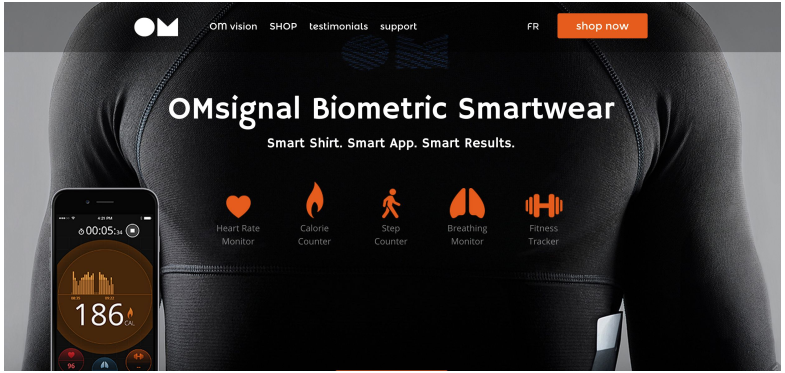
Institut
québécois
d'intelligence
artificielle

Mila

OMsignal Project
Block 2

Arsene Fansi-Tchango, PhD
Simon Blackburn, PhD

Company



1,8 billion+
heartbeats recorded

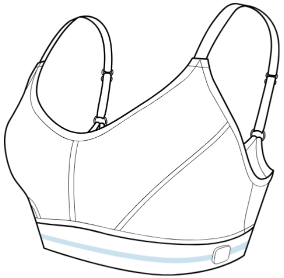
500 million+
breaths recorded

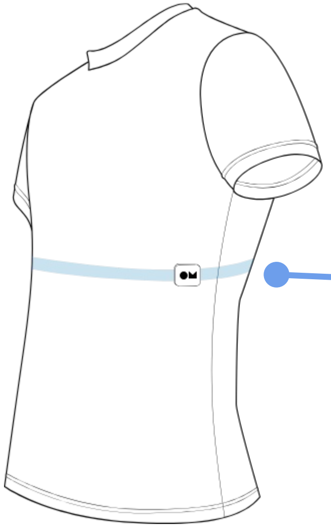
330 thousand+
hours of data collected

Make personal health and wellness central to our daily lives, through the world’s most advanced biosensing apparel platform.

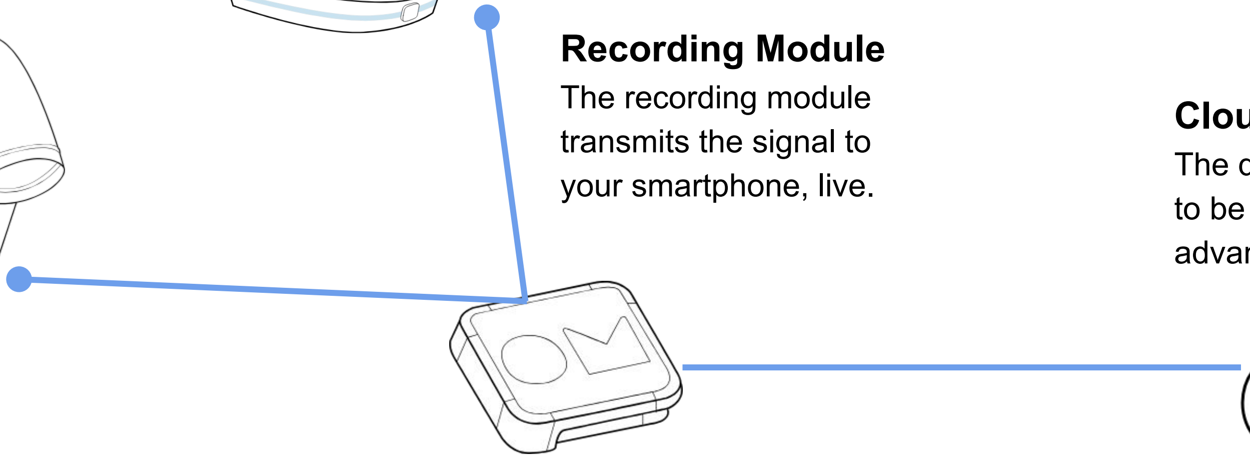
Technology

Garments
The apparel picks up the body's signals using strategically placed ECG, Respiration, and Physical Activity sensors.

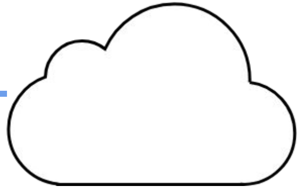




Recording Module
The recording module transmits the signal to your smartphone, live.



Cloud + AI
The data is sent to the Cloud to be further analysed using advanced algorithms and AI.

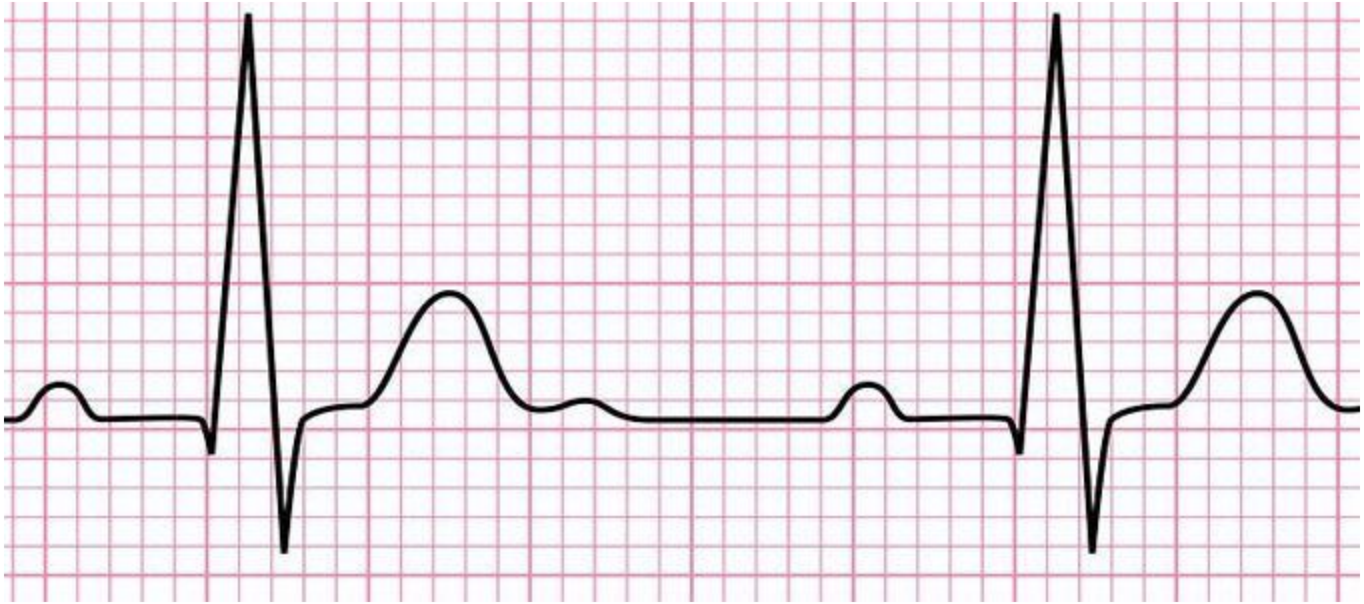


Operational Challenges

- **Easy** to collect unlabeled data
    - Huge amount of data captured under different conditions
        - running / walking / sitting / sleeping, etc...
        - different levels of signal to noise ratio

- **Hard** to label this data for supervised learning
    - Experts (e.g., medical doctors) are expensive
    - Time demanding
        - E.g., walk through all the samples of a signal

