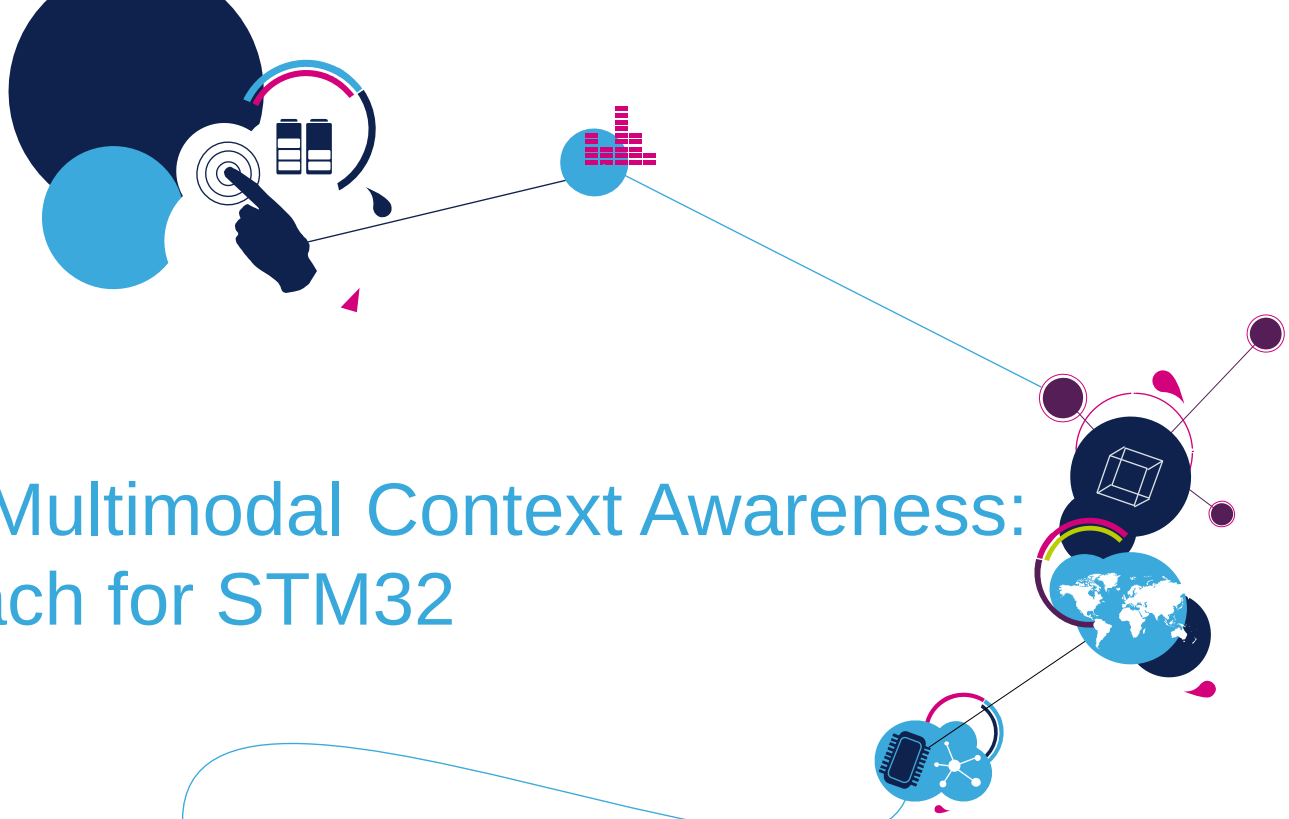


DNN-based Multimodal Context Awareness: a SW approach for STM32

Pierre Demaj
Laurent Folliot

Nov 2017

life.augmented

Introduction
Definitions

- “Context Awareness”: either scene recognition or activity recognition
- Example of Activity corpus:
  - Internally named “Activity”: Standstill / Walk / Run / Bike / Vehicle
- Examples of Scene corpus:
  - Internally named “Transports”: Motorbike / Vehicle / Bus / Rail (Tram + Subway + Train) / Plane
  - Internally named “Places”: Office / Home / Shop / Street / Nature
- The approach is said ‘Multimodal’ because potentially exploiting various families of sensors:
  - MEMS inertial sensors: accelerometer, gyroscope, magnetometer
  - Environment sensors: barometer, lightness, proximity
  - Audio: microphone

