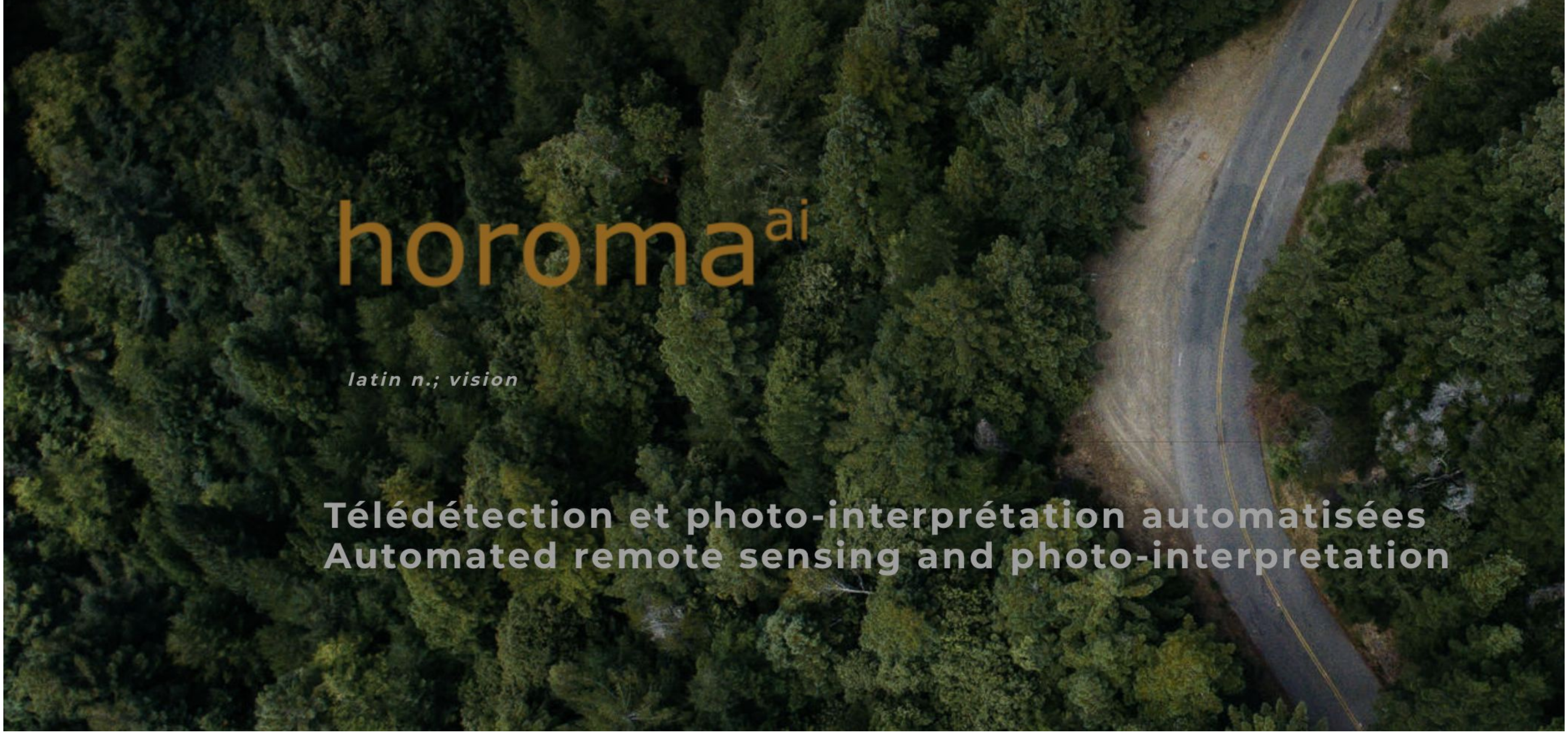
Institut québécois d'intelligence artificielle

Mila

Horoma project
Block 3

Francis Grégoire
Mathieu Germain


horoma ai
latin n.; vision
Télédétection et photo-interprétation automatisées
Automated remote sensing and photo-interpretation

Horoma project

- **Task:** Analysis of images of forest canopy to determine its tree specie.

