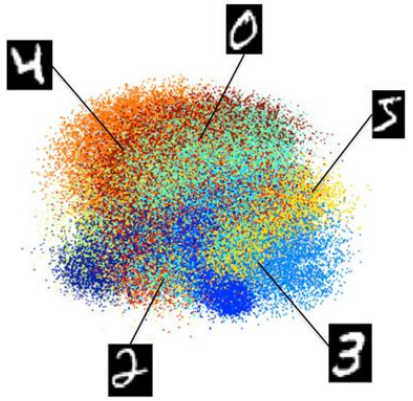
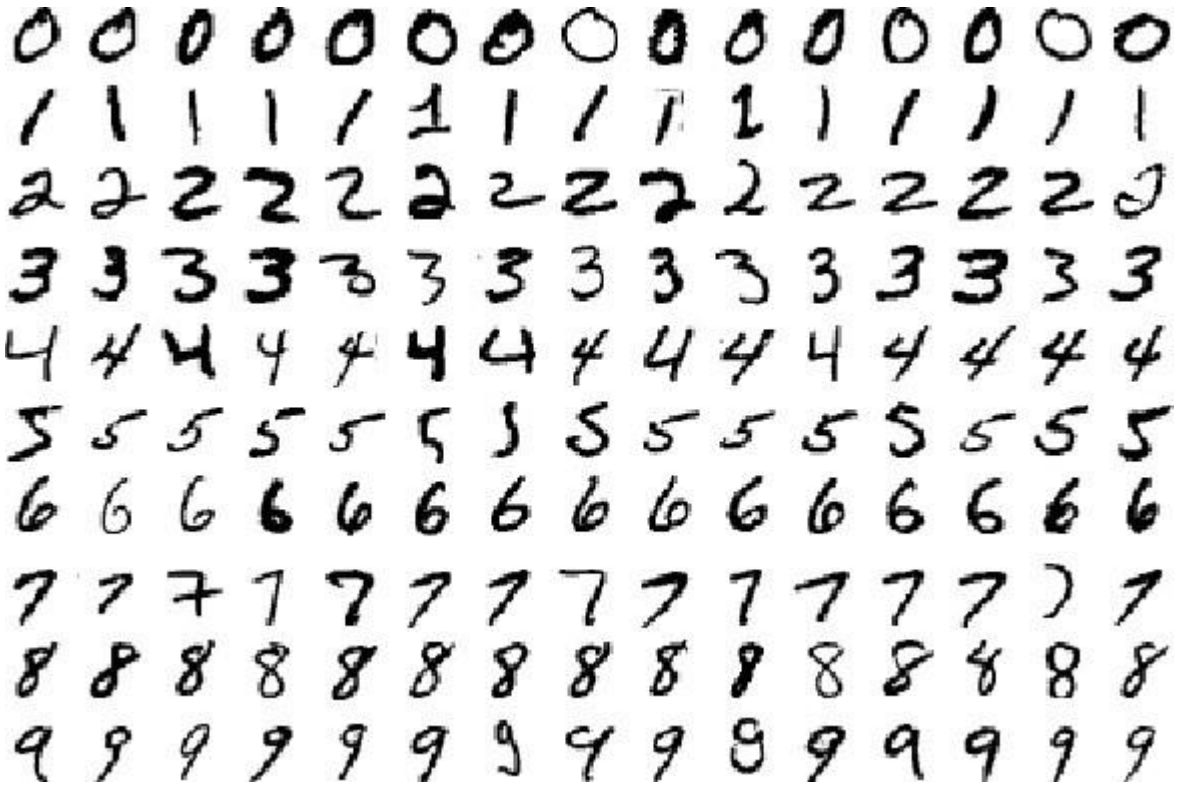
Dataset

MNIST Dataset

- Mixed National Institute of Standards and Technology
- Manuscript Digits from 0 to 9
- Black&White Images 28x28
- 60 000 training data
- 10 000 test data