Introduction

Requirements and assumptions

- Database creation for model learning:
  - Target model should be sensor manufacturer agnostic:
    - List of smartphones used: SSG S8/S7/S6/S5/S4, LG G5, HTC M8, Nexus 6P and Google Pixel.
- An important requirement:
  - To recognize the context independently from the device position
    - no usage of x/y/z raw data for instance, to avoid overfitting on device positioning.
- Available inputs on the subject:
  - a thesis between ST/Grenoble LIG: “Reconnaissance de scenes multi-modale embarquee” – David Blachon 2016
  - Exploit and challenge outcome of the thesis

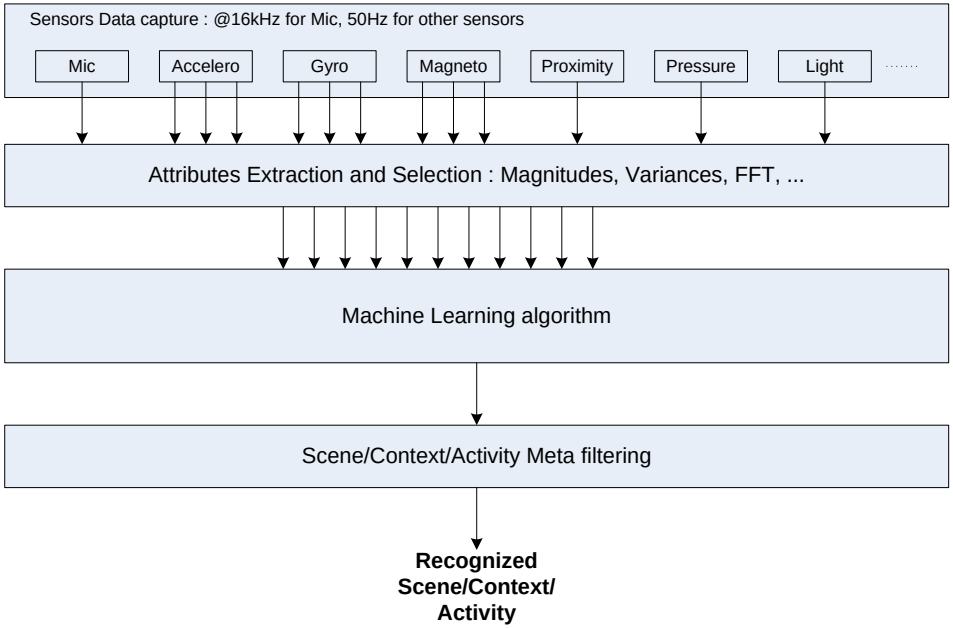
Introduction
Possible applications

- Smart devices improved capabilities:
  - Activity tracking:
    - Daily statistics on scenes/activities/most frequented places: ‘Walk’, ‘Run’, ‘Bike’, ‘Bus’, ‘Shop’…
    - Automatically starts measurements (speed, distance,…) when ‘Run’ is detected for example
  - Automatic smart settings:
    - Adjust phone sound profile depending on: ‘Flight’ (flight mode), ‘Mall’ (raise volume), ‘Meeting’ (mute + vibrator),’Vehicle’(Bluetooth)…
  - Awareness of location can help optimizing device battery life
    - GPS: can be switched-off when ‘Indoor’
    - WIFI: adapt scan settings upon coverage (configurable) conditions: ‘Nature’, ‘Vehicle’, ‘Home’, ‘Street’…

MCA processing chain

Overview

- Successive steps from sensors to scene detection:



- Attributes = Features = Descriptors:
  - how context is represented through various sensors/information measures and mathematical transformations
- Mathematical transformations on sensors values:
  - Temporal
    - Short-Term Energy, Amplitude…
  - Spectral
    - FFT, bandwidth estimators,…
  - Statistical
    - Any statistical moment in time and frequency

- Many attributes are generated and evaluated, but only a selection of them (depending on the Corpus) is used for the classification.

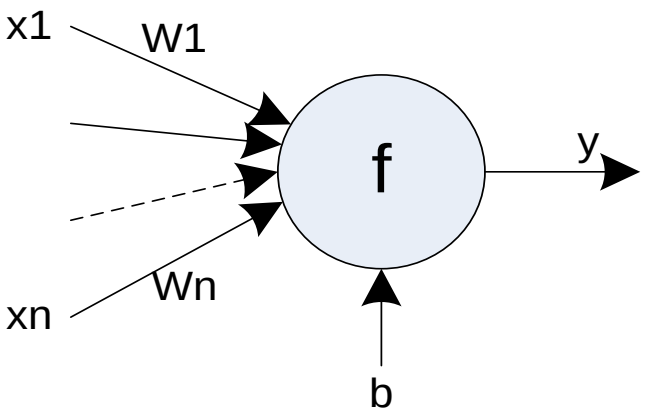
- In general, attributes extraction cannot be a-priori neglected in the total footprints of the solution: in particular when at high sampling data rate

