Walking/Running Classification

Advanced Operation System A.Y. 2018-2019

David Brellmann Shufan Zhang

July 16, 2019

1 Introduction

In this project we aim to create a neural network with Keras and implement it on Miosix embedded OS in order for it to be used on a microcontroller.

The microcontroller we use is the STM32F401RE of STMicroelectronics. It is integrated on a Nucleo-64 board. We also use the expansion board IKS01A2 which can be attached to the Nucleo board and which contains sensors to capture motion and environmental information such as temperature, humidity and acceleration.

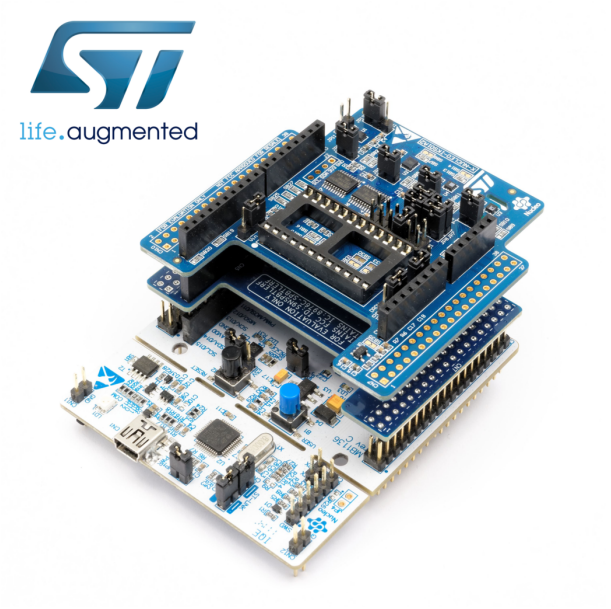

Figure 1: Boards

The application we aim to develop is to detect if a person is walking or running. This classification will be made possible by capturing the person's motion data and then analyzing it with neural network which will tell the person's activity.

The capture of motion data will be done by the sensor LSM6DSL containing an accelerometer and a gyroscope, each of which outputs three measurements since motion in 3-dimension space will be classified in three directions.

The first step of this project is to develop a database of walking/running with real life experience. After establishing the dataset we use it to train our neural network model with multiple experiments in order to determine the parameters of the neural network.

Once the neural network model created we convert it to optimized code by means of ST's software solution. Finally we implement this neural network library on Miosix to develop application with our microcontroller.

2 Creation of the training dataset

When you develop a machine learning model such as a neural network which accurately predicts whether the user walks or runs you need data. The collected data constitute a training dataset feeding the neural network.

2.1 Where to put the sensors?

To know whether the person runs or walks we first needed to find a place on human's body where each of these activities could be clearly distinguished in terms of sensor data. For example, putting the sensors in the pocket and collecting data from there is not a good solution because the sensors will hardly stay at the same positions. The perfect solution for us was collecting data on the hand wrists. We can now consider the possible cases with this solution:

- walking with the sensor on the right wrist
- walking with have the sensor on the left wrist
- running with have the sensor on the right wrist
- running with the sensor on the left wrist

We did not know at this time if the choice of the wrist may influence the classification.

2.2 What to collect?

After knowing where to collect the data we had to ask ourselves what to collect. Our board has an accelerometer, a gyroscope, a magnetometer, a humidity sensor and a pressure sensor. The most valuable data comes from two sensors: the accelerometer and the gyroscope. The accelerometer provides changes in device's velocity along 3 axes which is a crucial piece of information about how one moves his hand. The gyroscope delivers the rate at which a device rotates around a spatial axis which carries probably fewer insights for distinguishing running and walking activities.

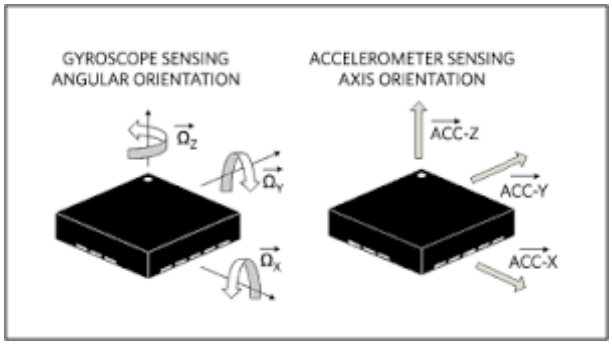

Figure 2: Accelerometer and gyroscope

Now our problem turned to this: how to get the data from our board? After some research about STM32CubeMX and drivers for our sensors, we discovered the Unicleo-GUI software (see Figure 3) developed by STMicroelectronics.