Example of results on a simple CNN

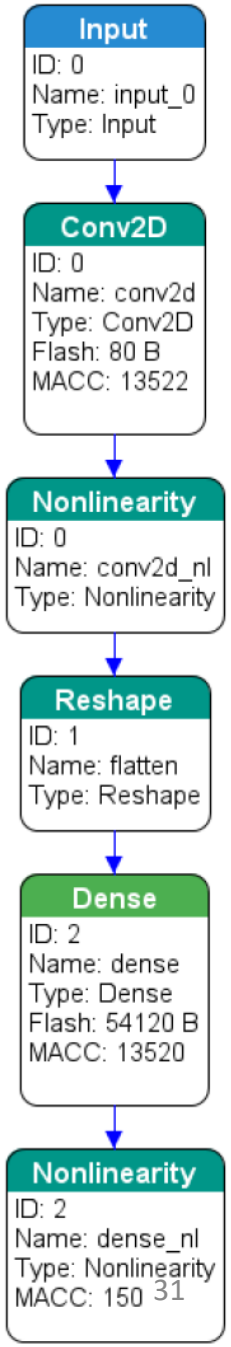
Results from tensorflow

Layer	Output shape	Number of parameters	Kernel (If conv2D)
Input	(28, 28, 1)	-	
Conv2D layers	(26, 26, 2)	20	3x3
Flatten	(1352)	–	
Dense layers	(10)	13,530	
Total trainable parameters	13,550		
Number of Epochs	3		

Results from STMCubeAI

	Type	Param #	MACC	MACC (%)	ROM (bytes)	ROM (%)	Bytes per Param
Layer 1	Conv2D	20	13 522	49,7	80	0,1	4
Layer 2	Dense	13530	13 520	49,7	54,120	99,9	4
…	DenseNL		150	0,6		0	
TOTAL			**27192**	**100**	**54,200**	**100**	**4**

Total memory
128kB RAM
1MB Flash



Validation on desktop

Results during inference	Number of inferences (test dataset)	Accuracy	RMSE	MAE	MACC	ROM (bytes)
Original model (result from TF)	250	96,3	0.069338	0.0131	27,192	54,200
Without compression		97,2	0.069338	0.013095	27,192	54,200
Compression X4		97,2	0.069323	0.013082	27,192	14,664
Compression X8		97,2	0.070925	0.013352	27,192	6,944

Validation on target

Results during inference	Number of inferences (test dataset)	Accuracy without compression	RMSE	Total latency (ms)	CPU cycles	Cycles / MACC
Original model (result from TF)	250	97,2	[redacted]			
Model validated on Laptop without compression		97,2				
Model validated on Target without compression		97,2	0.069338	5.413	433025	15.92
Model validated on Target with compression X4		97,2	0.069323	5.758	460646	16.94
Model validated on Target with compression X4		97,2	0.070925	5.462	436952	16.07

Labs sessions

- **Lab 1: installation and first execution of CNN on MCU**
  - **Validation on MNIST dataset**
  - **Learn a model and optimize it**
- Lab 2: optimization of CNN on MCU v1
  - Propose a model, learn and test
  - Application on MNIST and HAR dataset
- Break
- Lab 3: Apply to real data
  - Get data either from the shield or from the smartphone
  - Constitute a specific test set
  - Test onto the MCU
- Lab 4: Introduction to MicroAI
  - Directly export and generate code from TF

Session 2

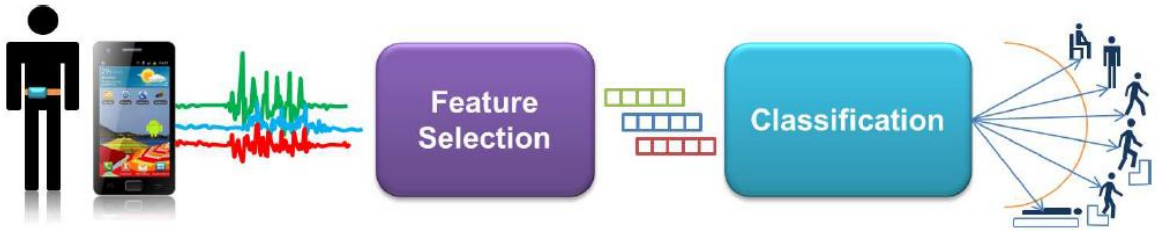
Labs sessions

- Session 1: installation and first execution of CNN on MCU
  - Validation on MNIST dataset
  - Learn a model and optimize it
- **Session 2: optimization of CNN on MCU v1 MNIST and HAR**
  - **Propose a model, learn and test**
  - **Application on MNIST dataset**
- Break
- Session 3: programmation of the MCU with MicroAI
  - Propose a model, learn and test
  - Application on HAR dataset
- Session 4: Apply to real data
  - Get data either from accelerometer
  - Constitute a specific test set
  - Test onto the MCU

