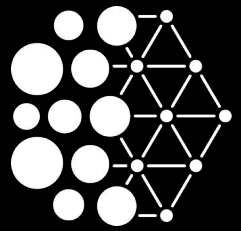
Institut
québécois
d'intelligence
artificielle



Mila

Block 3 - Humanware project

Margaux Luck, PhD
Jeremy Pinto

HumanWare

Over 25 years of service to people with vision loss

Recognized for its capacity to innovate
(Digital reading systems, talking book players, handheld talking GPS)

A worldwide network of distributors
(North America, Europe & Australasia)

150 employees making independence possible
(15% of our employees are also our clients)

#1 worldwide in blindness service

Products supporting over 25 languages
(including the Braille script)

Products available in more than 45 countries

Drummondville, Canada
Head Office
Operations Canada & USA

Rushden, Angleterre
Operations Europe,
Middle East & Africa

Longueuil, Canada
Marketing & R&D

Sydney, Australia
Operations Australasia

Humanware project

- **Task:** Detection of text in natural scenes, with its location and the ability to guide the user to get the full text (alignment) so that the text is interpretable by a recognition engine.

- **Use case:** Detection of house numbers, text on a bus stop sign, or text on a storefront (name, product details, opening hours, ...)

- **Constraints:** Execution time, online vs. offline, memory usage (in the case of a mobile application), etc.

Focus on door number detection

668

- Help blind persons find their way around
- Make sure that’s the right house
- (Long term) Facilitate micronavigation

Data (block 1 & 2)

The Street View House Numbers (SVHN) Dataset:

- Open-source
- ~200k street numbers
- bounding boxes and class labels for individual digits, giving about 600k digits total

Limitation:

- zoom on the numbers, lack of background
- no negative examples (i.e. no images without numbers)

[Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011.]

Mila

Data (block 3)

Synthetic dataset (kindly provided by ElementAI):

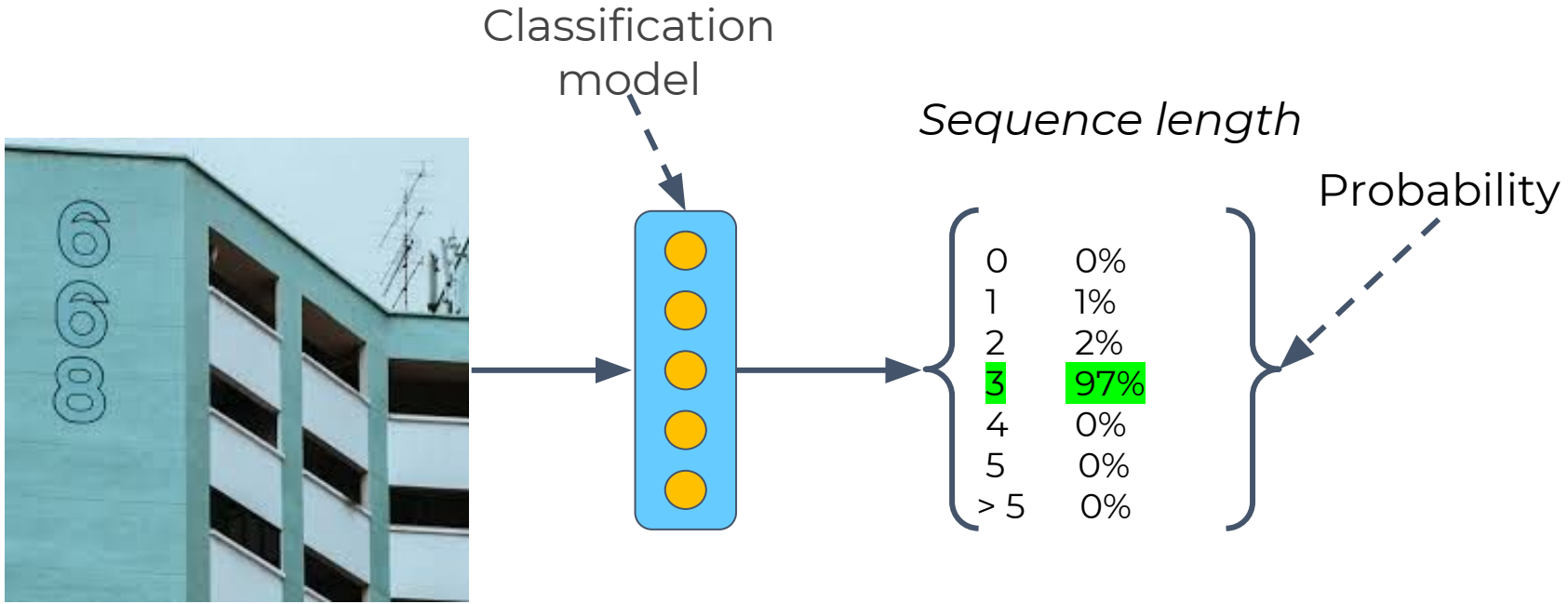
- Private for now
- 6k street numbers (5k train + 1k valid)
- bounding boxes, segmentation masks and class labels for sequence of digits

Limitation:

- Not really realistic
- Not enough diversity

... But different angles and more varied viewpoints than SVHN

Block 1: Sequence length classification



Block 2: Sequence digit recognition

Classification model

Sequence length

0	0%
1	1%
2	2%
3	97%
4	0%
5	0%
> 5	0%

Sequence number:

