EmbML Tool: supporting the use of supervised learning algorithms in low-cost embedded systems

Lucas Tsutsui da Silva, Vinicius M. A. Souza
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, São Paulo, Brazil
lucastsutsui@usp.br, vmasouza@icmc.usp.br

Gustavo E. A. P. A. Batista
School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
gbatista@cse.unsw.edu.au

This document only includes a full version for the result section of our paper. In order to see the published paper, please follow this link: https://ieeexplore.ieee.org/document/8995408

I. Result analysis

This section presents the results from the experiments with the selected datasets and microcontrollers[1]. First, we compare the performance of the classifiers produced by EmbML and the impact of using two fixed-point representations: *Q21.10* (FP32) and *Q11.4* (FP16). Next, we contrast the performance of the classifiers produced by our proposal against those generated by other related tools.

In all experiments, the microcontrollers read the test set from a microSD memory card. However, the time needed to read such information is not considered in the results. In some cases, the size of the classifiers surpassed the microcontroller's storage capacity, so we ignored them in our analysis.

A. Comparing our classifiers

We start with a sanity check. We compare the accuracy rates obtained by the classifiers produced by our tool running on the embedded devices with the results obtained by scikit-learn or WEKA on a desktop computer for the same test sets.

Table I shows the accuracy rates for each dataset and model running in a desktop and in a microcontroller with a classifier produced by our tool. For the sake of brevity, we report the results of SVM classifier using only the linear kernel, although our tool also supports the polynomial and RBF kernels. Also, we note in a preliminary evaluation that the use of such kernel provided the best results for the datasets used in the experiments.

Table I does not mention the microcontroller model since all results are the same, independent of the embedded device. As expected, the classifiers using floating-point representation (FLT) present the same accuracy rates than their desktop counterparts. There are two minor exceptions. In D1 with Decision Tree from scikit-learn, the accuracy reduces from 98.54% (in desktop) to 98.53% (with EmbML's classifier); and in D2 with MLP from WEKA, the accuracy increases from 89.19% (in desktop) to 89.26% (with EmbML's classifier and FLT). Another important observation is that, in most cases, there is not a significant change in accuracy when using the FP32, comparing to FLT representation. However, the FP16 representation can cause a notable reduction in accuracy value for most of the classifiers.

[1]For simplicity, we will refer to MK20DX256VLH7 as MK20DX and to MK66FX1M0VMD18 as MK66FX

TABLE I
Accuracy (%) for the classifiers.

Classifier	Version	D1	D2	D3	D4	D5	D6
Dec. Tree scikit-learn	Desktop	98.54	86.13	84.02	97.03	83.83	93.20
	EmbML/FLT	98.53	86.13	84.02	97.03	83.83	93.20
	EmbML/FP32	98.49	85.78	84.28	97.03	83.83	92.85
	EmbML/FP16	70.46	81.37	63.06	61.00	83.83	75.18
SVM scikit-learn	Desktop	90.51	92.11	88.83	80.02	36.74	98.58
	EmbML/FLT	90.51	92.11	88.83	80.02	36.74	98.58
	EmbML/FP32	86.64	21.05	88.92	35.27	36.41	98.58
	EmbML/FP16	50.00	91.82	83.08	18.45	9.59	48.35
MLP scikit-learn	Desktop	95.96	92.46	91.41	96.43	89.96	98.54
	EmbML/FLT	95.96	92.46	91.41	96.43	89.96	98.54
	EmbML/FP32	96.12	92.60	91.84	96.26	89.87	98.38
	EmbML/FP16	56.44	5.12	64.09	16.67	57.77	38.32
Log. Reg. scikit-learn	Desktop	98.18	90.97	84.19	98.06	71.51	98.25
	EmbML/FLT	98.18	90.97	84.19	98.06	71.51	98.25
	EmbML/FP32	98.15	90.90	84.11	46.17	71.75	98.28
	EmbML/FP16	50.00	90.90	83.42	18.45	40.38	98.12
J48 WEKA	Desktop	99.00	88.48	84.28	97.41	84.71	94.34
	EmbML/FLT	99.00	88.48	84.28	97.41	84.71	94.34
	EmbML/FP32	98.97	88.41	84.54	97.41	84.71	94.01
	EmbML/FP16	97.25	87.06	68.56	58.65	84.71	79.61
SVM WEKA	Desktop	98.39	91.96	91.75	97.13	80.67	98.38
	EmbML/FLT	98.39	91.96	91.75	97.13	80.67	98.38
	EmbML/FP32	98.40	92.32	91.92	97.13	80.61	98.48
	EmbML/FP16	89.97	81.44	71.39	22.35	78.34	16.73
MLP WEKA	Desktop	98.67	89.19	90.29	92.84	80.46	93.62
	EmbML/FLT	98.67	89.26	90.29	92.84	80.46	93.62
	EmbML/FP32	98.65	90.33	90.46	92.86	80.58	93.66
	EmbML/FP16	54.40	88.62	88.49	18.38	79.88	92.72
Log. Reg. WEKA	Desktop	97.71	91.61	89.00	98.97	73.00	97.35
	EmbML/FLT	97.71	91.61	89.00	98.97	73.00	97.35
	EmbML/FP32	97.65	91.54	87.97	98.35	72.72	97.35
	EmbML/FP16	50.06	67.57	17.96	34.86	40.81	94.40