UCA HAR Dataset

http://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

- 6 activity classes
- 30 subjects (19 to 48 years old)
- Data acquisition from a smartphone
- Laboratory conditions but subject perform the scenario freely
- Classification results using SVM up to 96% accuracy (but only 88% recall on *sitting*)

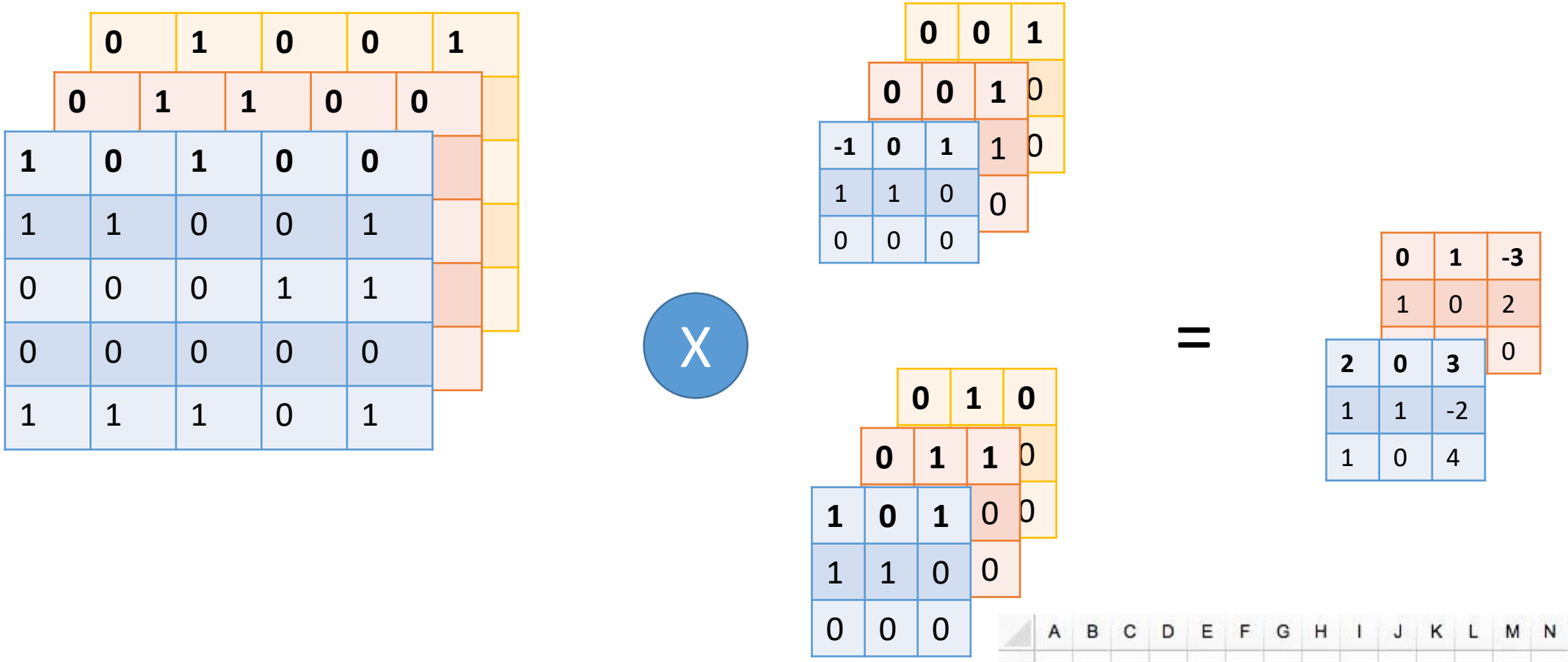


Data

- Median filter
- 3rd order 20Hz low-pass filter (99% of the signal energy is below 15Hz)
- Linear acceleration and gravity separated by 0.3Hz low-pass filter
- 2.56s windows at 50Hz = 128 samples
- 90-130 steps/min on average, at least 1.5 step/s = 2 step per window
- Available from **UC Irvine Machine Learning Repository**

