077	0.7615	0.9287	3541	
Model6	✓ Completed	Training completed	0.9186	0.8970	0.9375	0.9102	0.8890	0.9287	3541	
Model4	✓ Completed	Training completed	0.8425	0.7199	0.9375	0.8417	0.7332	0.9282	3541	
Model10	✓ Completed	Training completed	0.9100	0.8866	0.9385	0.8970	0.8756	0.9270	3541	
Model7	✓ Completed	Training completed	0.8801	0.8877	0.9300	0.8739	0.8775	0.9203	5969	
Model5	✓ Completed	Training completed	0.8860	0.8654	0.9021	0.8666	0.8493	0.8888	5969	

Actual

Predicted	(none)	jumping	running	sitting	standing	walking	Total
(none)	5.41 %	1.41 %	0.01 %	0.00 %	8.39 %	0.03 %	15.26 %
jumping	1.09 %	6.54 %	0.03 %	0.00 %	0.00 %	0.00 %	7.66 %
running	0.53 %	0.06 %	16.17 %	0.00 %	0.00 %	0.02 %	16.78 %
sitting	1.33 %	0.00 %	0.01 %	19.32 %	0.00 %	0.00 %	20.66 %
standing	1.17 %	0.05 %	0.01 %	0.00 %	10.89 %	0.00 %	12.12 %
walking	2.15 %	0.30 %	0.01 %	0.00 %	0.00 %	25.06 %	27.52 %
Total	11.68 %	8.38 %	16.23 %	19.32 %	19.28 %	25.11 %	Σ100.00 %

Display Unit: Percentage

Select cells in confusion matrix to populate observations table

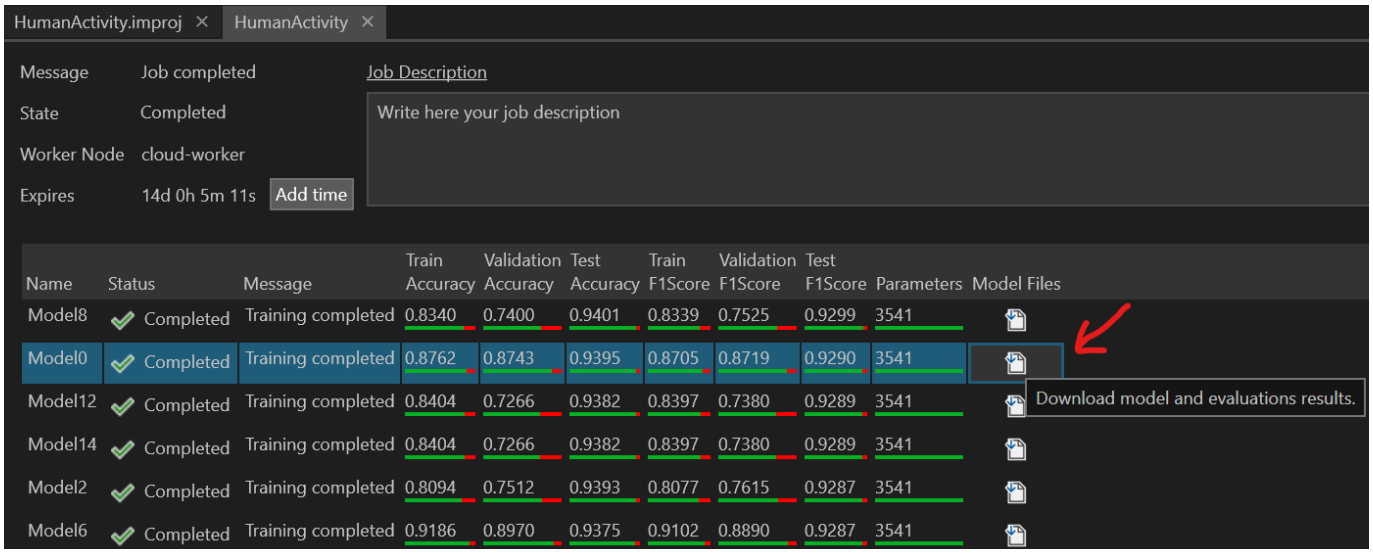
4.2.5 Optimize and package

Depending on the overall satisfaction with the results provided by the training there are a few options:

- Gather more data.
- Adjust some of the data.
- Modify the pre-processor options.
- Create or modify one or more models.
- Accept one of the models as-is.
- Use one of the models as input to the ModusToolbox™ ML configurator to perform additional optimization.

For any of the first four, simply go to the prior step, make adjustments at that point, and then proceed through again. The rest of this section will focus on the moving forward with a selected model as-is or as an input to the ModusToolbox™ ML configurator.

By clicking on the icon in the Model Files column for the model you want to proceed with, a menu appears allowing you to download the files.

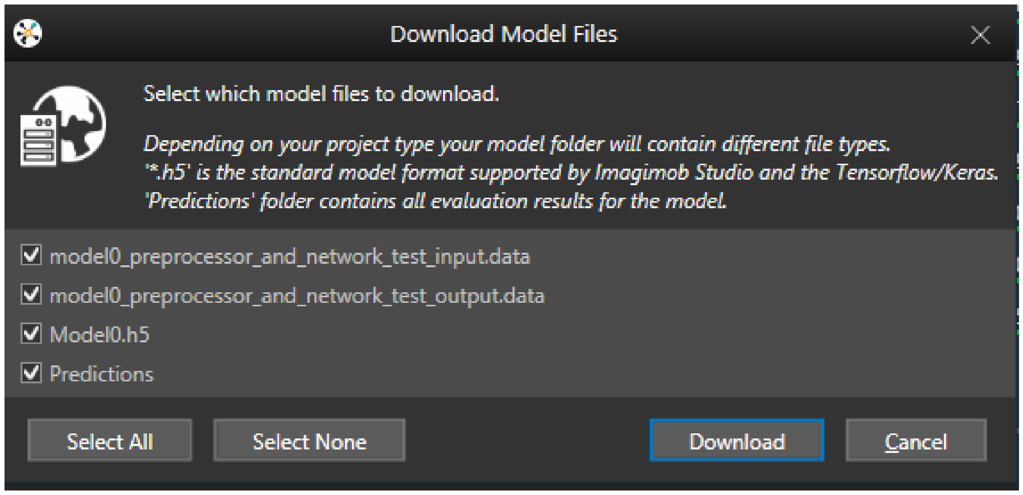


Message	Job completed
State	Completed
Worker Node	cloud-worker
Expires	14d 0h 5m 11s

