

Using DEEPCRAFT™ to create, train and deploy a model for motion recognition

Application note

Versions

Version	Date	Rationale
1.0	September 25, 2025	First release. Authors: ROJ, KOA

Legal disclaimer

The evaluation board is for testing purposes only and, because it has limited functions and limited resilience, is not suitable for permanent use under real conditions. If the evaluation board is nevertheless used under real conditions, this is done at one's responsibility; any liability of Rutronik is insofar excluded.

Table of contents

Introduction	3
Collect data	3
Preparation	3
Recording and live labeling	6
Machine learning project	8
Project creation	8
Data import	9
Pre-processing	11
Training	12
Live evaluation	15
Deployment	17
Generate source code	17
Create new Modus Toolbox project and use it	18
Problems	19
Deployment with 5 Hz model	24
Add FreeRTOS support to the project and use tasks	25

Introduction

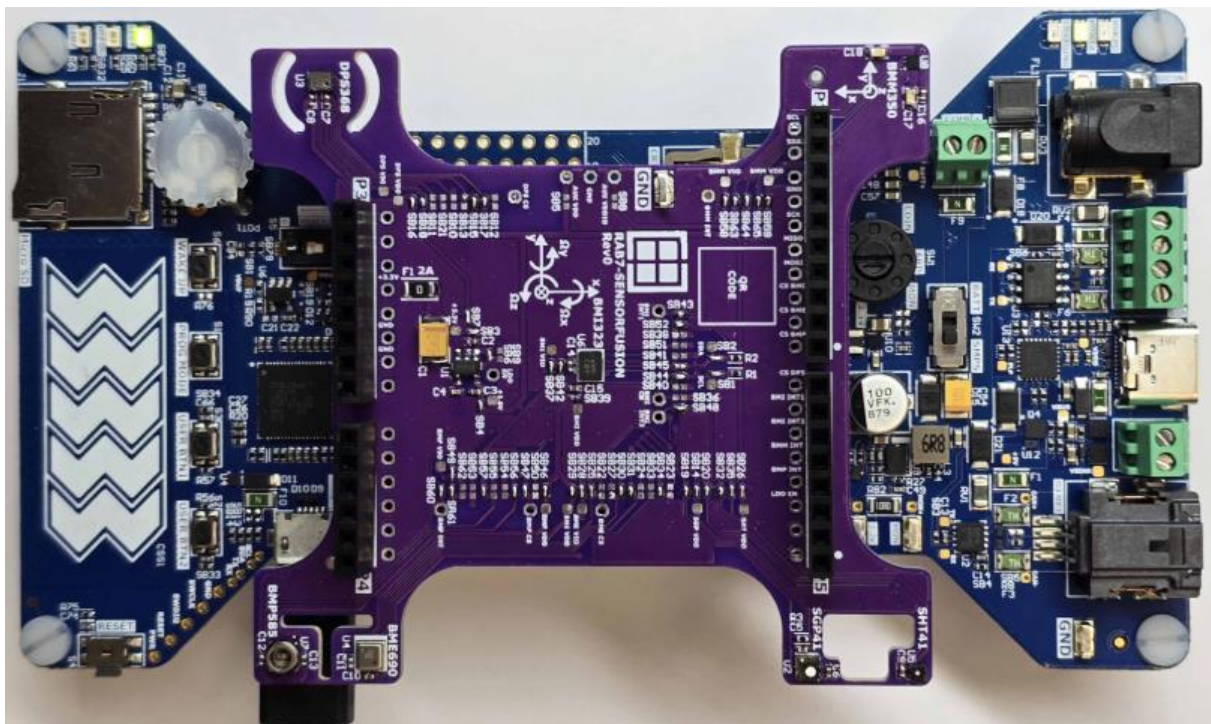
This application note describes how to develop a neural network for motion recognition and deploy it on the MCU using DEEPCRAFT™ software by Infineon.

The motion data is detected using RAB7 SensorFusion, the neural network is running on the MCU of RDK2.

Collect data

Preparation

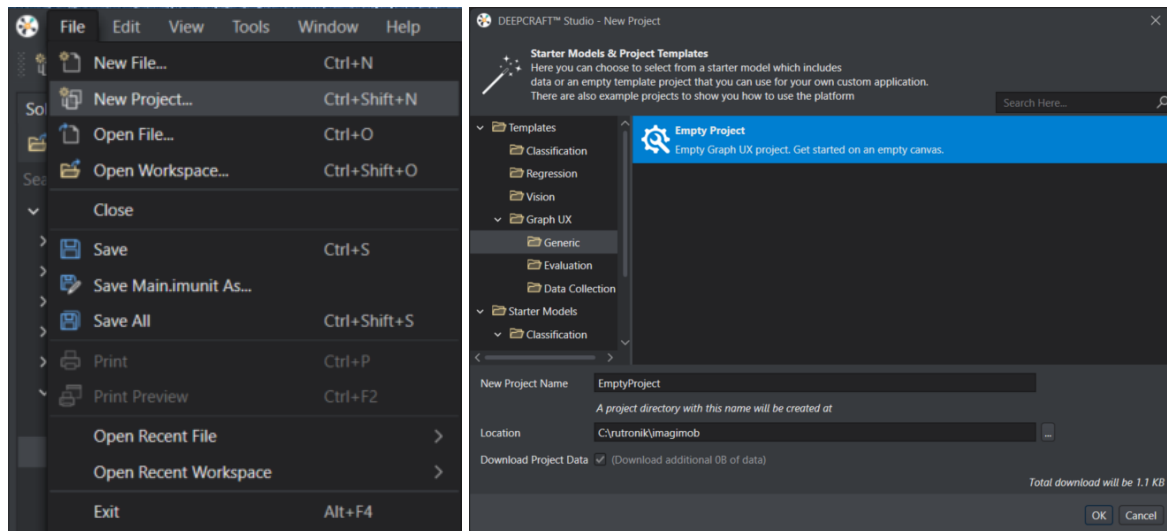
1. Get the required hardware: RDK2 + RAB7 SensorFusion.



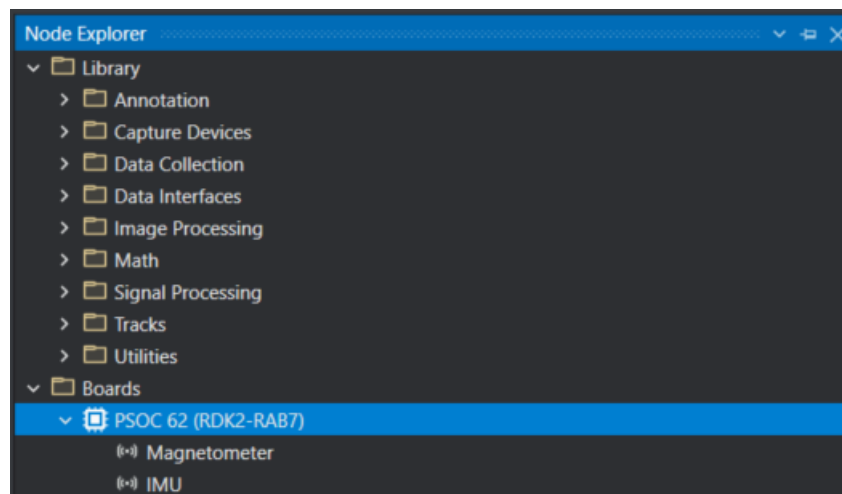
2. Run the following firmware on the RDK2:

https://github.com/RutronikSystemSolutions/RDK2_RAB7-SENSORFUSION-DEEPCRAFT-STREAMING-PROTOCOL

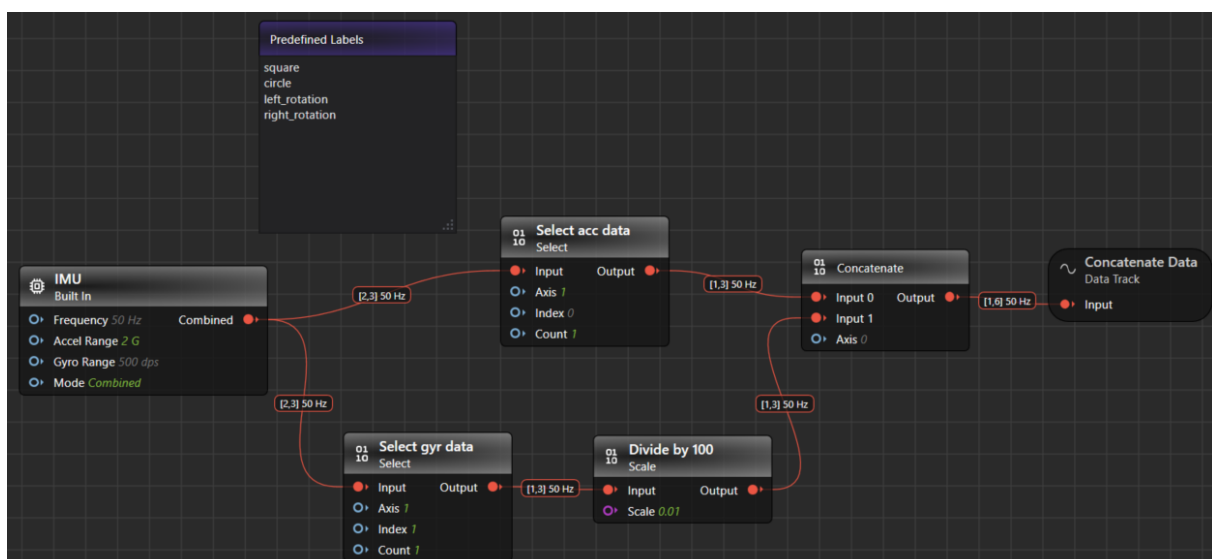
3. Run Deepcraft software and create a new **Graph UX** project:

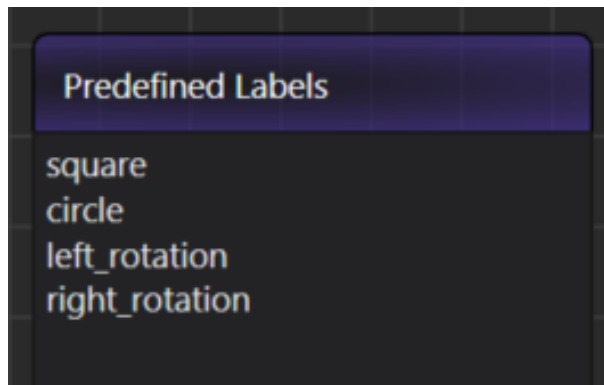


If your board is successfully detected, you will find it in the **Node Explorer**:



4. Drag and drop the **IMU** unit to the graph, and add following blocks (see details below):

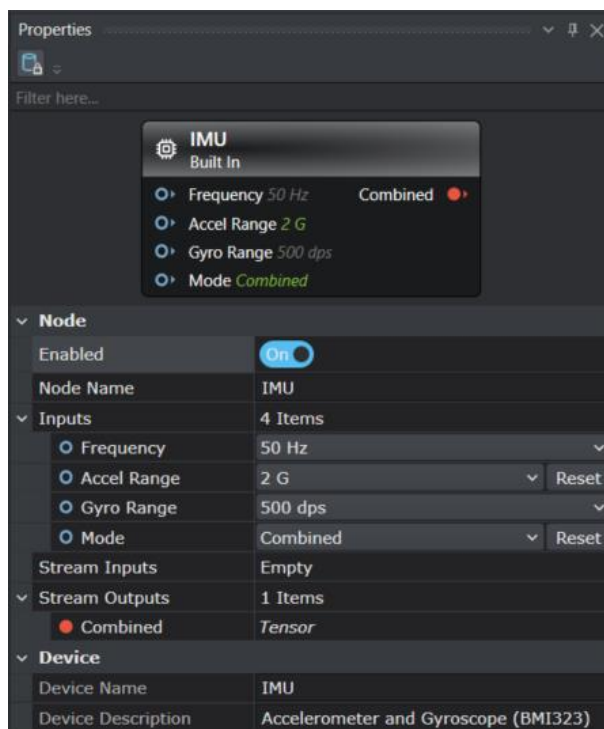




Predefined labels

This block defines the labels (different types of gesture in our case) that we wish to detect.