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine. International Workshop of Ambient Assisted Living (IWAAL 2012). Vitoria-Gasteiz, Spain. Dec 2012

Number of input channels : M = 3 Convolution Nb of Kernels : 2 Output channels : 2



2D and 1D Convolutions

Exemple of results on the HAR classes

No.	Static	Time (sec)	No.	Dynamic	Time (sec)
0	Start (Standing Pos)	0	7	Walk (1)	15
1	Stand (1)	15	8	Walk (2)	15
2	Sit (1)	15	9	Walk Downstairs (1)	12
3	Stand (2)	15	10	Walk Upstairs (2)	12
4	Lay Down (1)	15	11	Walk Downstairs (1)	12
5	Sit (2)	15	12	Walk Upstairs (2)	12
6	Lay Down (2)	15	13	Walk Downstairs (3)	12
			14	Walk Upstairs (3)	12
			15	Stop	0
				Total	**192**

e 1: Protocol of activities for the HAR Experiment.

	WK	WU	WD	ST	SD	LD	**Recall**
Walking	**492**	1	3	0	0	0	99%
W. Upstairs	18	**451**	2	0	0	0	96%
W. Downstairs	4	6	**410**	0	0	0	98%
Sitting	0	2	0	**432**	57	0	88%
Standing	0	0	0	14	**518**	0	97%
Laying Down	0	0	0	0	0	**537**	100%
Precision	96%	98%	99%	97%	90%	100%	**96%**

Table 4: Confusion Matrix of the classification results on the test data using the multi-class SVM. Rows represent the actual class and columns the predicted class. Activity names on top are abbreviated.