To analyse the time efficiency of the classifiers, let us consider two cases with small accuracy difference among the three numerical representations: SVM from WEKA in D1 (Fig. 1), and MLP from WEKA in D5 (Fig. 2). In Fig. 1, we observe that FP32 and FP16 reduce the classification time for ATmega2560 and MK20DX. However, in MK66FX, the FLT version still achieves shorter classification times than FP32 and FP16. The reason is that MK66FX has an FPU, so fixed-point representations do not make as much improvement in time results as they do in the other microcontrollers.

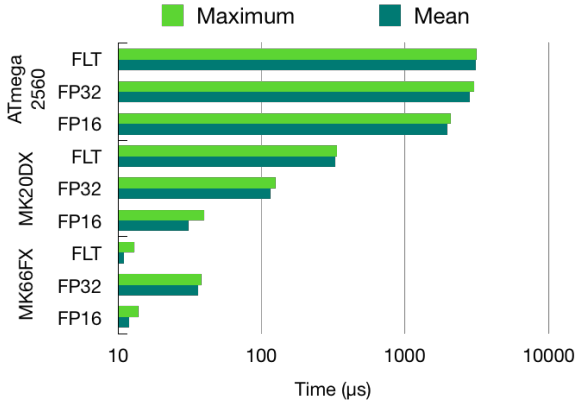


Fig. 1. Classification time comparison for SVM (from WEKA) in D1.

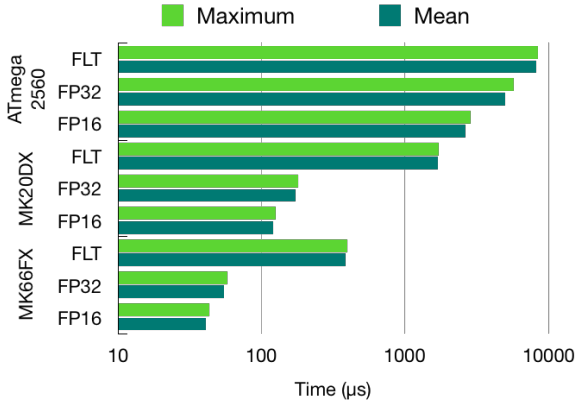


Fig. 2. Classification time comparison for MLP (from WEKA) in D5.

Fig. 2 shows the results for the MLP classifier from WEKA in D5. We observe that the FP32 versions consistently decreased classification time compared to FLT. Similarly, the FP16 versions improve time performance compared to FP32. Interestingly, in this particular case, the MK20DX classifier with FP32 representation outperformed the FLT version in a more robust hardware (MK66FX). Likewise, FP32 and FP16 improved the classifier performance on MK66FX compared with the FLT version even with the presence of an FPU in it.

B. Comparison with classifiers from other tools

In order to show that our tool produces competitive classifiers, we evaluated their performance against classifiers generated by other tools. To make a consistent comparison, we selected only models from other tools that have a correspondent in our tool. For example, we consider the MLP model from scikit-learn since both EmbML and emlearn support it. Also, we generated and evaluated the classifier versions with the same off-board-trained model and test set. For each classifier, we used the different datasets and platforms to compare mean classification time, shown in Table II, and memory consumption (SRAM + flash), shown in Table III. These tables highlight the best result for each dataset, and the symbol "-" means that the produced code was too large to fit in the microcontroller's memory.