- 1st digit
- 2nd digit
- 3rd digit
- 4th digit
- 5th digit

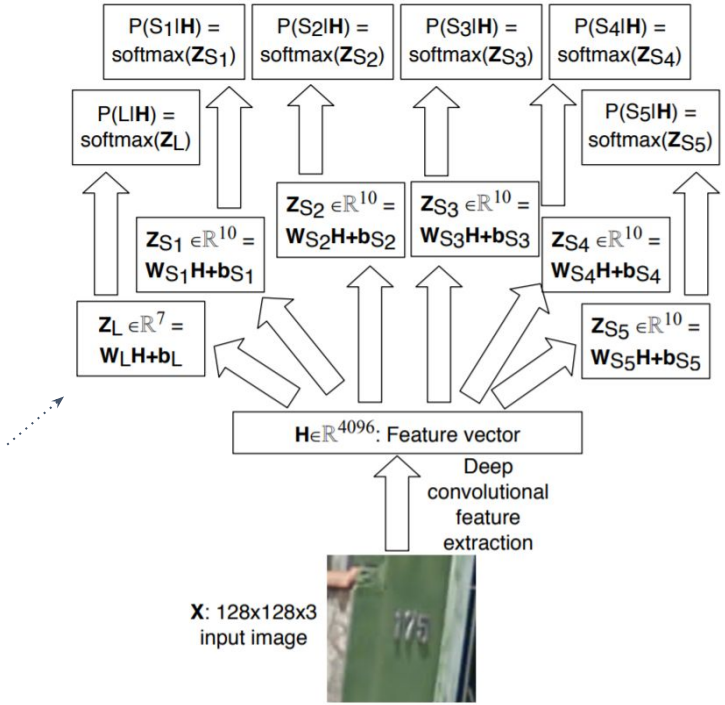
0	0%
1	0%
2	0%
3	0%
4	0%
5	0%
6	87%
7	0%
8	3%
9	0%

0	0%
1	0%
2	0%
3	0%
4	0%
5	0%
6	87%
7	0%
8	3%
9	0%

...

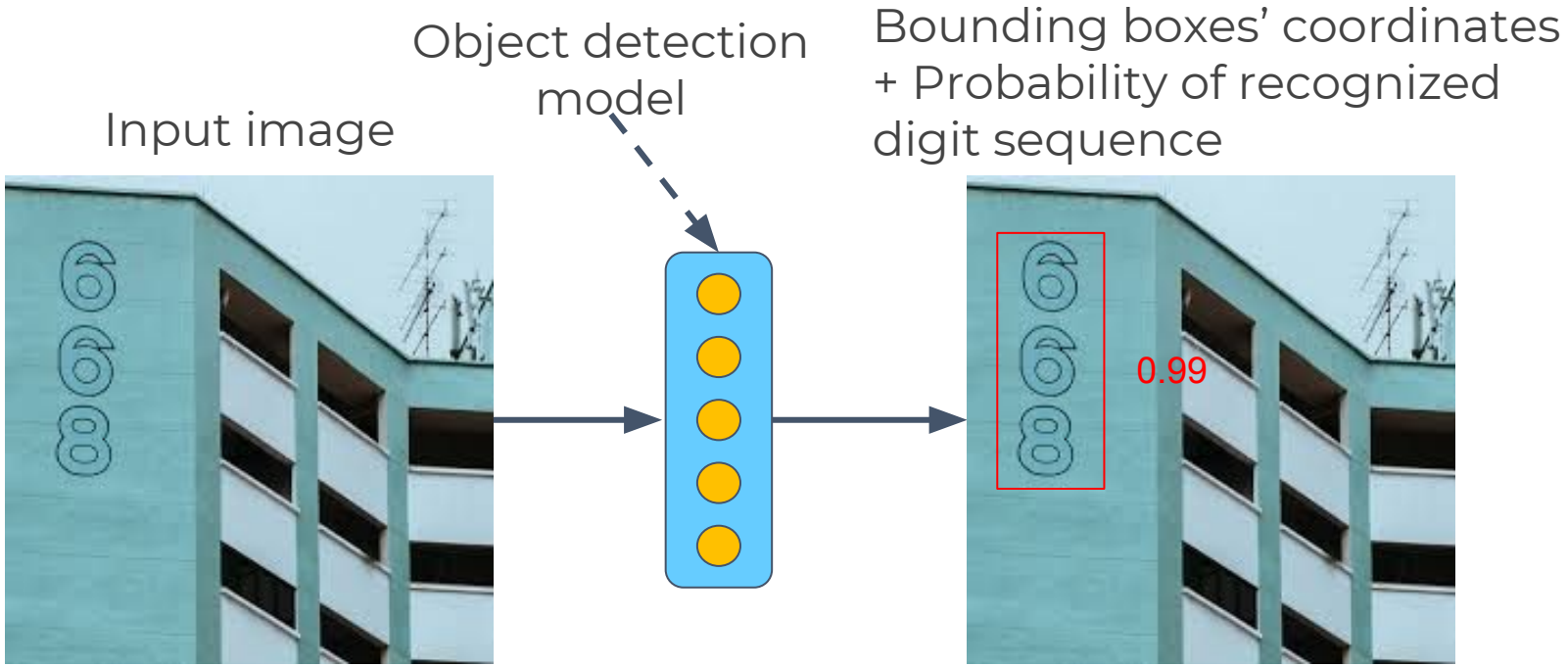
Block 2: Sequence digit recognition

Sequence numbers



Sequence length

Block 3: Object detection



Block 3: Construct the pipeline

Put together two models:

1) Object detection model.