MCA processing chain
Machine Learning Algorithms

- Machine Learning:
  - Problem of identifying to which set of categories/class/labels a new observation belongs to
  - Takes into account past experience to improve prediction reliability
- We have selected a supervised approach:
  - Supervised: category membership is known
- According to thesis outcome, we decided to investigate 2 promising approaches:
  - Decision trees and derivatives (DT/RF), recommended by the thesis
  - Deep Neural Networks (DNN), not enough considered in the thesis
- Although interesting results were obtained with DT family, from now on, this presentation will only focus on DNN

MCA processing chain
Neural Networks elements

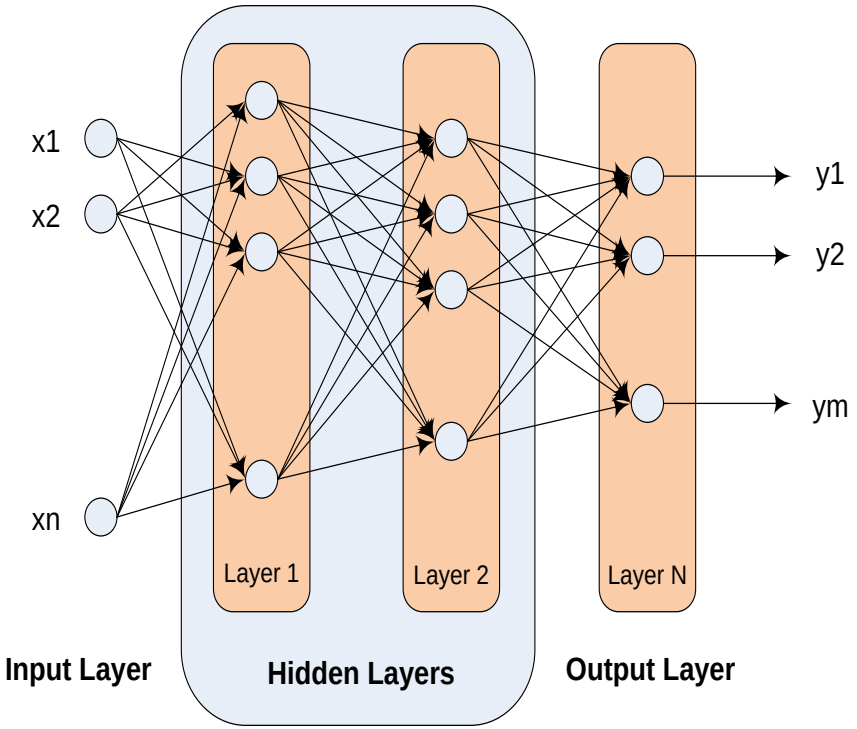
- Neural Network History
  - Perceptrons (Artificial Neuron) : 1960's
  - Multi-Layer Perceptrons (MLP) : 1980's
  - Deep Neural Networks (DNN) : 2010's
    - Fully connected DNN (FC-DNN) > 3 layers
    - Convolutional DNN (CNN)
    - Recurrent NN (RNN)
    - + others…
- Perceptron
  - Single Artificial Neural Network
  - $y = f(\sum_{i=1}^{n} w_i x_i - b)$
  - "f" is the sign function here
  - Training with Stochastic Gradient Descent

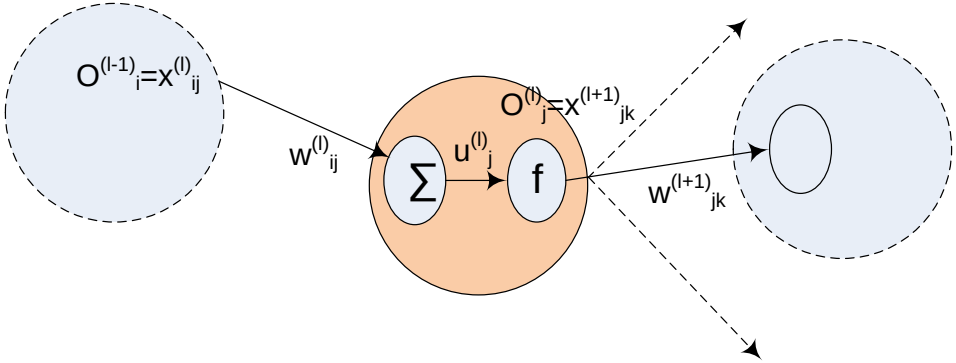


MCA processing chain

MLP principles (1)