TABLE II
COMPARISON OF MEAN CLASSIFICATION TIME (μs).

Classifier	Tool	ATmega2560 D1	D2	D3	D4	D5	D6
Dec. Tree scikit-learn	EmbML/FLT	107.30	83.18	96.48	88.71	-	-
	EmbML/FP32	66.95	52.54	62.07	56.04	-	-
	EmbML/FP16	**41.67**	38.07	36.72	**33.71**	-	-
	emlearn	55.28	**27.36**	**29.77**	56.69	**61.41**	**29.94**
	m2cgen	70.07	72.32	86.29	88.29	145.64	93.27
	sklearn-porter	153.42	-	-	-	-	-
SVM scikit-learn	EmbML/FLT	1379.43	4906.53	6086.57	14958.52	1504.24	-
	EmbML/FP32	868.96	5131.72	6428.47	13519.15	1471.69	-
	EmbML/FP16	**193.59**	**1578.91**	**1900.61**	**1495.83**	**197.09**	-
	m2cgen	755.16	4647.27	5750.71	14159.02	1375.07	**60566.52**
	sklearn-porter	792.77	4813.40	5953.19	15129.27	1520.61	-
MLP scikit-learn	EmbML/FLT	-	-	-	-	-	-
	EmbML/FP32	-	-	-	-	-	-
	EmbML/FP16	-	-	-	-	**16457.69**	-
	emlearn	-	-	-	-	-	-
Log. Reg. scikit-learn	EmbML/FLT	1382.83	4918.08	6111.57	15106.58	1508.19	-
	EmbML/FP32	885.33	5147.36	6410.91	10509.98	1485.17	-
	EmbML/FP16	**306.15**	**1576.63**	**1920.05**	**1550.10**	**346.67**	-
	m2cgen	758.45	4658.13	5775.24	14306.93	1378.42	**60806.55**
J48 WEKA	EmbML/FLT	46.49	47.58	53.37	71.44	-	-
	EmbML/FP32	29.33	29.96	33.14	43.81	-	-
	EmbML/FP16	**22.39**	**22.05**	**23.74**	**31.35**	**34.19**	**31.08**
	weka-porter	72.67	80.43	101.75	150.21	75.41	187.98

Classifier	Tool	MK20DX D1	D2	D3	D4	D5	D6
Dec. Tree scikit-learn	EmbML/FLT	19.99	14.92	17.10	16.44	17.14	18.10
	EmbML/FP32	7.01	5.31	6.15	5.86	6.12	6.49
	EmbML/FP16	**5.67**	4.99	**4.86**	**4.81**	**5.98**	**4.90**
	emlearn	11.90	**4.94**	5.90	11.93	12.70	5.97
	m2cgen	15.70	14.70	17.51	18.08	24.03	19.42
	sklearn-porter	36.62	28.64	33.20	31.73	-	37.08
SVM scikit-learn	EmbML/FLT	144.66	514.60	649.10	1534.63	170.58	6943.96
	EmbML/FP32	23.58	**67.38**	83.93	196.04	24.30	848.43
	EmbML/FP16	**22.62**	69.68	**82.87**	**180.19**	**22.50**	**824.25**
	m2cgen	84.68	480.67	631.03	1417.36	160.01	6623.55
	sklearn-porter	151.91	927.47	1165.81	2763.61	290.85	11991.95
MLP scikit-learn	EmbML/FLT	10035.30	15548.66	15825.15	30848.53	4012.01	131105.71
	EmbML/FP32	2211.37	3463.84	3514.89	6805.94	972.03	28508.80
	EmbML/FP16	**2080.93**	**3251.75**	**3309.49**	**6267.42**	**915.60**	**26291.28**
	emlearn	9002.85	14758.22	15085.99	29281.54	4101.68	-
Log. Reg. scikit-learn	EmbML/FLT	144.79	513.38	647.23	1530.07	170.05	6936.18
	EmbML/FP32	23.58	**67.39**	83.93	192.44	24.30	848.43
	EmbML/FP16	**23.13**	69.62	**83.15**	**180.09**	**23.12**	**832.10**
	m2cgen	84.75	502.57	628.97	1414.51	159.50	6614.93
J48 WEKA	EmbML/FLT	8.59	8.94	9.78	13.32	13.30	11.69
	EmbML/FP32	**3.40**	3.26	3.60	4.57	4.66	4.42
	EmbML/FP16	3.41	**3.11**	**3.32**	**4.33**	**4.63**	**4.21**
	weka-porter	12.37	12.02	13.59	18.52	16.74	18.26