It is a graphical user interface (only on Windows) for the the X-CUBE-MEMS1 and X-CUBE-MEMS-XT1 software expansions and STM32 Nucleo expansion boards (X-NUCLEO-IKS01A1, X-NUCLEO-IKS01A2 and X-NUCLEO-IKS01A3). The main objective of this application is to demonstrate the functionality of ST sensors and algorithms. The data collected with this application can be saved into csv files. To use this software, we had to download the ST-LINK drivers but no other tools or software were needed. Every measures are taken every milliseconds.

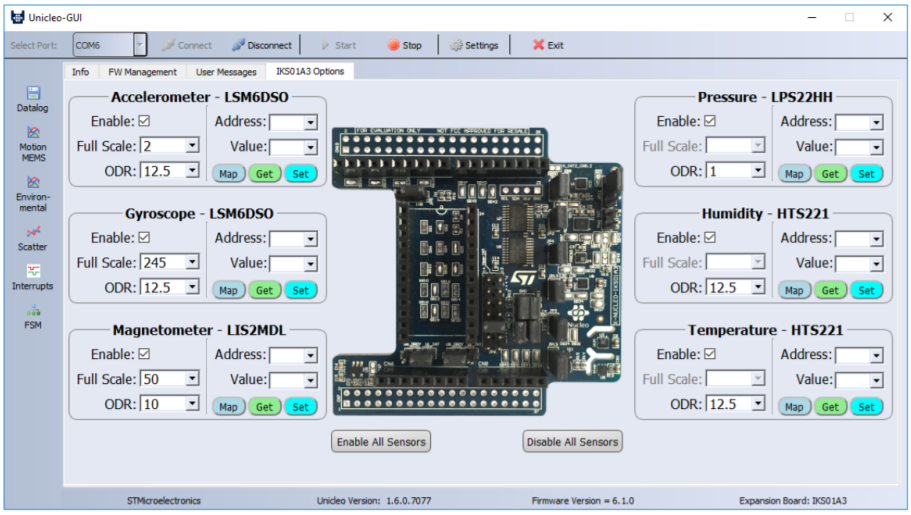

Figure 3: Unicleo-GUI

2.3 Where to collect?

The last problem for how to collect data was in which environment. Indeed, collecting data in an uncontrolled environment can not be a good idea : a continuous recording is required. In a city, when you are in movement, your are constantly interrupted by intersections, traffic lights, cars or other people. All these interruptions generate noises in our training dataset implying a weaker classifier. To this end, we have decided to run in a park (the Porta Venezia Park) to avoid all these possible interruptions.

For each possible case mentioned before we take measures for 5 minutes (in average) from each of us. After collecting walking and running data in the park thanks to the Unicleo-GUI software and by adding some information (such as who ran, which wrist was used...), we obtained a dataset like that:

	A	B	C	D	E	F	G	H	I	J	K
1	time	username	wrist	activity	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z	
2	15:52:43.68	david	1	0	363	1106	-23	51450	41650	-89530	
3	15:52:43.69	david	1	0	330	1022	-1	43330	40250	-89810	
4	15:52:43.70	david	1	0	292	970	17	35420	38990	-89390	
5	15:52:43.71	david	1	0	261	915	1	28070	38290	-87780	
6	15:52:43.72	david	1	0	240	858	-17	22050	38920	-86730	
7	15:52:43.73	david	1	0	217	815	-34	17570	40460	-86870	
8	15:52:43.74	david	1	0	202	773	-47	14980	42560	-87430	
9	15:52:43.75	david	1	0	195	736	-55	14560	44590	-88760	
10	15:52:43.76	david	1	0	197	715	-66	16310	46270	-91000	
11	15:52:43.77	david	1	0	206	705	-82	19530	47740	-93660	
12	15:52:43.78	david	1	0	219	700	-96	23380	48930	-96040	
13	15:52:43.79	david	1	0	240	697	-103	27930	49630	-97650	
14	15:52:43.80	david	1	0	266	701	-104	33460	49770	-98700	
15	15:52:43.81	david	1	0	284	725	-99	39340	48930	-99120	
16	15:52:43.82	david	1	0	301	761	-90	44170	46550	-98350	
17	15:52:43.83	david	1	0	324	800	-79	46970	42420	-95900	
18	15:52:43.84	david	1	0	339	836	-72	48300	37170	-91700	
19	15:52:43.85	david	1	0	354	858	-58	48160	31290	-85960	
20	15:52:43.86	david	1	0	373	888	-43	47040	24500	-79380	
21	15:52:43.87	david	1	0	394	927	-35	44730	17150	-71820	
22	15:52:43.88	david	1	0	413	950	-34	41160	9870	-63210	
23	15:52:43.89	david	1	0	429	960	-33	36890	3640	-53830	
24	15:52:43.90	david	1	0	442	964	-32	33040	-1610	-44660	
25	15:52:43.91	david	1	0	453	967	-28	30310	-5810	-36120	
26	15:52:43.92	david	1	0	465	978	-26	28350	-9800	-28350	
27	15:52:43.92	david	1	0	476	990	-22	26740	-13790	-21070	
28	15:52:43.94	david	1	0	488	999	-14	25270	-18340	-13580	
29	15:52:43.95	david	1	0	497	1006	-5	23450	-23590	-5740	
30	15:52:43.96	david	1	0	501	1009	-4	20860	-29050	2310	
31	15:52:43.97	david	1	0	495	1005	-11	17220	-33250	10570	
32	15:52:43.98	david	1	0	484	1008	-24	12460	-35420	18620	
33	15:52:43.99	david	1	0	471	1006	-33	1330	-35070	34020	
34	15:52:44.00	david	1	0	485	1033	-59	-3010	-34720	39620	
35	15:52:44.01	david	1	0	521	1077	-82	-5320	-35840	44380	
36	15:52:44.01	david	1	0	565	1115	-119	-5880	-37940	50190	
37	15:52:44.03	david	1	0	604	1152	-147	-5390	-39200	56840	
38	15:52:44.03	david	1	0	611	1157	-156	-4690	-38640	64610	
39	15:52:44.05	david	1	0	563	1108	-148	-4690	-35630	73500	
40	15:52:44.06	david	1	0	478	1029	-115	-6020	-30310	81270	
41	15:52:44.07	david	1	0	393	973	-68	-8470	-25130	86170	
42	15:52:44.08	david	1	0	320	932	-46	-11620	-21910	88970	
43	15:52:44.08	david	1	0	269	916	-48	-15470	-19740	89880	
44	15:52:44.10	david	1	0	251	919	-59	-19810	-18060	89040	
45	15:52:44.10	david	1	0	241	928	-82	-23940	-16450	87290	
46	15:52:44.12	david	1	0	245	941	-117	-27580	-14630	85050	
47	15:52:44.12	david	1	0	264	950	-151	-30660	-12810	81900	
48	15:52:44.14	david	1	0	293	961	-198	-32900	-11270	77980	