- **Main challenge:** Data labeling is made by human interpreters and is therefore costly and time consuming.

- **Goal:** Develop an unsupervised or semi-supervised machine learning system capable of predicting forest canopy properties with no or small amounts of labeled data.

Project summary

- **Overall task**:
  - Develop a model that predicts the labels of 32 x 32 pixel patches.
- **Block 1**:
  - Unsupervised training (i.e. labels are provided for the validation set; no labels are provided for the training set).
  - Structure data in a way that is useful to develop models for blocks 2 & 3.
  - Apply *K*-means clustering as a baseline model.
- **Block 2**:
  - Unsupervised training.
  - Develop more advanced clustering algorithms leveraging deep learning methods.
- **Block 3:**
  - Semi-supervised training (i.e. you are free to use labeled examples as training examples).
  - Develop semi-supervised classification or **advanced** unsupervised clustering algorithms.

Objectives

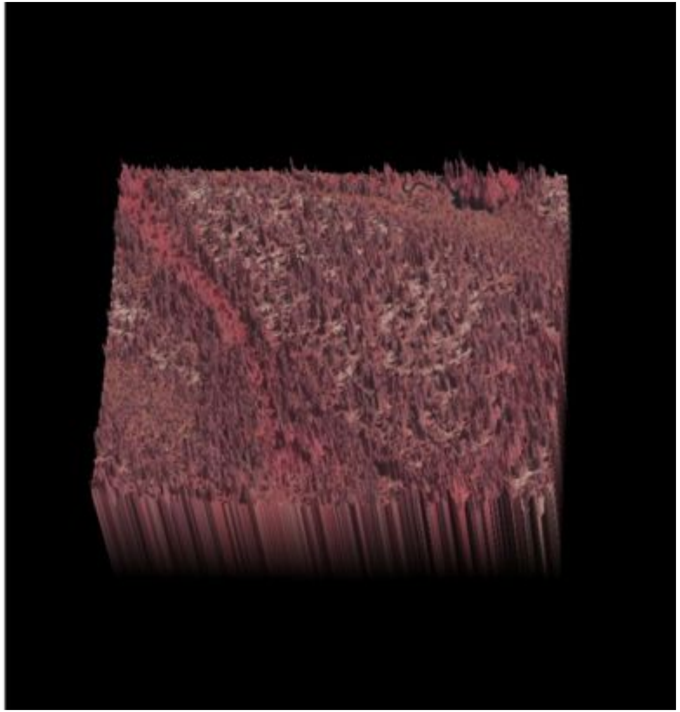
- We expect you to:
    - Explore and select relevant deep learning-based models.
    - Apply clustering algorithms.
    - Analyze your results.
    - Document your ideas and your experiments.
    - **PREPARE FINAL PRESENTATION (BLOCK 4) as soon as possible because you won't have time after block 3.**

Data (image)

- 3 Digital Surface Models (DSM) of forest areas in the Outaouais, a Western Quebec region.

- An image (or DSM) covers approximately 5km*3km (15km^2) of land, such that a single pixel represents an area of about 30cm*30cm.

- All images were taken during the same period of the year when trees are leafy (thus facilitating image labeling).

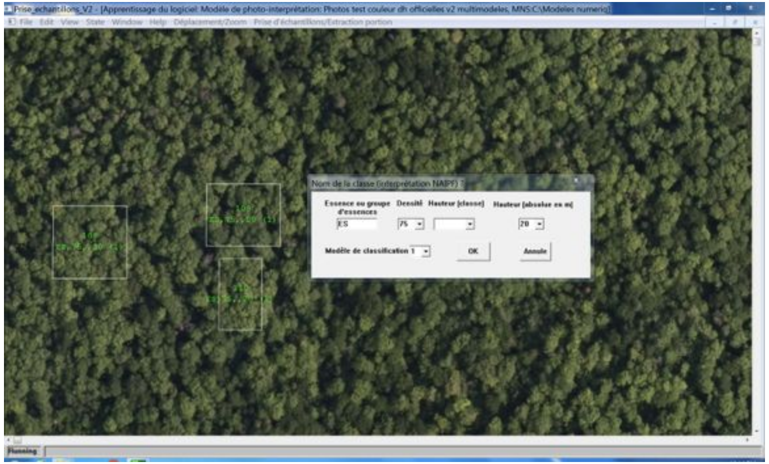


Mila

Data (image labeling)

- Each image was labeled by a different human interpreter.