- MLP: fully connected Feed-Forward





Input from neuron i (Layer l) to neuron j:

$$x^{(l)}_{ij} = o^{(l-1)}_{i}$$

Activation of neuron i (Layer l):

$$u^{(l)}_{j} = \sum_{i=1}^{n} w^{(l)}_{ij} x^{(l)}_{ij}$$

Output of neuron j (Layer l):

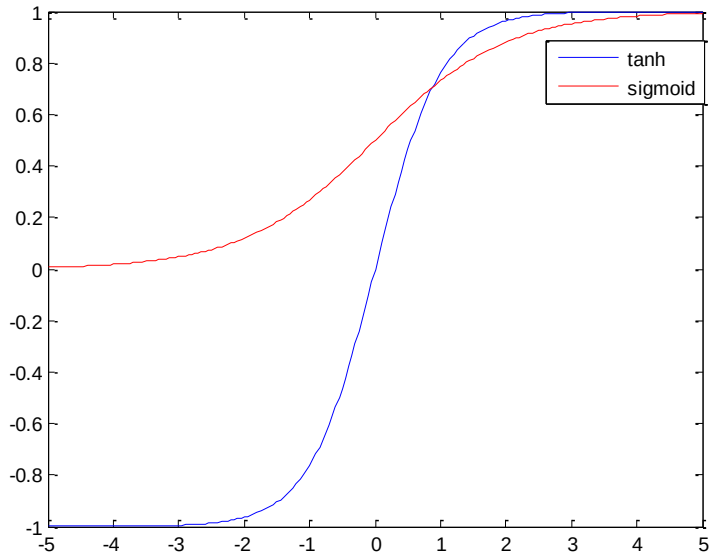
$$o^{(l)}_{j} = f(u^{(l)}_{j})$$

- Training leads to find the best $w^{(l)}_{ij}$ values to minimize error $E = \sum_j \sum_k (y_{jk} - t_{jk})^2$ and iteratively adapts weights.

MCA processing chain

MLP principles (2)

- Activation functions f:
  - Sigmoid, tanh, …



- Back Propagation algorithm used for training :
  - Feed forward to compute activations and outputs
  - Back Propagation to compute partial derivatives
    - Output layer L : $\frac{dE}{du^{(L)}_j} = (y_j - t_j) f'(u^{(L)}_j)$
    - Hidden layer l : $\frac{dE}{du^{(l)}_i} = \sum_j \frac{dE}{du^{(l+1)}_j} w^{(l+1)}_{ij} f'(u^{(l)}_i)$

      With : $\frac{dE}{dw^{(l)}_{ij}} = \frac{dE}{du^{(l)}_j} o^{(l-1)}_i$
    - Update : $w^{(l)}_{ij} \leftarrow w^{(l)}_{ij} - \lambda \frac{dE}{dw^{(l)}_{ij}}$

- A lot of parameters to tune :
  - NN architecture (nb layers, nb neurons, activation function choice, weights initialization,…)
  - Learning rate adaptation

- Limitations : increasing number of layers leads to numerical issues, saturated neurons and unstable model hard to train.

MCA processing chain
FC-DNN principles (1)

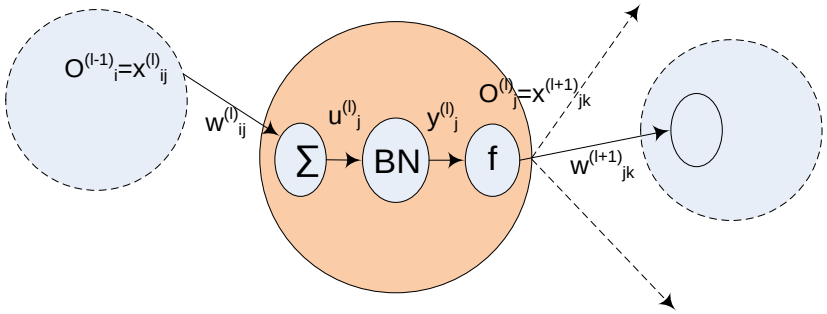
- Fully connected DNN re-use MLP training principles + some improvements (2010') to support deep architecture:
  - Deep (many layers, fewer neurons per layer => composition of simple functions) vs Shallow (few layers, many neurons => combination of simple functions)
  - Transfer function : LReLu + Normalization for hidden layers (also acts as a regularizer), SoftMax for output layer
  - Training : Backpropagation algorithm + mini-batch gradient descent
  - Learning Rate annealing (faster convergence)
  - RMS-Prop : adapts the learning rate to every parameters $w^{(l)}_{ij}$