Figure 4: Dataset

2.4 Content of the dataset

Currently, the dataset contains a single file (database.ods) which represents sensor data samples collected from accelerometer and gyroscope from our board in 1 millisecond interval. This data is represented by following columns (each column contains sensor data for one of the sensor's axes):

- acceleration x
- acceleration y
- acceleration z
- gyro x
- gyro y
- gyro z

There is an activity type represented by "activity" column which acts as label and reflects following activities:

- "0": walking
- "1": running

Apart of that, the dataset contains "wrist" column which represents the wrist where the device was placed to collect a sample:

- "0": left wrist
- "1": right wrist

Additionally, the dataset contains "time" and "username" columns which provide information about the time and user which collected these measurements.

3 Creation of the neural networks model

After collecting motion data, our task was to design a neural network model and train it with the data collected.

3.1 The design of the neural networks model

One difficulty of solving a problem of classification or more generally a problem of machine learning may be the variety of possible solutions. Selecting the most appropriate solution is critical because usually, we do not have time and resources to evaluate all of them. In this project, it was mandatory to use neural networks model especially using feedforward neural networks. This method is known to be efficient to solve classification problems.

We have decided to develop these models with the Keras framework with Python to design and train our model. This framework is designed with a focus for rapid experimenting and can be used on top of TensorFlow, CNTK, or Theano. Besides being compatible with STM32Cube.AI, Keras is also easy to use.

Figure 5: Keras

Our idea was to develop one neural network models for each of the six attributes. Therefore, we would have 6 classifiers (3 models for the accelerometer with three axes and 3 models for the gyroscope with three axes). Each classifiers gives us a result about the activity (0 if walking and 1 if running). To get the real activity, we take the majority vote of the 6 classifiers. One advantage with this solution is that we can have a weak classifiers and get good results. However, we do not know if all classifiers are equally important. Indeed, some directions of the gyroscope and accelerometer may be less significant than others for the predictions and more sensible to noises. We firstly designed one neural network and fed with the acceleration on the axes x.

Besides, the choice of making 6 classifiers comes from the fact that we can not have a matrix in input of our neural network. The input to feed the neural network can be only a column matrix.

3.2 Data structure to feed the neural network

In feedforward neural networks everything starts from the input layer. It is not possible to predict the type of motion activity based on a single sample of data but in using multiple data samples. Therefore, it is possible to detect some patterns in our training models.

A human needs on average between 1 and 2 seconds to make a single natural wrist movement. Here, our sensors measure every millisecond. Therefore we have a column matrix with 200 data. To generate the different training datasets used by our network models we decided to write a database_treatment.py script in Python. This script creates a csv for each sensor and each axe. On each line, we have 200 measures corresponding to a wrist movement and the 201th value corresponds to the activity measured (0 if walking and 1 if running). This script uses the previous dataset measured to generate these multiple datasets.