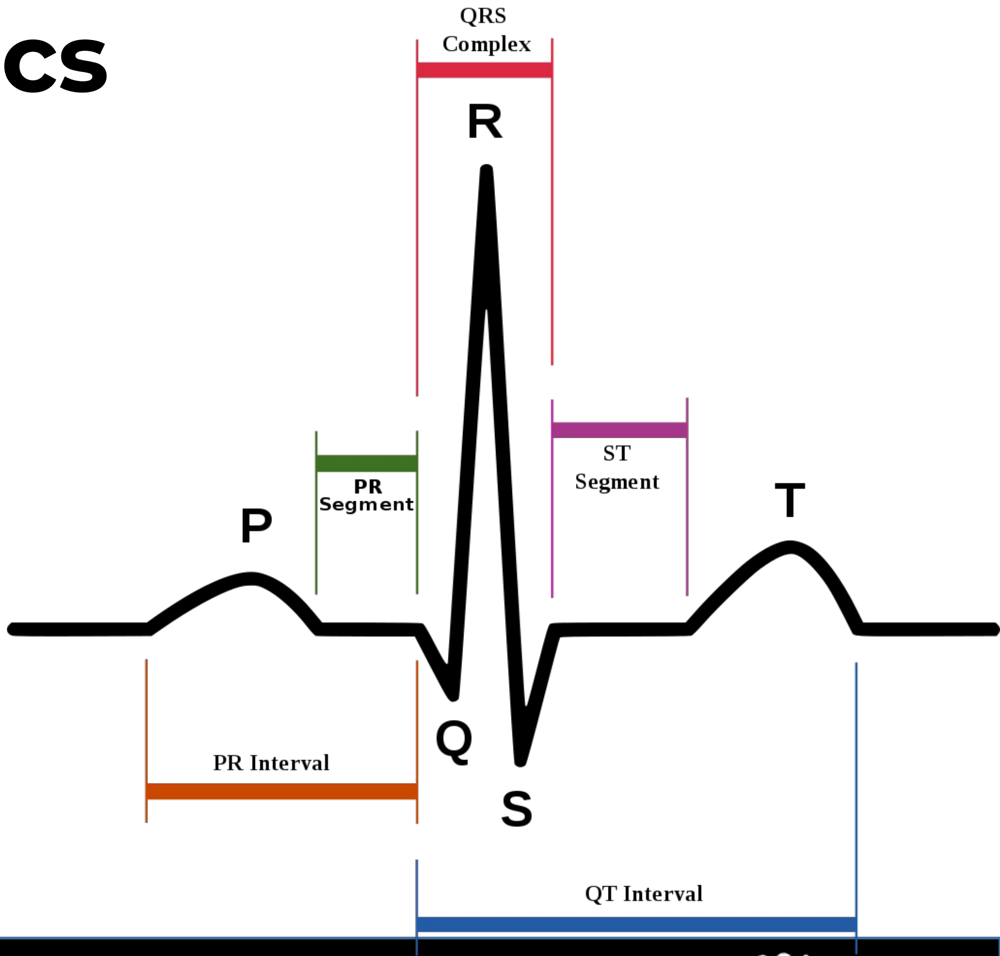
ECG Example (1 lead)



From http://www.onlinebiologynotes.com/electrocardiogram-ecg-working-principle-normal-ecg-wave-application-of-ecg/

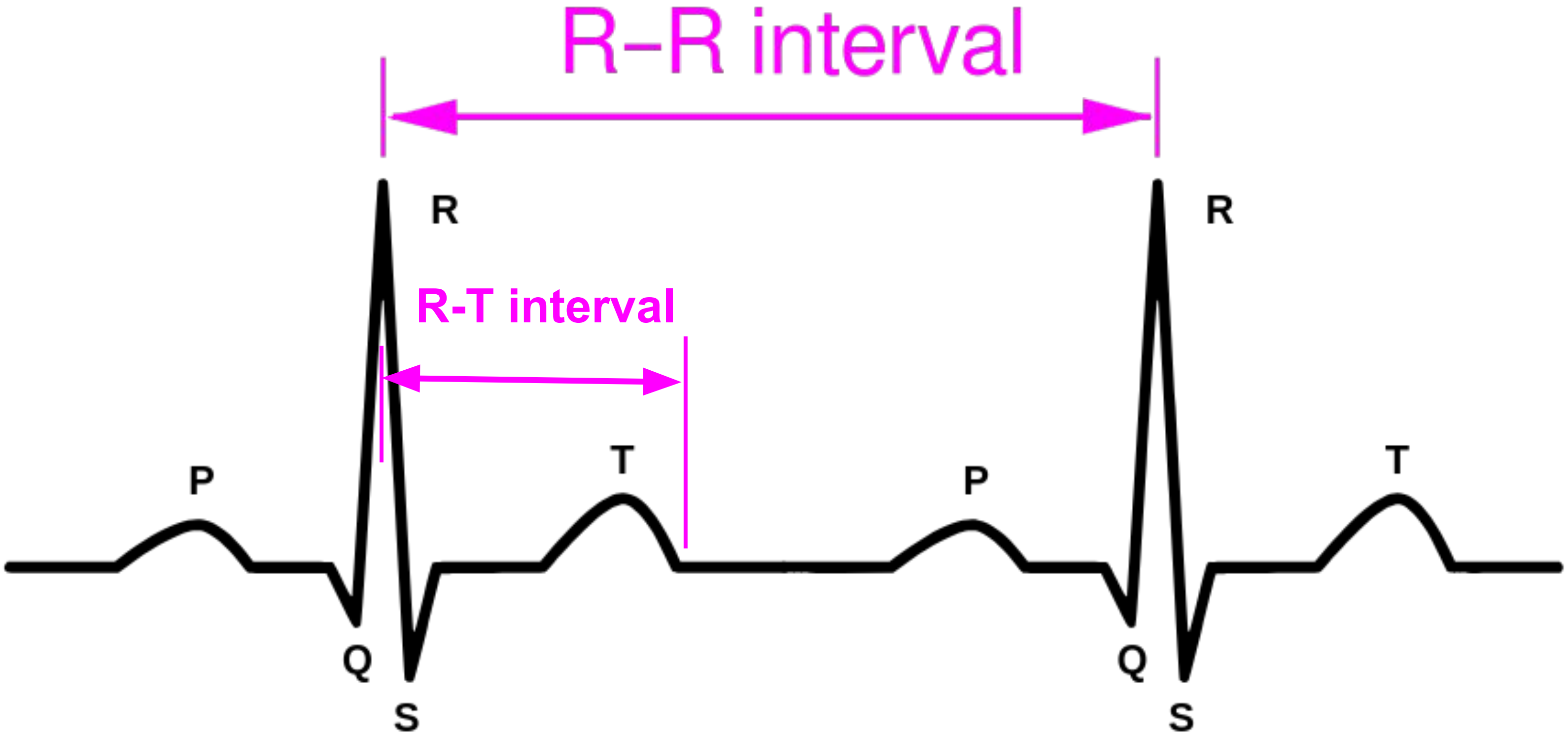
ECG Characteristics

- **Fiducial points**: P, Q, R, S, T
- **P-Wave**:
  - Indicates atrial depolarization (systole)
- **QRS wave:**
  - Represents the ventricular depolarization (systole)
- **T- wave:**
  - Indicates ventricular repolarization (diastole)
- **P-R interval:**
  - Represents the time required for an impulse to travel through the atria
- **S-T segment:**
  - Represents the time when ventricular fibres are fully depolarized



From https://en.wikipedia.org/wiki/Electrocardiography

ECG Characteristics



OMsignal Project

- **Overall goal**: develop an **unsupervised/semi-supervised** representation learning approach that produces representations useful for tasks that have little labeled data:
    - Identification of the user
    - Fiducial point distributional information
        - Mean of the PR-Interval (real value)
        - Mean of the RT-Interval (real value)
        - Standard deviation of the RR-Interval (real value)

Data

OMsignal MyHeart project:

- **Private** data
- **32** Participants
- ECG signals are divided into windows of 30 seconds each at 125 Hz (**3750** samples per window)
- Labeled data:
  - **15** windows for each participant are labeled
  - Among them, **5** windows are used as test data
  - The remaining **10** are provided as train/validation data

- Unlabeled data:
  - **657233** windows

Data Formats

- The data is provided as **numpy.memmap files:**
  - MILA_TrainLabeledData.dat: **160 x 3754**
  - MILA_ValidationLabeledData.dat: **160 x 3754**
  - MILA_UnlabeledData.dat: **657233 x 3750**
  - MILA_TestLabeledData.dat: **160 x 3754 (Blind)**

- The samples in each of train, validation and test originate from 3 different days.