2) Sequence digit recognition model (based on the block 2 model) and which is fed the bounding box predicted by the object detection model.

Mila

Block 3 instructions / expected timeline

	Mar 20 & 22	Mar 27 & 29	Apr 3 & 5	Apr 10 & 12
Tasks / Homework	• Read the Goodfellow et al. 2013 article • Read and understand the code and reports from block 2	• Understand and run Faster R-CNN object detection model on our data using facebook code	• Implement the pipeline • Visualize and discuss the results • Make recommendations for next steps	• Write a short report summarizing the work, and results • Prepare final presentation • (Peer-) Review of other teams' code
Objectives / Deliverables	• Select one code from block 2, along with its best model and report	• Have a running version of Faster R-CNN	• Pipeline ready • Visualize and discuss the results • Make recommendations for next steps	• Produce documented code and report summarizing the experimental work • Provide model for blind test set evaluation • Complete the peer code review • Have a solid draft of the final presentation

Evaluation grid for block 3

25% of the final score

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

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.

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

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

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

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 **train 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. It will not be the same script as block 2 and 3. 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**.

Global evaluation (after 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 4

Construct the pipeline

Model 1: Object detection model

Model 2: Sequence recognition (block 2 model)

House Number: 1877

Mila

Construct the pipeline

Mila

Data preparation model 1 (done by TAs)

- We preprocess the data so that it's in the format of the COCO dataset.
- In the case of this synthetic dataset, there is only one class, i.e. "House Number".
- Each annotation contains a bbox, as well as a segmentation (which is a polygon made from the bbox; you do not have to use it).
- More info on COCO format: http://cocodataset.org/#format-data

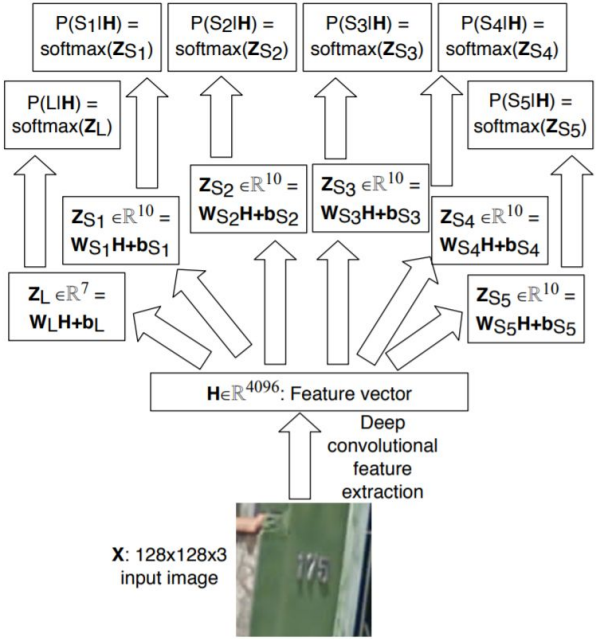
```
annotation{
   "id"                : int,
   "image_id"          : int,
   "category_id"       : int,
   "segmentation"      : RLE or [polygon],
   "area"              : float,
   "bbox"              : [x,y,width,height],
   "iscrowd"           : 0 or 1,
}

categories[{
   "id"                : int,
   "name"              : str,
   "supercategory"     : str,
}]
```

Data preparation model 2 (1)

- The current models from bloc 2 have been trained on SVHN.

- You can use the new synthetic data provided to improve or retrain the models.
  - You will need to preprocess the data accordingly.
  - We will provide you with the labels in SVHN style.



Data preparation model 2 (2)

- The model of block 2 can be fine tuned using the Element AI data.

- There are actually 4 options:
  - Train only on the Element AI data and ignore the SVHN data.
  - Train both on the Element AI and SVHN data.
  - Train first on the SVHN data and then fine tune on the Element AI data.
  - Train on the SVHN and Element AI data and then fine tune on the Element AI data.

- Useful link (don't limit yourself to this blog): https://docs.google.com/spreadsheets/d/1q9EHOg4QD662Ihtk7qRohF0oR4CRpK4mY1vQBWwDqu8/edit#gid=2102856037

Data preparation - To keep in mind

- Data preparation is dependent on the data set
- Consider standardizing the data
- Data augmentation can be (but is not always) useful
- Data difficulties / challenges:
  - Images and objects of different sizes
  - Adversarial examples
  - Partially visible objects
  - Similar / “confusable” classes
  - Bounding boxes need to be properly updated when doing some data transformations
  - Few or no examples for some of the classes in the training set
  - …

Object detection part (i.e., model 1)