Job Description

Write here your job description

Name	Status	Message	Train Accuracy	Validation Accuracy	Test Accuracy	Train F1Score	Validation F1Score	Test F1Score	Parameters	Model Files
Model8	✓ Completed	Training completed	0.8340	0.7400	0.9401	0.8339	0.7525	0.9299	3541	
Model0	✓ Completed	Training completed	0.8762	0.8743	0.9395	0.8705	0.8719	0.9290	3541	
Model12	✓ Completed	Training completed	0.8404	0.7266	0.9382	0.8397	0.7380	0.9289	3541	
Model14	✓ Completed	Training completed	0.8404	0.7266	0.9382	0.8397	0.7380	0.9289	3541	
Model2	✓ Completed	Training completed	0.8094	0.7512	0.9393	0.8077	0.7615	0.9287	3541	
Model6	✓ Completed	Training completed	0.9186	0.8970	0.9375	0.9102	0.8890	0.9287	3541	



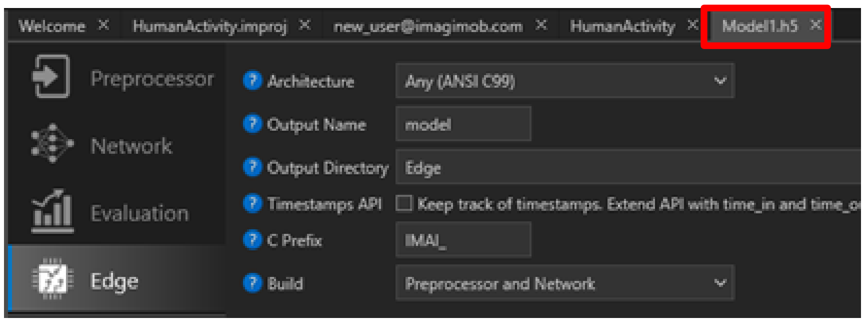
Select which model files to download.

Depending on your project type your model folder will contain different file types.
'.h5' is the standard model format supported by Imagimob Studio and the Tensorflow/Keras.*
'Predictions' folder contains all evaluation results for the model.

- ☑ model0_preprocessor_and_network_test_input.data
- ☑ model0_preprocessor_and_network_test_output.data
- ☑ Model0.h5
- ☑ Predictions

After downloading, the model (*.h5*) file can be opened in Imagimob Studio to view its details. This view provides four tabs:

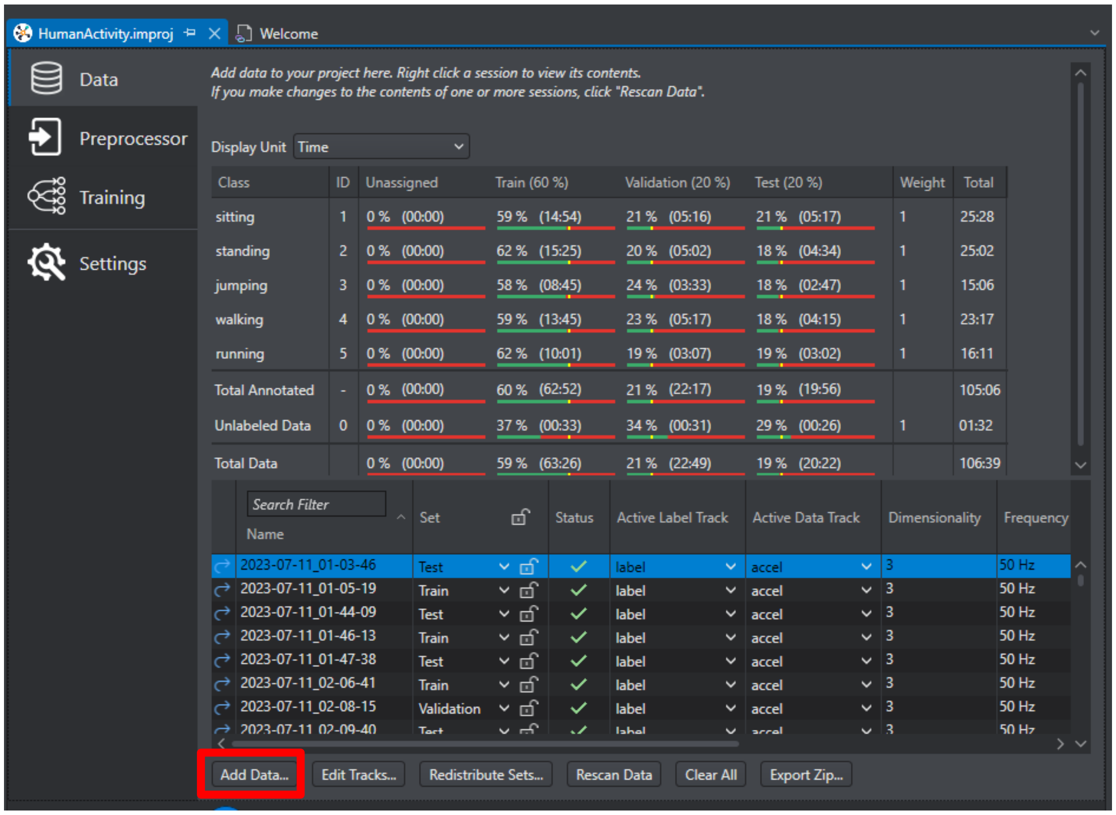
- Preprocessor: This is the same as the preprocessor information defined for the project
- Network: This provides details about the model's layers, activations and parameters
- Evaluation: This is the same performance data as was available from the cloud training
- Edge: This allows defining details about how the model should be deployed to the edge device



Architecture	Any (ANSI C99)
Output Name	model
Output Directory	Edge
Timestamps API	☐ Keep track of timestamps. Extend API with time_in and time_o
C Prefix	IMAI_
Build	Preprocessor and Network

4.2.6 Test the model

One of the items downloaded along with the Keras *.h5* model is a set of predictions. These can be used to evaluate the trained model by comparing the predictions the model makes vs. the original labels in the datasets. In Imagimob Studio, we can get a much more detailed view of the performance of a model by merging the model predictions as tracks into our sessions containing the original data. You can do this by using the data import tool of your project. Open the *.improj* file and navigate to the **Data** tab, then click the **Add Data** button.



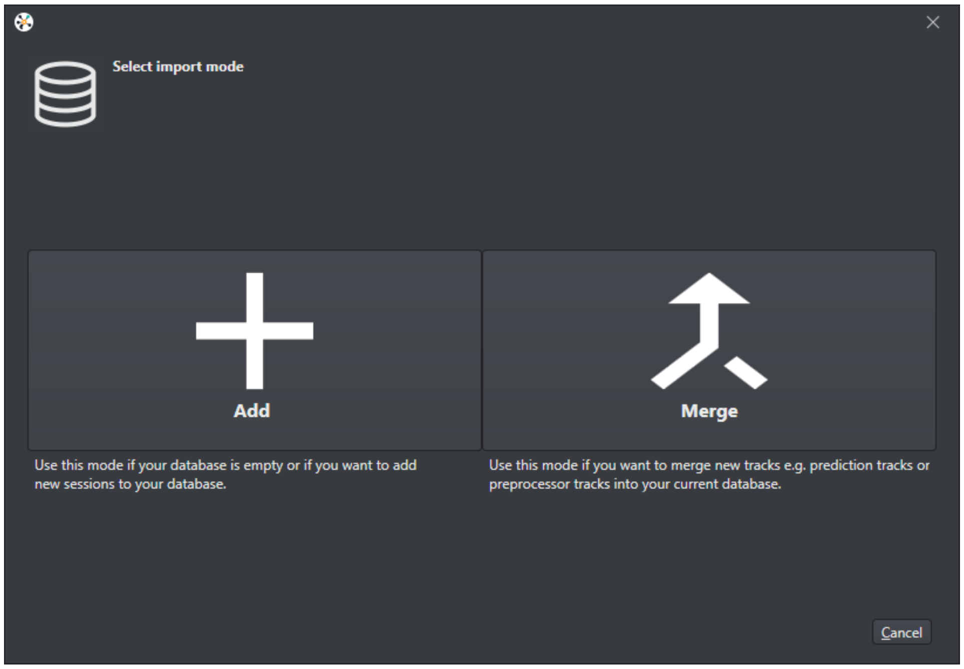
Add data to your project here. Right click a session to view its contents.
If you make changes to the contents of one or more sessions, click "Rescan Data".