FG	FH	FI	FJ	FK	FL	FM	FN	FO	FP	FQ	FR	FS	FT	FU	FV	FW	FX	FY	FZ	GA	GB	GC	GD	GE	GF	GG	GH	GI	GJ	GK	GL	GM	GN	GO	GP	GQ	GR	GS	GT
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
46	16	4	11	27	52	73	103	136	166	193	220	242	252	260	263	260	253	242	226	214	212	221	233	249	268	289	296	289	260	209	157	134	186	223	260	289	312	344	0
316	281	255	238	224	212	203	193	181	177	178	173	163	153	154	171	193	193	190	168	138	110	100	118	151	180	204	206	193	154	100	40	-4	-19	-8	21	73	219	289	0
195	158	138	127	123	121	118	107	93	80	78	87	119	158	182	183	165	131	77	25	-5	-15	-1	27	60	92	118	132	131	121	121	148	176	166	123	77	51	106	170	0
154	112	90	83	88	96	111	122	122	115	118	130	152	181	212	228	230	217	191	152	109	70	45	40	55	86	118	144	160	171	175	183	209	243	273	278	260	197	185	0
130	114	116	119	115	114	114	118	123	130	148	172	194	208	216	225	229	232	237	240	239	236	233	234	241	254	265	268	259	232	197	150	141	165	205	262	268	278	300	0
168	145	129	106	79	48	15	-9	-30	-42	-30	-35	-28	-20	-19	-16	-17	-16	-9	0	17	43	73	104	128	146	146	128	129	162	204	240	199	117	2	-88	-64	-6	74	0
184	157	135	115	93	75	59	53	57	69	89	116	153	192	225	254	276	283	278	263	249	251	256	265	285	314	342	358	358	345	346	381	431	467	388	267	143	37	-23	0
165	144	126	114	97	82	74	65	61	60	58	57	55	61	75	90	82	65	49	18	-17	-53	-82	-107	-123	-126	-115	-93	-66	-48	-41	-60	-91	-181	-193	-158	-105	-42	42	0
-71	-73	-88	-113	-127	-126	-124	-124	-127	-136	-150	-164	-170	-171	-163	-150	-142	-139	-139	-145	-154	-165	-175	-188	-207	-226	-240	-255	-260	-255	-248	-234	-218	-209	-225	-251	-293	-348	-405	0
-156	-190	-203	-197	-176	-144	-121	-120	-135	-150	-161	-161	-161	-169	-178	-187	-194	-201	-191	-184	-191	-201	-207	-206	-208	-204	-186	-172	-166	-164	-162	-155	-140	-145	-212	-299	-373	-412	-412	0
-154	-158	-171	-193	-207	-212	-220	-218	-211	-199	-186	-159	-161	-169	-172	-173	-166	-167	-151	-128	-101	-90	-84	-93	-105	-125	-152	-157	-135	-77	7	93	152	260	298	306	275	230	190	0
12	32	46	48	41	30	22	17	9	4	6	11	10	14	15	21	37	60	89	119	141	146	129	101	60	92	134	182	207	205	191	180	199	237	277	308	316	0
120	103	86	68	50	29	10	-8	-37	-69	-95	-106	-112	-150	-211	-264	-291	-280	-237	-182	-133	-90	-44	2	36	43	15	-33	-78	-77	-45	-48	-65	-55	-23	42	135	288	316	0
94	89	88	90	88	87	89	95	102	113	125	132	134	128	130	136	141	139	130	116	106	103	113	129	147	158	167	166	153	124	86	77	103	120	115	75	74	100	139	0
-4	-14	-15	-19	-32	-61	-71	-85	-96	-105	-108	-98	-80	-69	-63	-61	-69	-91	-110	-116	-116	-112	-107	-94	-84	-81	-96	-121	-154	-184	-205	-209	-178	-120	-61	-39	-62	-95	-112	0
-87	-129	-148	-142	-120	-102	-91	-80	-71	-60	-49	-43	-37	-34	-33	-33	-31	-33	-38	-37	-31	-20	-9	-2	0	-2	-6	-13	-5	18	63	218	277	275	200	86	-10	-78	-101	0
58	62	65	58	43	23	7	-1	0	11	14	10	1	-9	-15	-19	-25	-35	-40	-39	-27	-8	16	40	61	75	74	59	39	33	56	102	95	67	36	24	28	36	46	0
-58	-61	-60	-58	-64	-68	-71	-74	-72	-59	-41	-39	-50	-60	-64	-61	-50	-36	-18	-3	7	15	16	9	-11	-37	-42	-18	2	9	10	10	28	60	92	114	123	120	107	0
56	51	47	40	37	38	37	42	45	53	64	74	84	88	85	74	50	29	16	12	-5	0	28	32	70	119	159	182	148	205	226	225	260	286	294	288	283	271	242	0
95	95	92	82	73	64	55	50	58	70	76	71	49	13	-17	-35	-41	-40	-29	-7	18	42	52	44	19	-22	-61	-75	-58	-40	-41	-52	-47	-20	25	85	138	171	178	0
-6	26	51	81	102	111	116	126	130	129	120	111	102	95	84	72	66	62	61	56	36	5	-25	-4	27	48	81	127	190	266	329	362	354	326	305	282	252	223	195	0
117	111	122	140	151	161	163	134	112	80	50	29	22	34	52	71	90	106	109	97	78	80	119	156	173	187	177	141	114	106	104	107	130	169	205	231	252	262	271	0
1	15	33	46	48	41	29	13	2	-3	-7	0	36	61	79	94	109	118	119	136	185	238	229	142	13	-101	-169	-172	-120	-36	57	142	210	261	299	317	319	313	305	0