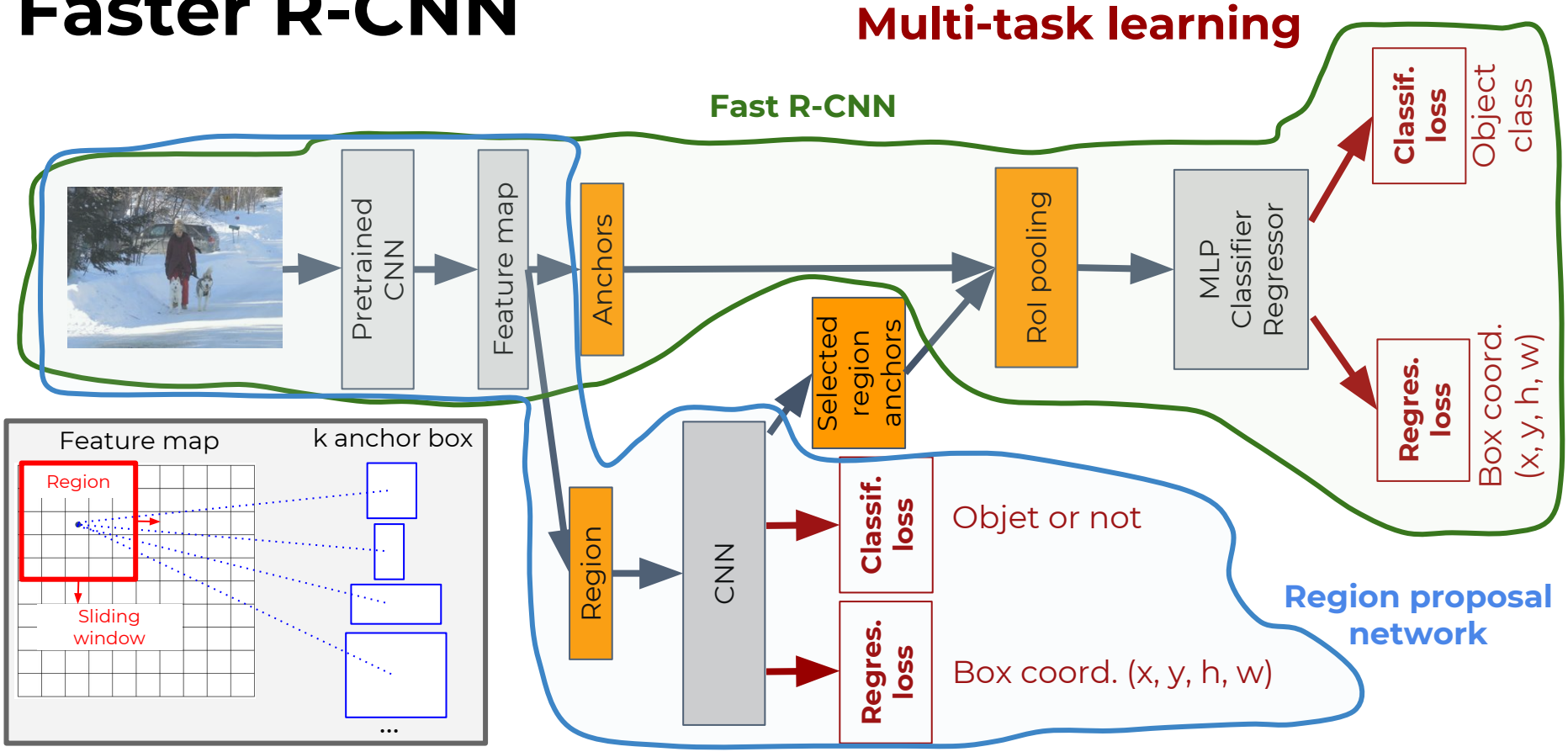
- Use facebook pytorch implementation

https://github.com/facebookresearch/maskrcnn-benchmark

- Don’t lose too much time to finetune the models (whose training can be long)

- Use Faster R-CNN

Faster R-CNN


Multi-task learning
Fast R-CNN
Pretrained CNN
Feature map
Anchors
RoI pooling
MLP Classifier Regressor
Classif. loss
Object class
Regres. loss
Box coord. (x, y, h, w)
Selected region anchors
Region
CNN
Classif. loss
Objet or not
Regres. loss
Box coord. (x, y, h, w)
Region proposal network
Feature map
k anchor box
Region
Sliding window
...

Other models:

- R-FCN (2016) : https://arxiv.org/pdf/1605.06409.pdf
- R-CNN (2014) : https://arxiv.org/pdf/1311.2524.pdf
- Fast R-CNN (2015) : https://arxiv.org/pdf/1504.08083.pdf
- YOLO (2016) : https://arxiv.org/pdf/1506.02640.pdf
- SSD (2016) : https://arxiv.org/pdf/1512.02325.pdf
- Mask RCNN (2017) : https://arxiv.org/pdf/1703.06870.pdf
- ...

Classification part (i.e., model 2)

- Done during blocks 1 & 2
- You need to adapt the data-loader to be able to take the output of model 1.
- We expect an end-to-end pipeline: i.e. given a raw image, return the bounding boxes as well as the digits contained.