Data Formats

- **Labels:** 4 last columns of labeled datasets
  - **PR_Mean**: 3751th column
  - **RT_Mean**: 3752th column
  - **RR_StdDev**: 3753th column
  - **UserID**: 3754th column

- Code to read/write numpy.memmap files into/from numpy.array **is provided**

OMsignal - Block 1 goal

- Build a simple supervised baseline, using only the limited amount of labeled data

OMsignal - Block 2 goal

- Find a way to leverage the unlabeled data and build a model for all four tasks

Block 2 - Leveraging Unlabeled Data

Instructions / Expected timeline

	2019/02/11 week	2019/02/18 week	2019/02/25 week	2019/03/11 week
Tasks / Homework	• Review code and reports from block 1 • Understand how to use TensorboardX • Code data loader for unlabeled data	• Identify and start implementing multi-task solutions for incorporating unlabeled data into the training process	• Continue implementing multi-task solution, leveraging the unlabeled data • Hyper parameter fine tuning	• Write a short report summarizing the work, and results • (Peer-) Review of other teams' code
Objectives / Deliverables	• Have a clear understanding of the data • Data loader for unlabeled data	• Understanding of solutions for incorporating unlabeled data into the training process	• Multi-task model with unlabeled data (beginning of week after spring break)	• Produce documented code and report summarizing the experimental work • Provide model for blind test set evaluation • Complete the peer code review

Deadlines

- Each team needs to provide the deliverable (report + code + best model) corresponding to a block at the latest on **Friday 11:59pm** of the last week of the block.

- Any block deliverable that is provided past **Friday 11:59pm** of the last week of a block will automatically get 0% for the peer evaluation.

Deadlines

- Any block deliverable that is provided past **Tuesday 11:59pm** following the last week of a block will automatically get 0% for the UdeM evaluation.

- Peer evaluation must be completed by **Monday 11:59pm** following the last week of a block.

Evaluation for Block 2

25% of the final score

- 10% Code review [5% of averaged peer evaluation + 5% UdeM]
- 12% Report evaluation [UdeM]
- 3% Model performance evaluation on blind test set [UdeM]

Code review - Peer evaluation

- **10% of the final score** [5% of average peer evaluation + 5% UdeM]
- Random assignation of code reviews
- The code provided by a team will be evaluated by at **least 2 other teams**

Code quality (peer evaluation + UdeM evaluation)	8
Coherent and modular code/file organization (e.g. data processing, model definition, model training, model inference are in different files/modules; no code duplication)	/1
Code respects the PEP8 standard	/1
Comments are relevant (see article)	/1
Proper management of input arguments in the training script (see argparse, python fire, configparser)	/1
Proper utilization of GitHub (e.g. branching, relevant commits and messages, usage of pull request)	/1
Meaningful variable and function names	/1
Executable scripts with a “main” function (see article)	/1
Reproducible experiments (e.g. seed)	/1

Report Evaluation

Introduction	2
Introduction to the project	/1
Brief introduction to the methods that will be used in the report	/1
Methodology	6
Description of the algorithms and the experiments (including hyperparameter fine tuning (if appropriate), etc.)	/3
Data description and data selection (train/valid/test, number of samples, shape/structure of data points)	/3
Results and discussion	6
Presentation of results (tables, figures, etc.)	/2
Discussion of results	/4
Conclusion	2
Recommendation for next steps	/1
Summary of project state (what was done, what needs to be done)	/1
Quality of the report	2
Report format (title with team member names, clear sections, flow between sections, figures and tables titled, axes titled, etc.)	/1
Report is short and to the point (5-7 pages including references, font size 11)	/1

- **12% of the final score**
- 5-7 pages
    - Including figures, tables, and references
- Single column
- Font size 11
- Use the NeurIPS LaTeX format (if comfortable with LaTex)

Mila

Blind Test Set Evaluation

- **3% of the final score**.
- If the best model provided by a team crashes or provides results that are statistically worse than those of the baseline model provided by the TAs, the team gets 0%.
- Otherwise, if the best model provided by a team is statistically equivalent to the baseline model, the team gets 1%.
- Otherwise, if the best model provided by a team is statistically better than the baseline model:
    - The team gets 3% if the model is the best performing one or is statistically equivalent to the best performing model provided by another team.
    - Otherwise, the team gets 2%.

Code Execution - Blind Test Set Evaluation

- The **test dataset** will be structured as follows:
    - A **numpy.memmap** file containing unlabeled samples of shape **160X3750**.

- You will not have access to the **test set** and we will be executing your code on the **test set** ourselves.

- We will provide explicit instructions and examples for you to enhance an evaluation skeleton script that will be provided to you. You will need to complete this script. We reserve the right to give 0 if we cannot execute your code.

- Tips: **do not shuffle the test set.** Predictions should be made sequentially.

Official Evaluation Metrics

- Classification task
  - Macro Average Recall Score (sklearn.metrics.recall_score)

- Regression tasks
  - Kendall Correlation Score for each task (scipy.stats.kendalltau)

- Overall Score:
  - All individual scores are clipped at zero
  - Geometric mean of the scores of the 4 tasks

- The code of the scoring function will be **provided**

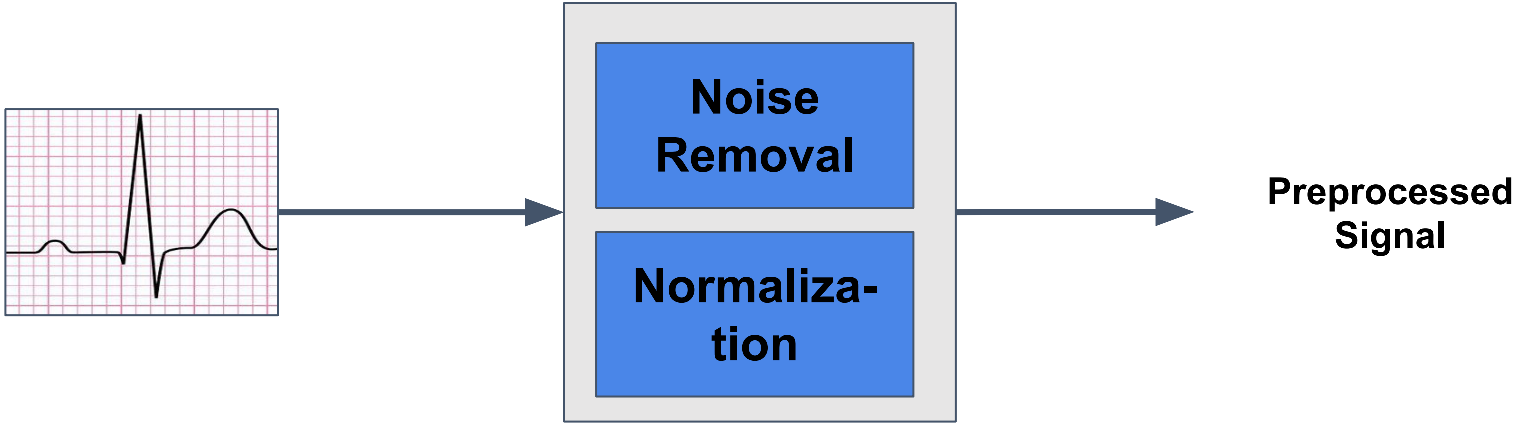
Institut
québécois d'intelligence
artificielle

Mila

OMsignal Project
Data Preprocessing

Data Preprocessing

- The collected ECG signals come with some noise and different levels of amplitude (running vs walking)
- **Needs preprocessing**



Data Preprocessing

- **Noise Removal**
  - Remove baseline wander (low frequency noise) with a moving average

- **Normalization**
  - Normalize the amplitude of the signal within a moving window

- The code for the preprocessing is **provided**.

Mila

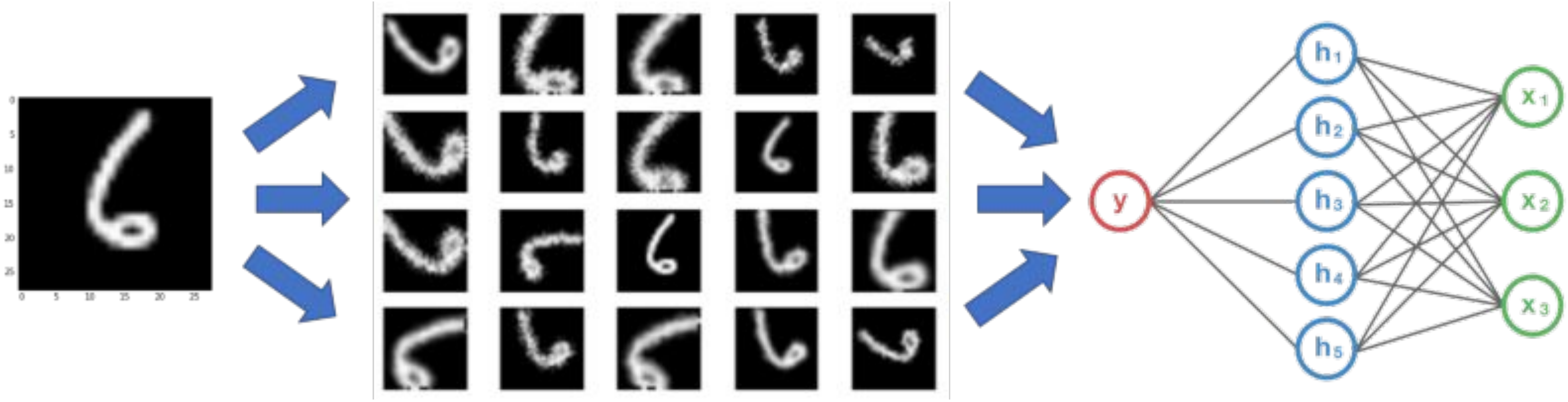
Institut
québécois d'intelligence
artificielle