37	29	14	5	-1	-8	-24	-48	-65	-79	-94	-114	-138	-167	-197	-225	-248	-225	-175	-102	-35	5	27	40	57	85	133	190	236	288	338	370	390	397	391	377	371	365	358	0
105	95	87	86	90	102	123	149	189	228	261	279	273	250	211	169	138	149	183	210	252	289	320	344	356	366	357	358	365	360	352	344	340	324	307	288	275	262	260	0
83	122	141	107	66	64	81	109	140	160	162	158	155	148	154	124	68	13	15	75	154	97	55	121	141	190	231	241	313	345	339	359	366	352	347	335	322	332	366	0
126	134	149	169	187	197	204	201	189	167	152	163	191	229	240	188	81	-34	-102	-117	-89	-37	22	83	181	210	231	238	234	229	232	247	271	295	316	342	368	390	412	0
111	114	120	124	120	111	106	79	27	-10	-15	-9	-53	-96	-119	-113	-71	-1	77	133	180	228	264	283	285	277	238	228	241	268	292	304	296	290	290	293	306	326	360	0
155	162	147	113	72	51	58	69	62	49	50	79	130	180	209	221	226	216	195	163	134	118	96	83	94	122	127	148	178	219	280	353	427	504	585	658	704	707	702	0
134	71	25	-5	-13	10	73	137	210	289	360	398	411	425	433	436	439	432	432	423	408	385	376	382	385	410	444	486	523	547	566	579	594	610	631	661	693	715	723	0
276	256	235	220	201	185	181	181	184	188	200	228	258	274	289	290	281	290	306	332	377	425	493	516	541	561	573	578	582	598	614	630	632	624	602	560	518	509	544	0
171	198	211	213	206	173	121	68	43	57	82	105	108	118	139	151	157	161	172	202	245	311	346	381	416	456	486	499	504	501	507	498	502	501	500	491	461	433	421	0
39	106	145	172	196	208	203	189	181	187	206	244	269	256	234	211	200	206	219	243	313	344	385	423	452	484	520	556	582	604	630	653	667	668	657	625	583	556	576	0
92	116	141	174	209	236	259	294	322	372	450	497	504	497	480	444	406	395	399	425	442	468	479	488	498	500	507	508	505	521	544	561	580	598	607	604	590	562	531	0
171	197	219	232	237	233	233	234	238	246	254	263	286	305	306	308	316	332	357	385	440	483	490	506	540	564	590	604	617	628	637	638	623	590	540	497	485	523	601	0
128	161	172	165	147	139	134	133	129	135	146	154	168	191	210	282	273	240	279	326	375	426	482	522	556	595	640	679	698	691	670	643	620	623	691	776	855	890	825	0
247	259	280	301	326	347	386	414	451	470	484	489	480	477	471	474	563	571	587	628	643	641	633	641	662	693	733	773	806	815	780	719	657	610	604	637	663	679	683	0
250	215	180	156	143	141	151	165	176	186	209	247	293	394	442	483	523	558	606	663	611	605	617	632	642	631	624	619	620	612	593	603	670	769	860	879	784	625	468	0
254	246	244	241	250	267	272	257	247	240	252	286	327	374	436	515	586	611	658	666	658	653	672	710	744	764	766	746	711	651	577	507	467	489	560	619	639	618	551	0
200	191	182	186	201	226	241	251	262	266	265	274	307	351	389	439	492	507	529	565	587	619	642	660	687	711	724	734	714	688	630	562	499	491	551	622	653	650	598	0
191	231	266	295	309	316	318	332	341	340	335	328	328	342	367	395	428	460	487	497	524	461	478	471	467	474	482	494	520	550	579	605	622	622	596	537	475	433	425	0
357	355	320	280	235	181	125	97	91	92	97	100	110	135	167	206	254	308	364	411	457	504	546	581	603	611	612	601	588	608	620	614	601	600	573	537	490	452	454	0
298	300	294	279	254	233	213	199	190	201	215	205	234	263	290	335	400	467	532	602	666	701	697	667	624	587	575	591	604	622	631	619	584	537	483	438	416	431	472	0
282	275	255	230	195	153	120	109	119	153	185	212	277	318	341	359	383	397	391	391	398	409	422	443	467	488	499	508	501	476	433	385	373	420	486	527	528	458	347	0
41	7	-20	-40	-52	-53	-41	-16	16	48	87	102	116	132	148	169	192	216	242	239	253	256	242	226	204	173	127	75	34	27	72	133	156	155	138	114	81	50	23	0
10	48	83	117	152	198	234	251	255	264	265	258	258	265	274	299	326	345	345	339	336	361	396	421	413	384	325	256	195	127	64	9	-36	-63	-70	-56	-34	-24	-25	0