They will be used later, during the live labeling process.



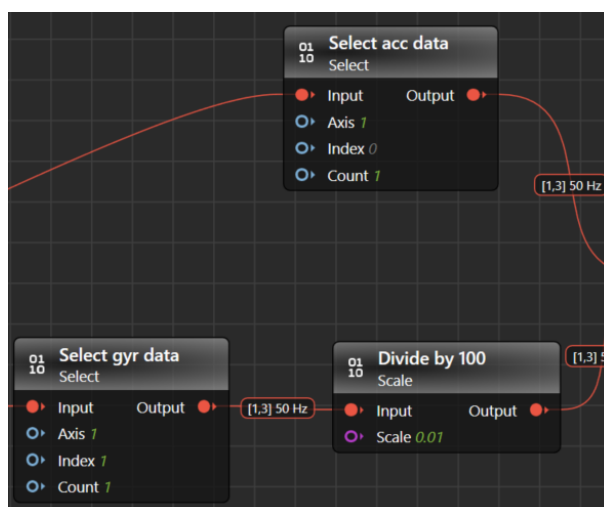
IMU

This block represents the IMU sensor (BMI323) of the RAB7 SensorFusion board.

It generates a matrix of dimension [2,3] at a frequency of 50 Hz.

The sensor generates 3 values for accelerometer and 3 values for gyroscope, that gives the [2,3] dimension.

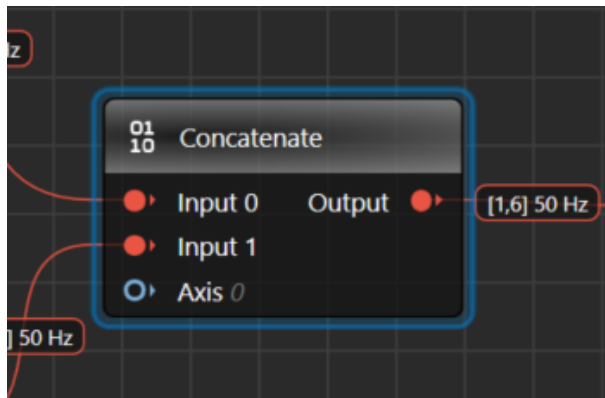
You can change the parameters in the **Properties** view.



The IMU generates a Matrix of dimension [2,3] containing **accelerometer data** (column 0) and **gyroscope data** (column 1).

The gyroscope values are much bigger than the accelerometer values.

It is good practice in machine learning problems, to normalize the data (i.e. give the same importance), therefore we **divide the gyroscope values by 100** (and only the gyroscope values).



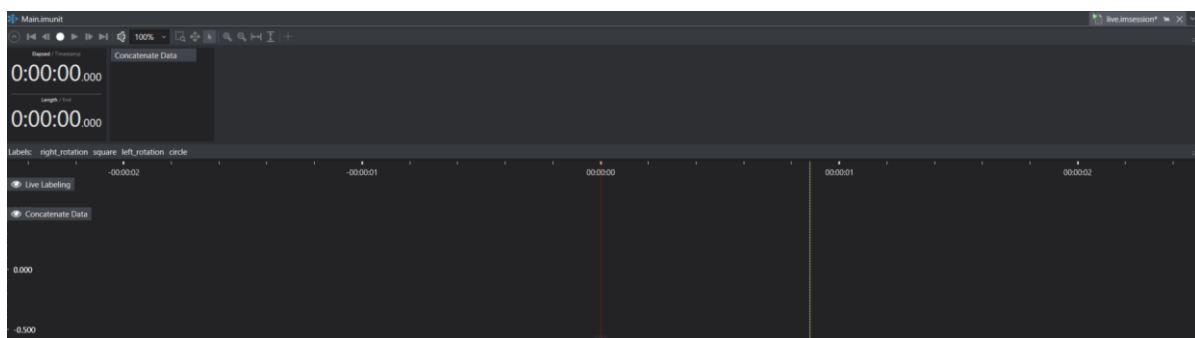
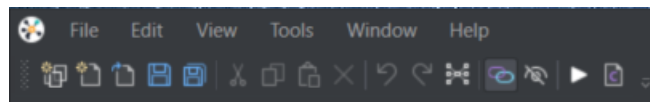
When the values are scaled, we have one array of 3 values containing gyroscope values and one array of 3 values containing accelerometer values.

To be stored and used, we need to **concatenate** those 2 arrays inside one.

The output dimension is an array containing 6 values.

Recording and live labeling

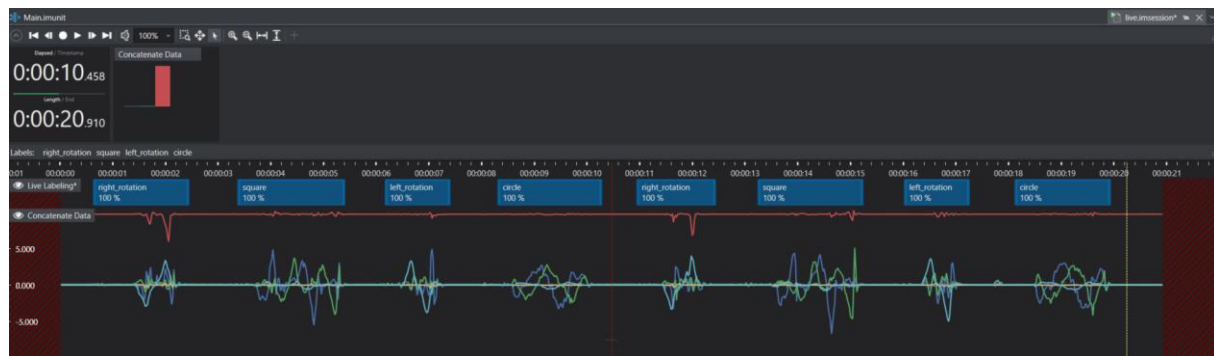
1. Press the **Start** button to start the recording/live labeling. This will open a "live.imsession" file.



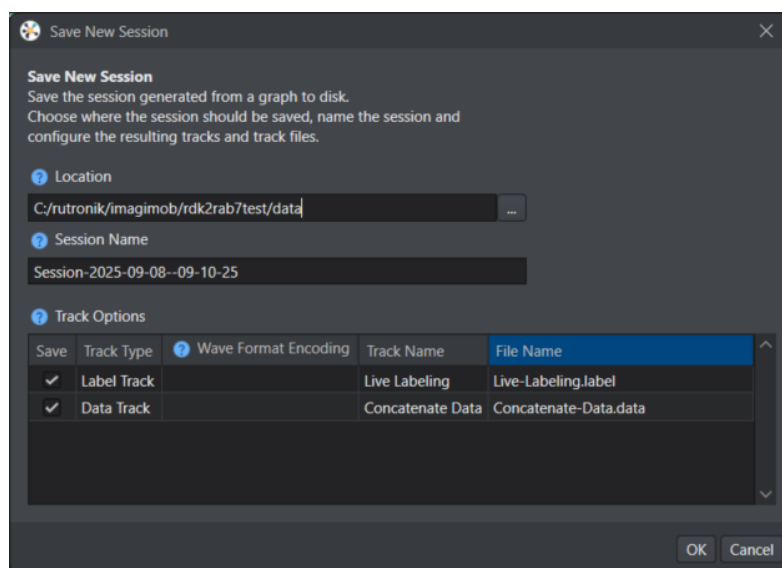
2. Click on the recording button to start the recording process.
3. To label a gesture, click on one of the labels in the bar, perform the gesture and then click again on the label.



Example for a session containing multiple gestures:



4. Save the session to import it later in the machine learning project.



Save New Session

Save the session generated from a graph to disk.
Choose where the session should be saved, name the session and configure the resulting tracks and track files.

Location
C:/rutronik/imagimob/rdk2rab7/test/data

Session Name
Session-2025-09-08--09-10-25

Track Options

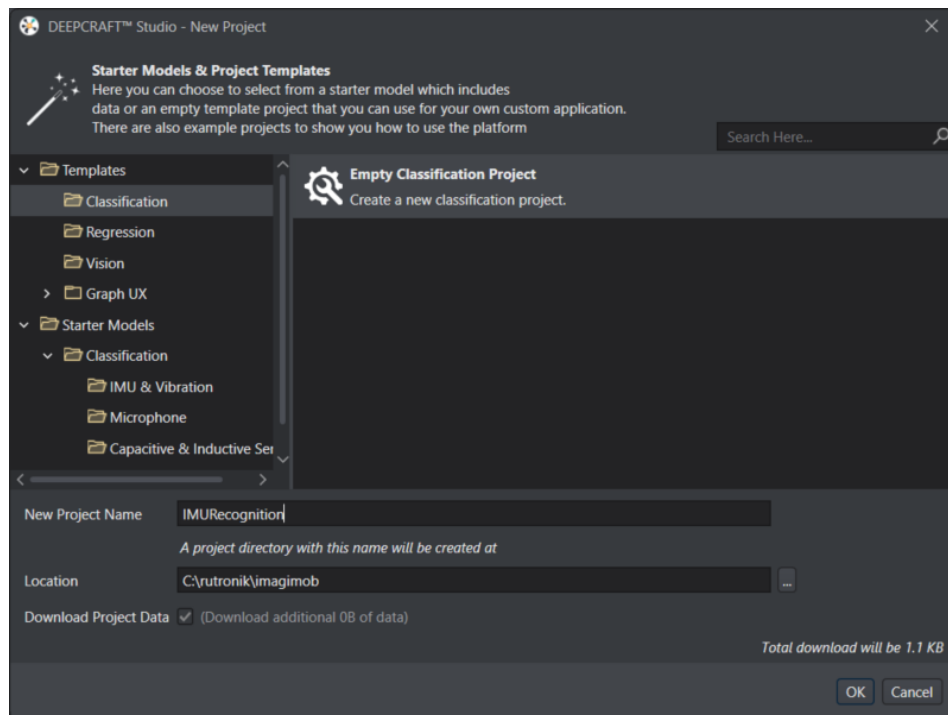
Save	Track Type	Wave Format Encoding	Track Name	File Name
<input checked="" type="checkbox"/>	Label Track		Live Labeling	Live-Labeling.label
<input checked="" type="checkbox"/>	Data Track		Concatenate Data	Concatenate-Data.data

OK Cancel

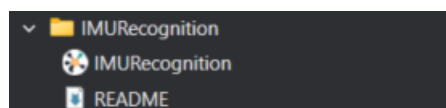
Machine learning project

Project creation

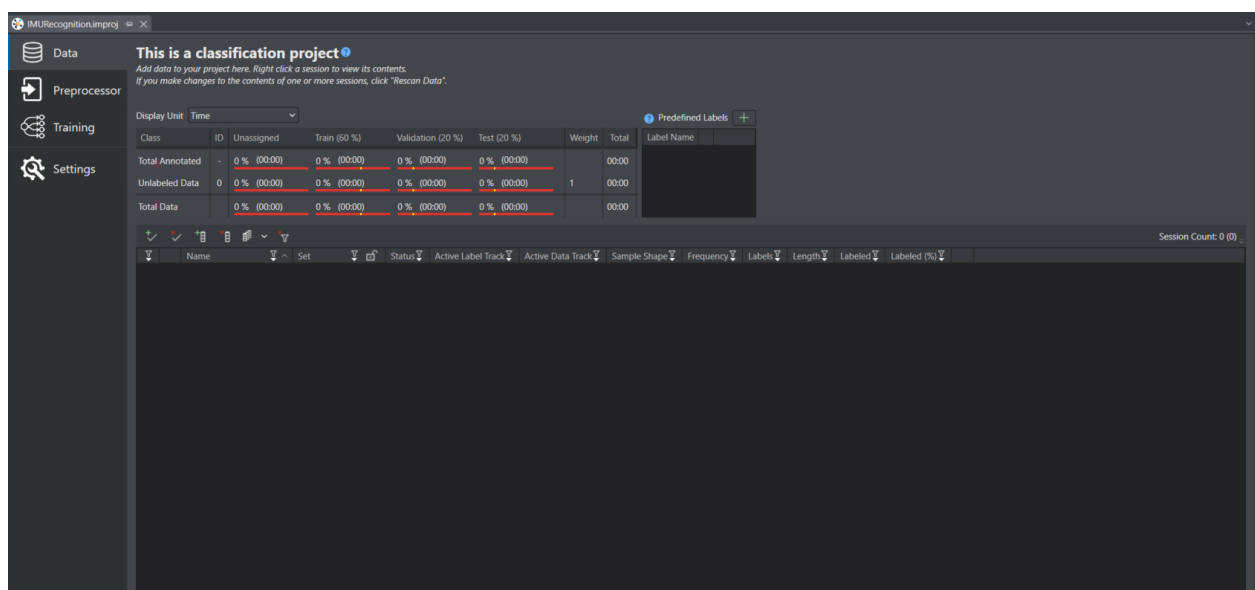
Create a new project using **File - New project**.