Mila

OMsignal Project
Data Augmentation

Data Augmentation

- DL (Deep Learning) in general requires large datasets for robustness
- How to get more data?



https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

- **offline** augmentation vs **online** augmentation

Mila

(Possible) Data Augmentation for ECG

- **Signal shift**
  - Shift elements along the temporal axis (assuming periodic boundary conditions)
- **Partial noise addition**
  - Replace at most 2 second length sub-windows with controlled noise (matching the mean and the variance of the underlying samples)
- **Upside-Down inversion**
  - Negate the values of the signal
- Other ideas …

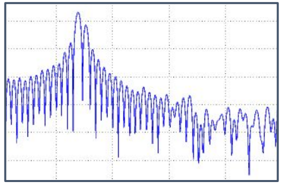
Institut
québécois d’intelligence
artificielle

Mila

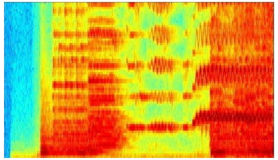
OMsignal Project
Data Transformation

Data Transformations

It can be interesting to supply different representations of the data to a network to improve performance. For ECG signals, one can consider:



- **Fast Fourier Transform**



- **(log) Spectrogram**
- Other ideas ...
- Code for these transformations is **provided**.

Mila

Homework

- Implement your data loader for the labeled/unlabeled datasets
  - The code should provide the ability to perform **online** data augmentation if needed

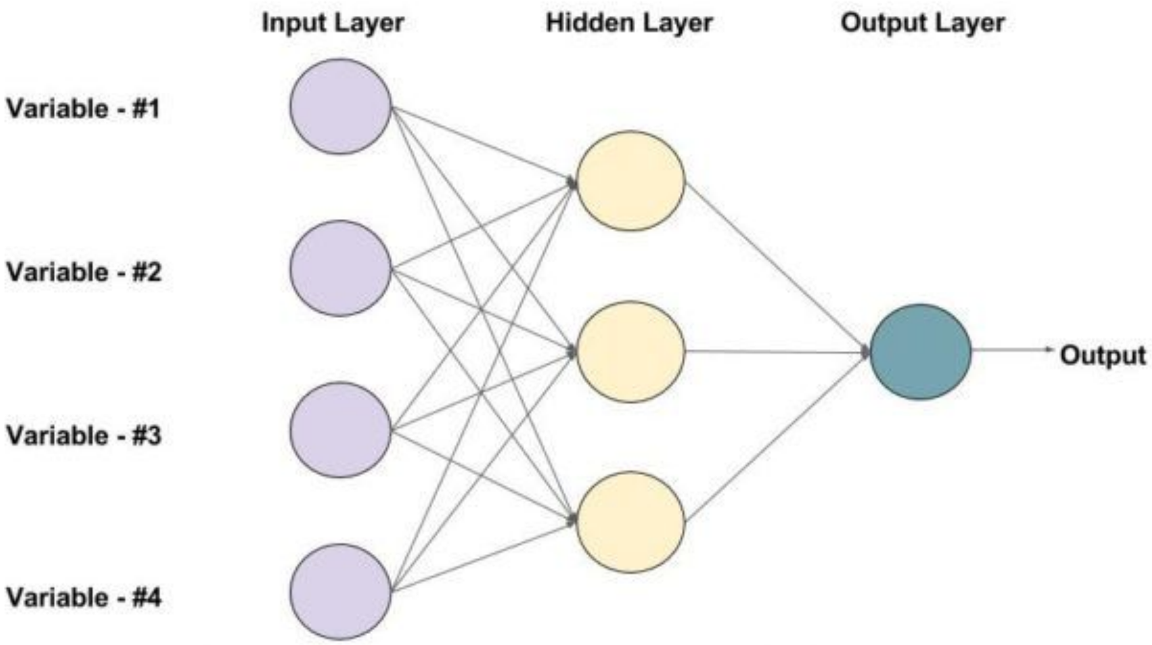
Institut
québécois d'intelligence
artificielle

Mila

OMsignal Project
Models

Basic Models for 1D Signals

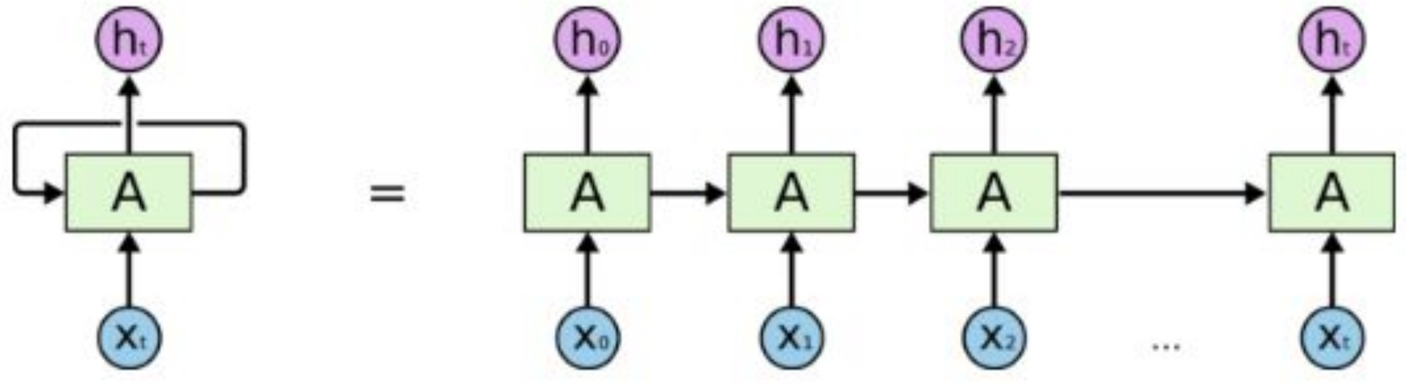
- **MLP**
  - The size of the input is fixed: 3750



An example of a Feed-forward Neural Network with one hidden layer (with 3 neurons)

Basic Models for 1D Signals

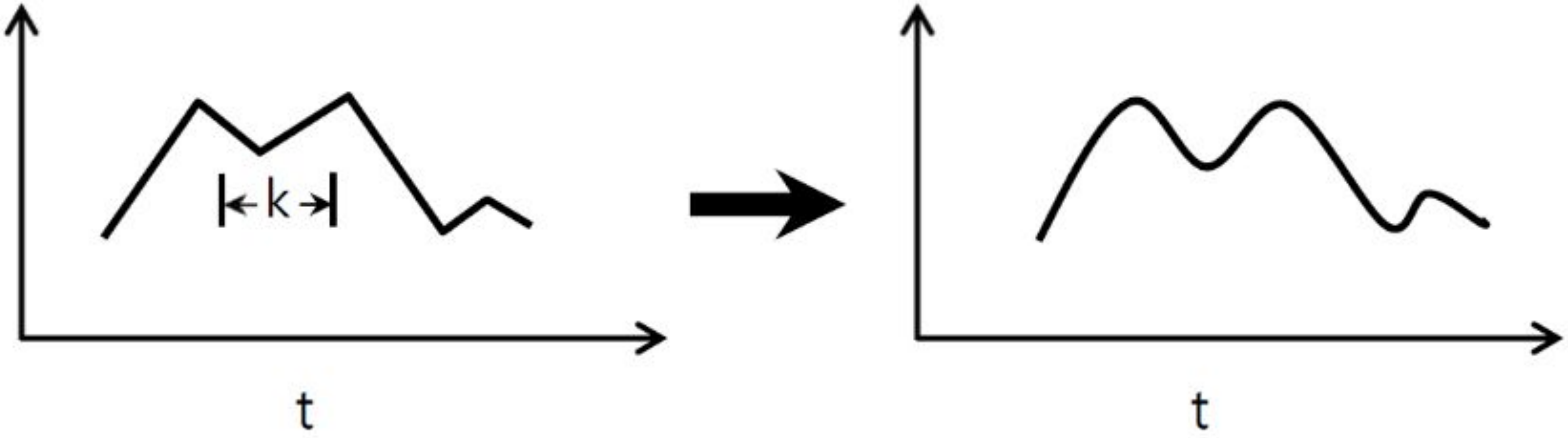
- **Recurrent Neural Networks**
  - Process the signal as a sequence



An unrolled recurrent neural network.