Classifier	Tool	MK66FX D1	D2	D3	D4	D5	D6
Dec. Tree scikit-learn	EmbML/FLT	2.74	2.21	2.76	2.48	2.73	2.66
	EmbML/FP32	2.52	1.90	2.30	2.13	2.41	2.50
	EmbML/FP16	2.10	1.91	1.88	1.93	2.41	1.96
	emlearn	**0.83**	**0.53**	**0.45**	**1.11**	**1.52**	**0.78**
	m2cgen	1.87	1.55	2.05	2.04	3.02	2.54
	sklearn-porter	13.24	10.45	12.07	11.24	11.91	12.90
SVM scikit-learn	EmbML/FLT	4.54	17.95	22.46	**54.04**	6.77	**169.87**
	EmbML/FP32	8.78	25.61	31.75	73.81	9.02	343.15
	EmbML/FP16	8.70	26.85	31.94	70.19	8.72	324.37
	m2cgen	**2.65**	**13.40**	**16.68**	58.73	**4.05**	309.35
	sklearn-porter	42.03	262.24	329.48	777.69	86.80	3344.85
MLP scikit-learn	EmbML/FLT	654.26	983.07	994.53	1925.95	259.21	9149.05
	EmbML/FP32	852.23	1331.26	1349.90	2621.73	359.29	11892.27
	EmbML/FP16	794.89	1275.43	1294.96	2550.54	342.55	11296.79
	emlearn	**349.82**	**548.57**	**560.88**	**1058.44**	**174.33**	**5171.48**
Log. Reg. scikit-learn	EmbML/FLT	4.54	17.95	22.46	**54.04**	6.77	**169.88**
	EmbML/FP32	8.82	25.63	31.75	71.88	9.03	343.09
	EmbML/FP16	8.87	26.84	31.97	70.23	8.95	326.14
	m2cgen	**2.65**	**13.67**	**16.66**	58.73	**4.05**	309.36
J48 WEKA	EmbML/FLT	1.61	1.47	1.66	2.07	1.96	1.84
	EmbML/FP32	1.24	**1.05**	**1.22**	1.70	1.58	1.54
	EmbML/FP16	**1.16**	1.13	1.23	**1.53**	1.48	**1.39**
	weka-porter	1.55	1.45	2.05	3.26	**1.36**	4.33

The time results in Table II show that EmbML's classifiers perform best with ATmega2560 and MK20DX. However, with MK66FX, they are frequently beaten by other classifiers. The explanation again is that MK66FX has an FPU, so fixed-point representations are likely not to improve the performance. Moreover, many of the runtimes for MK66FX are tiny (under $10\mu s$), so significant improvements in classification time may not represent significative gains for most applications.

Table III shows that the FP16 versions have the lowest memory usage in most of the cases. However, the overhead imposed by the other representations (FLT and FP32) is small. From

TABLE III
COMPARISON OF MEMORY CONSUMPTION (KB).

Classifier	Tool	ATmega2560 D1	D2	D3	D4	D5	D6
Dec. Tree scikit-learn	EmbML/FLT	20.98	21.78	22.24	24.59	-	-
	EmbML/FP32	21.47	22.27	22.88	25.08	-	-
	EmbML/FP16	20.05	20.64	21.16	**22.77**	-	-
	emlearn	**18.82**	**17.23**	**19.82**	26.39	**52.55**	26.42
	m2cgen	22.81	20.92	21.45	23.87	85.82	**24.50**
	sklearn-porter	24.39	-	-	-	-	-
SVM scikit-learn	EmbML/FLT	16.27	17.73	18.24	22.00	16.09	-
	EmbML/FP32	16.75	18.20	18.86	22.46	16.44	-
	EmbML/FP16	16.11	**16.86**	**17.26**	**18.98**	**15.87**	-
	m2cgen	17.04	27.91	30.54	51.93	18.39	**178.79**
	sklearn-porter	**15.96**	17.90	18.41	22.16	16.21	-
MLP scikit-learn	EmbML/FLT	-	-	-	-	-	-
	EmbML/FP32	-	-	-	-	-	-
	EmbML/FP16	-	-	-	-	**23.49**	-
	emlearn	-	-	-	-	-	-
Log. Reg. scikit-learn	EmbML/FLT	16.27	17.73	18.24	22.00	16.09	-
	EmbML/FP32	16.75	18.20	18.86	22.46	16.44	-
	EmbML/FP16	**16.11**	**16.86**	**17.26**	**18.98**	**15.87**	-
	m2cgen	17.04	27.91	30.54	51.93	18.39	**178.79**
J48 WEKA	EmbML/FLT	17.54	18.31	19.09	21.12	-	-
	EmbML/FP32	18.00	18.79	19.57	21.59	-	-
	EmbML/FP16	**17.50**	**17.95**	**18.55**	**20.07**	**28.22**	**20.82**
	weka-porter	18.70	21.20	23.19	27.71	41.90	29.59