- Interpreters only labeled subsections of an image, with each subsection containing trees of the same overall specie.



Mila

Data (pixel)

- Each pixel of an image has 3 values associated with it:
    - **RGB colors** (3 values). Those values are in [0, 255].

Data (pixel patches)

- 32 x 32 pixel patches were extracted from labeled image subsections.
- **Inputs**: each pixel in a 32 x 32 pixel patch has 3 dimensions.
- **Outputs**: each 32 x 32 pixel patch has 1 label: tree specie.

Data (input format)

- Inputs are provided as binary numpy.memmap files in **float32**.

- Memory-mapped files are used for accessing small segments of large files on disk, without reading the entire file into memory:
    - **train_x.dat**: 152,000 x 32 x 32 x 3.
    - **train_labeled_x.dat**: 228 x 32 x 32 x 3 (used to label clusters).
    - **valid_x.dat**: 252 x 32 x 32 x 3 (used to evaluate your models).

- You also have access to files containing overlapped patches to increase the size of your datasets (pixel patches with ~50% overlap):
    - **train_overlapped_x.dat**: 548,720 x 32 x 32 x 3.
    - **train_labeled_overlapped_x.dat**: 635 x 32 x 32 x 3.
    - **valid_overlapped_x.dat**: 696 x 32 x 32 x 3.

Mila

Data (input format)

You are not forced to use this split (e.g. 228 labeled examples as training data).
For example, you could do cross-validation or random subsampling and use nearly every labeled example as training data.

- Inputs are provided as binary numpy.memmap files in **float32**.
- Memory-mapped files are used for accessing small segments of large files on disk, without reading the entire file into memory:
  - **train_x.dat**: 152,000 x 32 x 32 x 3.
  - **train_labeled_x.dat**: 228 x 32 x 32 x 3 (used to label clusters).
  - **valid_x.dat**: 252 x 32 x 32 x 3 (used to evaluate your models).
- You also have access to files containing overlapped patches to increase the size of your datasets (pixel patches with ~50% overlap):
  - **train_overlapped_x.dat**: 548,720 x 32 x 32 x 3.
  - **train_labeled_overlapped_x.dat**: 635 x 32 x 32 x 3.
  - **valid_overlapped_x.dat**: 696 x 32 x 32 x 3.

Data (output format)

- Outputs are provided as text files (can be easily read from a terminal).

- **train_labeled_y.txt**: contains 228 tree species (2 characters).
- **valid_y.txt**: contains 252 tree species (2 characters).

- **train_labeled_overlapped_y.txt**: contains 635 tree species (2 characters).
- **valid_overlapped_y.txt**: contains 696 tree species (2 characters).

- The *i*-th value in ***filename*_y.txt** is associated to the *i*-th pixel patch in ***filename*_x.dat**.

Data (output format)

You are not forced to use this split (e.g. 228 labeled examples as training data).
For example, you could do cross-validation or random subsampling and use nearly every labeled examples as training data.

- Outputs are provided as text files (can be easily read from a terminal).

- **train_labeled_y.txt**: contains 228 tree species (2 characters).
- **valid_y.txt**: contains 252 tree species (2 characters).

- **train_labeled_overlapped_y.txt**: contains 635 tree species (2 characters).
- **valid_overlapped_y.txt**: contains 696 tree species (2 characters).

- The *i*-th value in ***filename_y.txt*** is associated to the *i*-th pixel patch in ***filename_x.dat***.