Display Unit: Time

Class	ID	Unassigned	Train (60 %)	Validation (20 %)	Test (20 %)	Weight	Total
sitting	1	0 % (00:00)	59 % (14:54)	21 % (05:16)	21 % (05:17)	1	25:28
standing	2	0 % (00:00)	62 % (15:25)	20 % (05:02)	18 % (04:34)	1	25:02
jumping	3	0 % (00:00)	58 % (08:45)	24 % (03:33)	18 % (02:47)	1	15:06
walking	4	0 % (00:00)	59 % (13:45)	23 % (05:17)	18 % (04:15)	1	23:17
running	5	0 % (00:00)	62 % (10:01)	19 % (03:07)	19 % (03:02)	1	16:11
Total Annotated	-	0 % (00:00)	60 % (62:52)	21 % (22:17)	19 % (19:56)		105:06
Unlabeled Data	0	0 % (00:00)	37 % (00:33)	34 % (00:31)	29 % (00:26)	1	01:32
Total Data		0 % (00:00)	59 % (63:26)	21 % (22:49)	19 % (20:22)		106:39

Name	Set	Status	Active Label Track	Active Data Track	Dimensionality	Frequency
2023-07-11_01-03-46	Test	✓	label	accel	3	50 Hz
2023-07-11_01-05-19	Train	✓	label	accel	3	50 Hz
2023-07-11_01-44-09	Test	✓	label	accel	3	50 Hz
2023-07-11_01-46-13	Train	✓	label	accel	3	50 Hz
2023-07-11_01-47-38	Test	✓	label	accel	3	50 Hz
2023-07-11_02-06-41	Train	✓	label	accel	3	50 Hz
2023-07-11_02-08-15	Validation	✓	label	accel	3	50 Hz
2023-07-11_02-09-40	Test	✓	label	accel	3	50 Hz

In the dialog, select to **Merge** the prediction data with the existing data sets.

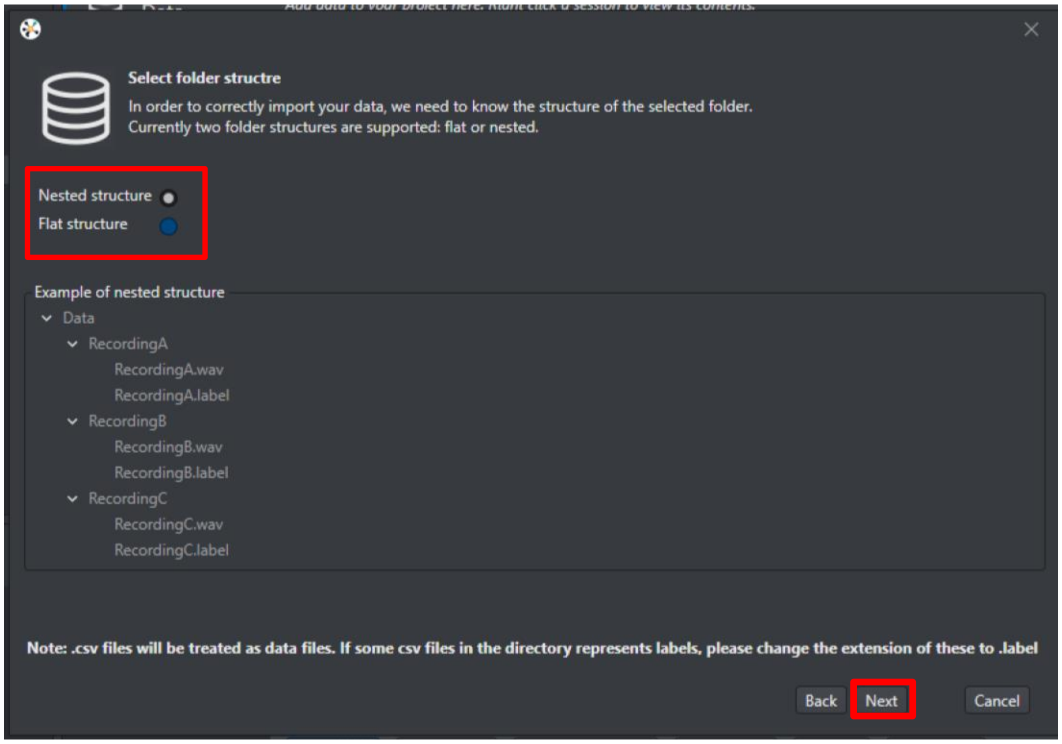


Use this mode if your database is empty or if you want to add new sessions to your database.

Use this mode if you want to merge new tracks e.g. prediction tracks or preprocessor tracks into your current database.

This will then bring up a navigation window to pick what data to merge in. For this, navigate to the Predictions folder for the model that was downloaded above. This will be in the project's Model directory. Once selected, click **Select Folder**.

In the subsequent dialog, select the appropriate structure of the prediction folder, typically it has the "Nested" structure and click **Next**.