Classifier	Tool	MK20DX D1	D2	D3	D4	D5	D6
Dec. Tree scikit-learn	EmbML/FLT	39.02	39.46	39.70	40.99	53.47	42.64
	EmbML/FP32	39.59	40.02	40.26	41.55	54.03	43.21
	EmbML/FP16	**38.86**	39.18	39.36	**40.30**	**50.13**	**41.23**
	emlearn	39.27	**38.66**	**39.23**	42.79	56.70	43.29
	m2cgen	41.65	40.42	40.68	42.11	86.95	43.18
	sklearn-porter	55.23	63.30	68.21	81.83	-	78.19
SVM scikit-learn	EmbML/FLT	37.66	38.44	38.70	40.70	37.56	52.60
	EmbML/FP32	37.11	37.88	38.13	40.13	37.00	51.97
	EmbML/FP16	**36.86**	**37.24**	**37.38**	**38.37**	**36.80**	**44.29**
	m2cgen	39.59	44.92	46.37	57.55	40.45	136.74
	sklearn-porter	38.02	41.50	42.52	49.84	38.62	92.13
MLP scikit-learn	EmbML/FLT	55.85	64.66	65.05	90.70	44.94	261.60
	EmbML/FP32	55.23	64.03	64.43	90.08	44.31	260.97
	EmbML/FP16	**45.68**	**50.42**	**50.62**	**63.45**	**40.56**	**148.92**
	emlearn	60.16	70.02	70.41	96.27	50.29	-
Log. Reg. scikit-learn	EmbML/FLT	37.66	38.44	38.70	40.70	37.56	52.60
	EmbML/FP32	37.11	37.88	38.13	40.13	37.00	51.97
	EmbML/FP16	**36.86**	**37.24**	**37.38**	**38.37**	**36.80**	**44.29**
	m2cgen	39.59	44.92	46.37	57.55	40.45	136.74
J48 WEKA	EmbML/FLT	**38.68**	**39.12**	**39.51**	40.66	45.59	42.27
	EmbML/FP32	39.24	39.68	40.07	41.22	46.16	41.45
	EmbML/FP16	38.90	39.22	39.52	**40.35**	**44.32**	**39.82**
	weka-porter	39.27	40.05	41.55	43.73	54.95	44.80

Classifier	Tool	MK66FX D1	D2	D3	D4	D5	D6
Dec. Tree scikit-learn	EmbML/FLT	43.34	43.78	44.01	45.32	57.79	46.90
	EmbML/FP32	41.96	42.39	42.63	43.93	56.40	45.52
	EmbML/FP16	**41.23**	**41.55**	**41.74**	**42.68**	**52.51**	**43.54**
	emlearn	42.26	41.60	42.23	46.10	60.07	46.85
	m2cgen	44.27	42.67	43.04	44.66	92.94	45.98
	sklearn-porter	58.36	66.36	71.33	84.95	273.01	81.31
SVM scikit-learn	EmbML/FLT	40.73	41.57	41.82	43.82	40.69	55.66
	EmbML/FP32	39.42	40.18	40.44	42.44	39.31	54.34
	EmbML/FP16	**39.16**	**39.55**	**39.68**	**40.68**	**39.11**	**46.66**
	m2cgen	39.64	42.98	43.73	51.60	39.94	105.73
	sklearn-porter	41.15	44.69	45.71	52.96	41.81	95.25
MLP scikit-learn	EmbML/FLT	59.11	67.79	68.18	92.45	48.07	263.28
	EmbML/FP32	57.54	66.41	66.80	92.45	46.68	263.28
	EmbML/FP16	**47.99**	**52.80**	**53.00**	**65.82**	**42.94**	**151.29**
	emlearn	61.53	71.39	71.79	97.64	51.66	271.86
Log. Reg. scikit-learn	EmbML/FLT	40.73	41.57	41.82	43.82	40.69	55.66
	EmbML/FP32	39.42	40.18	40.44	42.44	39.31	54.34
	EmbML/FP16	**39.16**	**39.55**	**39.68**	**40.68**	**39.11**	**46.66**
	m2cgen	39.64	42.98	43.73	51.60	39.94	105.73
J48 WEKA	EmbML/FLT	41.55	40.61	41.00	42.14	48.47	43.70
	EmbML/FP32	41.61	40.61	41.00	42.21	48.54	43.76
	EmbML/FP16	41.27	**40.21**	**40.51**	**41.34**	**46.69**	**42.13**
	weka-porter	**40.27**	40.92	42.23	44.29	51.07	45.85