Data (regions id)

- To split your own labeled datasets efficiently, you have access to files containing ids representing the pixel subregion where each pixel patch has been extracted from images:
  - **train_regions_id.txt**.
  - **train_overlapped_regions_id.txt**.

  - **train_labeled_regions_id.txt**.
  - **train_labeled_overlapped_regions_id.txt**.

  - **valid_regions_id.txt**.
  - **valid_overlapped_regions_id.txt**.

- Tip: you should limit the number of shared regions between your train and valid sets.

Data (tips)

- This is a hard learning task.
- Find clever ways to exploit the information that is given to you to determine the number of clusters.
- You should document your different approaches and explain your ideas in your report.
- It’s important to think beyond the numbers and to analyze your results.

Block 3 instructions & expected timeline

	Mar 20 & 22	Mar 27 & 29	Apr 3 & 5	Apr 10 & 12
Tasks	• Read about relevant deep learning-based models (see slide 42 and semi-supervised section) • Select code from Block 2 • Start preparing the final presentation	• Develop deep learning-based models • Apply classification or clustering algorithms on latent representations	• Finish deep learning-based models • Evaluate models against the best performing model from Block 2 using labeled examples	• Write a short report summarizing the work, and results • Peer review the code of other teams • Finish final presentation
Objectives / Deliverables	• Understand the data • Understand deep learning-based models	• Have a clear idea on which models to explore • Build code on top of code from Block 2	• Models and predictions ready • Visualize and discuss the results	• Produce documented code and report summarizing the experimental work • Provide model for blind test set evaluation • Complete peer code review • Final presentation ready

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 grid for block 3

- **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

- **12% of the final score**.
- 5-7 pages (including figures, tables, and references).
- Single column.
- Font size 11.
- Use the NeurIPS LaTeX format.

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 a description of the approaches used to fine tune the hyperparameters, select the best "model" using checkpointing, 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.). Note that this should include: • A comparison with results from the previous block. • Figures showing the loss value across epochs/checkpoints and models (using tensorboard).	/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

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 set** will be structured identically to the **valid set**.
- 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.

Global evaluation (for block 4)

Content of the presentation	/5
Description of the project	/1
Description of the solutions adopted	/1
Presentation of the achievements	/1
Identification of major problems	/1
Synthesis of findings and recommendations	/1
Format of the presentation	/3
The presentation is clear and structured	/1
Figures and tables are adequate to present the results	/1
Respect of time	/1
Questions period	/1
The answers to the questions are precise and clear	/1

25% for final presentation in front of companies (15 min presentation + 5 min questions).

We recommend that you start working on the final presentation during block 3.

Official evaluation metric

- For each patch, you need to predict its tree specie.
- Use ground truth labels found in **valid_y.txt** to calculate the F1 score of your predictions (see sklearn.metrics.f1_score).
- An evaluation script will be provided.

Informative evaluation metrics

- You have to explore widely used metrics for clustering performance evaluation such as **Normalized Mutual Information** (NMI) and **adjusted Rand index** (ARI) (see scikit-learn documentation).

- F1 score of tree specie:

$F1_s$ = `sklearn.metrics.f1_score(specie_pred, specie_true)`

Mila

Precision, recall, F1 score

- **Precision** is the proportion of correctly classified examples among all classified examples.

- **Recall** is the proportion of correctly classified examples among all examples in the validation/test set.

- **F1 score** is the harmonic mean of precision and recall.

Institut
québécois d’intelligence
artificielle

Mila

Autoencoders

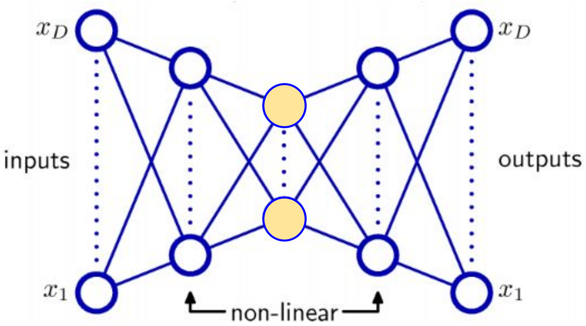
Unsupervised learning

- We assume that a data generating process contains internal variables that are not observed, but that still influence the data.
- These unobserved variables are called hidden variables or latent variables.
- We postulate that there is a latent variable $\mathbf{z}_i$ associated with each training example $\mathbf{x}_i$.