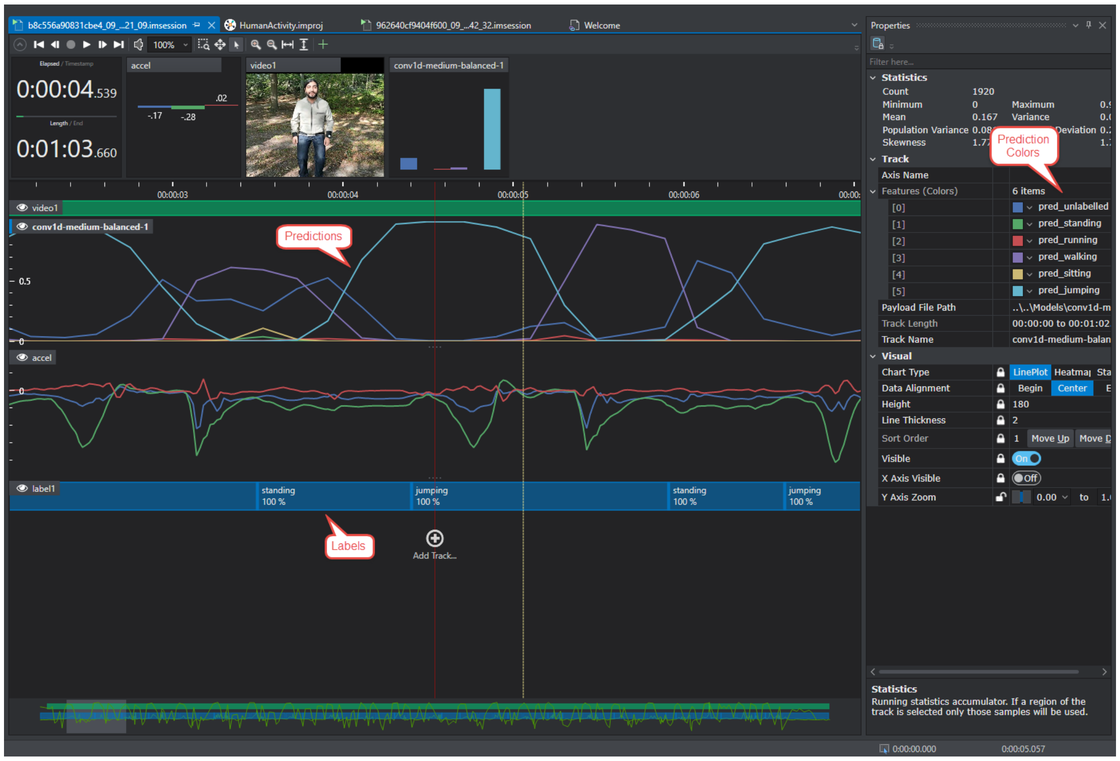


In order to correctly import your data, we need to know the structure of the selected folder. Currently two folder structures are supported: flat or nested.

Note: .csv files will be treated as data files. If some csv files in the directory represents labels, please change the extension of these to .label

We can see that the predictions from the *Predictions* folder we downloaded will be merged as tracks into the corresponding local sessions. In the dialog asking to select data and label tracks, click **Next**.

Now it's finally time to see how the imported model predictions compare to the labels from the training data. Open a session by double-clicking a session in the table of the **Data** tab of your project.



Properties

Statistics	
Count	1920
Minimum	0
Mean	0.167
Population Variance	0.08
Skewness	1.77
Track	
Axis Name	
Features (Colors)	6 items
[0]	🟦 pred_unlabelled
[1]	🟩 pred_standing
[2]	🟥 pred_running
[3]	🟪 pred_walking
[4]	🟨 pred_sitting
[5]	🟦 pred_jumping
Payload File Path	..\..\Models\conv1d-m
Track Length	00:00:00 to 00:01:02
Track Name	conv1d-medium-balan
Visual	
Chart Type	LinePlot
Data Alignment	Center
Height	180
Line Thickness	2
Sort Order	1
Visible	On
X Axis Visible	Off
Y Axis Zoom	0.00 to 1.

Statistics
Running statistics accumulator. If a region of the track is selected only those samples will be used.

In the main window, there is a combined plot of the labels, the model predictions (here "conv1d_medium_balanced_1"), and the original data. The model predictions are shown as solid lines with different colors depending on the class. The properties window shows what color maps to what prediction. This allows easily comparing the three 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 possibly the label is not correct. If the label and data both seem correct, it may indicate that the model was not trained on enough data to be as robust as needed.

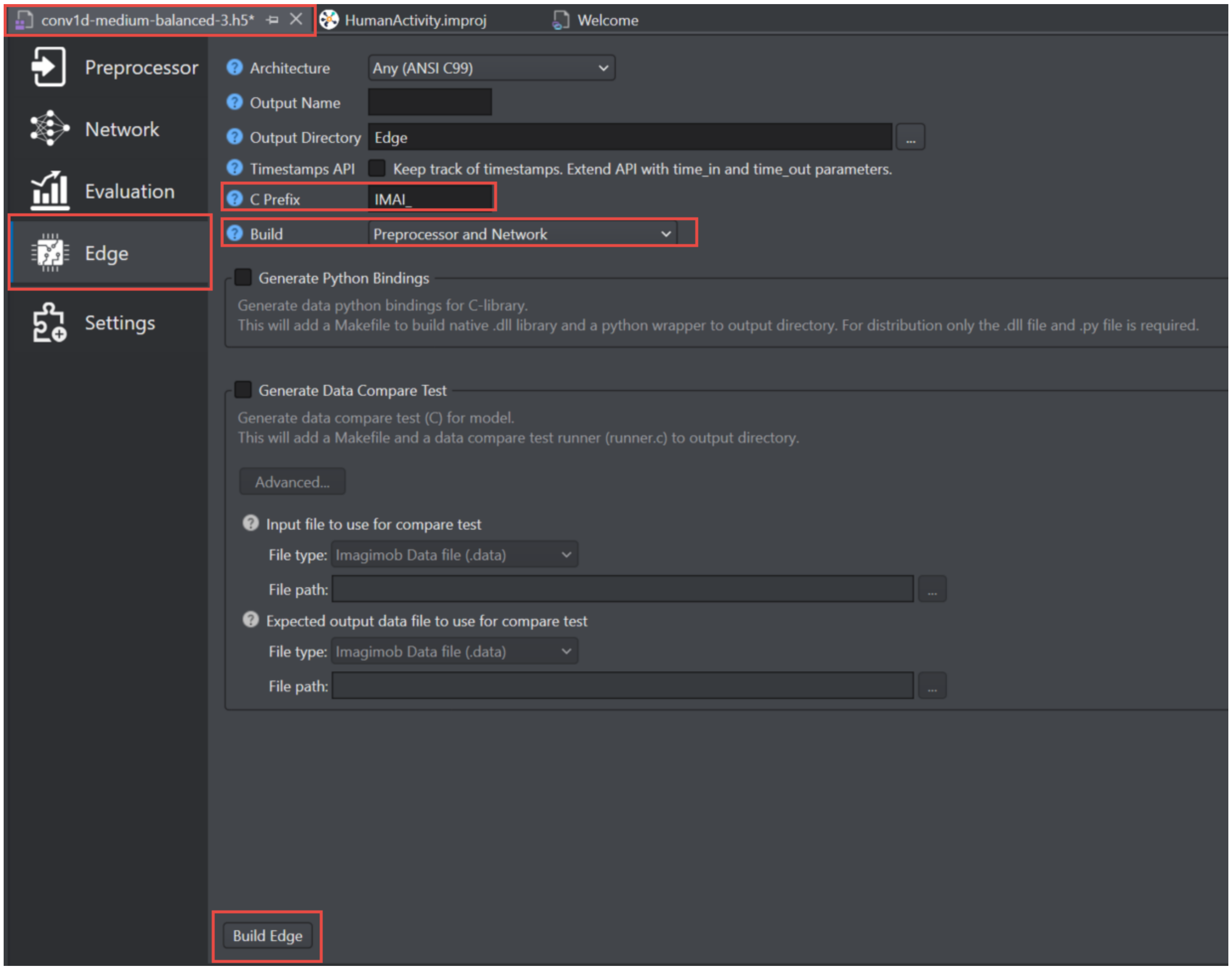
We can optionally hide some data tracks to analyze other tracks more easily.