both tables, we can observe that the decision tree classifiers achieve the best classification times and intermediate results in memory consumption. On the other hand, the MLP classifiers have the highest values for these two metrics.

The analysis of time and memory usage can lead to some incorrect conclusions if not accompanied by accuracy rates, as the fixed-point representations impact the classifiers accuracies. Even though the FP16 representation obtains the best results of classification time and memory usage for most of the cases, we have previously shown in Table I that it frequently reduces the accuracy to unacceptable values. The same happens to some classifiers generated by other tools. For instance, the decision tree (from scikit-learn) classifier, produced by emlearn for the D1 dataset, presents the best memory usage and second best mean classification time running in the ATmega2560. However, its accuracy is 73.46%, considerably lower than the value of 98.53% achieved with the classifier produced by EmbML for this same case using FLT.

Therefore, some specific cases were selected to make a comparative analysis between the classifiers that also includes their accuracy values. We present results of Decision Tree and SVM classifiers since a higher number of tools supports them. Also, we preferred scenarios in which analyzing only time and memory can be deceiving, *i.e.*, when classifiers reach good results in classification time or memory usage but have poor accuracy values.

Fig. 3 presents a comparison of mean classification time per instance and accuracy for the EmbML's classifiers and those produced by other tools. In Fig. 3(a) and Fig. 3(b), EmbML's classifiers obtain the best time performances – between those with the highest accuracies – for ATmega2560 and MK20DX and for MK66FX they are relatively near the best. In Fig. 3(c), the opposite occurs, EmbML's classifiers achieve the best time performance with MK66FX, and they are close to the best in ATmega2560 and MK20DX.

Fig. 4 presents comparisons of memory usage and accuracy. Specifically, Fig. 4(a) and Fig. 4(b) present examples in which the EmbML's classifiers reach the lowest memory usage for the classifiers with the highest accuracies running in MK20DX and MK66FX, while their results maintain relatively close to the lowest in ATmega2560. For the case presented in Fig. 4(c), we can consider all classifiers, since there is a small accuracy variation (less than 0.5%). Then, the EmbML's classifiers achieve the best memory performance in all microcontrollers.

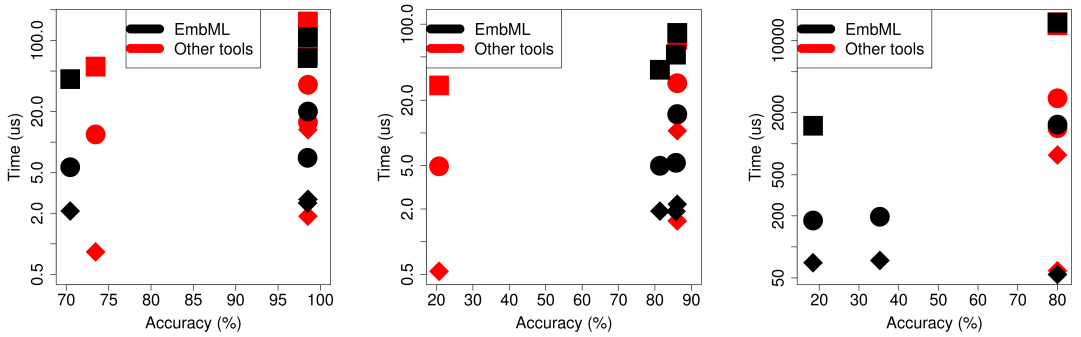
For reproducibility reasons, we make public a repository[2] that contains the supplementary material for this paper. It includes the implementation of EmbML and detailed results from our experiments.