As we want to identify a gesture type, we create a **Classification** project.

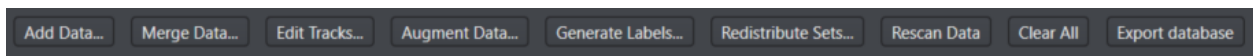


The new project is empty. We now need to add the data [we collected earlier](#).

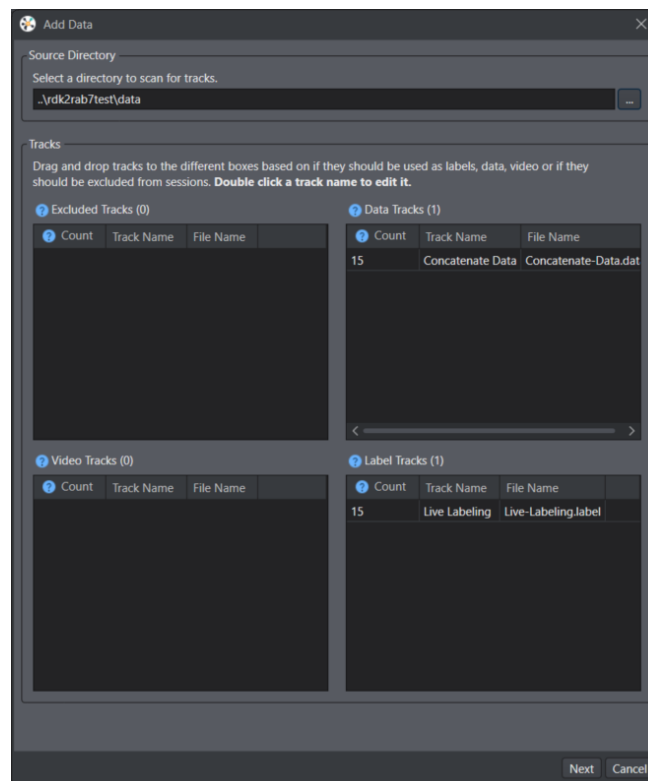


Data import

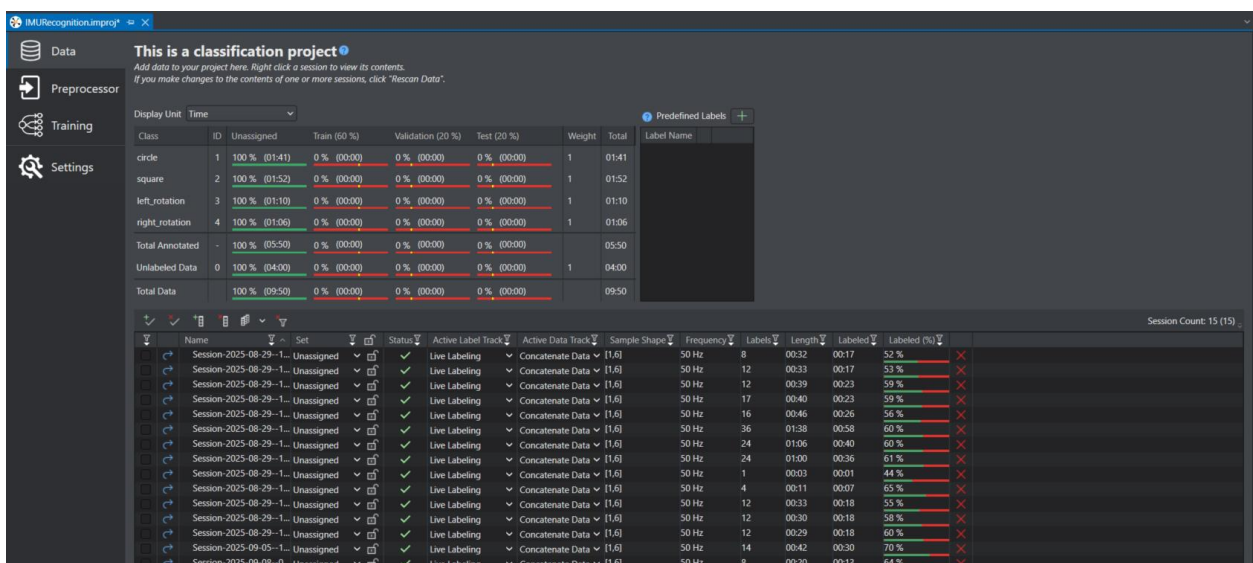
Click on the **Add Data...** button.



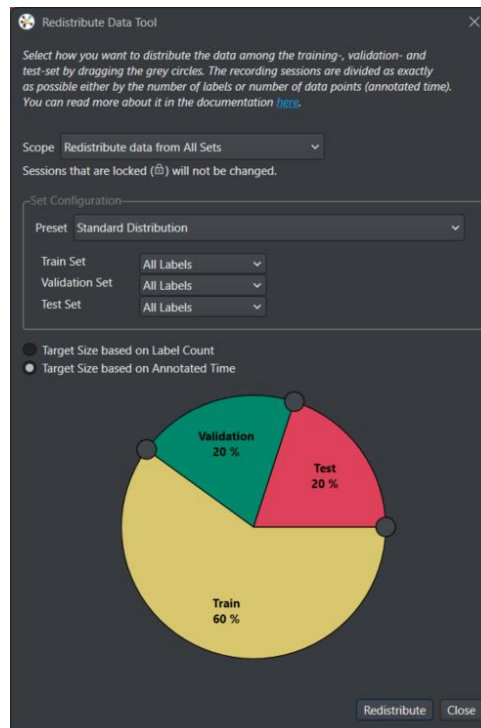
Select the directory storing your data using the [...] button. The software will scan the directory and find the sessions your recorded. In our folder, 15 sessions were stored.



The data is imported inside the project. The **Set** value for every session is set to **Unassigned**.



Click on **Redistribute Sets...** button to let Deepcraft Studio assign the sessions automatically. After clicking on **Redistribute** button, each session should have an assigned **Set** (Validation, Train or Test).



Train. Used during the training process itself. The dataset is used to compute/fit the parameters of the model. If using a dense layer for example, it will be used to update the weights of the layer.

Validation. Used after each “epoch” to evaluate (roughly) the model performance. Good to detect overfitting for example.

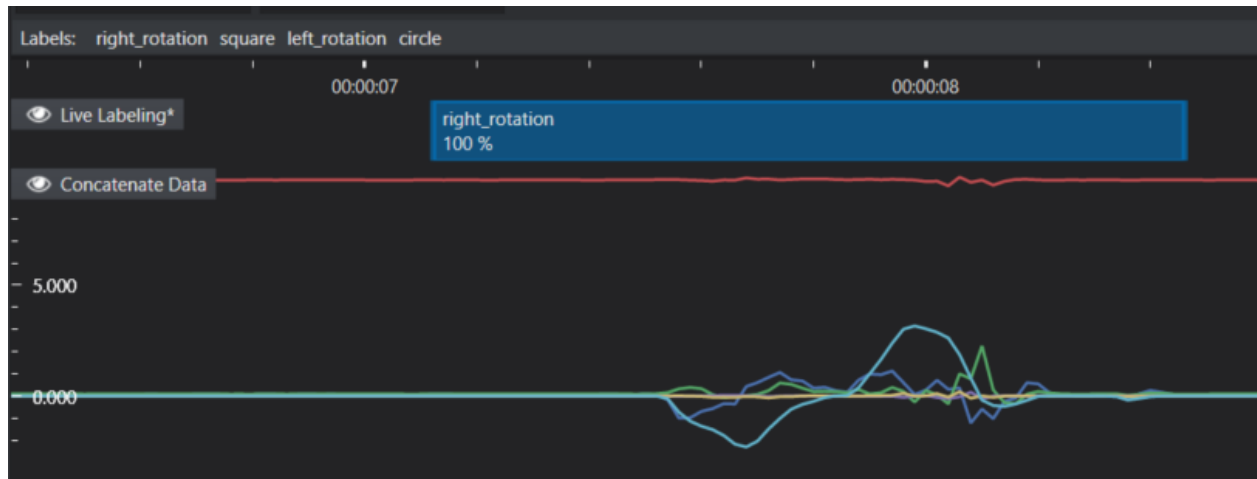
Test. Once the model has been trained, the “Test” dataset is used to evaluate it, using data that have never been seen during the training process. Here you will see how good the model behaves on new values.

Display Unit		Time						Predefined Labels	
Class	ID	Unassigned	Train (60 %)	Validation (20 %)	Test (20 %)	Weight	Total	Label Name	
circle	1	0 % (00:00)	46 % (00:46)	22 % (00:22)	32 % (00:32)	100	01:41		
square	2	0 % (00:00)	45 % (00:50)	23 % (00:25)	33 % (00:36)	100	01:52		
left_rotation	3	0 % (00:00)	63 % (00:44)	15 % (00:10)	22 % (00:15)	100	01:10		
right_rotation	4	0 % (00:00)	61 % (00:40)	15 % (00:10)	23 % (00:15)	100	01:06		
Total Annotated	-	0 % (00:00)	52 % (03:01)	20 % (01:08)	29 % (01:40)		05:50		
Unlabeled Data	0	0 % (00:00)	50 % (02:00)	22 % (00:51)	29 % (01:08)	1	04:00		
Total Data		0 % (00:00)	51 % (05:01)	20 % (02:00)	29 % (02:48)		09:50		

To put more focus on the annotated data, change the **Weight** parameter to 100 for all classes. This will let the training process focus on the “labeled” data.

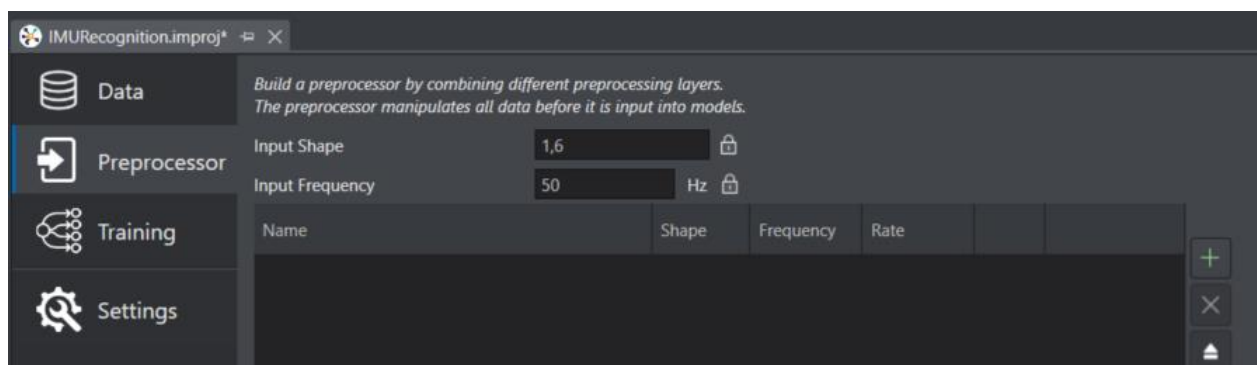
Pre-processing

The input of our model are the values coming from the accelerometer and gyroscope (6 values generated at a frequency of 50Hz).