4.2.7 Test on the device

The final step to verify the model is to test on the target device to make sure it behaves as expected. Imagimob models are fully self-contained and do not require the ModusToolbox™ Machine Learning Configurator, but it can be used to perform quantization and optimization if desired.

4.2.7.1 Use Imagimob model directly

If you want to use the Imagimob model directly, go to the tab containing the Keras *.h5* model and click **Build Edge** to generate a C model that can be used in your firmware. By selecting **Preprocessor and Network** you get *.c* and *.h* files that include any data preprocessing that you selected, such as a sliding window and the model itself.



Architecture: Any (ANSI C99)
Output Name:
Output Directory: Edge
Timestamps API: Keep track of timestamps. Extend API with time_in and time_out parameters.
C Prefix: IMAI_
Build: Preprocessor and Network

Generate Python Bindings
Generate data python bindings for C-library.
This will add a Makefile to build native .dll library and a python wrapper to output directory. For distribution only the .dll file and .py file is required.

Generate Data Compare Test
Generate data compare test (C) for model.
This will add a Makefile and a data compare test runner (runner.c) to output directory.

Input file to use for compare test
File type: Imagimob Data file (.data)
File path:
Expected output data file to use for compare test
File type: Imagimob Data file (.data)
File path:

The Imagimob Machine Learning Deploy (*mtb-example-ml-imagimob-deploy*) code example is available to drop in your model for testing.

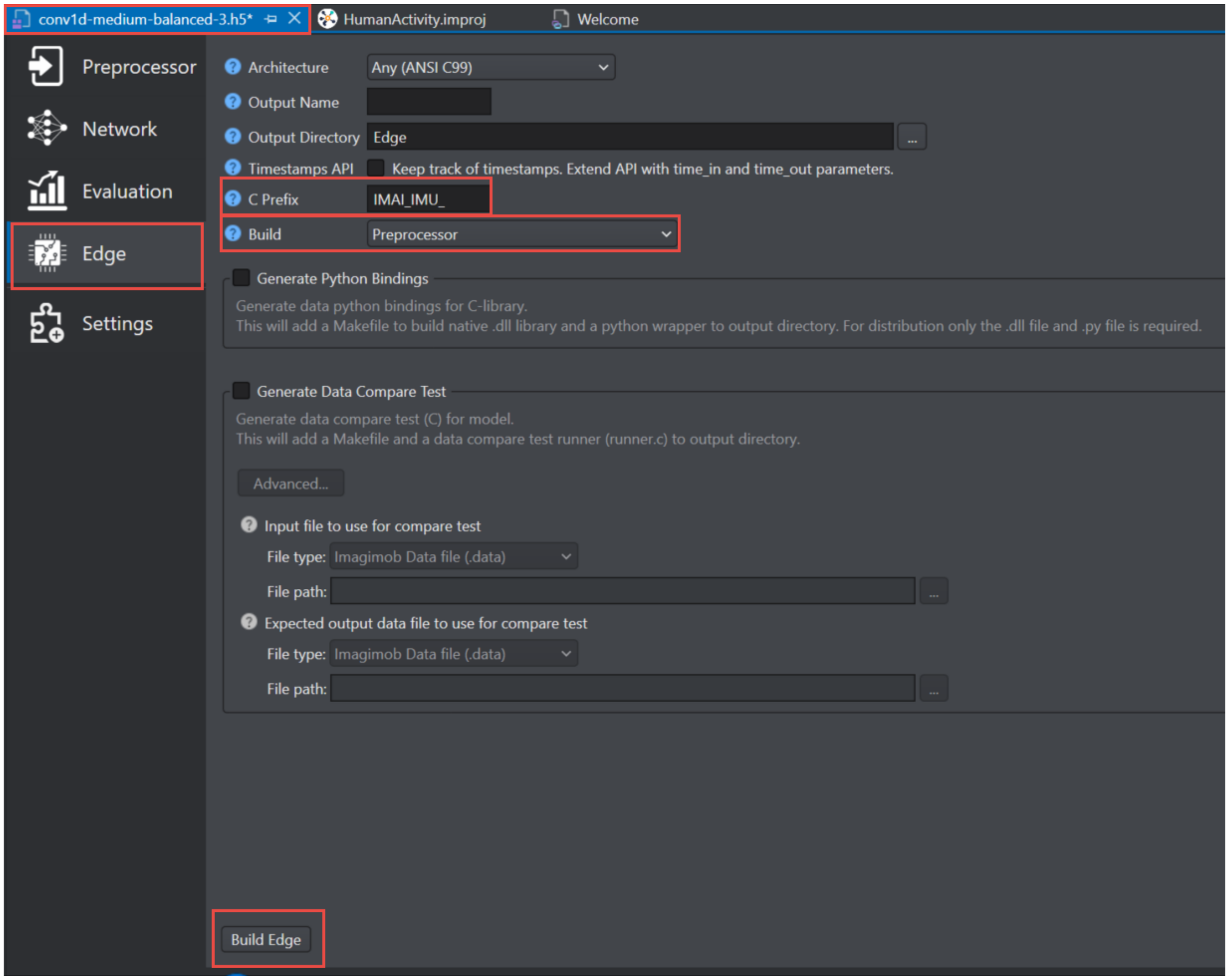
The code example assumes that the **C prefix** for the models is **IMAI_** as shown above.

Once you have the *.c* and *.h* files from Imagimob Studio, copy them to the *source* directory of the deploy application and replace the *model.c* and *model.h* files.

4.2.7.2 Use ModusToolbox™ ML configurator to optimize the model

If you want to try further optimization and quantization using the ModusToolbox™ ML Configurator, you can import the Keras *.h5* model from Imagimob into the ModusToolbox™ ML Configurator and optimize it as you learned in prior chapters.

In this case, you will use the model that is created by the ModusToolbox™ ML Configurator, but you still will need any preprocessing that was done to the data in Imagimiob, such as a sliding window. You can get that by selecting just **Preprocessor** when building files for the edge.



Architecture: Any (ANSI C99)
Output Name:
Output Directory: Edge
Timestamps API: Keep track of timestamps. Extend API with time_in and time_out parameters.
C Prefix: IMAI_IMU_
Build: Preprocessor

Generate Python Bindings
Generate data python bindings for C-library.
This will add a Makefile to build native .dll library and a python wrapper to output directory. For distribution only the .dll file and .py file is required.

Generate Data Compare Test
Generate data compare test (C) for model.
This will add a Makefile and a data compare test runner (runner.c) to output directory.