- Transfer functions:
  - Leaky Rectified Linear Unit for hidden layers : avoid dead neurons, no saturation
  - Softmax for output layer : forces the output to represent a probability distribution across output neurons (mutually exclusive classes) :
    - $f(x_k) = \frac{e^{x_k}}{\sum_{j=1}^{L} e^{x_j}}$ and $f'(x_k) = f(x_k)(1 - f(x_k))$
  - Cross-Entropy cost function used (negative log probability of the correct answer):
    - $E = -\sum_j t_j log(\frac{e^{u_j}}{\sum_{i=1}^{L} e^{u_i}})$
    - $with\ simple\ gradient : \frac{dE(w)}{du_j} = \sum_i \frac{dE(w)}{dy_i}\frac{dy_i}{du_j} = y_j - t_j$

MCA processing chain
FC-DNN principles (2)

- Mini-batch gradient descent:
  - Faster than global batch, and less sensitive to local minimum, lower variance of the parameter updates compared to stochastic gradient descent, …

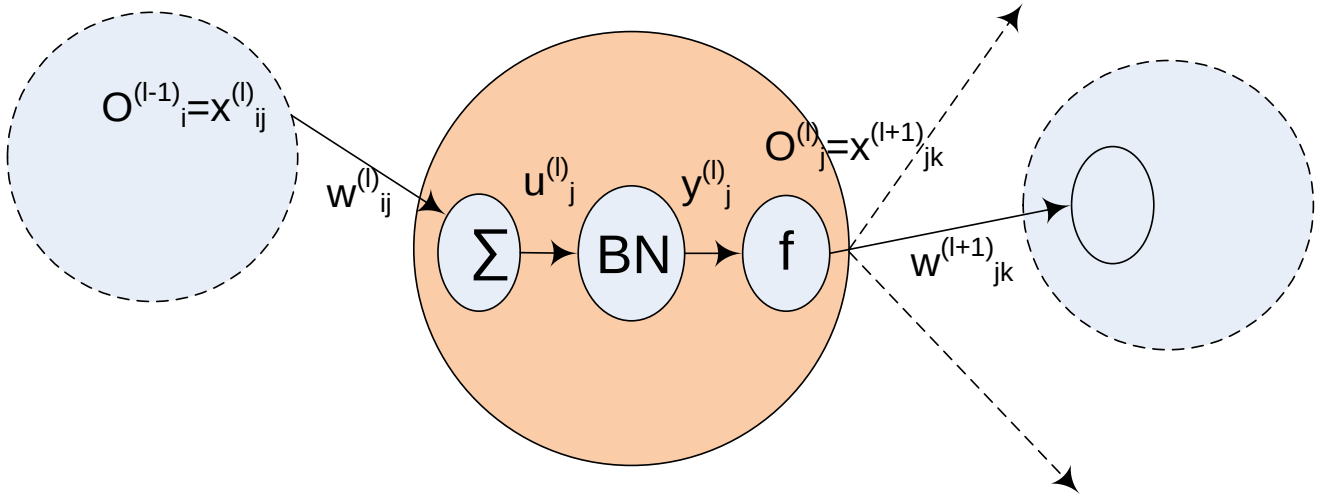
- Mini-Batch Normalization layers (2015') required with LReLu usage. Goal here is to have a zero mean, unit variance as input of all neurons of the mini-batches:
  - $\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} u_i$: *Mini-batch mean*
  - $\sigma^2{}_B \leftarrow \frac{1}{m}\sum_{i=1}^{m}(u_i - \mu_B)^2$: *Mini-batch variance*
  - $\hat{u}_i \leftarrow \frac{u_i - \mu_B}{\sqrt{\sigma^2{}_B + \epsilon}}$: *Normalize*
  - $y_i \leftarrow \gamma\,\hat{u}_i + \beta \equiv BN_{\gamma,\beta}(u_i)$: *Scale and shift*
  - *γ and β are trained during back-propagation thanks to chain rule and computational graphs technics.*



- RMSprop : Adapts the learning rate to the parameters for each neuron:
  - $w^t{}_i \leftarrow w^{t-1}{}_i - \frac{\lambda}{\sqrt{G^{t-1}{}_i + \epsilon}} \Delta_w E(w^{t-1}{}_i)$
  - $G^t{}_i \leftarrow \gamma\, G^{t-1}{}_i + (1-\gamma)\,(\Delta_w E(w^{t-1}{}_i))^2$