Figure 6: acceleration$_x$.csv

3.3 Model architecture

After building the data structure to train our model, we needed to know the design of our model and more specifically the number of hidden layers, the number of neurons per layer and the activation functions.

To determine all these different parameters, we wrote a script python with Keras to test some possible configurations and compute their efficiencies. This script is divided into many functions:

- The function create_model() creates our neural network with the global variables nb_layers and nb_neurons corresponding respectively to the number of layers and the number of neurons. We can also modify the activation function in the builder Dense().

- The function LossHistory used by the callbacks and saving the logs for the training.

- The main function calls create_model() to create a model and trains it with a train dataset that is a subset of the dataset defined in the previous section. The model is tested with a test dataset (subset of the dataset different from in the previous section and disjointed from the train dataset) and a 10-fold cross-validations. We selected categorical cross-entropy loss function for its ability to increase a network's learning speed independently of defined learning rate. ADAM optimizer was chosen for our model for computational efficiency and delivering adequate results for the kind of problem we were trying to solve.

The idea was to compare the robustness of different models according different number of hidden layers, number of neurons per layer and activation functions. Thereby, for each activation function of Keras, we built models with one and two hidden layers and the number of neurons per layers varied between 5 and 45. A t-test for example would be more efficient but also more difficult to compute. The aim here was just to give us an approximate idea of the architecture of the model. After many computations, the search procedure shows that the tanh function for hidden layers and softmax functions for the output layer were the best in our case with 2 hidden layers and 35 neurons per layers. With these parameters we have in average a prediction accuracy of 93% with a variance of 2%.

We tested this model with the acceleration's data on the x axis. It works also for the accelerations on the other axis but not for the gyroscope because we do not have good performances.

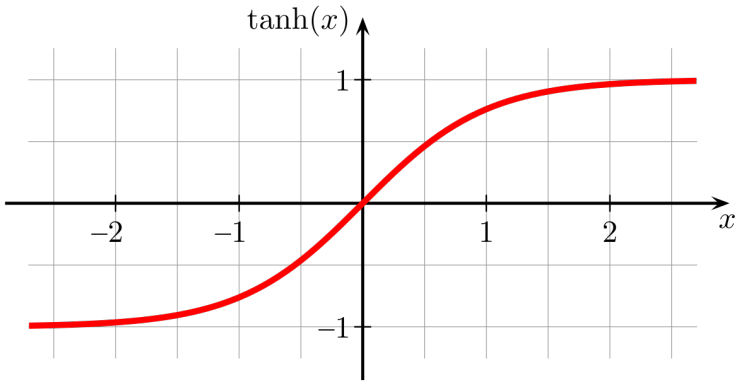

Figure 7: tanh function

4 Conversion of neural network model

Since our neural network model is written in Python, it is necessary to convert it into C++ code compatible to the utilization on our STM32 microcontroller developed by the company STMicroelectronics.

The company developed a software named STM32CubeMX that allows to generate C++ code according to the specific microcontroller being used and its configuration. And there exists an extension in this software called STM32Cube.AI that allows to convert a pre-trained neural network model into a C++ library that will run on the microcontroller.

Figure 8: STM32Cube.AI

After saving our neural network model into a .h5 file we loaded it into STM32CubeMX with the selection of the microcontroller STM32F401 provided on our Nucleo-64 board. After analyzing we found out that our model didn't take much memory space, we then chose not to compress it to avoid potential performance loss.

Figure 9: Memory space taken by neural network model

With the artificial intelligence application of the software being used, a UART-based link is expected. For our microcontroller, it is USART2 that supports the Virtual COM port. Therefore we activated it by configuring the pins PA2 and PA3 respectively for transmission and reception(see Figure 10).