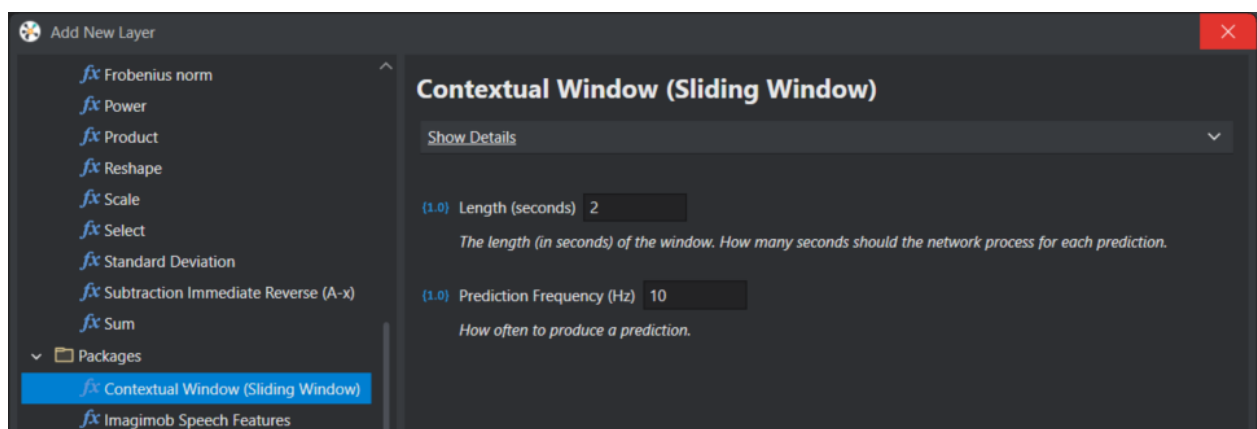


We want to detect a gesture. To do that, we need to evaluate how these 6 values behave during a certain amount of time, that is called a window in machine learning vocabulary.

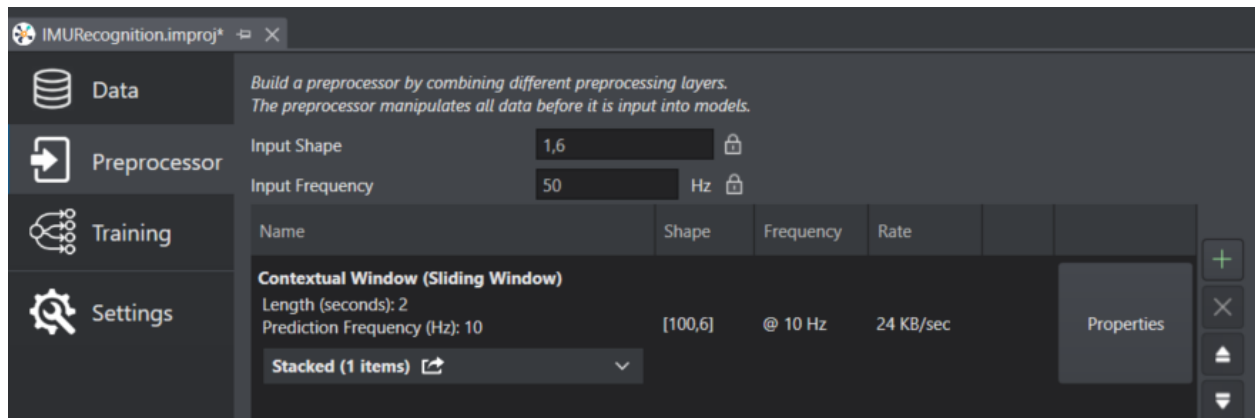
In the **Preprocessor** tab, click on the **[+]** button.



Then select **Contextual window**, enter a **Length** in seconds and a **Prediction frequency** (i.e. how often do we want the model to check for a gesture). Since our gestures have a maximum duration of around 2 seconds, we select a window of 2 seconds.



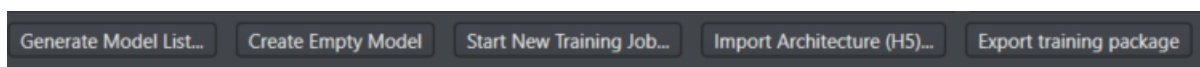
The result:



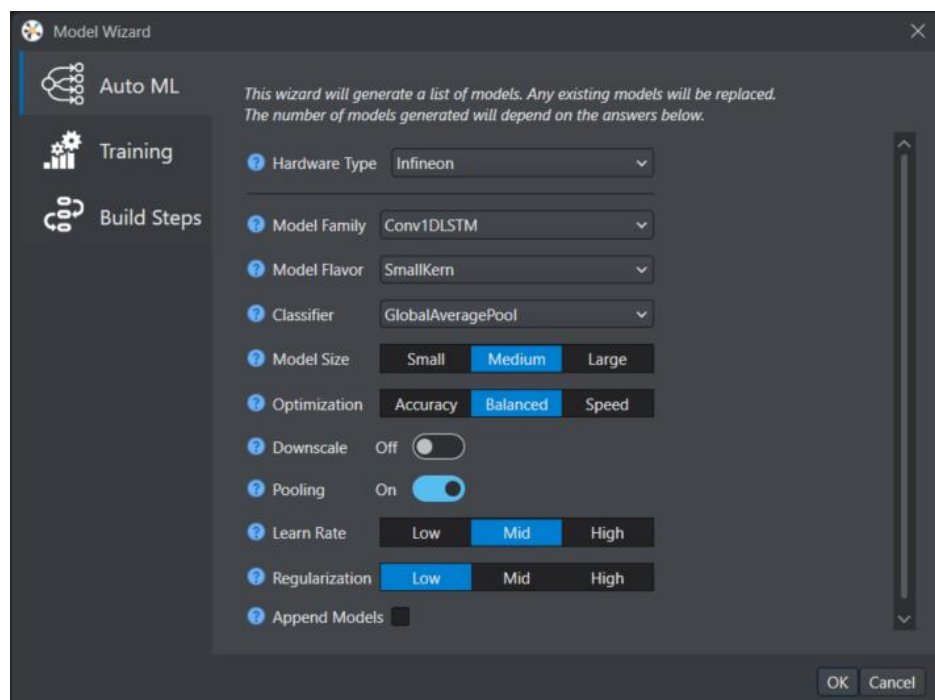
Training

Model creation

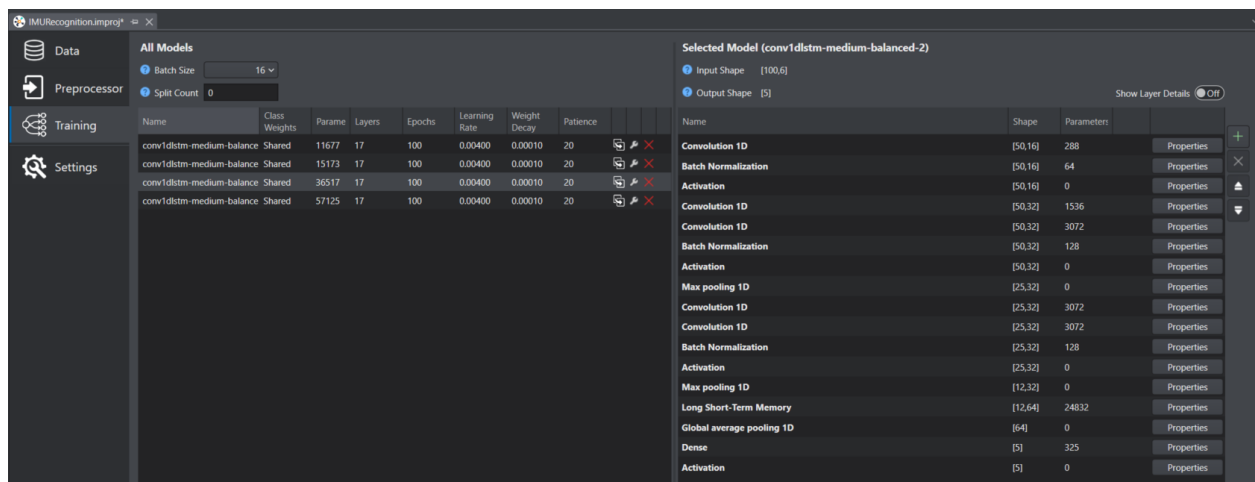
The first step is to create the topology of our model. Deepcraft Studio has a great feature **Generate Model List...** that helps to start from scratch:



Since we are monitoring the behavior of a signal during the time, I set **Model family** = **Conv1DLSTM**. You can also choose **Conv1D** or **Conv2D** for this type of problem.



After clicking on **OK**, Deepcraft Studio will generate some model topologies. Click on it to see its details.

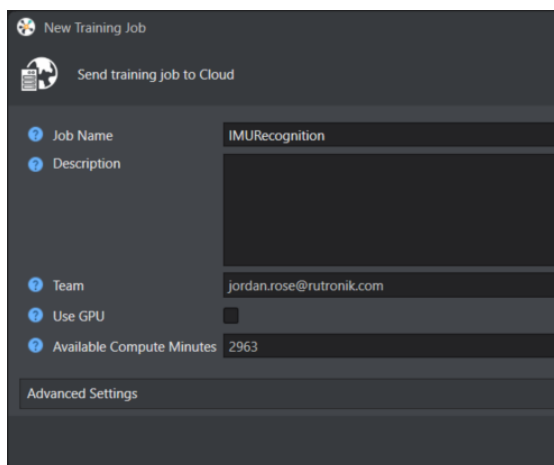
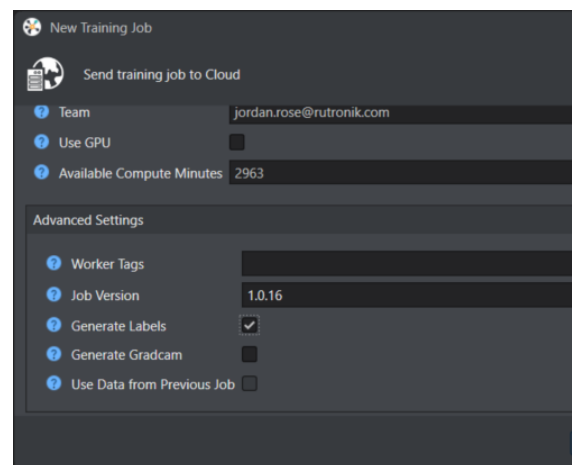


Name	Class Weights	Param	Layers	Epochs	Learning Rate	Weight Decay	Patience
conv1d_lstm-medium-balance	Shared	11677	17	100	0.00400	0.00010	20
conv1d_lstm-medium-balance	Shared	15173	17	100	0.00400	0.00010	20
conv1d_lstm-medium-balance	Shared	36517	17	100	0.00400	0.00010	20
conv1d_lstm-medium-balance	Shared	57125	17	100	0.00400	0.00010	20

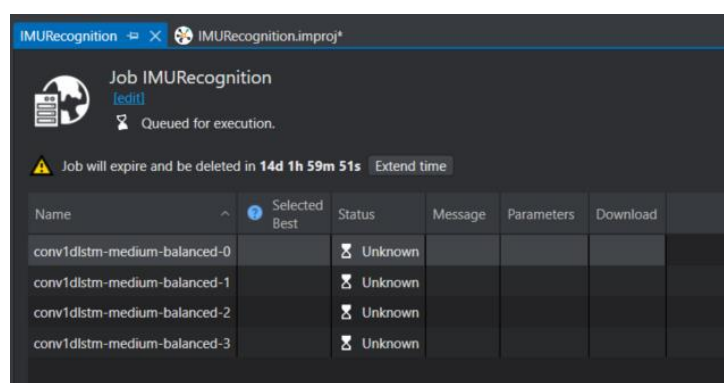
Name	Shape	Parameters
Convolution 1D	[50,16]	288
Batch Normalization	[50,16]	64
Activation	[50,16]	0
Convolution 1D	[50,32]	1536
Convolution 1D	[50,32]	3072
Batch Normalization	[50,32]	128
Activation	[50,32]	0
Max pooling 1D	[25,32]	0
Convolution 1D	[25,32]	3072
Convolution 1D	[25,32]	3072
Batch Normalization	[25,32]	128
Activation	[25,32]	0
Max pooling 1D	[12,32]	0
Long Short-Term Memory	[12,64]	24832
Global average pooling 1D	[64]	0
Dense	[5]	325
Activation	[5]	0

Training process

Press **Start New Training Job...** to start the training process.