Mila

Scores used for block 3

- We split the task into two independent subtasks:
  - Finding the right bounding box containing a house number (model 1: object detection)
  - Detecting the sequence number in the box (model 2: classification)

Reminder : classification scores / metrics

- Accuracy = (TP + TN) / (TP + FP + FN + TN)
- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1-score = 2 * (precision * recall) / (precision + recall)

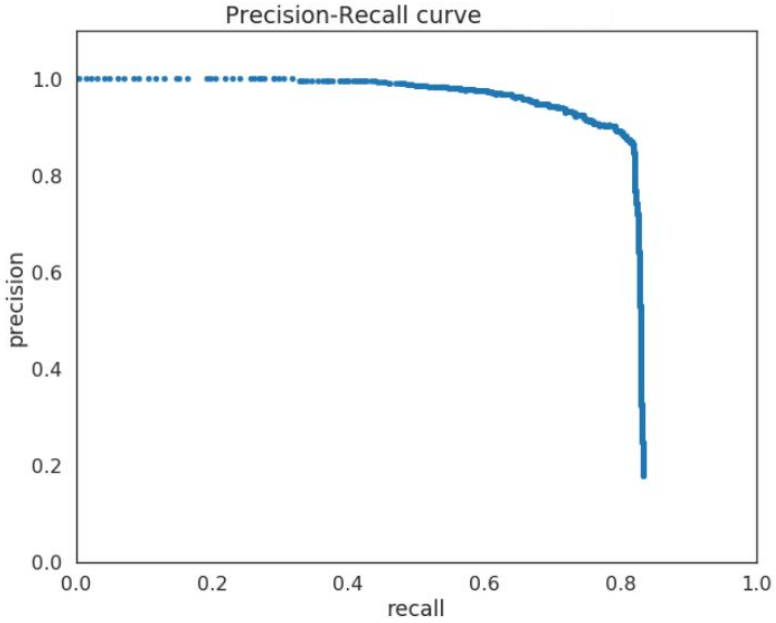
Where

- TP = true positives
- TN = true negatives
- FP = false positives
- FN = false negatives

Scores used in object detection (AP)

- For a given bbox, we compute the Average Precision (AP)

$$AP = \frac{1}{11} \sum_{\text{Recall}_i} \text{Precision}(\text{Recall}_i)$$



Mila

Scores used in object detection (IoU)

- To determine how well the predicted box fits, we use the Intersection over Union metric

$$IoU = \frac{\text{area of overlap}}{\text{area of union}}$$

Mila

Scores used in object detection (mAP)

- To calculate the mean average precision (mAP), we calculate the average precision as a function of varying IoU tolerance thresholds. For COCO, j=10

$$AP = \frac{1}{11} \sum_{\text{Recall}_i} \text{Precision}(\text{Recall}_i)$$

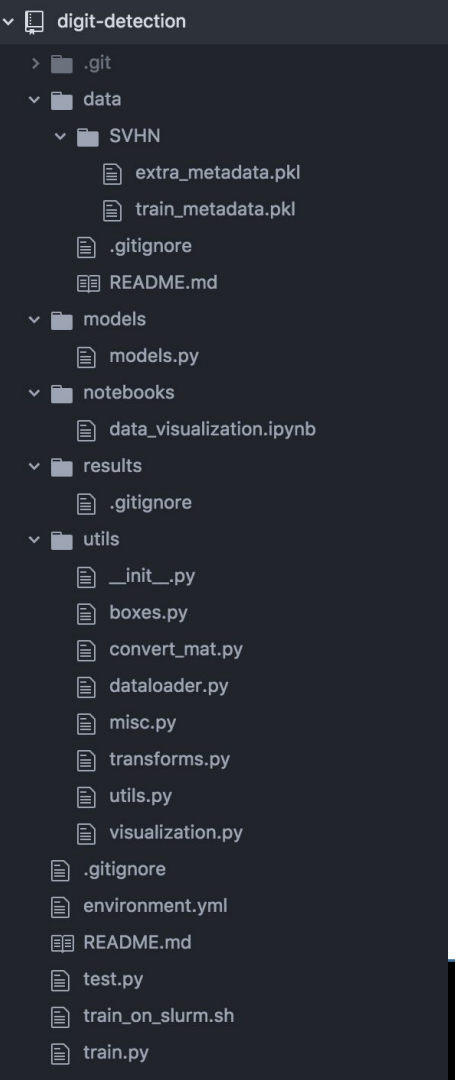
$$mAP = \frac{1}{N} \sum_{j=1}^{N} AP(IoU = j/N))$$

Scores used in object detection (implementation)

- All of these metrics are already implemented in pycocotools and in the maskrcnn-benchmark repo for the coco dataset. Use the same metrics!

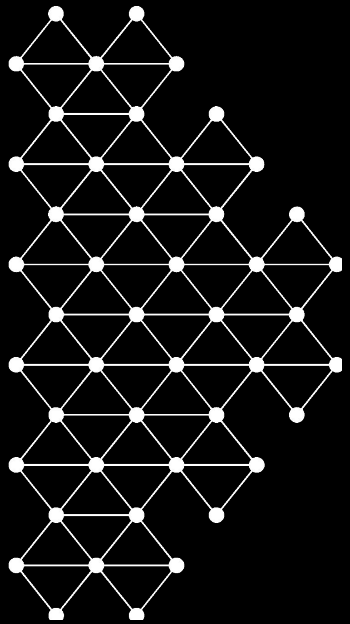
Official evaluation metrics

- Sequence length accuracy (for block 1)
  - # correct sequence length predictions / total # sequences
- Sequence transcription accuracy (for block 2)
  - # correct sequences / total # of sequences
- Sequence transcription accuracy (for block 3)
  - # correct sequences / total # of sequences