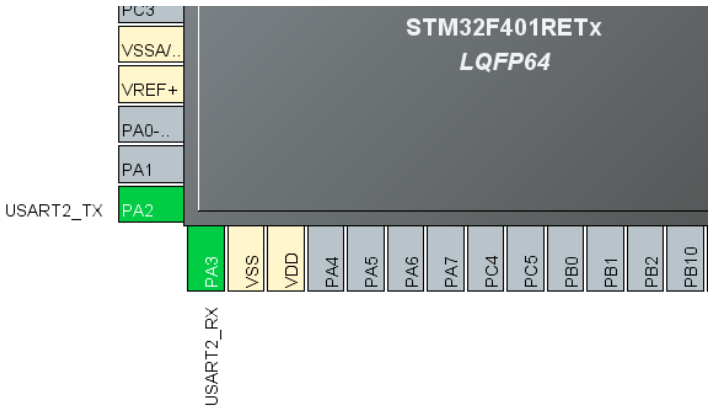

Figure 10: Pins configuration for USART2

The next step is to increase the speed to the maximum supported frequency, that is 84 MHz. The CubeMX automatically changes the clock source and configures the PLL to achieve the chosen frequency(see Figure 11).

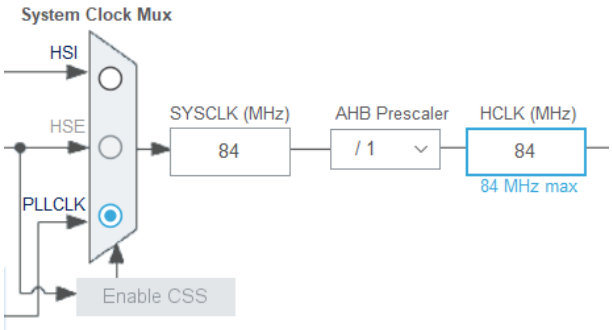

Figure 11: Clock configuration

After setting USART2 for AI application, we executed the generation, which outputs a folder containing the library. The key elements are the follow ones:

- Network_runtime.a: library that will be used by the AI core component

- estimator_data.h and estimator_data.c (with "estimator" the name of our project): containing all the weights of the neural network
- estimator.h and estimator.c: containing the functions that will be used in our application to call the neural network

In estimator.h we find the most important functions: ai_estimator_create, ai_estimator_init and ai_estimator_run. The last one is actually used for calling the neural network.

5 Implementation on Miosix

With the library generated, we can call the neural network in our program. But first of all we have to write the drivers to establish communication between the sensor and the computer. Once we can receive on the computer the measuring results coming from the sensor, we can pass this data into our neural network and therefore carry out the classification.

5.1 Transmission of sensor data to computer

There are existing drivers provided on the ARM platform Mbed for our sensor LSM6DSL. Application of our board expansion board with this sensor can be directly developed using these drivers on the platform.

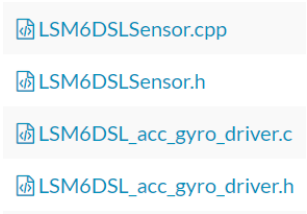

Figure 12: Sensor drivers

However, in order to implement application on our microcontroller, certain modifications have to be made in the drivers and the I2C protocol has to be written by ourselves (the header file LSM6DSL.h calls the I2C whose source is not provided).

The modifications consist in the suppression of the parts of code concerning SPI protocol, since we will use I2C for the microcontroller to communicate with the sensor. These modifications are rather simple to carry out.

The major effort lies in the implementation of I2C, with which the write/read operation follows stringent procedures. In short, the read/write process is the follow:

1. Send the device address
2. Send the register address
3. Read from that register or write to that register

The documentation of Miosix provides the basic function calls that we can use to implement I2C transmission protocol.

First of all, since I2C works with two wires, the SDA(data line) and SCL(clock line), we configure two pins of the microcontroller to these wires(Figure 13).

Writing data to an I2C device:

```
typedef Gpio<GPIOB_BASE,8> scl; // PORTB8
typedef Gpio<GPIOB_BASE,9> sda; // PORTB9
typedef SoftwareI2C<sda,scl> i2c;
```

Figure 13: Pins configuration I2C

First we send the start bit by calling "i2c::sendStart();", after that, we address our desired device by sending the device address + the write bit(0). After that, we are sending the address of the register in the memory of the slave device in which we want to write that data. Then we can send the data. Last but not least, we send our desired value, and issue a STOP signal.