Input file to use for compare test
File type: Imagimob Data file (.data)
File path:
Expected output data file to use for compare test
File type: Imagimob Data file (.data)
File path:

When you want to test the optimized model on the device, you can use the Imagimob Machine Learning MTBML Deploy (*mtb-example-ml-imagimob-mtbml-deploy*) code example. The difference is that you use the C models that the ML configurator creates from the Keras *.h5* model instead of the C files that came directly from Imagimob Studio. The C files created from Imagimob are still used, but only for preprocessing of the data before sending it to the inference engine.

The code example assumes that the **C prefix** for the models is **IMAI_IMU_** (shown above) for motion detection applications and **IMAI_PDM_** for audio applications.

In the code example, the Imagimob API is used to run the pre-processing (e.g. `IMAI_IMU_enqueue`) while the ModusToobox™ API is used to run the inference engine (e.g. `mtb_ml_model_run`).

Once you have the Keras *.h5* model and the *.c/.h* files from Imagimob, copy them into the MTBML deploy application's *Edge/IMU* or *Edge/PDM* directory.

The code example supports hardware-specific optimizations including model quantization, sparsity support, and optimization for NPUs including NN-Lite or U55e.

4.2.8 Deployment

Once you have the final model, you will want to deploy it in your end application. You can either add your application's functionality to the code example that was used to verify the model previously or you can integrate the model into another application. Either way, it is useful to review how the model is used in the code example.

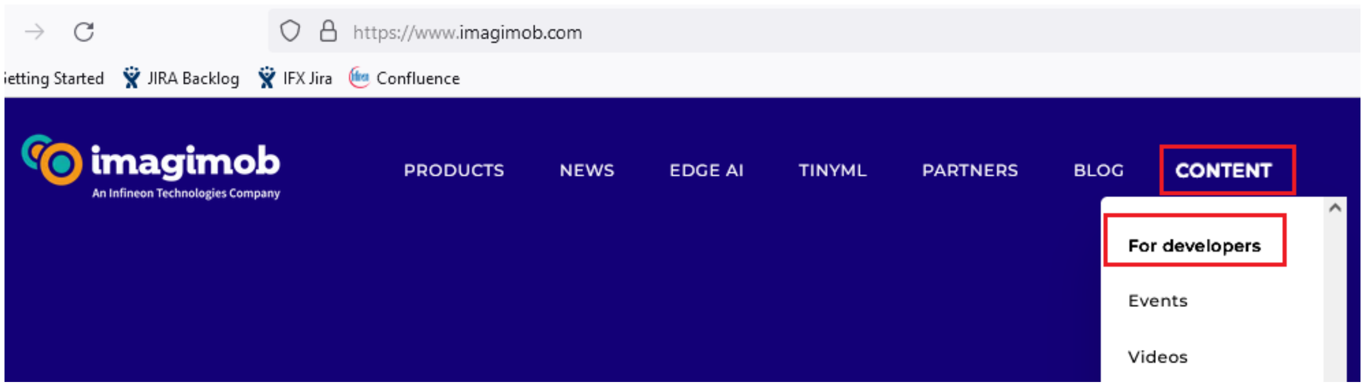
You can find documentation on the Imagimob API used to interface with the model from the website: https://developer.imagimob.com/edge-optimization/edge-api.

4.3 Getting Started

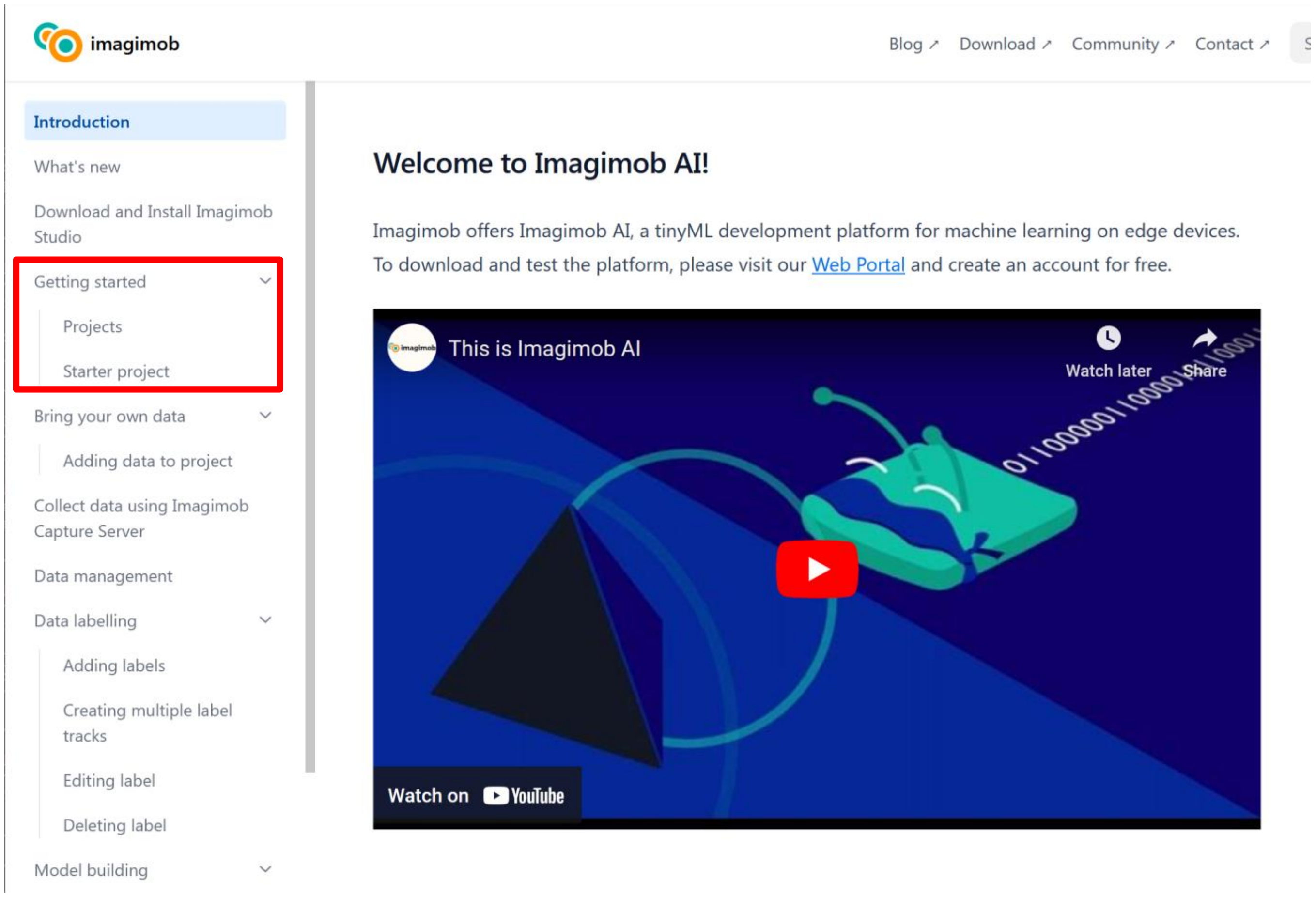
The Imagimob website has lots of useful documentation. Rather than repeating that information here, let's take advantage of their existing documentation. First, go to the Imagimob main webpage:

https://www.imagimob.com/

From that page, select **Content** and then **For Developers**.