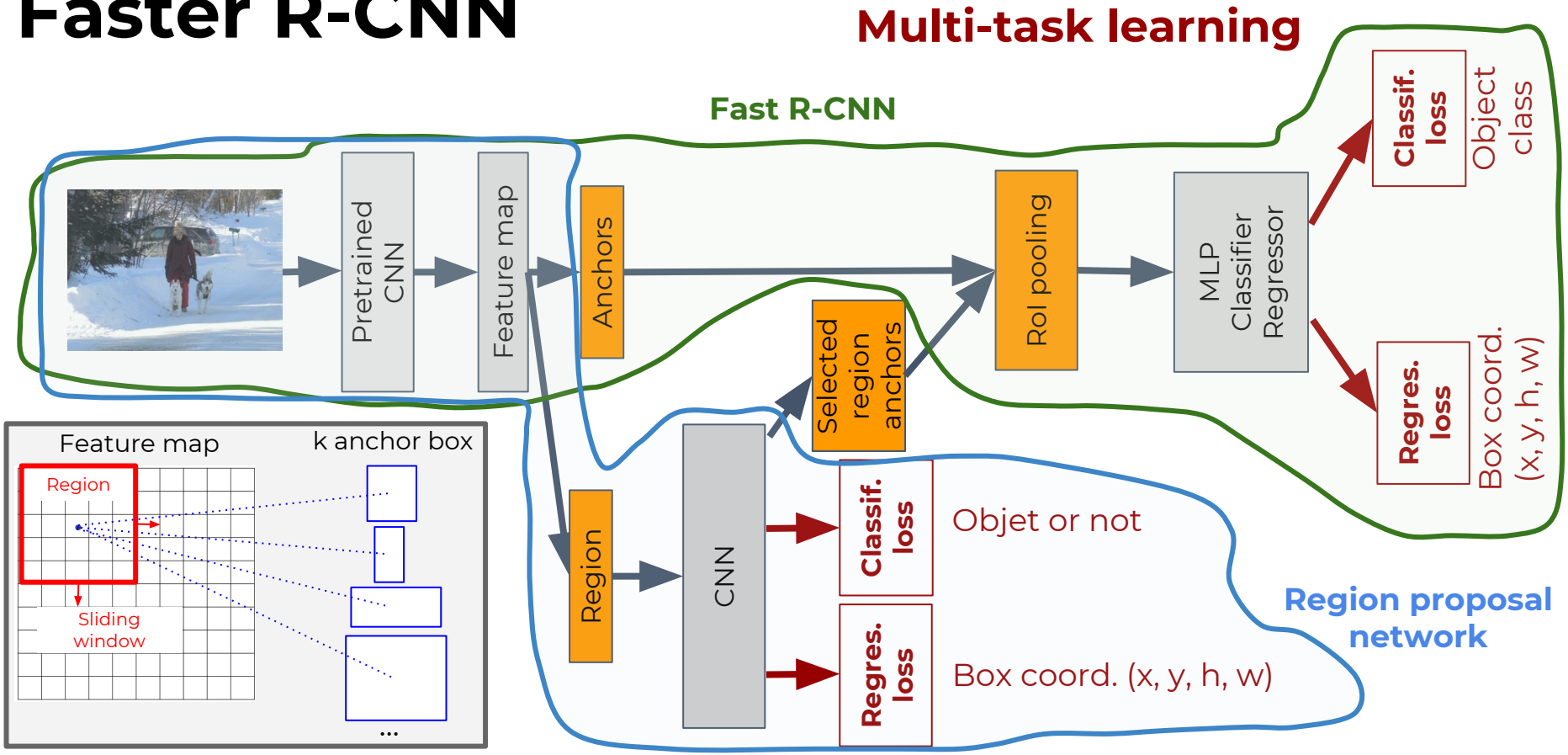
How the code should be organized

- 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)