This process leads us to the following code:

```
bool write(uint8_t* pBuffer, uint8_t deviceAddr, uint8_t registerAddr, uint16_t numByteToWrite)
{
    i2c::init(); // Initialasation of I2C
    int i=0;
    for(i=0;i<numByteToWrite;i++)
    {
        i2c::sendStart(); // START
        if (i2c::send((unsigned char)deviceAddr)) // Send device(slave) address + 0
        {
            if (i2c::send((unsigned char)(registerAddr+i))) // Send register address
            {
                if(!i2c::send((unsigned char)*(pBuffer+i))) // Send data
                    return false;
            }
            else
                return false;
        else
            return false;
    }
    i2c::sendStop(); // STOP
    delayUs(10);
    return true;
}
```

Figure 14: Function write

Reading data from an I2C device:

Again, we are issuing the start bit, and sending the device(slave) address with the write bit(0). This is because we first want the master to know from which slave we want to read data. After sending the slave address, we send the register from which we want to read. Now, we issue a repeated start, and we are ready to read. We send the slave device address, but this time with the read bit(1). Now the slave device will send us the value from the register which we pointed to.

This process leads us to the code in Figure (Figure 13)

Once the I2C protocol implemented, the driver is finished and ready to be used and we can call the functions provided in the drivers in the main.cpp to receive the data(Figure 16).

```
bool read(uint8_t* pBuffer, uint8_t deviceAddr, uint8_t registerAddr,  uint16_t numByteToRead)
{
    i2c::init(); // Initialasation of I2C
    int i=0;
    for (i=0;i<numByteToRead;i++)
    {
        i2c::sendStart(); // START
        if (i2c::send((unsigned char)deviceAddr)) // Send device(slave) address + 0
        {
            if (i2c::send((unsigned char)(registerAddr+i))) // Send register address
            {
                i2c::sendRepeatedStart(); // Repeated Start
                unsigned char sl_addr=(unsigned char)(deviceAddr+1); // Send device(slave) address + 1
                if(i2c::send((unsigned char)sl_addr))
                    *(pBuffer+i)=i2c::recvWithNack(); // Read data
            }
            else
                return false;
        }
        else
            return false;
    }
    i2c::sendStop(); // STOP
    delayUs(10);
    return true;
}
```

Figure 15: Function read

```
acc_gyro = new LSM6DSLSensor(LSM6DSL_ACC_GYRO_I2C_ADDRESS_HIGH); // I2C address
acc_gyro->enable_x(); // Enable accelerometer
acc_gyro->enable_g(); // Enable gyroscope
acc_gyro->get_x_axes(axes); // Get data from accelerometer
acc_gyro->get_g_axes(axes); // Get data from gyroscope
```

Figure 16: Code reading data from sensor

5.2 Configuration of Makefile

Before we can generate the application with our code, we still have to modify the makefile to allow the compilation.

Assumingly the modification only consists of adding the files necessary for the program as follows:

```
##
## List here your source files (both .s, .c and .cpp)
##
SRC = main.cpp
SRC += LSM6DSLSensor.cpp
SRC += LSM6DSL_acc_gyro_driver.c
```

Figure 17: Adding source files

However, when compiling, we encountered a C standard conflict, that is the “for” loop is only allowed in C99 mode, which is not the case of our compiler. The solution is to add “-std=gnu99” in the “CFLAGS” section of the Makefile(Figure 18).

Other issues we encountered consist of lack of files which we added gradually to the Makefile to solve them.

```
CFLAGS   := $(CFLAGS_BASE)   -I$(CONFPATH) -I$(CONFPATH)/config/$(BOARD_INC)  \
            -I. -I$(KPATH) -I$(KPATH)/arch/common -I$(KPATH)/$(ARCH_INC)      \
            -I$(KPATH)/$(BOARD_INC) $(INCLUDE_DIRS) \
            -std=gnu99
```

Figure 18: Compile in C99

5.3 Utilization of neural network library

As mentioned earlier, there are several functions provided by the library generated from the neural network that are crucial to the creation, initialization and running of the neural network in our environment.

Unfortunately, calling them in our main.cpp is not sufficient to let them work. We have encountered numerous errors when trying to put them into use. They include but are not limited to: lack of files, undefined variables, illogical function outputs, etc.

After long investigation and mending, we appear to have eliminated most of the bugs and reach this one:

"Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"

This is where we had to stop for not being able to find solution.

6 Conclusion

Thanks to this project, we have been able to learn the concept of neural networks, their implementations and utilisations with Keras and STM32Cube.AI. Besides, we wrote drivers to collect data from our sensors using I2C protocol on Miosix. Unfortunately, because of a lack of time, we did not success in finishing this project.

This project, even if very interesting, was maybe too challenging for us because the timeline was very tight for people who had no specific knowledge on machine learning techniques and drivers. Our last problem was to modify Miosix files or the Makefile to correctly compile our neural network model on Miosix.

However, despite this problem we are confident that our code works. If the problem had been solved, it would have been interesting to design the other network models for the others axis of the accelerometer and for the gyroscope and take the majority vote to predict the activity. Moreover, we could use more appropriate techniques to know how much layers and neurons per layer are needed by our model. Finally, we can try to solve this problem with other methods such as fuzzy logic systems or other classification methods like the Support Vector Machines, logistic regression, etc.