Basic Models for 1D Signals

- **Convolution 1D**
  - Process the signal on a windows basis. Apply the same filter on each window.



Models

- Other types (including combinations of the previously described models) can be considered

- Some transformed signals (e.g. spectrogram) might require other types of models

Institut
québécois d’intelligence
artificielle

Mila

OMsignal Project
Regularization

Regularization

- **Useful** to avoid overfitting on the training data

- **Examples of techniques**:
    - Dropout
    - Batch Normalization
    - Instance Normalization
    - ...

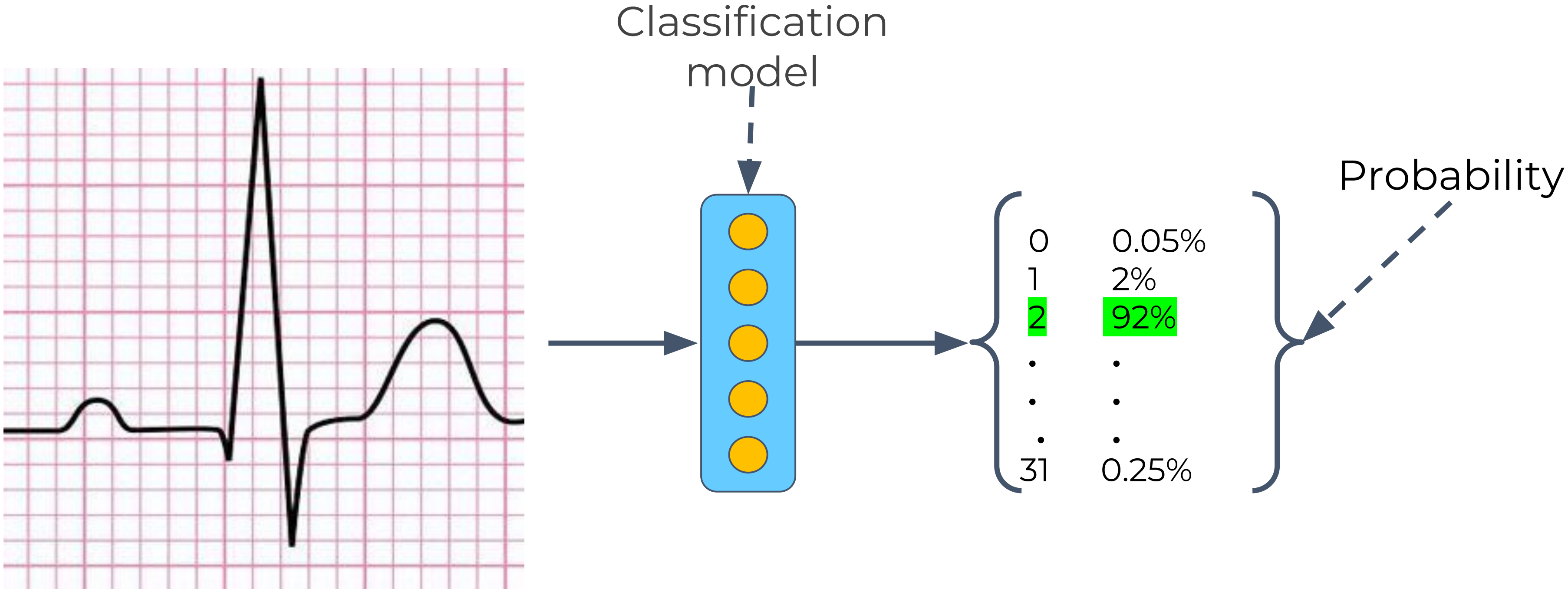
- All are **readily available** in PyTorch.