Reminder

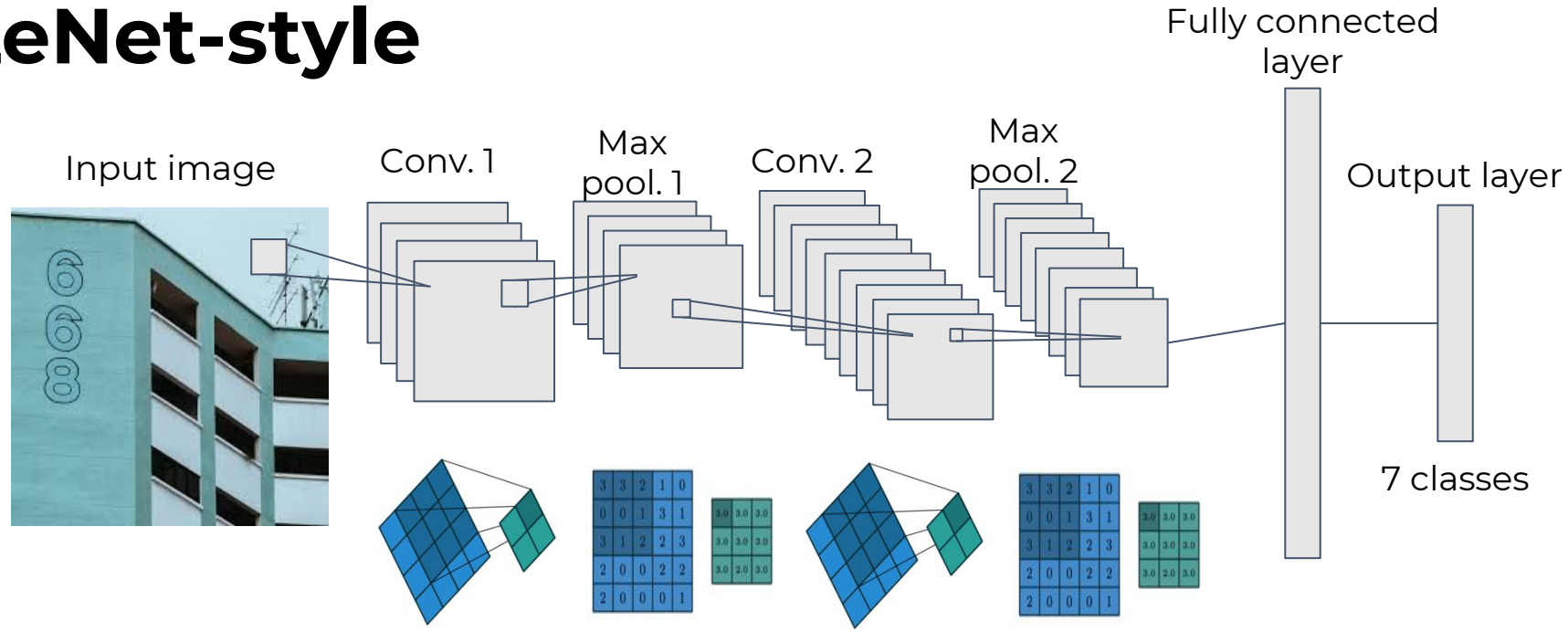
Faster R-CNN



Faster R-CNN (2016) : https://arxiv.org/pdf/1506.01497.pdf
https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8

Mila

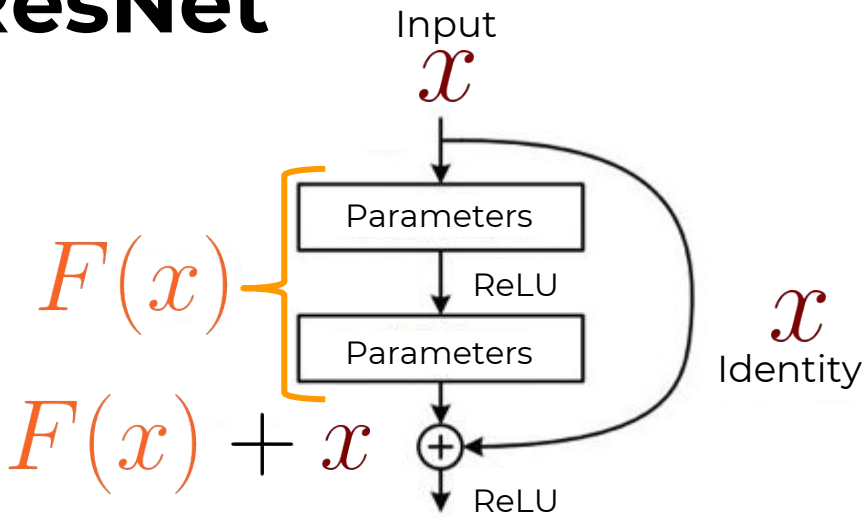
Model example: LeNet-style



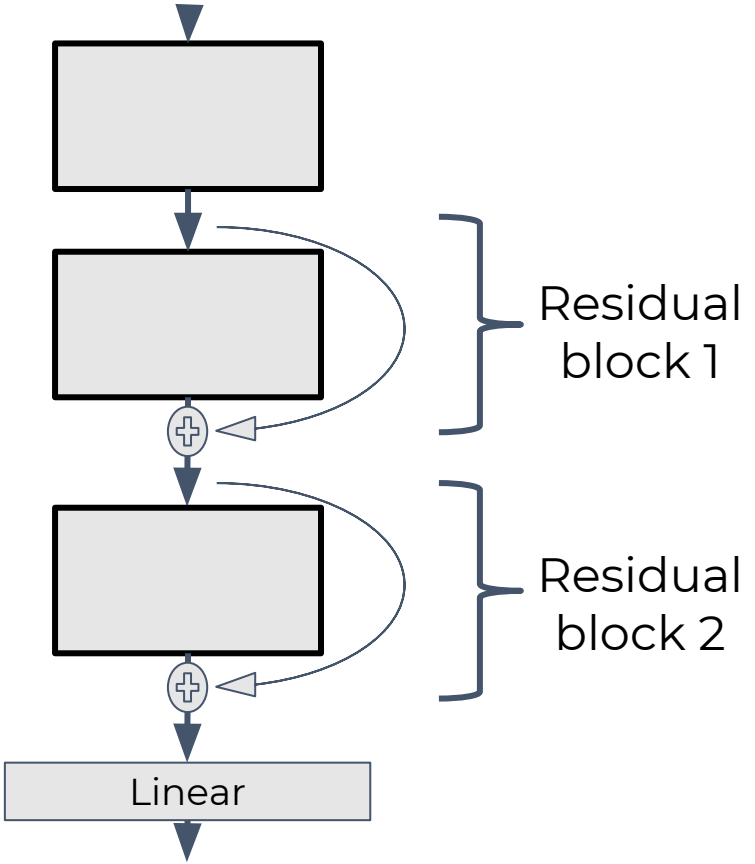
LeNet (1998) : http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