The developer page includes links for Getting Started, Data management, Data labelling, Model building, and Edge optimization.

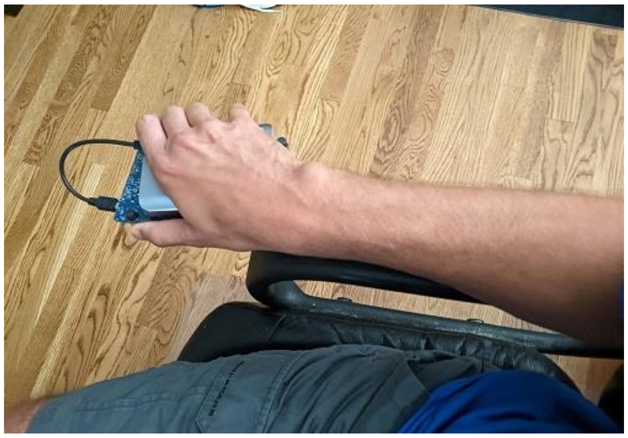


Welcome to Imagimob AI!

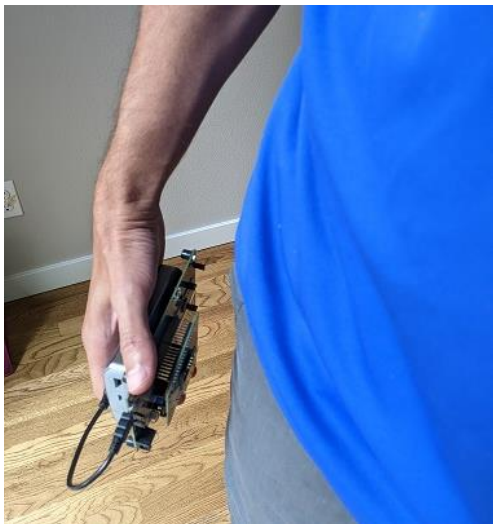
Imagimob offers Imagimob AI, a tinyML development platform for machine learning on edge devices. To download and test the platform, please visit our Web Portal and create an account for free.

The documentation provides a tutorial that goes through the entire Imagimob flow for one of the starter projects. The Human activity recognition project provides a good introduction as it uses the motion sensor to detect movement of the kit. It will detect the following movement conditions: sitting, standing, walking, running, jumping.

The orientation of the kit is important for proper motion detection. The existing data was collected based on the orientations described below. For sitting and standing the positions are shown below. For walking, running, and jumping the board is held the same as standing with the normal arm movements associated with the respective activity.



* Sitting: Kitprog USB facing forward, shield toward the ground



* Standing: Kitprog USB facing forward, shield toward the body

4.3.1 Account types

If you don’t already have one, you will create a free Imagimob evaluation account in the first exercise. This account never expires and supports many of Imagimob's tools and features limited to 300 cloud compute units on Imagimobs's servers. Don't worry – that’s plenty of time for completing the exercises in this chapter. For additional account options, visit: https://www.imagimob.com/contact-sales.

4.4 Exercises

Exercise 1: Install Imagimob tools

Note: *If you already did the installation in Chapter 1 you can skip this exercise.*

☐ 1. Create an Imagimob account: https://account.imagimob.com/signup.

☐ 2. Download and install Imagimob Studio.

Note: *Imagimob Studio is only available for Windows.*

☐ 3. Clone the Imagimob Capture Server https://bitbucket.org/imagimob/captureserver.

☐ 4. Install Python if it is not already installed on your system. It is recommended that you use Python version 3.11 or later.

Note: *It is recommended to go to https://www.python.org/downloads/ to download the latest version.*

☐ 5. Go to the *captureserver* repo directory and run `python setup.py install`.

Note: *Use the Command Prompt for this step if you are using Windows. Remember that is uses "\" instead of "/" for directory separation and "dir" instead of "ls" to list directory contents.*

☐ 6. Download and install the Imagimob Capture App on an Android phone.

Note: *The Imagimob Capture App is only available for Android.*

Exercise 2: Complete the Imagimob getting started tutorial

In this exercise, you will try out the full Imagimob flow by following the online walkthrough of the tools at: https://developer.imagimob.com/. It uses the full Imagimob development flow to create and train the model. Labeling is done using a manual labelling session.

The exercise will use the Human Activity starter project described at https://developer.imagimob.com/getting-started/starter-project. It already contains a set of data, labels, and models, but for learning, you will go through each of the steps to add additional data, label it, retrain the models, and test the best model on the device.

Note: *Detailed documentation for each step can be found along the left side of the https://developer.imagimob.com/ site. You can also refer to the Workflow section in this document.*

Collect data

☐ 1. Create the Imagimob Machine Learning Data Collection (mtb-example-ml-imagimob-data-collection) example in ModusToolbox™. Name the application **ch05_ex02_imagimob_collection**.

☐ 2. In the file *Makefile*, verify that the `DEFINES` variable has the correct shield selected.

☐ 3. If you are using a Mac, you may have to modify the following line in the *Makefile* to work with `csh` instead of `bash`. The necessary addition is a pair of single quotes.

Old: `PREBUILD+=sed -i 's/UINT8_C(0xD1)/UINT8_C(0xD8)/' bmi160/bmi160_defs.h;`
New: `PREBUILD+=sed -i  ''  's/UINT8_C(0xD1)/UINT8_C(0xD8)/' bmi160/bmi160_defs.h;`

☐ 4. Program the application to your kit. This will allow the kit to be used to collect accelerometer data for use in adding additional data to the Imagimob projects.

☐ 5. Use the ModusToolbox™ data collection code example with the Python `captureserver` to collect data for sitting and standing.