Autoencoders

- A basic autoencoder is a feedforward neural network trained to reproduce its inputs (i.e. outputs = inputs).
- The size of the hidden layer is smaller than the input’s size.
- This leads to reconstruction errors, which we seek to minimize during training.
- We obtain a latent representation of the inputs in the hidden layer.
- Traditionally used for dimensionality reduction or feature learning.

Autoencoders



- An autoencoder can be viewed as consisting of two parts:
  - An **encoder** that maps the inputs to a latent representation in the hidden layer.
  - A **decoder** which creates a reconstruction of the inputs from the hidden layer.

Autoencoders

- Reference: Chapter 14 - Autoencoders of the Deep Learning book.
- https://www.deeplearningbook.org/contents/autoencoders.html
- Intuitive introduction to VAE [blog][video]
- Overview of multiple types of autoencoders

Baseline block 2

- Vanilla autoencoder.

- Hyperparameter found via extensive hyperparameter search.

- K-means clustering done at the end of each training epoch.

- Apply early stopping based on F1 score.

- 1. Train autoencoder on train set.
  2. Label clusters on a subset of the valid set.
  3. Evaluate model on leftover of the valid set.

- Bottleneck hidden size = 3.

- Number of clusters = 150.

- See code for details.

Institut
québécois d'intelligence
artificielle

Mila

K-means clustering

K-means clustering

- Most widely used clustering algorithm.
- We consider $\mathbf{z}_i$ as a discrete latent variable.
- The inference of $\mathbf{z}_i$ is known as clustering because it leads to assigning each training example $\mathbf{x}_i$ to a group among K, where $\mathbf{z}_i$ represents the assigned group (cluster).
- We use one-hot encoding to model $\mathbf{z}_i$.
- e.g. if $K = 5$ and $\mathbf{z}_i$ is associated to group 4, then $\mathbf{z}_i = [0, 0, 0, 1, 0]$ ($z_{i4} = 1$).

K-means clustering

- We suppose that there are K prototypes $\mathbf{u}_1, \ldots, \mathbf{u}_K$ which represent the means of the clusters (centroids).

- The dimension of a prototype $\mathbf{u}_k$ is equal to the dimension of $\mathbf{x}_i$.
- Each $\mathbf{x}_i$ belongs to the cluster with the nearest prototype (i.e. shortest Euclidean distance).

- The loss function of K-means clustering minimizes the average distance between the training examples and their associated $\mathbf{u}_k$:

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} \|\mathbf{x}_i - \mathbf{u}_k\|^2$$

K-means clustering

- The iterative learning process corresponds to finding the prototypes $\mathbf{u}_k$ as well as the associations $\mathbf{z}_i$ that minimize J.

- It is common practice to initialize the $\mathbf{u}_k$ by randomly choosing K training examples and using these as initial values.

K-means clustering

- Finding the optimal $\mathbf{z}_i$:
  - By knowing the $\mathbf{u}_k$ we can optimize all $\mathbf{z}_i$ independently.
  - J is minimized if $z_{ik} = 1$ with the closest $\mathbf{u}_k$ ($\mathbf{z}_i = [z_{i1}, z_{i2}, \dots, z_{iK}]$) and 0 otherwise.
  - The optimal solution is:

$$z_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mathbf{u}_j\|^2 \\ 0 & \text{else} \end{cases}$$

K-means clustering

- Finding the optimal $\mathbf{u}_k$:
  - By knowing the $\mathbf{z}_i$ we can minimize J by setting the derivative w.r.t. $\mathbf{u}_k$ equal to zero:

$$2\sum_{i=1}^{N} z_{ik}(\mathbf{x}_i - \mathbf{u}_k) = 0$$

  - The optimal solution is:

$$\mathbf{u}_k = \frac{\sum_i z_{ik}\mathbf{x}_i}{\sum_i z_{ik}}$$

  - The optimal $\mathbf{u}_k$ is the mean of all examples associated to the cluster k.