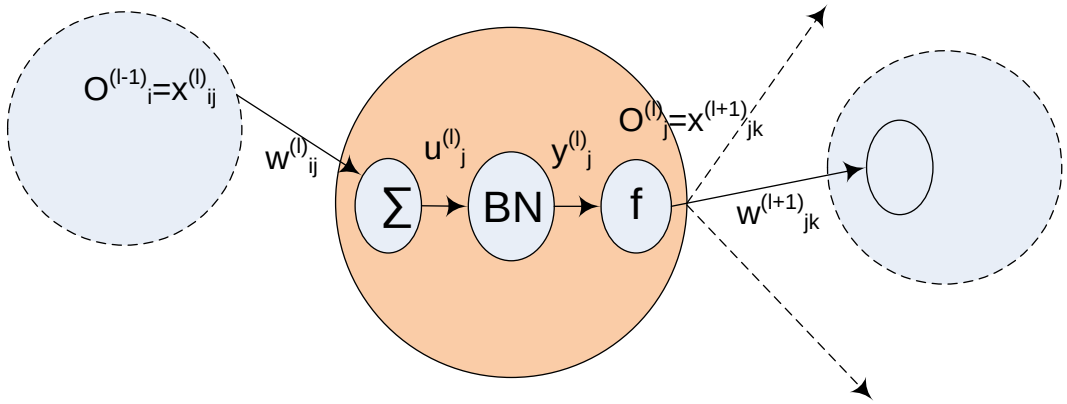
MCA processing chain
FC-DNN decoding

- DNN decoding summary to be done from input layer to output layer :
  - Gets input or previous layer (l) output data: $x^{(l)}_{ij} = o^{(l-1)}_{i}$
  - Calculates activation of each neuron of layer l: $u^{(l)}_{j} = \sum_{i=1}^{n} w^{(l)}_{ij} x^{(l)}_{ij}$
  - Centers and normalizes each neuron activation : $\hat{u}^{(l)}_{j} = \frac{u^{(l)}_{j} - \mu^{(l)}_{j}}{\sqrt{\sigma^{(l)}_{j}{}^{2} + \epsilon}}$
  - Normalized output adapted to different distributions : $y^{(l)}_{j} = \gamma^{(l)}_{j} \hat{u}^{(l)}_{j} + \beta^{(l)}_{j} \equiv BN_{\gamma,\beta}(\hat{u}^{(l)}_{j})$
  - Calculates output of each neuron of layer l : $o^{(l)}_{j} = f(y^{(l)}_{j})$
    - With f() is Leaky ReLu transfer function for hidden layers, and SoftMax for output layer