Note: *See the code example's README.md for instructions on how to use the data capture server to collect data.*

Note: *If you have more than one camera, you can use `--opencv-camera-index` with a value other than 0. The second camera will typically have an index of 1.*

Import data, set up models, train and export

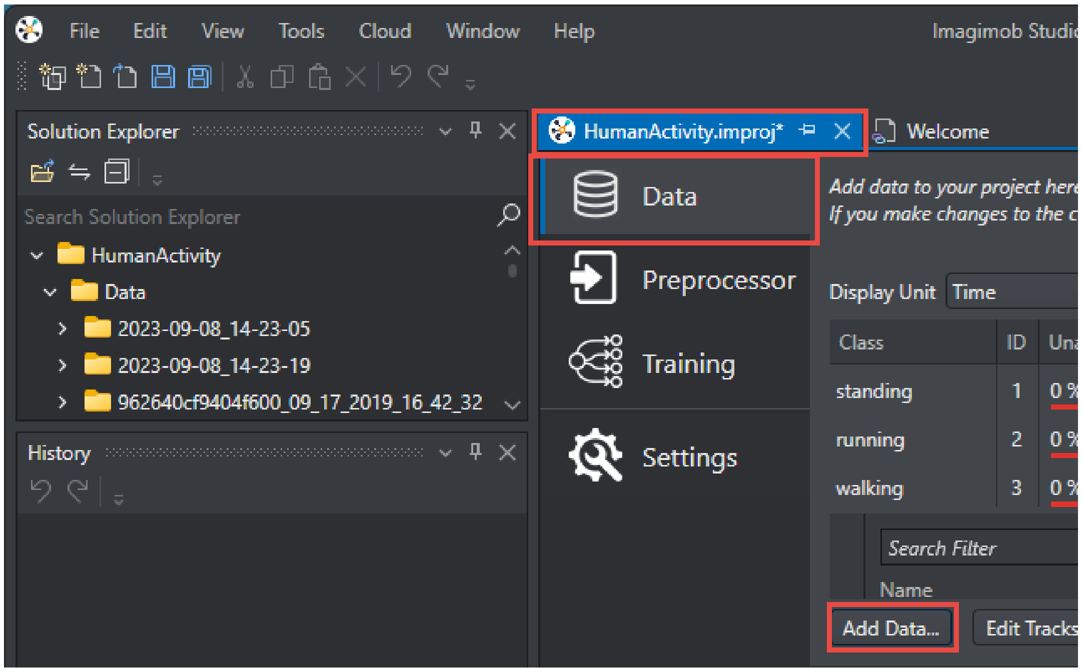
☐ 6. Create a new application in Imagimob Studio based on the **Human Activity** project.

Note: *Details can be found at: https://developer.imagimob.com/getting-started/workspace-project.*

☐ 7. Copy the data you collected in the previous section into the Imagimob project directory structure.

Note: *The default directory for the Imagimob project is C:\Imagimob\HumanActivity\Data*

☐ 8. Open the project file (**.improj*) and select the **Data** tab. Add the new data to the project by using the **Add Data** button at the bottom of the **Data** tab window.



☐ 9. Click the **Add** button from the **Select import mode** page and the select the **Data** folder. Select **Nested structure** and click **Next**.

☐ 10. Click **OK** in the Batch Import window. You will see a message that there are some segments with the same name already in the database. That’s because we told it to search the entire Data directory so any sessions that were already in the project are found again. Just click **Don’t Replace** to add just the new sessions.

☐ 11. When the import is done, open the *.imsession* file for each new data session and add the appropriate labels.

Note: *Hold the left mouse button down and drag to select an area to add a label. Then right click in the label track and select **New Label.** The starting time and duration will be set to the region you selected.*

Note: *Once a label is created, you can drag it around or drag its endpoints to change the duration.*

Note: *You can scroll by holding the left mouse button and dragging. The bar at the bottom of the screen shows which section of the file is currently shown.*

Note: *The play head is always centered in the window so if you want to play back from a different location, just scroll the screen.*

☐ 12. Redistribute the data by clicking on **Redistribute Sets** at the bottom of the **Data** tab window.

☐ 13. Set up data pre-processing on the **Preprocessor** tab.

Note: *In this case, just review the pre-processing that is already configured.*

☐ 14. Generate models from the **Training** tab by clicking **Generate Model List** at the bottom of the window.

Note: *Leave the values as-is but look at the different tabs (Auto ML, Training, Build Steps) to see the possibilities.*

☐ 15. Train models by clicking **Start New Training Job** at the bottom of the **Training** tab window.

☐ 16. Open the job when prompted to view training progress and results.

Note: *You can also use **Cloud > Connect to Imagimob Cloud or** the **Open Cloud** button at the top right corner of the window and then double-click the job name to see the progress.*

Note: *When you connect to the cloud, there is a tab to see how many credits you have remaining in your account.*

☐ 17. Download the best model back into Imagimob Studio by clicking the **Download** button for the model you want to use.

☐ 18. Evaluate the model in Imagimob Studio.

Note: *Double-click the .h5 file in the Models/<name>/Predictions directory and select the **Evaluation** tab.*

☐ 19. Generate the *.c/.h* model files for use on the device.

Note: *Switch to the **Edge** tab. Review the selections and the click **Build Edge**.*

Note: *On Windows, the default location for the downloaded edge model is C:/Imagimob/<project name>/Models/<model name>/Edge.*

Test on the device

☐ 20. Create the Imagimob Machine Learning Deploy (mtb-example-ml-imagimob-deploy) example in ModusToolbox™. Name the application **ch05_ex02_imagimob_model**.

☐ 21. Replace the *model.c* and *model.h* files in the application's *source* directory with the files that you downloaded.

☐ 22. Program the application to your kit and make sure the model is running properly.

Note: *Open a UART terminal with a baud rate of 115200 to see the model's predictions as you move the kit to mimic various motions.*

Exercise 3: Optimize the model via ModusToolbox™ ML

In this exercise, you will take the model generated above and run it through the ModusToolbox™ Machine Learning tools to further optimize it for the target device.

☐ 1. Create the Imagimob Machine Learning MTBML Deploy (mtb-example-ml-imagimob-mtbml-deploy) example in ModusToolbox™. Name the application **ch05_ex03_model_optimization**.

☐ 2. Take the Kara's h5 model file generated from the exercise above and add it to the newly created application.

Note: *The Human Activity model is in the application under Edge/IMU/human_activity.h5.*

☐ 3. Open the Machine Learning Configurator and make sure it is pointing to the model you copied in above.

☐ 4. Change the model to using 8-bit quantization instead of float.

☐ 5. **Generate Source** for the model to work with Tensor Flow Light Micro (TFLM).

☐ 6. Optional: Validate the model on the desktop or on the target to confirm it is working as expected.

☐ 7. Program the design onto the board and try it out to ensure it works as expected.

☐ 8. Compare the performance of the optimized model vs what was obtained in the exercise above. The size of the model is easy to compare just looking at the binary size. The performance can be measured by starting a timer when the inference operations begin and stopping it when they finish.

Note: *The optimized model makes use of specialized kernels for ARM devices and quantizes the model to use integer operations instead of floating point.*

Exercise 4: Run some application examples

In this exercise, you will try out some other application examples provided by Imagimob.

The various applications showcase different features of Imagimob such as model creation using full AutoML (which you used in exercise 2), the advanced tuning UI for more control, and the Imagimob Capture App/Server for complete control over the data.

☐ 1. Review the list of starter projects via Imagimob Studio. Try out one or more examples to see how different features of Imagimob are used. A few possibilities that may interest you are:

- The **Keyword Spotter** project recognizes keywords with a microphone and MCU/CPU.
- The **Fall Detection** project detects a fall with an IMU.
- The **Acoustic Recognition** project uses a microphone to sample and identify specific acoustic events.

Published by

Infineon Technologies AG

81726 Munich, Germany