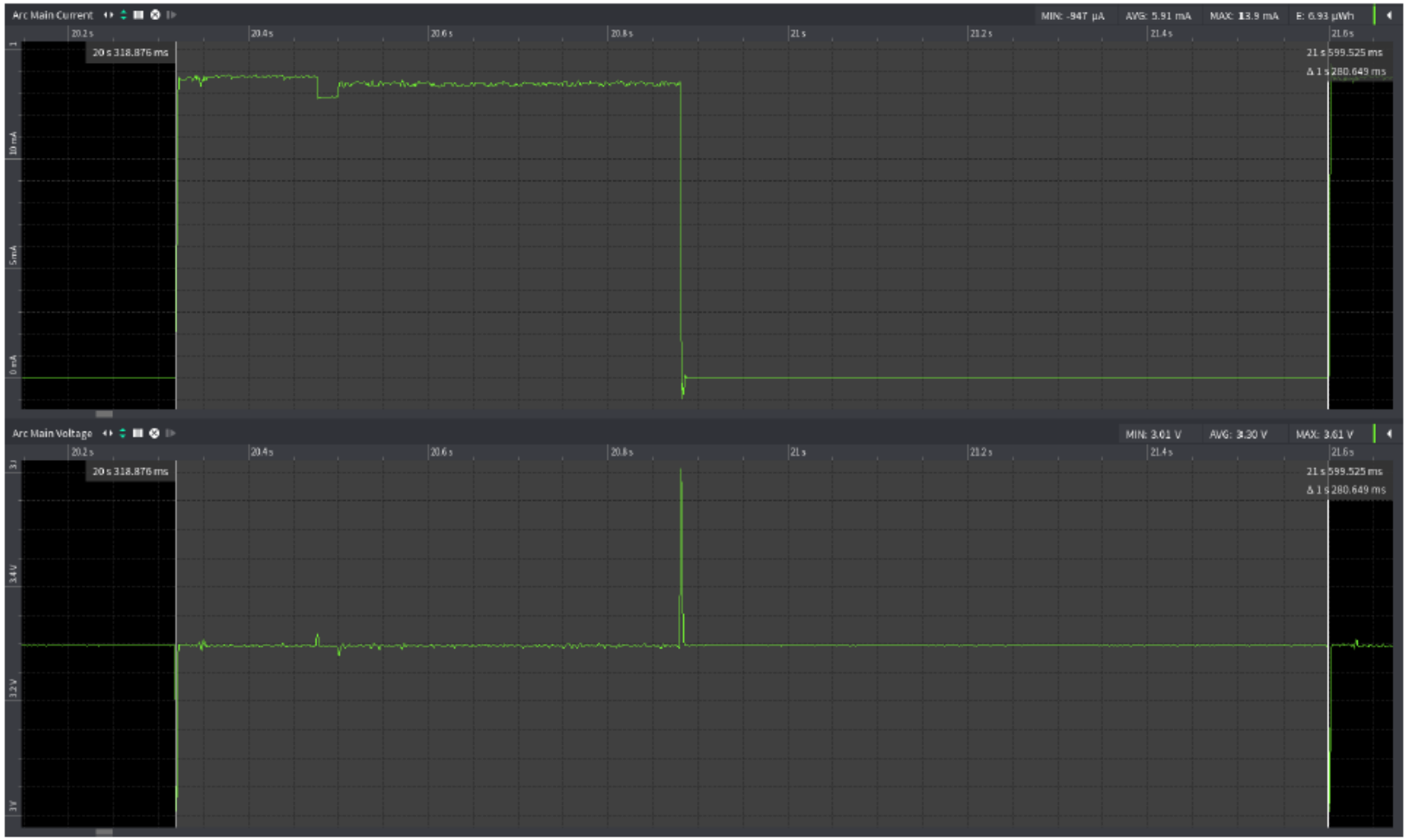
Mean accuracy for several methods

Supervised Machine Learning		Unsupervised Machine Learning		Neural Networks	
Nearest Neighbours[2]	91.0	K-means[2]	52.1	PNN[3]	92.4
Decision Tree[2]	87.4	Mini Batch K-means[2]	50.7	RNN[3]	95.1
Random Forest[2]	91.4	Spectral Clustering[2]	57.8	BPNN[3]	94.7
SVM[2]	90.7	Gaussian Mixture[2]	49.8	Deep Convolutional NN [4]	95.75
AdaBoost[2]	40.8	DBSCAN[2]	16.4	Stacked Autoencoder [5]	92.16
Naive Bayes[2]	88.9			Denoising Autoencoder [5]	90.50
QDA[2]	90.0			CNN (3 layers)*[6]	92.71
Multi-Class SVM [1]	96			LSTM (standard / bidirectional)[6]	89.1 / 89.40
OVO multiclass linear SVM with majority voting [7]	96.4			MLP[6]	86.83
A sparse kernelized matrix learning vector quantization model [8]	96.23				
Confidence-based boosting algorithm Conf-AdaBoost [9]	94.33				
Two-Stage Continuous Hidden Learning Markov Models [10]	91.76				

Example of results on different topologies on 2 different MCU

N°	Avg time(s)		Accuracy (%)	ROM (B)		Avg power (mW)		Battery runtime (h)	
	Ambiq	STM		Ambiq	STM	Ambiq	STM	Ambiq	STM
M1	0.085	0.0051	84.87	108270	103780	0.66	0.20	136.27	452.50
M2	0.095	0.0074	85.94	109340	104776	0.74	0.29	121.88	312.86
M3	0.117	0.0126	88.03	140647	136140	0.91	0.49	99.30	184.07
M4	1.249		88.94	469450		9.66		9.32	
C1	0.071		77.95	79222		0.55		163.85	
C2	0.078	0.0025	87.11	81820	80052	0.61	0.10	148.62	930.76
C3	0.081	0.0030	88.40	80487	78696	0.63	0.11	143.32	768.43
C4	0.085	0.0052	89.89	153956	150256	0.66	0.20	136.50	449.68
C5	0.113	0.0103	90.26	222916	221336	0.88	0.40	102.81	225.07
C6	0.166	0.0220	91.46	366876	365296	1.29	0.85	70.00	105.72
C7	0.352	0.0633	92.58	554124	552616	2.72	2.45	33.06	36.78
C8	0.601	0.1185	92.88	758348	808920	4.65	4.58	19.36	19.64