K-means clustering

- The learning process consists of iterating the inference of $\mathbf{z}_i$ and the update of $\mathbf{u}_k$ until the convergence of J:

1. Initialize $\mathbf{u}_k$

    2. Repeat until convergence {

        Find the optimal $\mathbf{z}_i$

        Find the optimal $\mathbf{u}_k$

    }

- This procedure is guaranteed to converge because J is positive and at each iteration, either J decreases or does not change.

K-means clustering

PLAY WITH ME!

Institut
québécois d’intelligence
artificielle

Mila

Recommended advanced unsupervised clustering papers

Recommended papers (non-exhaustive)

- Foundations:
  - Masci et al. - Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction
  - You are invited to search for relevant papers😎.
- Recent approaches:
  - Xie et al. - Unsupervised Deep Embedding for Clustering Analysis
  - Yang et al. - Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering
  - Caron et al. - Deep Clustering for Unsupervised Learning
  - Chazan et al. - Deep Clustering Based on a Mixture of Autoencoders
- Survey:
  - Min et al. - A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture

Institut
québécois d’intelligence
artificielle

Semi-supervised learning

Arsene Fansi-Tchango

Semi-supervised learning

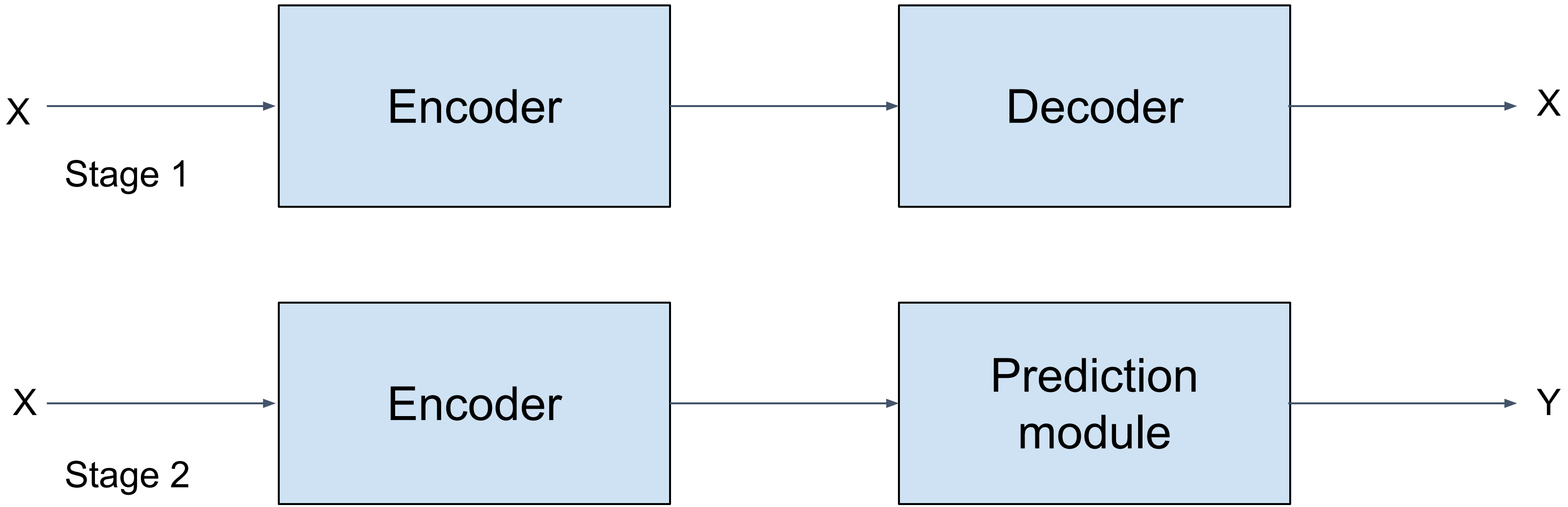
- You are free to explore semi-supervised approaches.