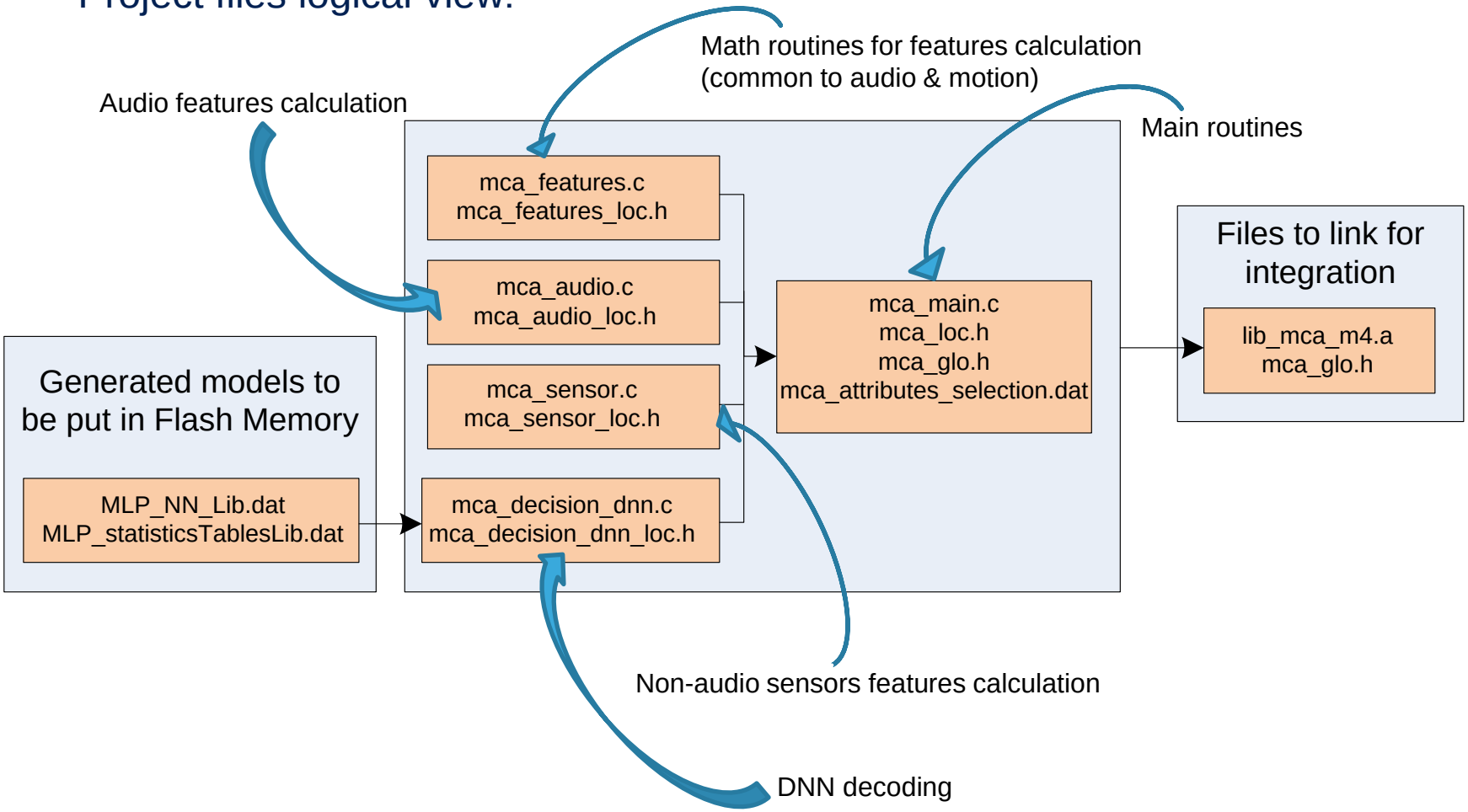
MCA processing chain
FC-DNN Continuous Learning

- DNN batch continuous learning summary:
  - Decodes first a batch of input vectors, and calculates output error gradient $d_{out}^{(L)} = \frac{dE}{do^{(L)}_j}$ of each vector – some intermediates calculations are also needed during the back-propagation.
  - Back propagation of the error from output to input layer:
    - $d_{out}^{(l-1)} = \frac{dE}{do^{(l-1)}_i} = \sum_j \frac{dE}{do^{(l)}_j} \frac{do^{(l)}_j}{dy^{(l)}_j} \frac{dy^{(l)}_j}{du^{(l)}_j} \frac{du^{(l)}_j}{do^{(l-1)}_i} = \sum_j \frac{dE}{du^{(l)}_j} \frac{du^{(l)}_j}{do^{(l-1)}_i}$ since $\frac{dE}{du^{(l)}_j} = \frac{dE}{do^{(l)}_j} \frac{do^{(l)}_j}{dy^{(l)}_j} \frac{dy^{(l)}_j}{du^{(l)}_j}$
    - When batch processing is completed, adapts Weights, BNL scale (γ) and BNL shift (β) values with a gradient descent:
      - $w^{(l)}_{ij} \leftarrow w^{(l)}_{ij} - \lambda \sum_{batch} \frac{dE}{dw^{(l)}_{ij}} = w^{(l)}_{ij} - \lambda \sum_{batch} \frac{dE}{du^{(l)}_j} \frac{du^{(l)}_j}{dw^{(l)}_{ij}} = w^{(l)}_{ij} - \lambda \sum_{batch} \frac{dE}{du^{(l)}_j} o^{(l-1)}_i$
      - $\gamma^{(l)}_i \leftarrow \gamma^{(l)}_i - \lambda \sum_{batch} \frac{dE}{d\gamma^{(l)}_i}$
      - $\beta^{(l)}_i \leftarrow \beta^{(l)}_i - \lambda \sum_{batch} \frac{dE}{d\beta^{(l)}_i}$



MCA on STM32
EWARM Library development

- Project files logical view:



MCA performances
Footprints

- Footprints primarily depend on:
  - Scene Corpus
  - Target performances
  - Topology of the selected machine learning algorithm
  - Selected attributes
- Footprints also depend on the task scheduling:
  - How frequently do we need to estimates the selected attributes?
- Footprints eventually depend on final application:
  - How frequently do we need to update the scene/activity detection?
  - What are the requirements in terms of
    - Latency?
    - Scene stability?
    - Transition detection and handling?