Mila

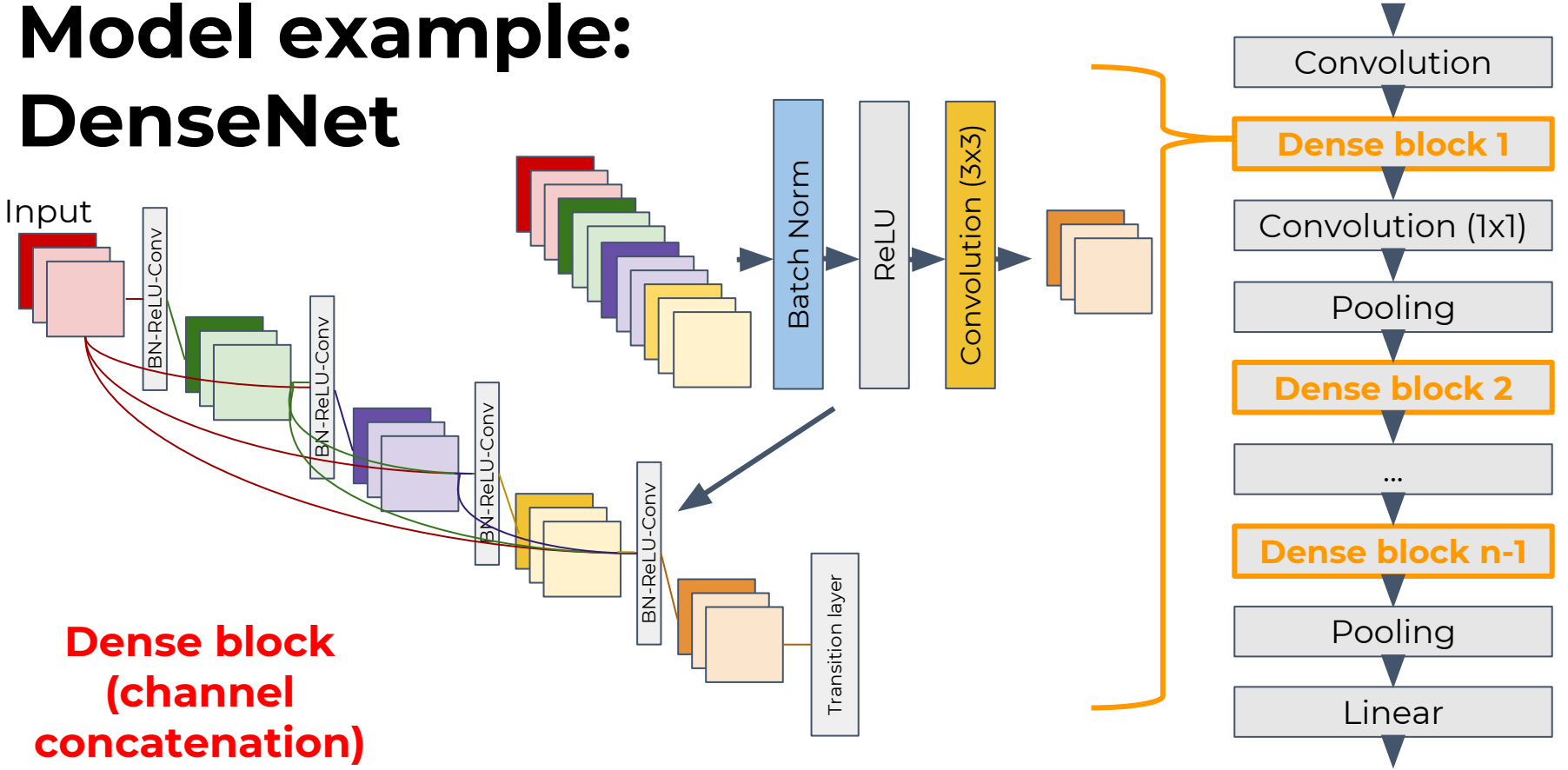
Model example: ResNet



Residual block (element-wise addition)



Model example: DenseNet



Dense block (channel concatenation)

Other examples :

- AlexNet (2012) : http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf
- VGG (2014) :

  https://arxiv.org/abs/1409.1556
- Inception (2014) :

  https://arxiv.org/abs/1409.4842
- ...

Implementation example

Very often the code already exists!

```
import torchvision.models as models
import torch.nn as nn

# ResNet
resnet18 = models.resnet18(pretrained=False)
resnet34 = models.resnet34(pretrained=False)
resnet50 = models.resnet50(pretrained=False)
resnet101 = models.resnet101(pretrained=False)
resnet105 = models.resnet152(pretrained=False)

# DenseNet
densenet121 = models.densenet121(pretrained=False)
densenet169 = models.densenet169(pretrained=False)
densenet161 = models.densenet161(pretrained=False)
densenet201 = models.densenet201(pretrained=False)
```

Think about redefining the output layer!

```
model = models.resnet18(pretrained=True)
num_ftrs = model.fc.in_features
num_classes = 447
model.fc = nn.Linear(num_ftrs, num_classes)
```