- See reference document made by Arsène for the OMSignal project:

https://github.com/mila-iqia/ift6759/blob/master/projects/omsignal/slides/omsignal-syllabus-block2.pdf

Incorporating unlabeled data

- Unsupervised pretraining.



Incorporating unlabeled data

- Unsupervised pretraining.
  - **Autoencoder**
    - 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.

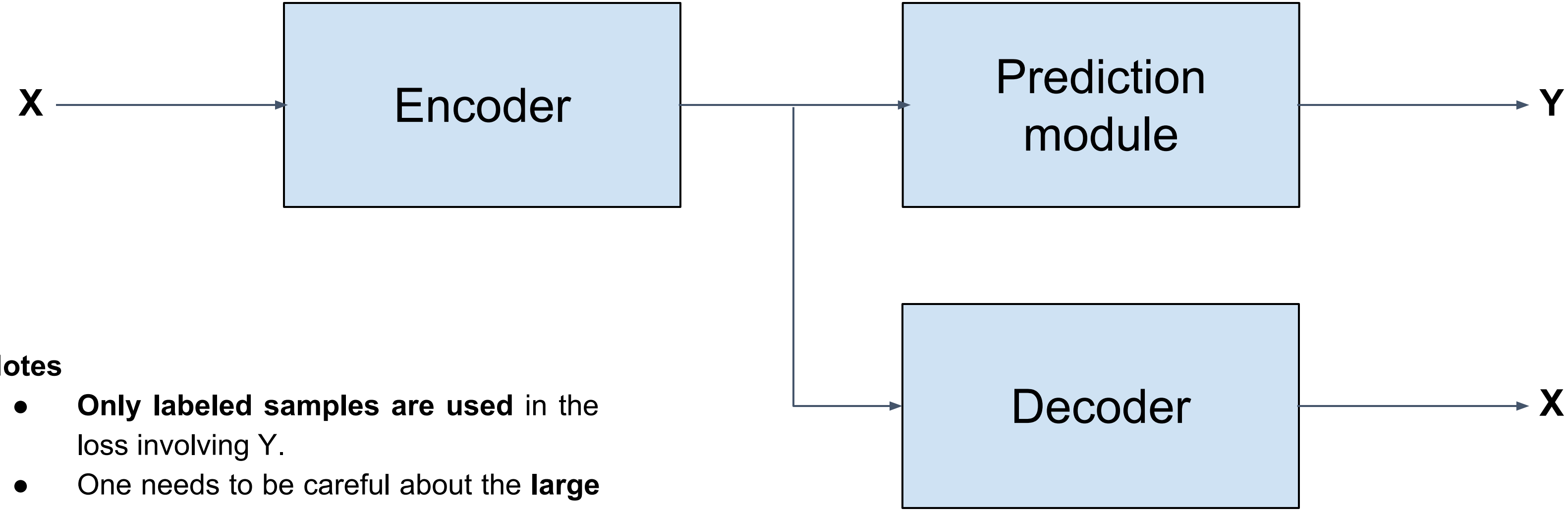
Incorporating unlabeled data

- **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.

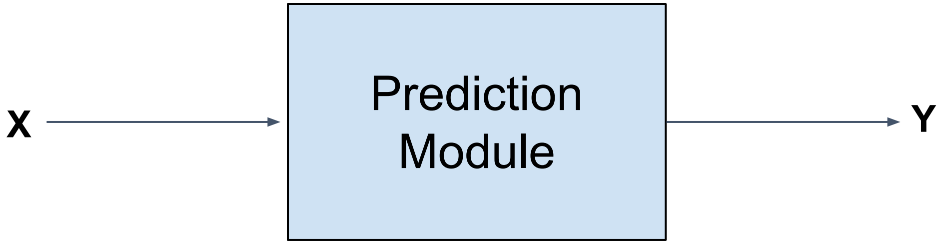


Notes

- **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: 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.