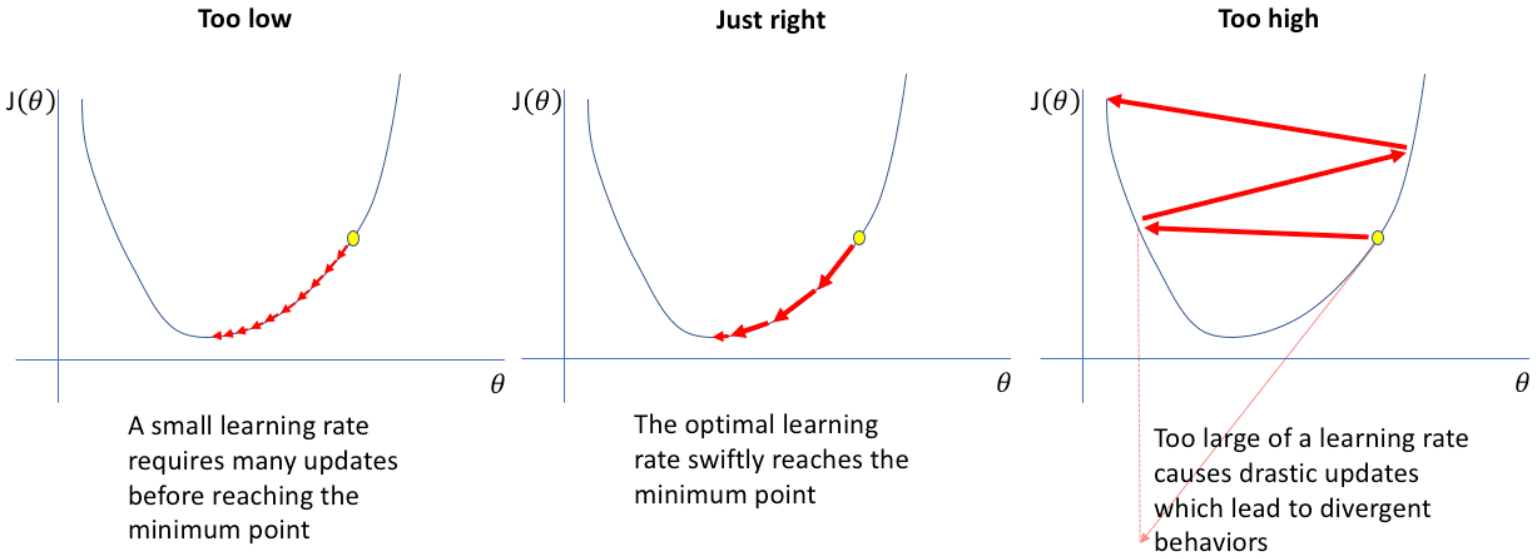
Power consumption measurement

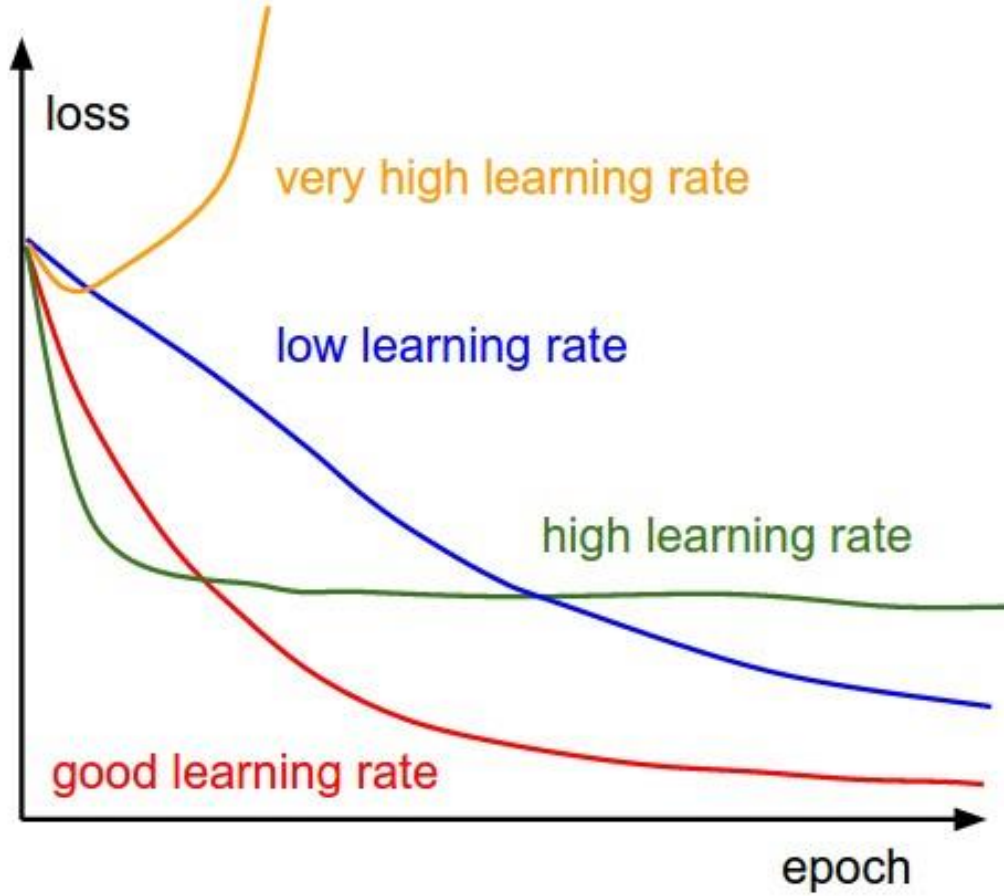


How get the best results ?