MCA performances

Performances

- Performances depend on a lot of parameters:
  - Training database size and balance
  - Validation database size and balance
  - Scene Corpus
  - Chosen DNN architecture/Memory size
    - Initialization and Convergence parameters
    - Number of layers
    - Number of neurons
    - Sensitivity to over fit
  - Attributes relevance and selection
- Performances must be consolidated after integration and tests on the field.

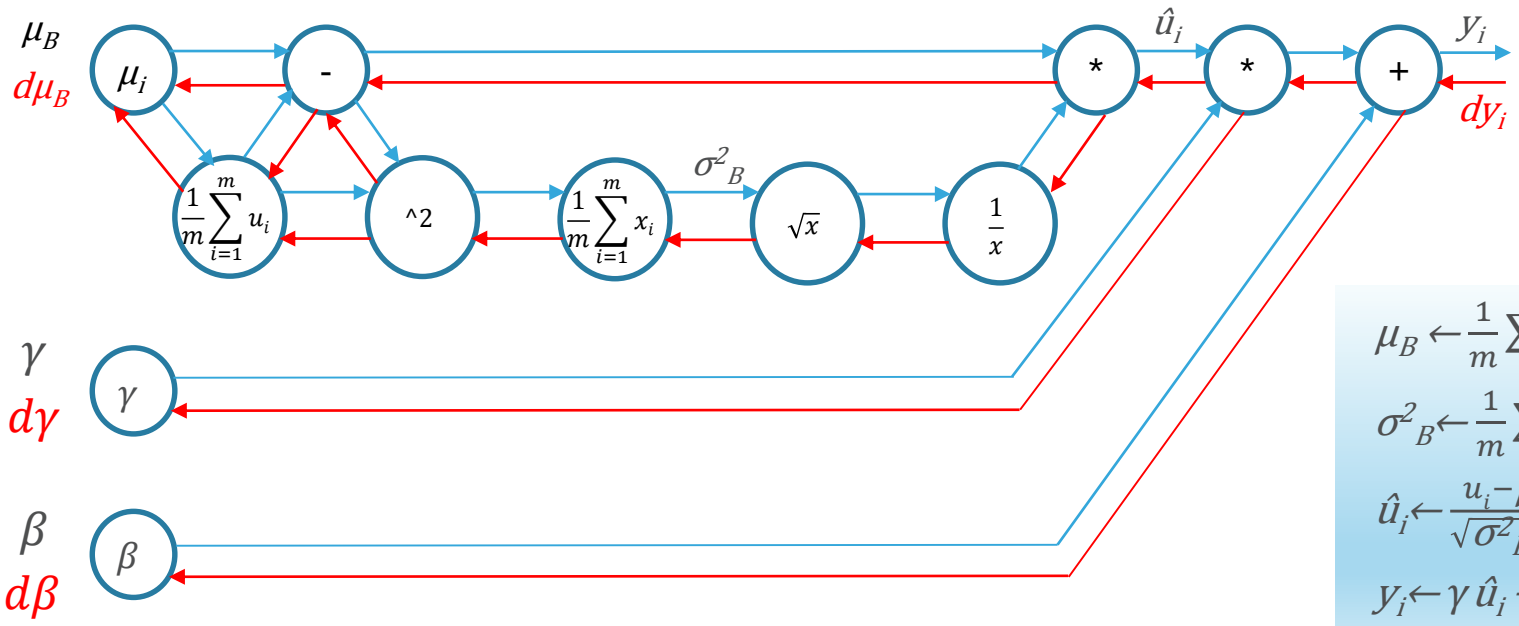
Thank you!

BACKUP Slides

MCA processing chain

FC-DNN continuous learning (2)

- Adaptation with BNL adds some complexity to propagate error gradient. Computational graph and chain rule are applied here to calculate $\frac{dE}{d\gamma^{(l)}_i}$, $\frac{dE}{d\beta^{(l)}_i}$ and $\frac{dE}{du^{(l)}_j}$.



$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} u_i$$

$$\sigma^2{}_B \leftarrow \frac{1}{m}\sum_{i=1}^{m}(u_i - \mu_B)^2$$

$$\hat{u}_i \leftarrow \frac{u_i - \mu_B}{\sqrt{\sigma^2{}_B + \epsilon}}$$

$$y_i \leftarrow \gamma\,\hat{u}_i + \beta \equiv BN_{\gamma,\beta}(u_i)$$