Institut
québécois d'intelligence
artificielle

Mila

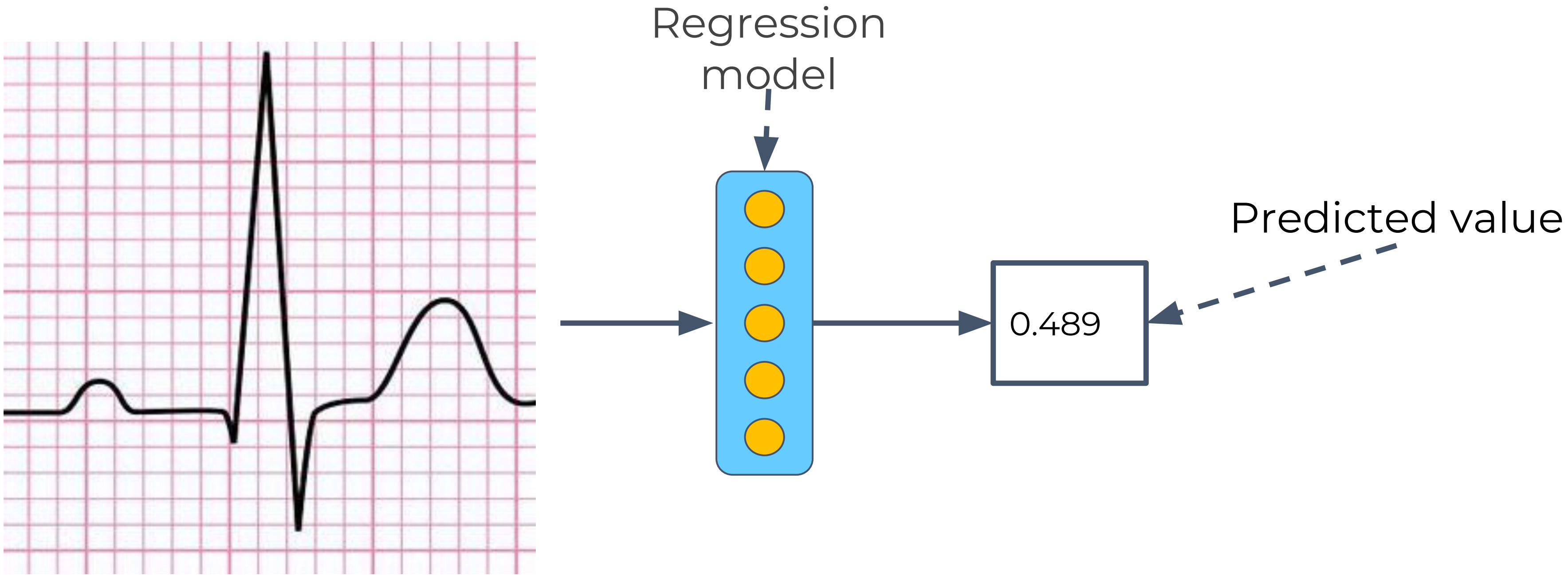
OMsignal Project
Multi-Task Learning

User Identification Task



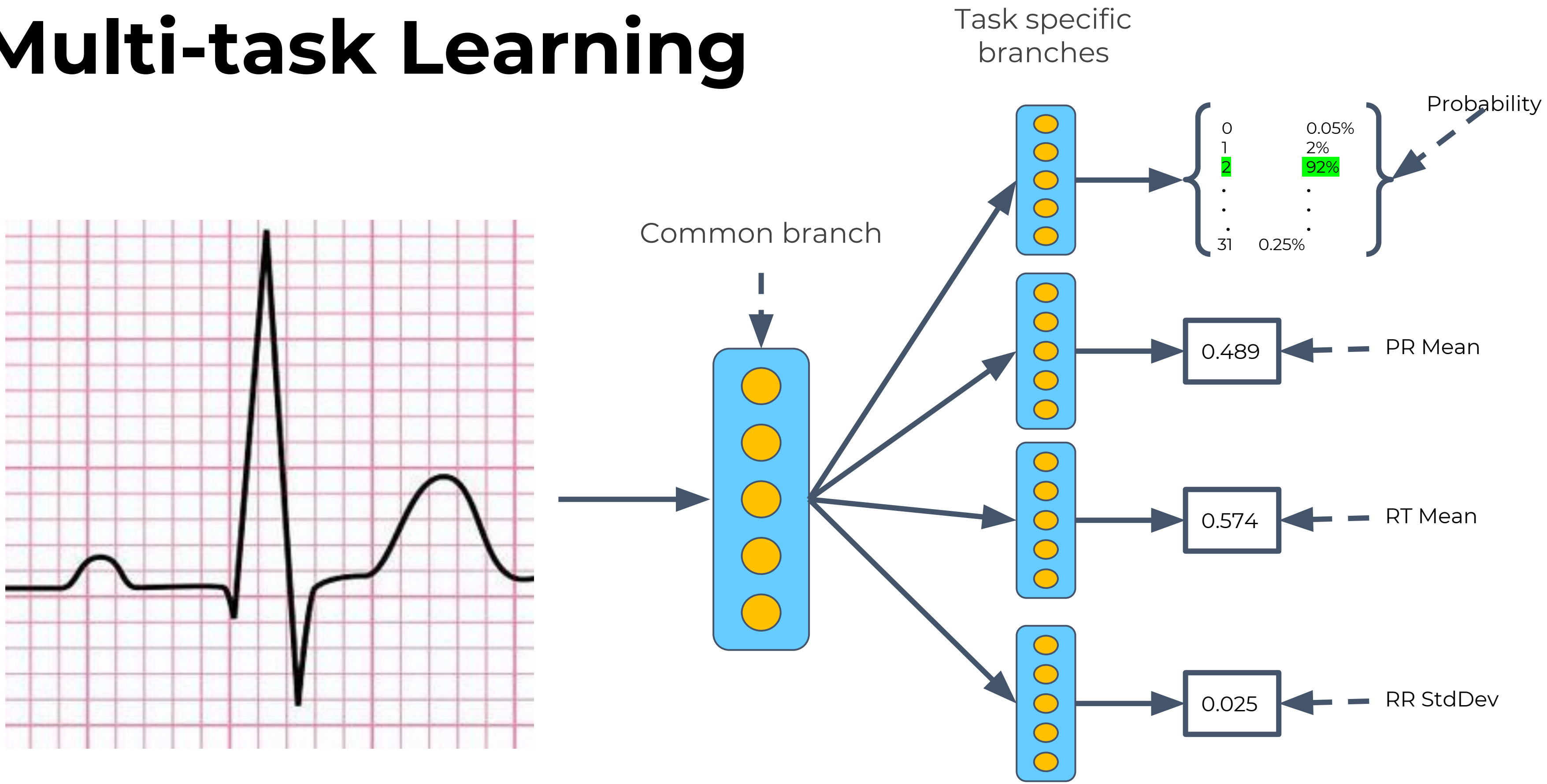
Regression Tasks

- Applicable for the prediction of the fiducial point statistics: **PR Mean, RT Mean, RR StdDev**



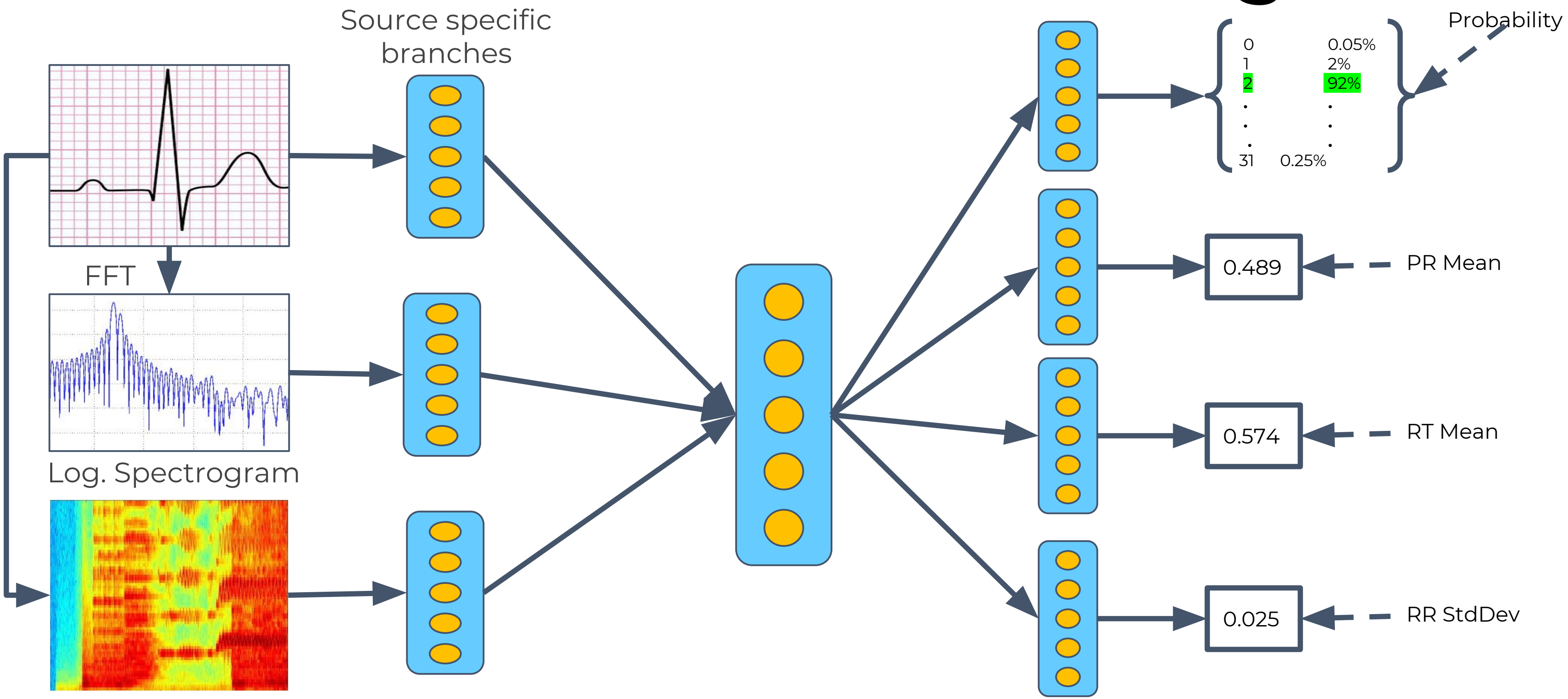
Mila

Multi-task Learning



Mila

Multi-source Multi-task Learning



Mila

Institut
québécois d'intelligence
artificielle

Mila

OMsignal Project
Current Baseline
From Block 1

Baseline from Block 1

- Supervised setting only
- 160+160 data points from training/validation set
- Data augmentation:
  - Random noise

$$s(t) \to s(t) + \delta(t) \qquad \delta(t) \sim \mathcal{N}(0, 1)$$

  - Signal inversion

$$s(t) \to -s(t)$$

  - Time shift

$$s(t) \to s(t - T)$$

Mila

Baseline from Block 1

- Tested architecture:
  - MLP
  - RNN / LSTM
  - CNN 1D (best performance)
- Loss function defined as a direct sum of the single-task losses

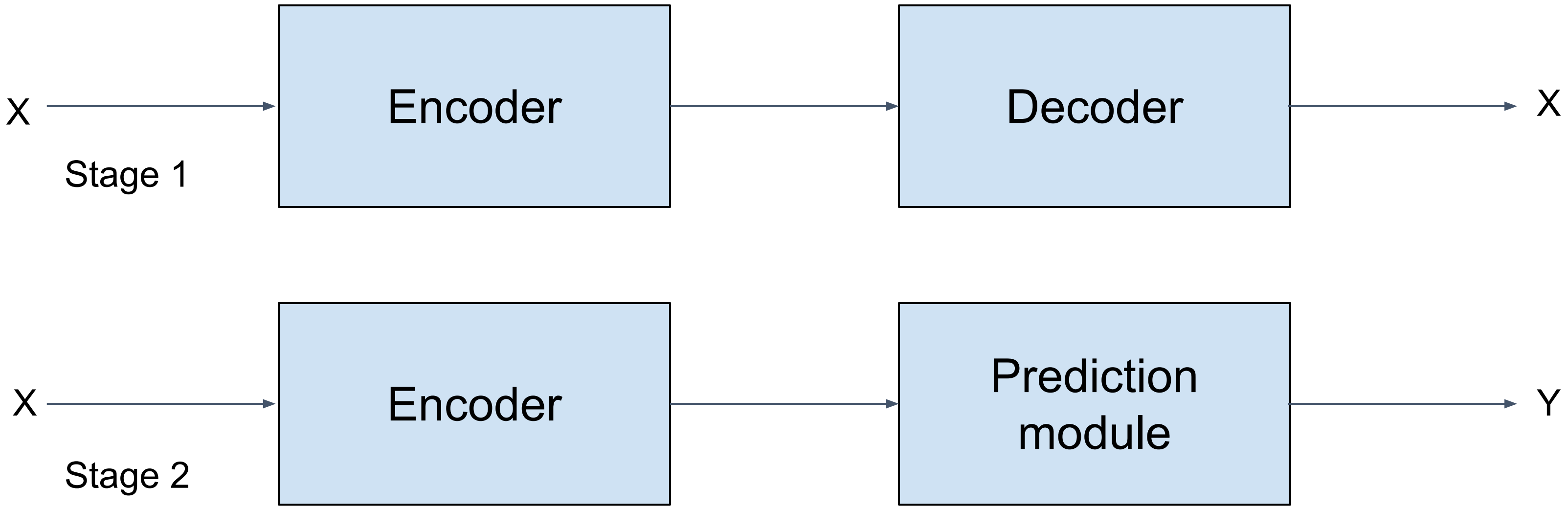
Institut
québécois d'intelligence
artificielle