In **Advanced Settings**, check **Generate Labels** if you want to verify how the training process uses your data.



Name	Selected	Status	Message	Parameters	Download
conv1d_lstm-medium-balanced-0	Best	Unknown			
conv1d_lstm-medium-balanced-1		Unknown			
conv1d_lstm-medium-balanced-2		Unknown			
conv1d_lstm-medium-balanced-3		Unknown			

Your data will be uploaded onto the Deepcraft server, and the training will be proceeded on the Deepcraft server.

IMURecognition - X IMURecognition.improj*

Job IMURecognition
[edit](#)
Job started

⚠ Job will expire and be deleted in 14d 1h 59m 47s [Extend time](#)

Name	Selected Best	Status	Message	Train Accuracy	Validation Accuracy	Train F1Score	Validation F1Score	Train Best Loss	Validation Best Loss	Parameters	Download
conv1d1stm-medium-balanced-0		Started	Training 0/100 epochs completed	0.2137	0.0446	NaN	NaN	6.0265	3.4172	11677	
conv1d1stm-medium-balanced-1		Started	Training 0/100 epochs completed	0.4092	0.0071	NaN	NaN	5.1881	4.5394	15173	
conv1d1stm-medium-balanced-2		Started	Training 0/100 epochs completed	0.2266	0.0348	NaN	NaN	5.7209	3.1145	36517	
conv1d1stm-medium-balanced-3		Started	Training 0/100 epochs completed	0.4033	0.0705	NaN	NaN	5.5106	3.0557	57125	

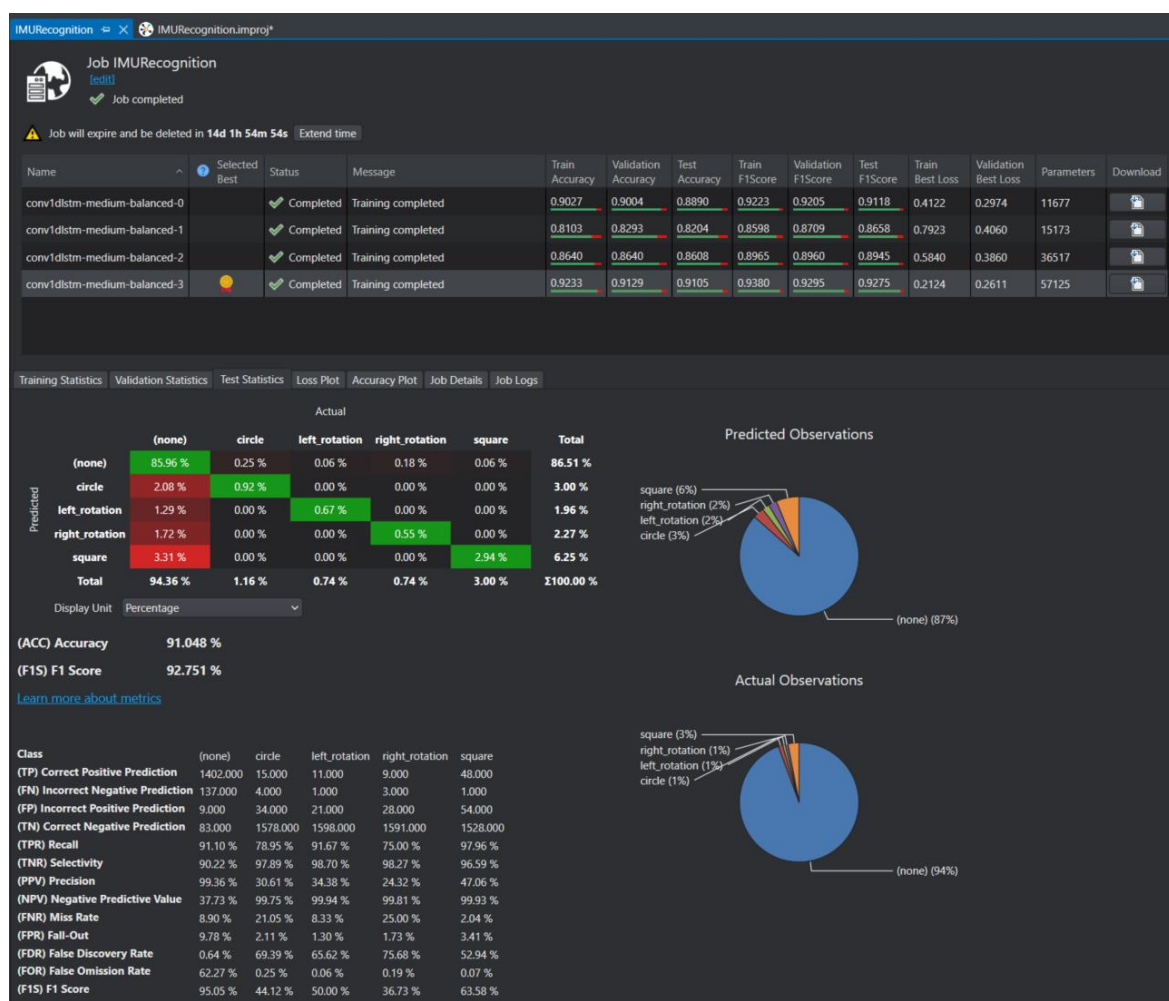
IMURecognition - X IMURecognition.improj*

Job IMURecognition
[edit](#)
Job started

⚠ Job will expire and be deleted in 14d 1h 59m 58s [Extend time](#)

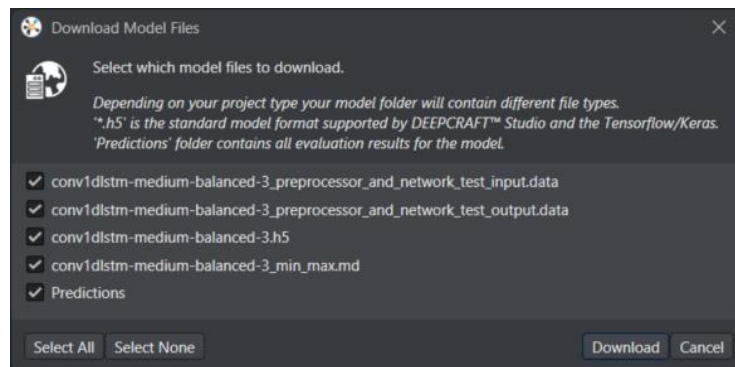
Name	Selected Best	Status	Message	Train Accuracy	Validation Accuracy	Train F1Score	Validation F1Score	Train Best Loss	Validation Best Loss	Parameters	Download
conv1d1stm-medium-balanced-0		Started	Training 2/100 epochs completed	0.6585	0.3098	NaN	NaN	1.6249	1.3137	11677	
conv1d1stm-medium-balanced-1		Started	Training 4/100 epochs completed	0.7863	0.7884	NaN	NaN	1.0342	0.4060	15173	
conv1d1stm-medium-balanced-2		Started	Training 2/100 epochs completed	0.6260	0.0304	NaN	NaN	2.5019	3.1145	36517	
conv1d1stm-medium-balanced-3		Started	Training 2/100 epochs completed	0.7259	0.0527	NaN	NaN	1.3713	3.0557	57125	

When the training is completed, the results will be displayed.

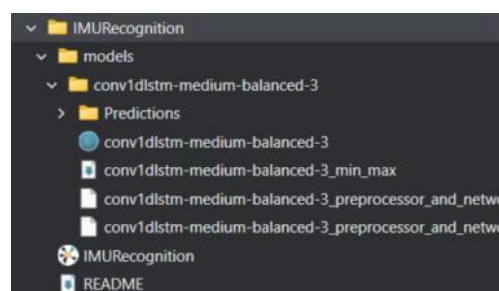


Predicted observations are the outputs returned by the model during the training. **Actual observations** show true values based on the label data. The model with the best “test” accuracy is marked with a “gold medal”.

Use the buttons in the **Download** column to download the models you want.



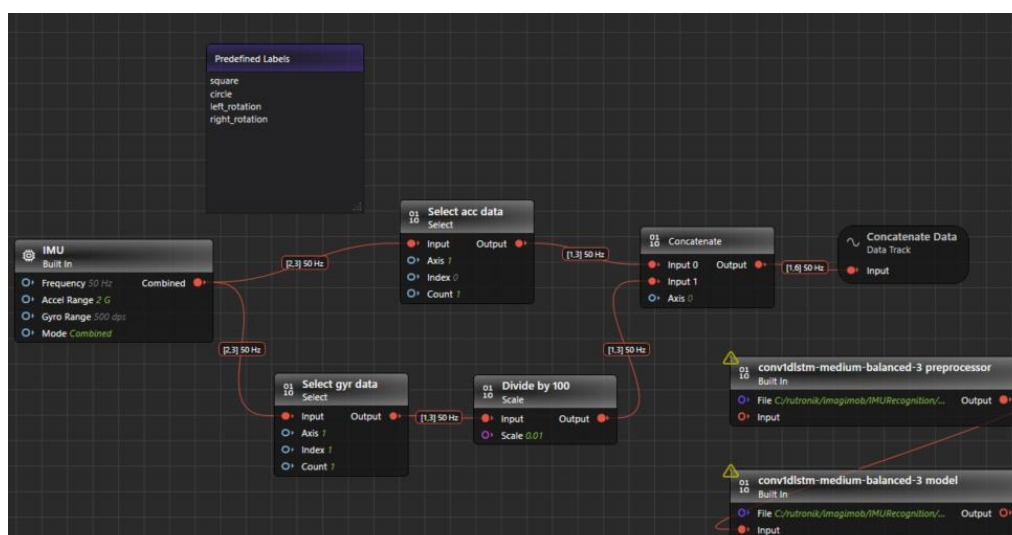
Choose the **models** directory inside your project (good practice) to store the trained model.