II. CONCLUSION

In this article, we presented the *EmbML* software tool that implements a pipeline to develop classifiers for low-powered embedded systems. It begins with off-board-training a classifier in a desktop or server computer using popular software packages or libraries as WEKA or scikit-learn. EmbML converts the classifier into a carefully crafted C++ code with specific support for embedded hardware, such as the avoidance of unnecessary use of main memory and implementation of fixed-point operations for non-integer numbers.

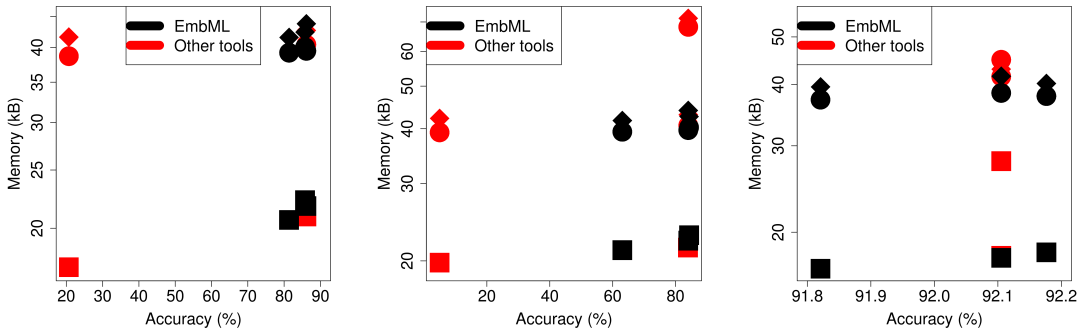
In our experimental evaluation, we have empirically demonstrated that the classifiers produced by EmbML maintain the

[2]https://github.com/lucastsutsui/paper-ictai-2019



(a) Dec. Tree - scikit-learn - D1 (b) Dec. Tree - scikit-learn - D2 (c) SVM - scikit-learn - D4

Fig. 3. Accuracy and mean classification time comparison. In the graphics, each ○ symbol represents the result achieved with ATmega2560, □ represents results from MK20DX, and ◊ represents results from MK66FX.



(a) Dec. Tree - scikit-learn - D2 (b) Dec. Tree - scikit-learn - D3 (c) SVM - scikit-learn - D2

Fig. 4. Accuracy and memory consumption comparison. In the graphics, each ○ symbol represents the result achieved with ATmega2560, □ represents results from MK20DX, and ◊ represents results from MK66FX.

accuracy rates obtained with the training tools, and improve classification time by using fixed-point representations. Moreover, we conducted a comparative analysis with classifiers produced by similar tools. We showed that EmbML generates classifiers with competitive accuracy, classification time, and memory usage running in different sorts of microcontrollers.

Besides the proposal of EmbML, another contribution of this paper is the analysis with benchmark data, different hardware, and other tools. Since the previously proposed non-commercial tools are available without a more detailed description and comparisons, our contributions can help both researchers and practitioners.

Future work will assess the classifiers produced by EmbML in an intelligent trap for real-time classification and capture of flying mosquitoes [1], [2]. We also plan to analyze the impact of piecewise linear approximation for activation functions in MLP classifiers.

Acknowledgment

This study was financed by São Paulo Research Foundation (FAPESP) in the grant numbers #16/04986-6 and #18/05859-3, the Brazilian National Council for Scientific and Technological Development (CNPq) in the grant numbers 306631/2016-4 and 166919/2017-9, and the United States Agency for International Development (USAID), grant AID-OAA-F-16-00072.

References

[1] V. M. A. Souza, D. F. Silva, and G. E. A. P. A. Batista, "Classification of data streams applied to insect recognition: Initial results," in *BRACIS*, 2013, pp. 76–81.

[2] D. F. Silva, V. M. A. Souza, D. P. W. Ellis, E. J. Keogh, and G. E. A. P. A. Batista, "Exploring low cost laser sensors to identify flying insect species," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 1, pp. 313–330, 2015.