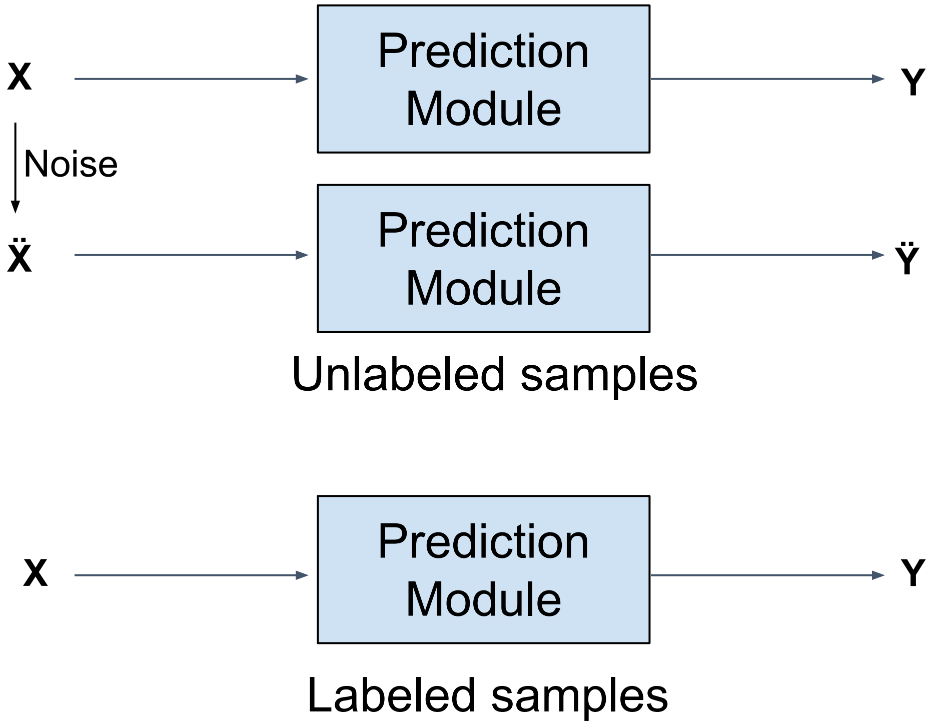
Semi-supervised learning

- **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

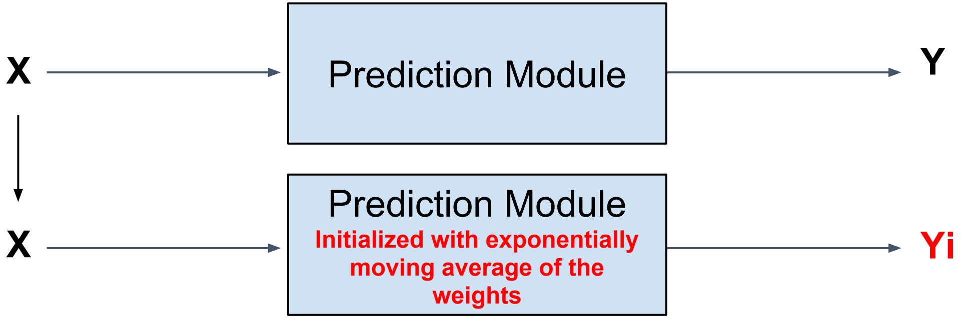
Class 2: Stochastic perturbations



Semi-Supervised learning

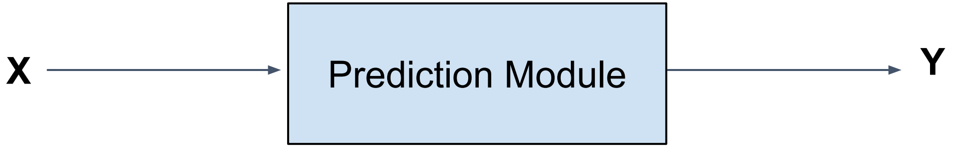
Class 2: Mean teacher

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



Unlabeled samples

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



Labeled samples

Semi-Supervised learning

Class 2: Virtual adversarial training