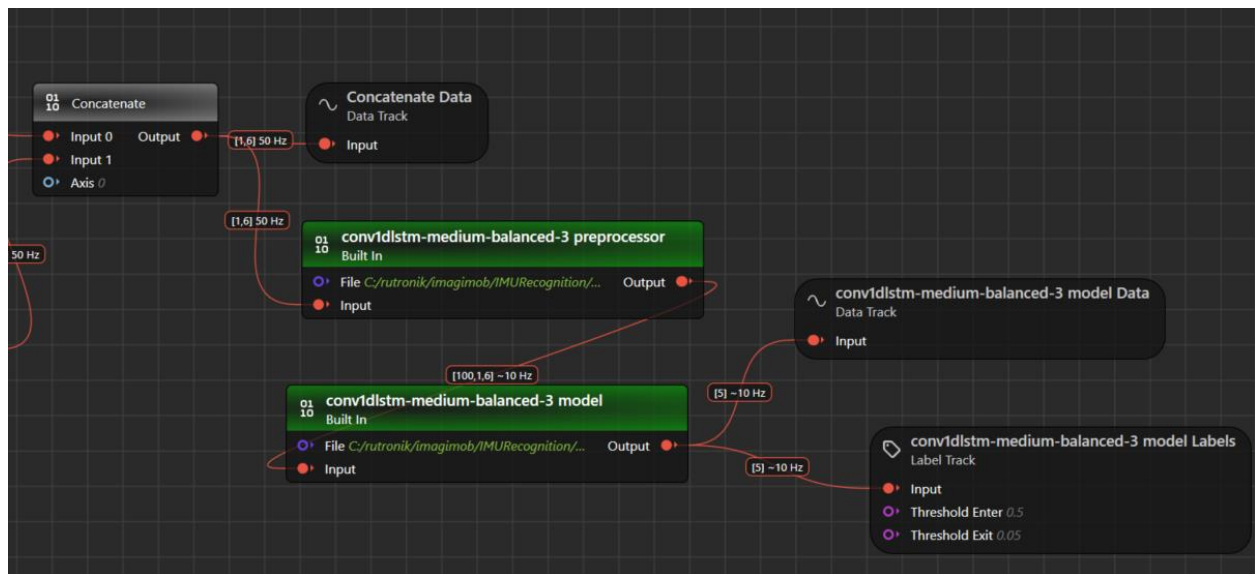
Live evaluation

Now we can evaluate a trained model live using Deepcraft Studio.

Open the “Main.imunit” file you created in the “Collect data” chapter. Using drag and drop, add the model file “conv1dlstm-medium-balanced-3” into the “Main.imunit” file.

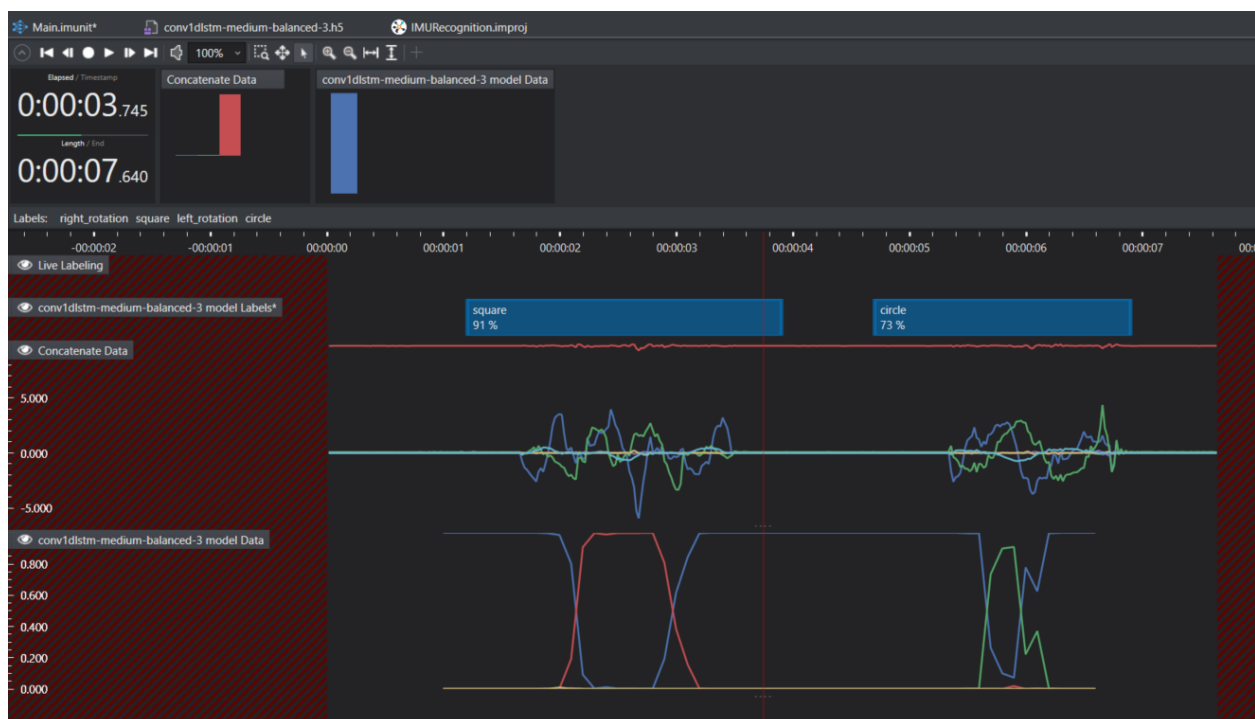


Connect it to your data and add a **Data track** and a **Label track**.



Click on the **Start** and then on the **Record** button. Do some gestures using your board.

You should see something like this. Here the model successfully detected a “square” and a “circle”:

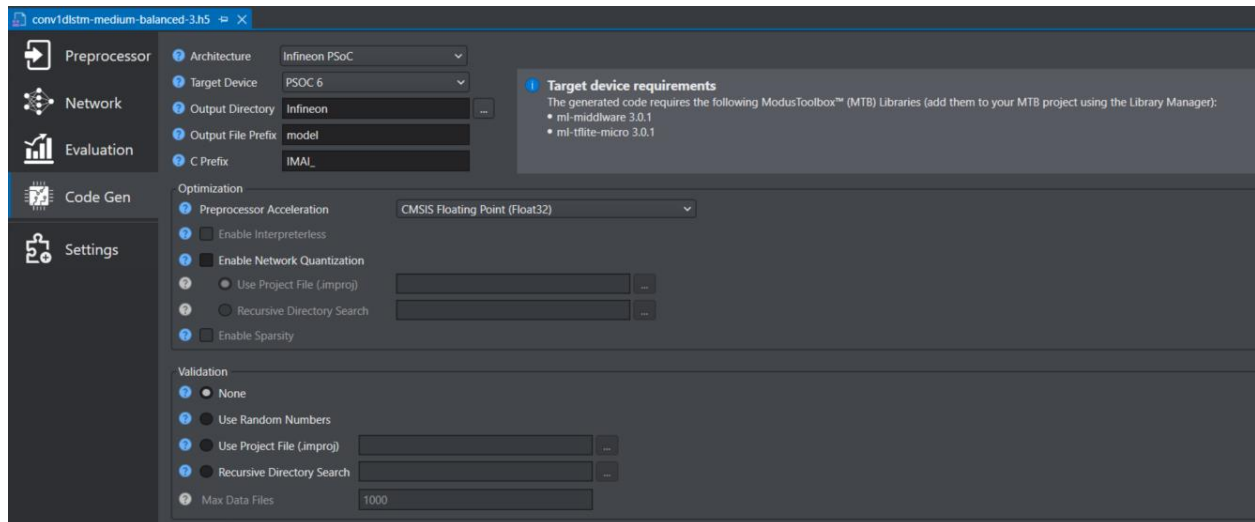


The values of the model outputs (between 0 and 1) are shown below. They represent the probability of a gesture being recognized by the model. The color of the curve is equal to the kind of gesture. For example, when the square gesture was performed, the model showed that its probability raised up to 91% (red line).

Deployment

Generate source code

To generate a code for the MCU, go to the **Code Gen** tab.



Press **Generate Code**.

[Generate Code](#)
[Cancel Build](#)
[Show Progress in Console](#)
[How do I use the generated code?](#)
[License Agreement](#)

After code generation process finishes, you will see some reports.

code_generation_report.md IMURecognition.improj conv1dstm-medium-balanced-3.h5

Model performance and validation report

Source model: C:/rutronik/imagimob/IMURecognition/models/conv1dstm-medium-balanced-3/conv1dstm-medium-balanced-3.h5
Generated: 2025-09-08 15:09:14

Memory usage

Model	Model memory (Bytes)	Scratch memory (Bytes)
float	236,796	18,432

Latency

Layer	Cycles
CONV_2D	222,443
CONV_2D	844,816
CONV_2D	1,468,804
MAX_POOL_2D	51,774
CONV_2D	1,468,675
CONV_2D	2,716,652
MAX_POOL_2D	47,098
MEAN	104,013
FULLY_CONNECTED	3,961
SOFTMAX	0
TOTAL	6,928,235

Validation

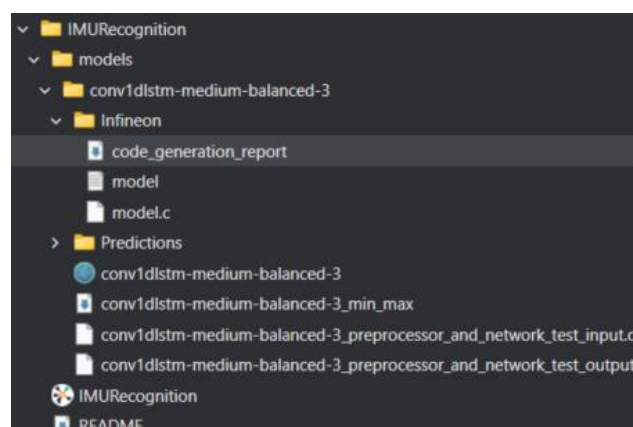
Validation data source: None

```

1 | Model performance and validation report
2 **Source model:** C:/rutronik/imagimob/IMURecognition/models/conv1dlstm-medium-balanced-3/conv1dlstm-medium-balanced-3.h5
3 **Generated:** 2025-09-08 15:09:14
4
5 ### Memory usage
6 | Model | Model memory (Bytes) | Scratch memory (Bytes) |
7 | :--- | :--- | :--- |
8 | float | 236,796 | 18,432 |
9
10 ### Latency
11 | Layer | Cycles |
12 | :--- | :--- |
13 | CONV_2D | 222,443 |
14 | CONV_2D | 844,816 |
15 | CONV_2D | 1,468,804 |
16 | MAX_POOL_2D | 51,774 |
17 | CONV_2D | 1,468,675 |
18 | CONV_2D | 2,716,652 |
19 | MAX_POOL_2D | 47,098 |
20 | MEAN | 104,013 |
21 | FULLY_CONNECTED | 3,961 |
22 | SOFTMAX | 0 |
23 | **TOTAL** | **6,928,235** |
24
25 ### Validation
26 **Validation data source:** None
27

```

The source code is also available in the project.



Create new Modus Toolbox project and use it

Use this GitHub project to run the source code on your MCU:

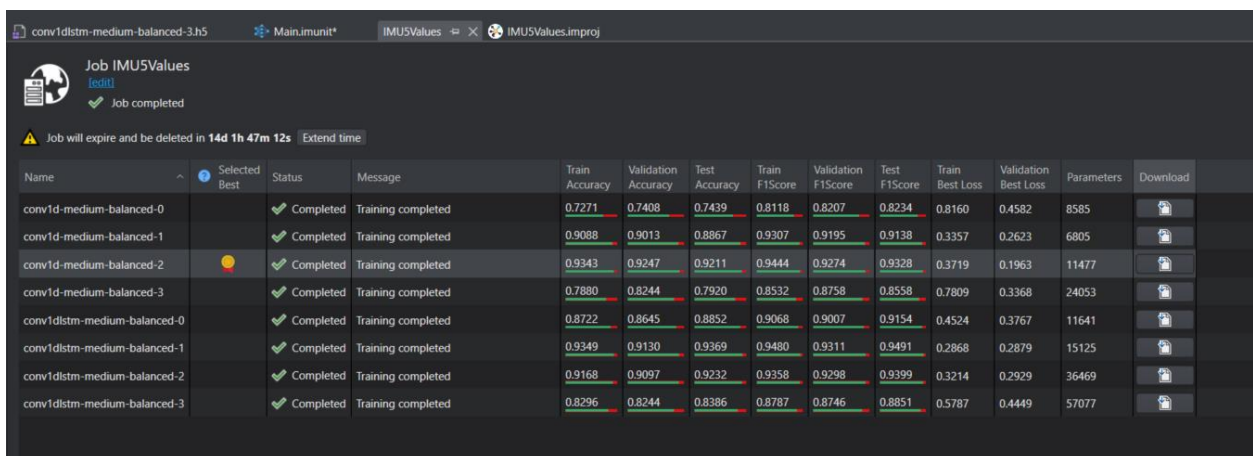
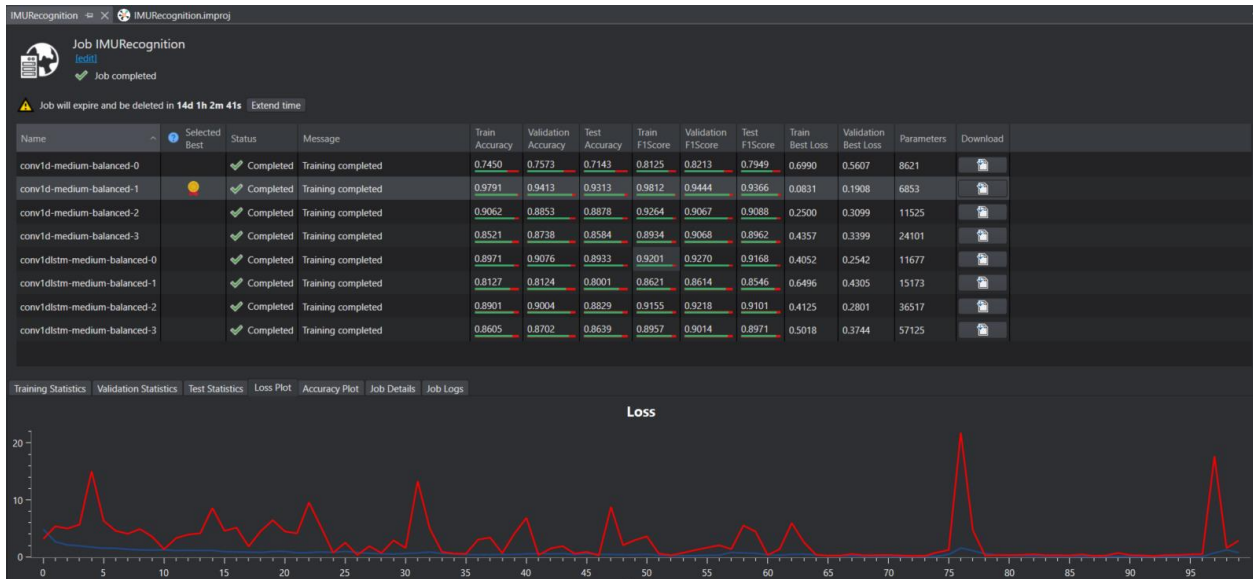
https://github.com/RutronikSystemSolutions/RDK2_RAB7-SENSORFUSION-DEEPCRAFT-DEPLOY

You'll find more details in the project README.

Problems

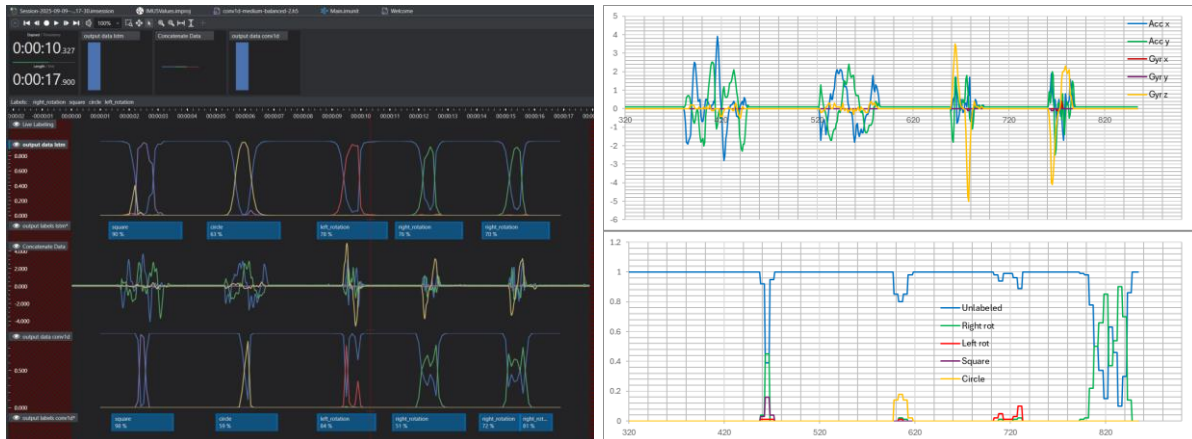
Model not performing well on embedded target

We tried to use Conv1D architecture (instead of LSTM). At the end LSTM showed itself as a more efficient architecture in this case.



Problem with deployment

The models work well inside Deepcraft Studio, but when trying to deploy them on the RDK2, I get some strange results.

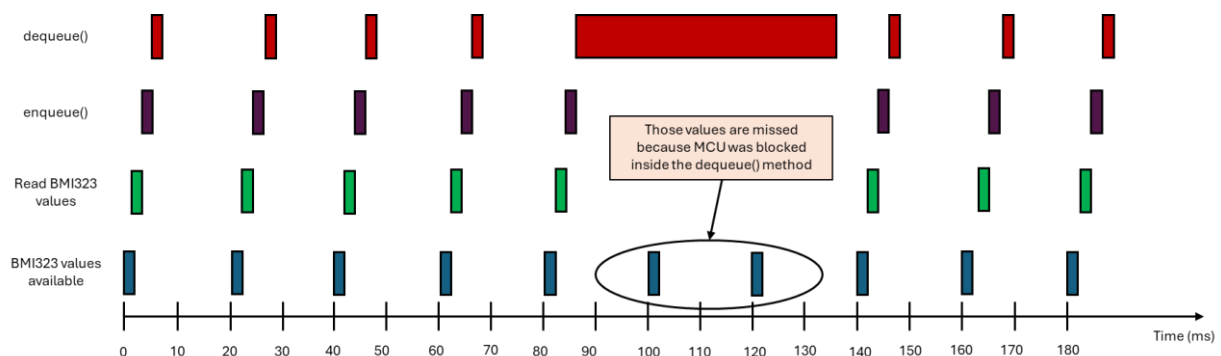


It turned out that the computation time, to perform an inference is about 52ms, but the sampling frequency of the accelerometer is 50 Hz (20ms). We were dropping some values, which leads to wrong model behavior.

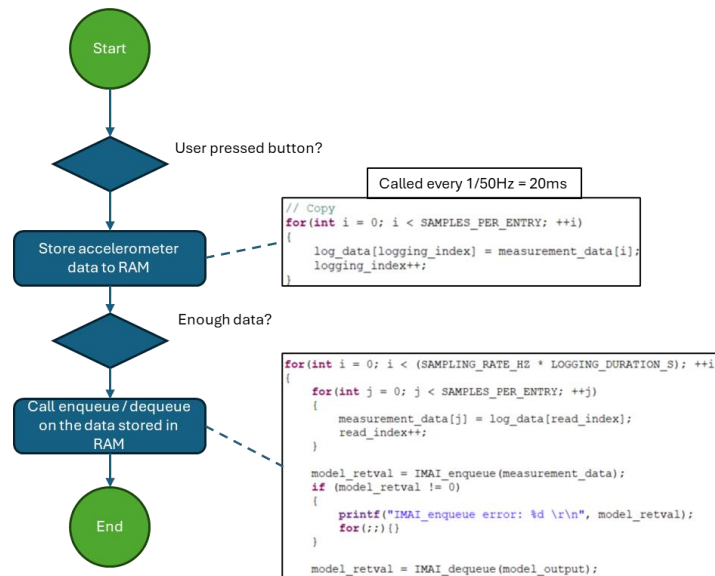
```
start_time = clock_get_tick();
{
    dequeue_result = IMAI_dequeue(model_output);
}
stop_time = clock_get_tick();

if (dequeue_result == 0)
{
    printf("Dequeue (%d) takes: %d ms\r\n",
        dequeue_result,
        (int)((float)(stop_time - start_time) / (float)CLOCK_TICK_PER_SECOND) * 1000.f);
}
```

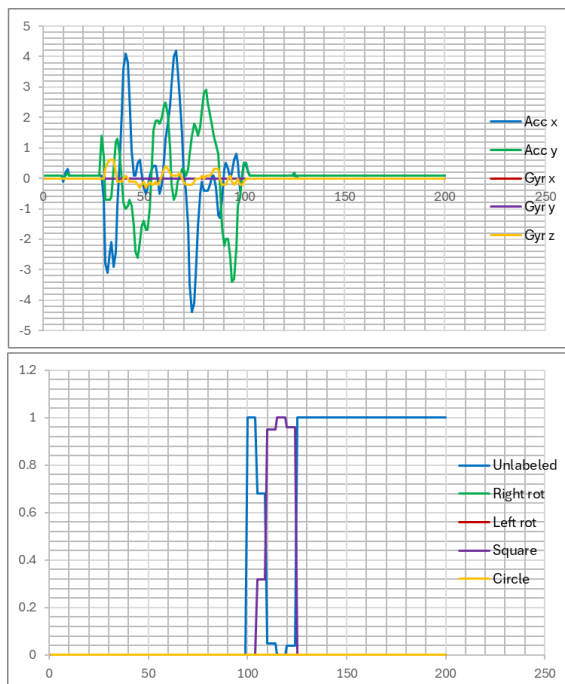
```
Dequeue (0) takes: 52 ms
Dequeue (0) takes: 52 ms
Dequeue (0) takes: 52 ms
Dequeue (0) takes: 52 ms
Dequeue (0) takes: 52 ms
```



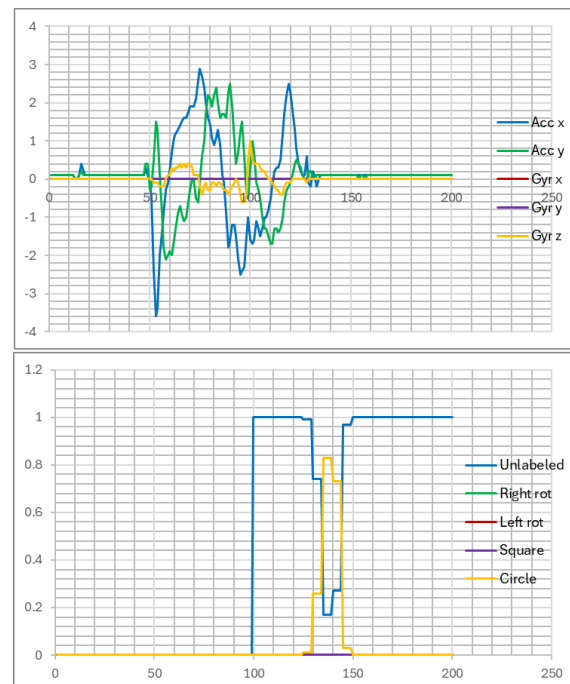
To check the assumption, I tried following:



The results look better this way:

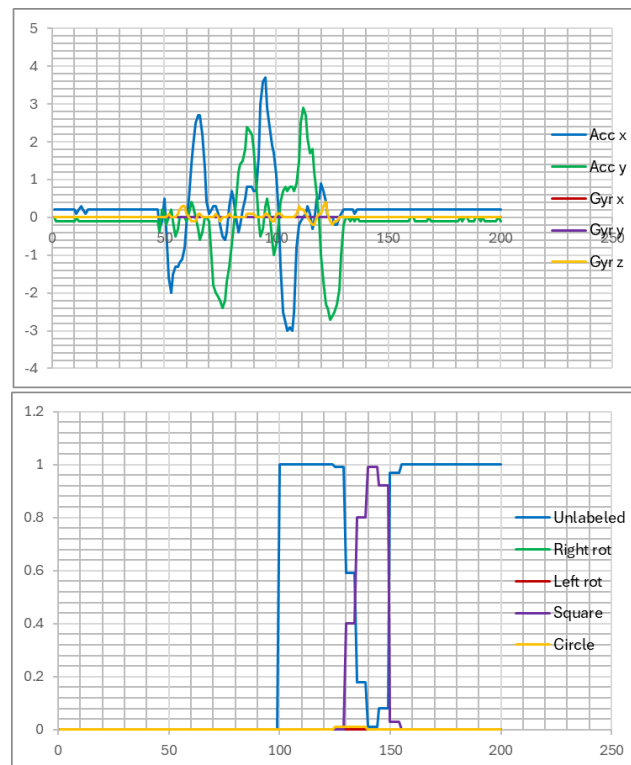


Square motion



Circle motion

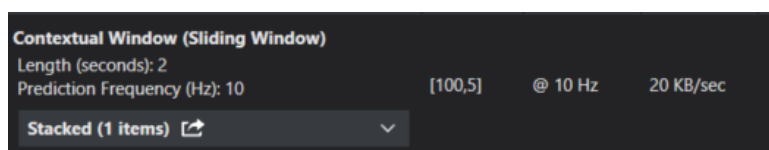
LSTM model also working (see below):