Mila

OMsignal Project
Incorporating Unlabeled data

Incorporating Unlabeled data

- Unsupervised Pretraining



Incorporating Unlabeled data

- **Unsupervised Pretraining**

  - **Auto-encoder**
    - Can be a **denoising** version if **suitable function** for altering the data is available

  - **Supervised tasks**
    - Use Encoder AS-IS
    - Or fine-tune the encoder weights

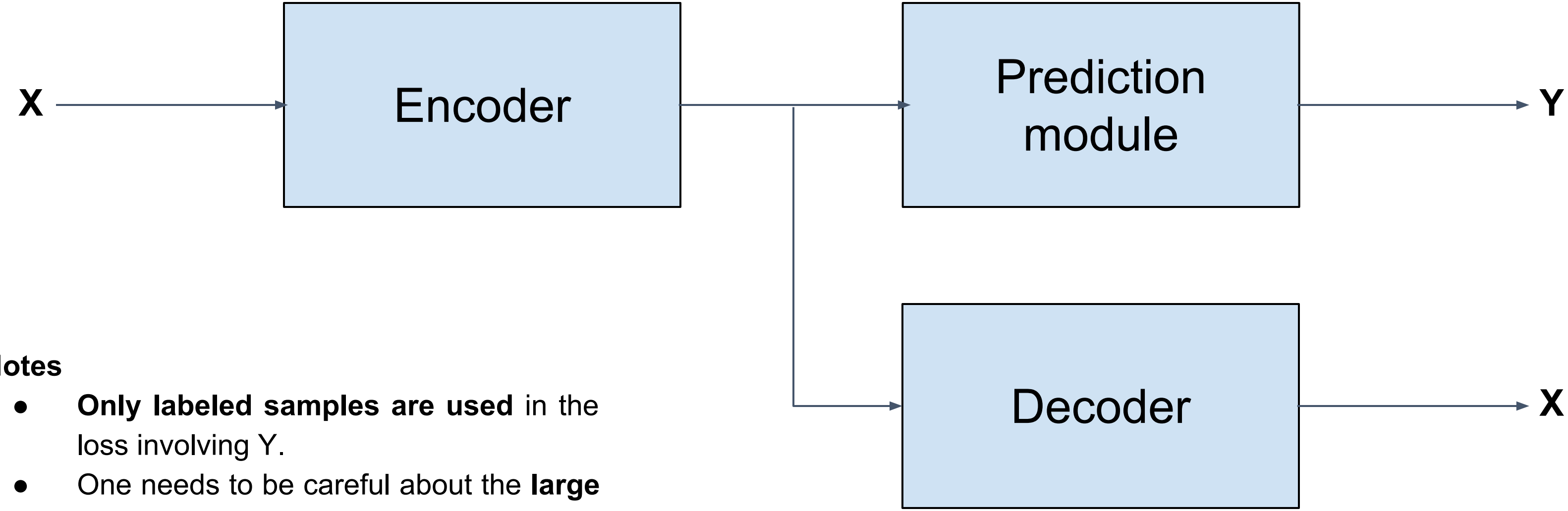
Incorporating Unlabeled data

- **Semi-supervised setting:** use the unlabeled and the labeled data **jointly** in order to train a global architecture.

- Several ways to approach semi-supervised learning in the literature

Semi-Supervised Learning

- **Class 1:** a branch is added in the network for handling unlabeled data.

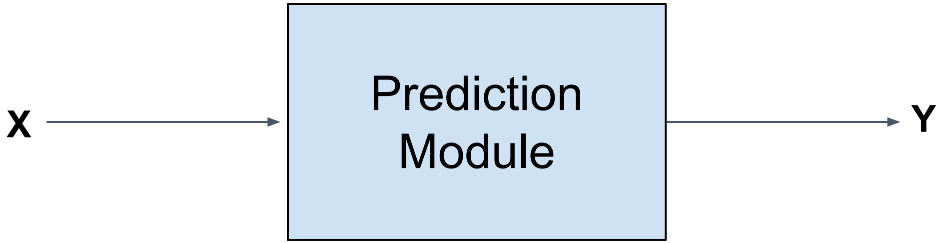


Notes

- **Only labeled samples are used** in the loss involving Y.
- One needs to be careful about the **large discrepancy in the amount of training data for both tasks**.

Semi-Supervised Learning

- **Class 2:**
  - Leave the model unchanged as for the fully supervised setting.



Some hypotheses are made regarding the outputs of unlabeled data. These hypotheses lead to additional loss terms that are added to the loss of the supervised tasks as **regularized terms**.

Semi-Supervised Learning

- **Class 2: Entropy Based Approaches**
  - A simple loss term for the unlabeled data is added to encourage the network to make **“confident” (low-entropy)** predictions for all examples, regardless of the actual class predicted.

The “entropy minimization” term **- sum_{1:k} fθ(x)k log fθ(x)k** is added to the supervised loss (where k is the number of classes in a classification task, and **θ** are the weights of the model)

Mila

Semi-Supervised Learning

- **Class 2: Pseudo Labeling**
  - Produce **“pseudo-labels”** for the unlabeled data using the prediction function itself

  - Pseudo-labels with corresponding class probability **larger** than a predefined threshold are used as **targets for a standard supervised loss** function applied to unlabeled data.

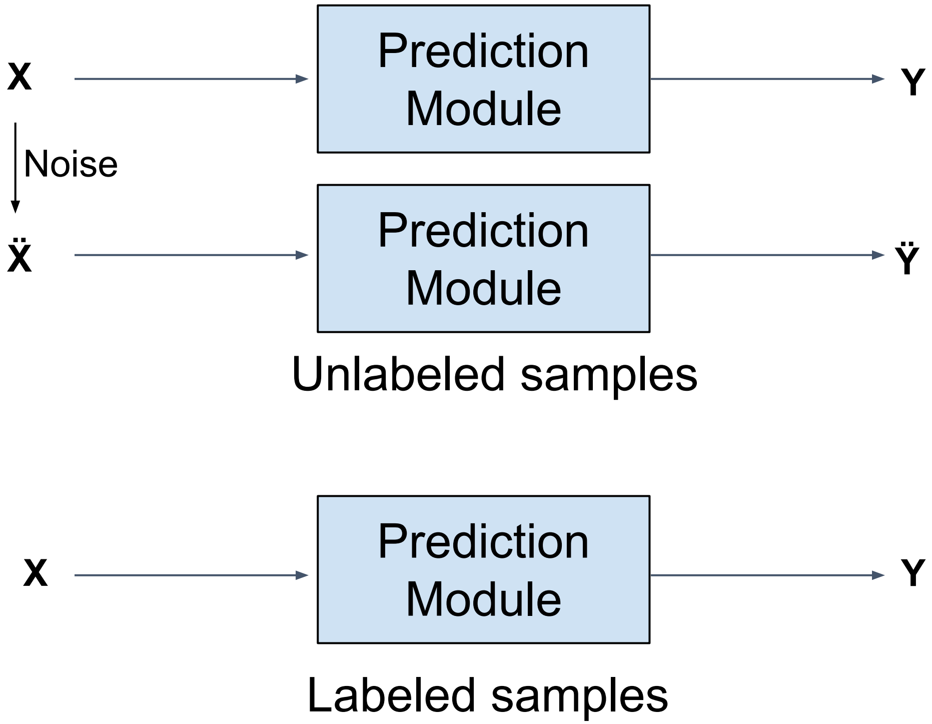
Semi-Supervised Learning

- **Class 2: Consistency regularization**
  - **Intuition**: Realistic perturbations **$X \rightarrow \ddot{X}$** of unlabeled data points **X** should not significantly change the output of **$f\theta(X)$** of the networks.