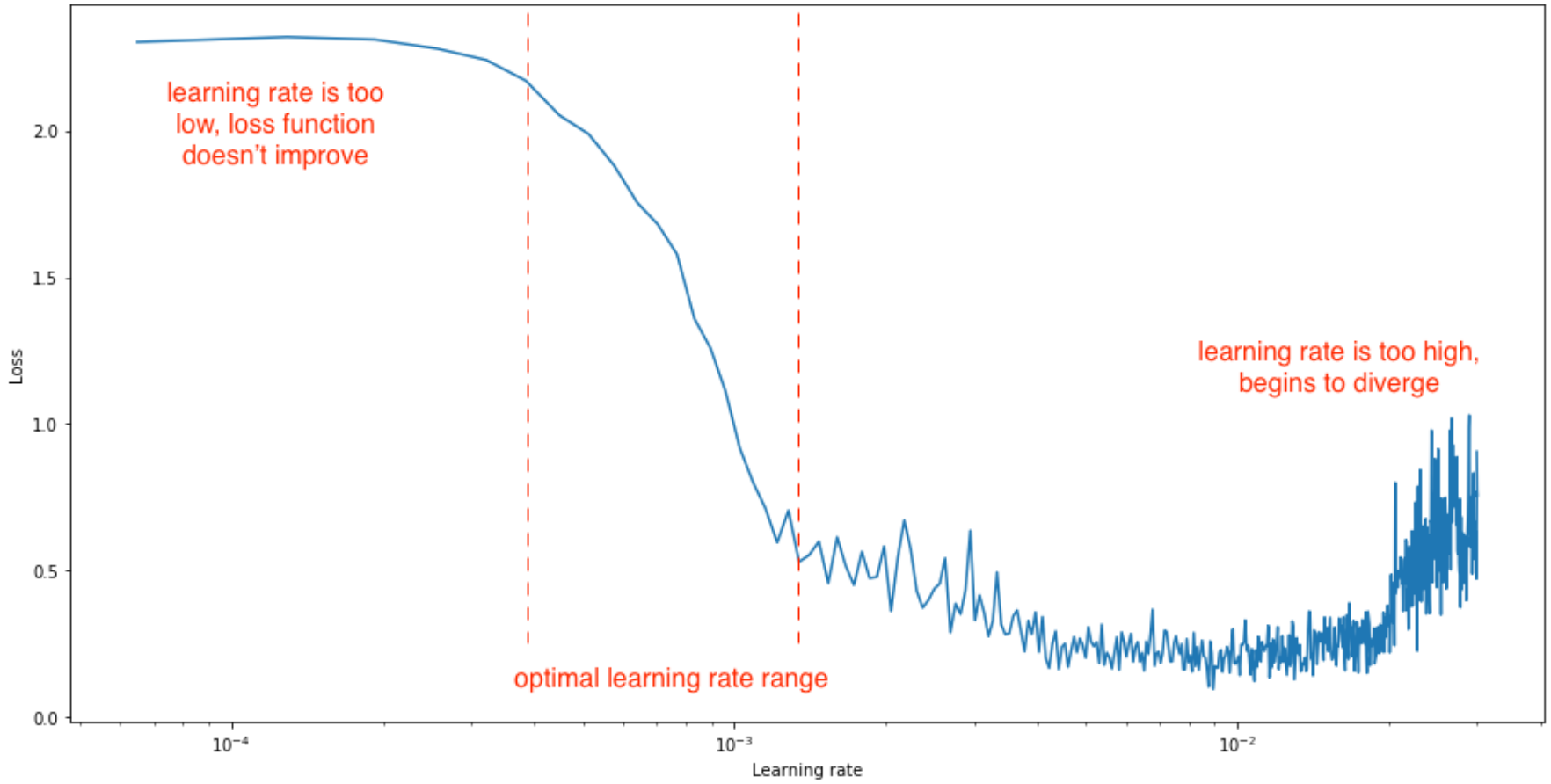
Epochs and learning rate

Often conducting to exploding gradient problem => Loss = NaN





Importance of learning rate tuning

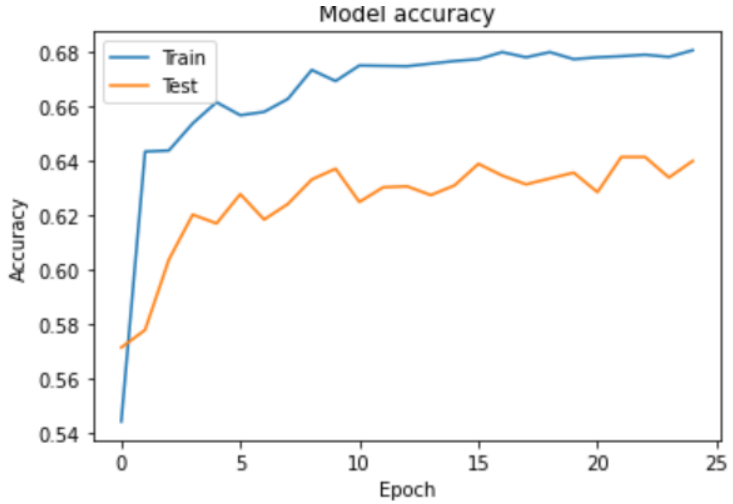


Advanced methods would modify the LR during learning in an adaptive manner…

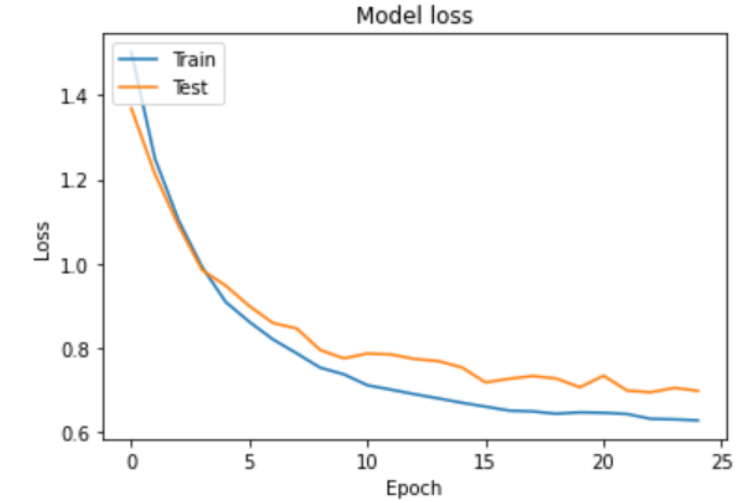
Plotting training evolution

history = model.fit(x_train, y_train, epochs=25, validation_data=(x_test, y_test))

```
# Plot training & validation accuracy values
plt.plot(history.history['categorical_accuracy'])
plt.plot(history.history['val_categorical_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```




```
# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



Labs sessions

- Session 1: installation and first execution of CNN on MCU
  - Validation on MNIST dataset
  - Learn a model and optimize it
- Session 2: optimization of CNN on MCU v1 MNIST and HAR
  - Propose a model, learn and test
  - Application on MNIST dataset
- Break
- **Session 3: programmation of the MCU with MicroAI**
  - **Propose a model, learn and test**
  - **Application on HAR dataset**
- **Session 4: Apply to real data**
  - **Get data either from accelerometer**
  - **Constitute a specific test set**
  - **Test onto the MCU**