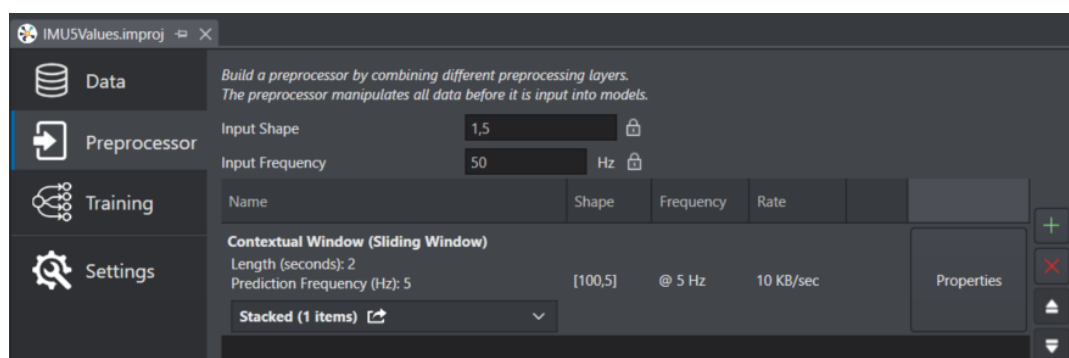
When using the LSTM model, the inference takes 101ms (2 times what is needed for the model that uses Conv1D).

```
Inference takes: 101 ms
No data
No data
No data
No data
Inference takes: 101 ms
```

In the pre-processor configuration, we have a prediction frequency of 10 Hz. Since our inference time is slower, we need to reduce the prediction frequency.



Change it to 5Hz and train again:



Results of the new training:

IMUSValues X IMUSValues.improj

Job IMUSValues
[\[edit\]](#)
 Job completed

⚠ Job will expire and be deleted in 14d 1h 59m 2s [\[Extend time\]](#)

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	0.8457	0.8462	0.8555	0.8739	0.8688	0.8768	0.6456	0.3618	8585	
conv1d-medium-balanced-1		✔ Completed	Training completed	0.8324	0.8428	0.8269	0.8704	0.8682	0.8671	0.6509	0.4122	6805	
conv1d-medium-balanced-2		✔ Completed	Training completed	0.8673	0.8595	0.8670	0.8883	0.8749	0.8891	0.7031	0.3266	11477	
conv1d-medium-balanced-3		✔ Completed	Training completed	0.9454	0.8997	0.8970	0.9510	0.9037	0.9102	0.3616	0.3340	24053	
conv1dstm-medium-balanced-0		✔ Completed	Training completed	0.9230	0.9064	0.9256	0.9337	0.9185	0.9351	0.3324	0.2479	11641	
conv1dstm-medium-balanced-1		✔ Completed	Training completed	0.9279	0.8896	0.9328	0.9375	0.9077	0.9414	0.3125	0.2791	15125	
conv1dstm-medium-balanced-2		✔ Completed	Training completed	0.9261	0.9064	0.9328	0.9362	0.9203	0.9412	0.2929	0.2775	36469	
conv1dstm-medium-balanced-3	🔴	✔ Completed	Training completed	0.9325	0.9097	0.9413	0.9413	0.9220	0.9481	0.2798	0.3064	57077	

Not enough space

```

CDT Build Console [RDK2_RAB7-SENSORFUSION_DEEPCRAFT_Deploy]
Constructing build rules...
Build rules construction complete

=====
= Building application =
=====
Generating compilation database file...
-> ./build/compile_commands.json
Compilation database file generation complete
Building 914 file(s)
Compiling main.c -DARM_MATH_DSP -DARM_MATH_LOOPUNROLL -DCOMPONENT_APP_RDK2 -DCOMPONENT_CAT1 -DCOMPONENT_CAT1A -DCOMPONENT_CM0P_SLEEP -DCOMPONENT_CM4 -DCOMPONENT_CM4_C
Linking output file rdk2-rab7-sensorfusion-deepcraft-deploy.elf
c:/users/roj030/modustoolbox/tools_3.3/gcc/bin/./lib/gcc/arm-none-eabi/11.3.1/../../../../arm-none-eabi/bin/ld.exe: address 0x10089c68 of C:/rutronik/mtw33_mlteststoremove/RDK2_RAB7-SENSOF
c:/users/roj030/modustoolbox/tools_3.3/gcc/bin/./lib/gcc/arm-none-eabi/11.3.1/../../../../arm-none-eabi/bin/ld.exe: C:/rutronik/mtw33_mlteststoremove/RDK2_RAB7-SENSOF
c:/users/roj030/modustoolbox/tools_3.3/gcc/bin/./lib/gcc/arm-none-eabi/11.3.1/../../../../arm-none-eabi/bin/ld.exe: address 0x10089c68 of C:/rutronik/mtw33_mlteststoremove/RDK2_RAB7-SENSOF
c:/users/roj030/modustoolbox/tools_3.3/gcc/bin/./lib/gcc/arm-none-eabi/11.3.1/../../../../arm-none-eabi/bin/ld.exe: region 'flash' overflowed by 41800 bytes
collect2.exe: error: ld returned 1 exit status
make[1]: *** [./mtb_shared/core-make/release-v3.4.1/make/core/build_v1.mk:288: C:/rutronik/mtw33_mlteststoremove/RDK2_RAB7-SENSORFUSION_DEEPCRAFT_Deploy/build/APP_RDK2_RAB7-SENSOF
make: *** [./mtb_shared/core-make/release-v3.4.1/make/core/main.mk:420: secondstage build] Error 2
"C:/Users/ROJ030/ModusToolbox/tools_3.3/modus-shell/bin/make CY_MAKE_IDE=eclipse CY_IDE_TOOLS_DIR=C:/Users/ROJ030/ModusToolbox/tools_3.3 CY_IDE_BT_TOOLS_DIR=-j22 --c
16:41:46 Build Failed. 6 errors, 0 warnings. (took 30s.950ms)

```

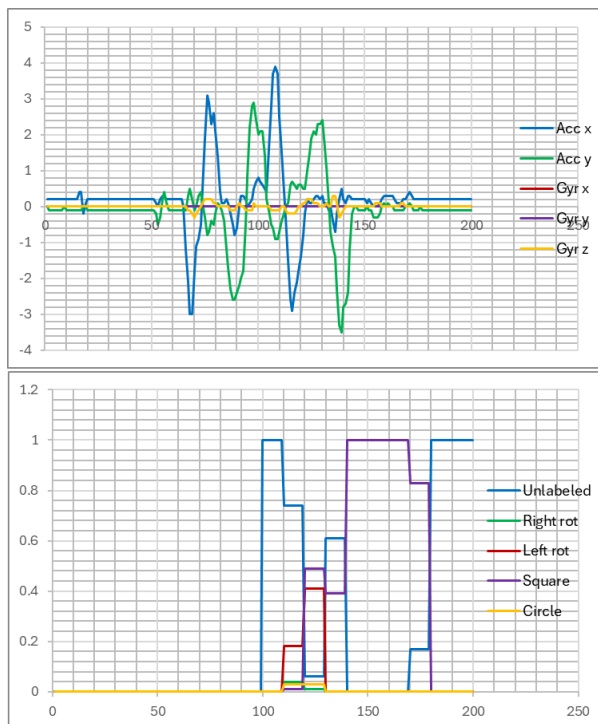
The solution is to use the external flash.

To be updated.

Deployment with 5 Hz model

```
Inference takes: 96 ms
No data
No data
No data
No data
No data
No data
No data
No data
No data
Inference takes: 96 ms
```

Data acquisition with 50 Hz and one inference every 10 data -> 5 Hz.

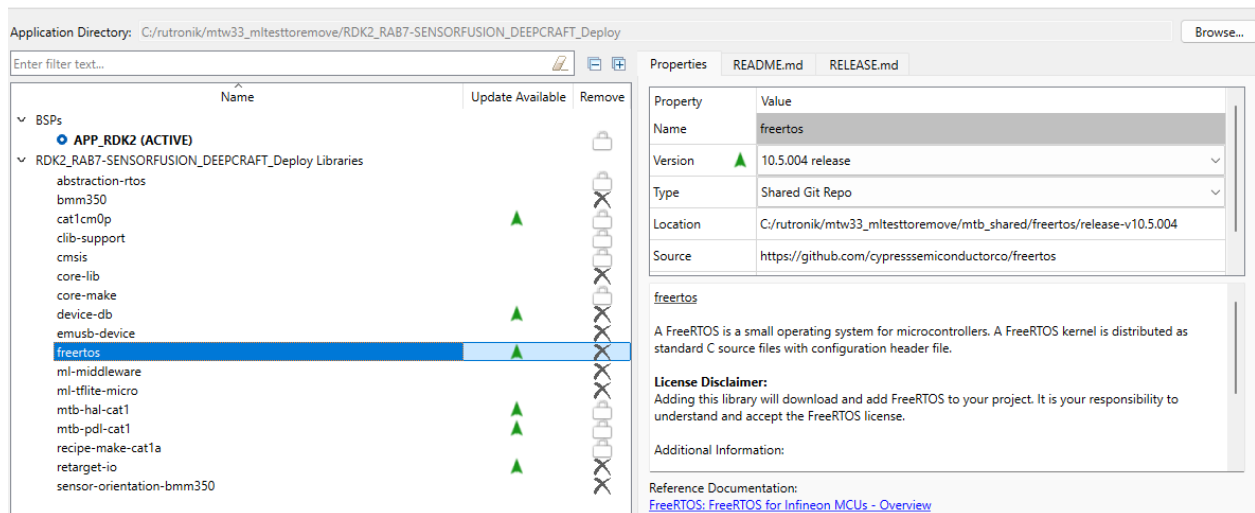


First, we collect data, then run inference on it. The result looks ok.

Model needs to be retrained again with more data (left rotation detected before the square motion).

Add FreeRTOS support to the project and use tasks

A good practice to avoid problem of inference time is to use FreeRTOS.



Makefile:

```
COMPONENTS+=FREERTOS
COMPONENTS+=RTOS_AWARE
```

Configuration files:

```
> FreeRTOSConfig.h
> FreeRTOS-openocd.c
```

We will use one task to collect the data (maximum priority) and one task to perform the inference (lower priority). Data transfer between those 2 tasks is done using a xQueue.

```
/**
 * @brief Used to transfer BMI323 to the inference task
 */
QueueHandle_t transfer_queue;

transfer_queue = xQueueCreate( QUEUE_MAX_ELEMENTS_COUNT, IMAI_DATA_IN_COUNT * sizeof(float));
```

Inference task:

```
if (xQueueReceive(transfer_queue, (void*)data, portMAX_DELAY) != pdPASS)
```

Data collection task:

```
if (xQueueSend(transfer_queue, (void*) raw_data, (TickType_t) 0) != pdPASS)
```

The test results look good.

```
RDk2 RAB7 Sensor Fusion - Model deployment - V1.0
BMI2xx I2C Interface Initialized.
BMI323 config hw
Inference task - start
bmi323: all good
data_collection_task: Start loop
unlabelled
square
unlabelled
square
unlabelled
circle
unlabelled
```