Minimize $d(f\theta(X), f\theta(\ddot{X}))$ where $d(\cdot, \cdot)$ measures a distance between the prediction function outputs, e.g. mean squared error or Kullback-Leibler divergence.

Semi-Supervised Learning

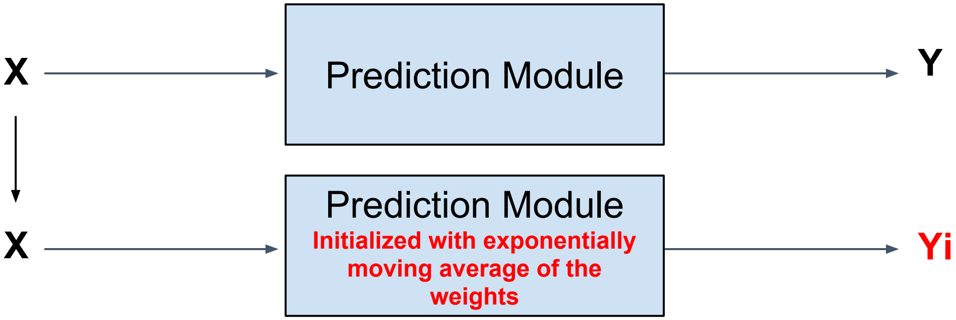
Class 2: Stochastic Perturbations



Semi-Supervised Learning

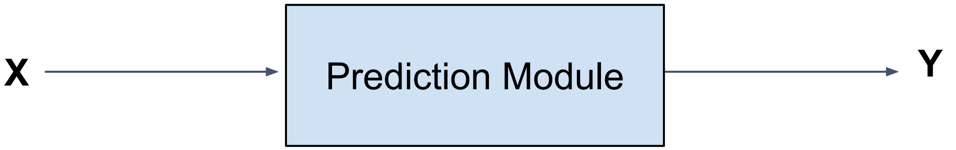
Class 2: Mean Teacher

$$S_t = \begin{cases} Y_1, & t = 1 \\ \alpha \cdot +(1-\alpha) \cdot S_{t-1} & t > 1 \end{cases}$$



$$S_t = \begin{cases} Y_1, & t = 1 \\ \alpha \cdot Y_t + (1-\alpha) \cdot S_{t-1}, & t > 1 \end{cases}$$

Unlabeled samples

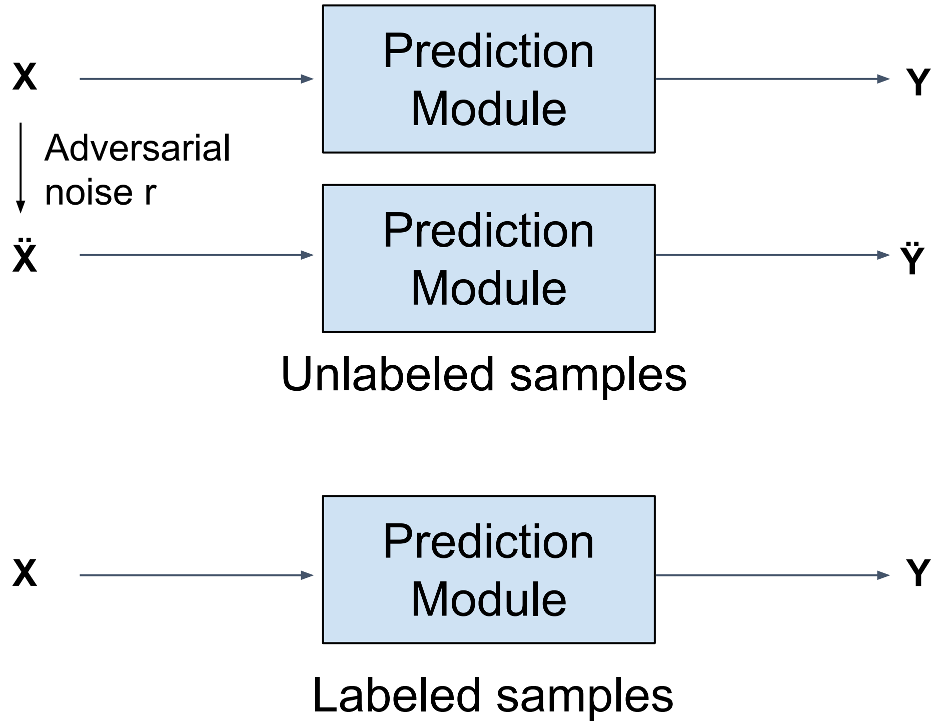


Labeled samples

Mila

Semi-Supervised Learning

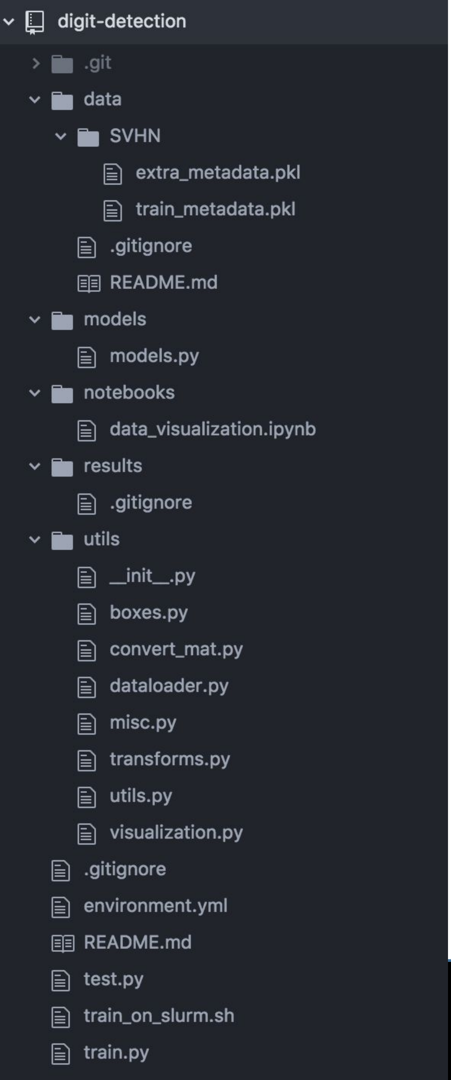
Class 2: Virtual Adversarial Training



Homework

Implement your model

Mila



How the code should be organized

- Have separate folders: config, data, models, notebooks, results, trainer, utils ...
- In each folder, separate the code into several files to simplify reading
- Use meaningful names for your directories and files
- Avoid code duplication => use object-oriented programming (e.g. parent and child classes)

Mila

Running a Deep Learning experiment

- Exploration of hyperparameters to obtain the best model

Script to generate hyperparameter candidate

Hyperparameter 00 → Performance metrics 00

Hyperparameter 01 → Performance metrics 01

... → ...

Each experiment must produce enough logs to:

- diagnose errors and suspicious behaviour

- allow the selection of a set of hyperparameters that is considered the best to generalize to new data

Implementation - To keep in mind (1)

General advice

- See what other people are doing and use it as inspiration.

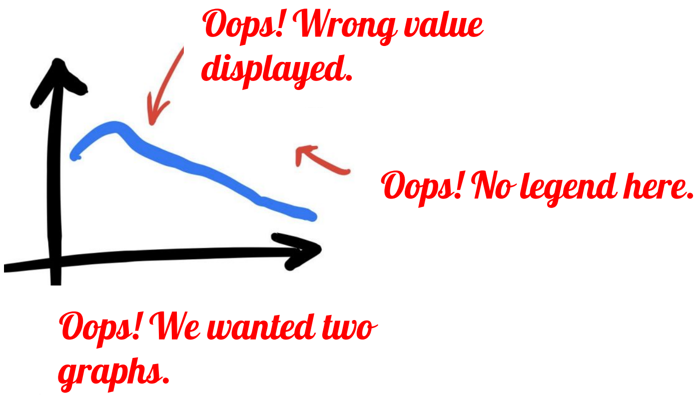
- Develop your own style from this.

Implementation of a deep learning model

- Reuse as much as possible models already available online. Search for "model zoo".

Implementation - To keep in mind (2)

- When you run several large-scale experiments, you must store the data necessary to generate graphs / figures in case you need to modify / update them.



- An experiment that can be interrupted in the middle and then restarted without being affected by the interruption is much easier to manage.

Quebec
Artificial
Intelligence
Institute

Mila

Université
de Montréal