Institut
québécois
d'intelligence
artificielle

Mila

Block 2
Door number detection project

Margaux Luck, PhD
Jeremy Pinto

Door number detection 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

Mila

Data

The Street View House Numbers (SVHN) Dataset:

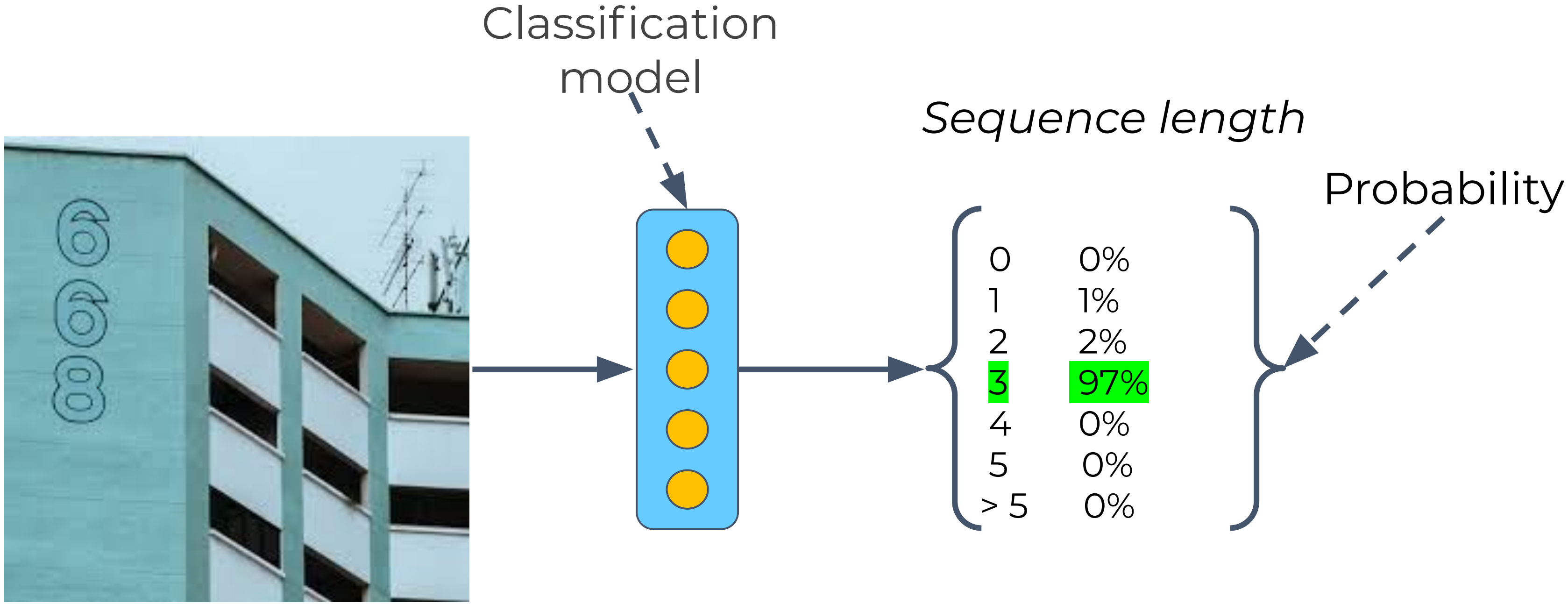
- Open-source
- ~200k street numbers
- bounding boxes and class labels for individual digits, giving about 600k digits total

Limitation:

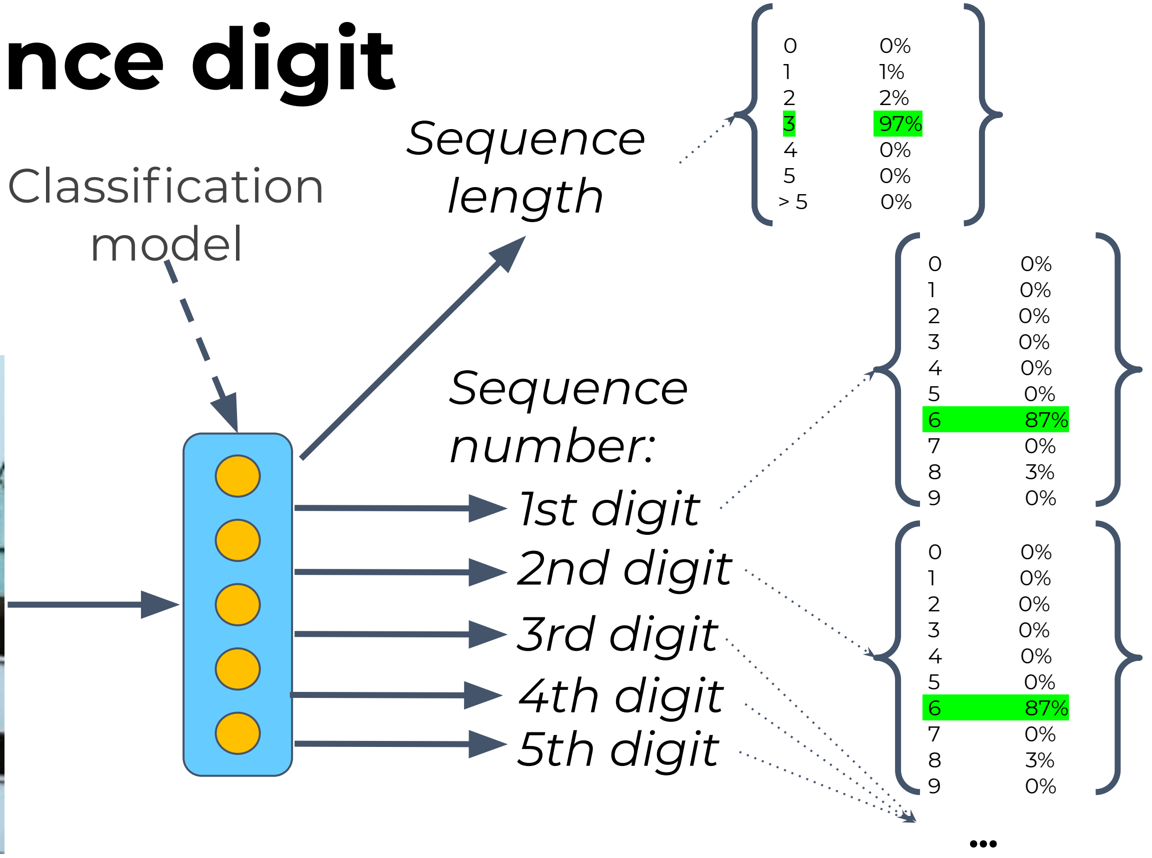
- zoom on the numbers, lack of background
- no negative examples (i.e. no images without numbers)

[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.]

Block 1: Sequence length classification

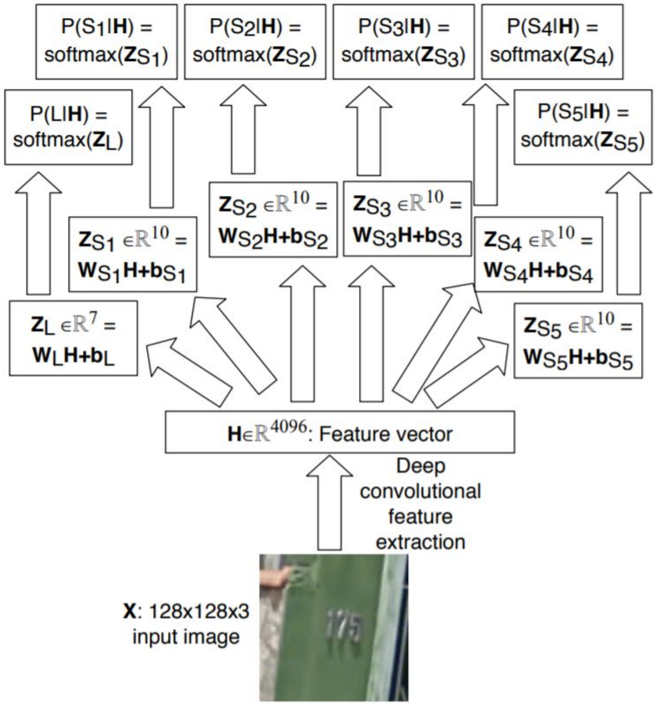


Block 2: Sequence digit recognition



Block 2: Sequence digit recognition

Sequence numbers (TODO)



Sequence length (Already done)

Block 2 instructions / expected timeline

	Feb 13 & 15	Feb 20 & 22	Feb 27 & Mar 1	Mar 13 & Mar 15
Tasks / Homework	• Read the [Goodfellow et al. 2013] article • Read and understand the code and reports from block 1	• Implementation of the multi-task learning loss	• Implementation and comparison of various models • Search for hyper-parameters, use of TensorboardX • Comparative performance grids	• Write a short report summarizing the work, and results • (Peer-) Review of other teams' code
Objectives / Deliverables	• Select one code from block 1, along with its best model and report	• Multi-task learning model for the sequence digits recognition	• Find the best set of hyper-parameters, visualize the models' training, … • Best model ready (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

Evaluation grid for block 2

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

Mila

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 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

- **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. You will need to complete this script. We reserve the right to give 0 if we cannot execute your code.
- **If you use the test set labels for predictions, you will get 0. We will audit your code.**

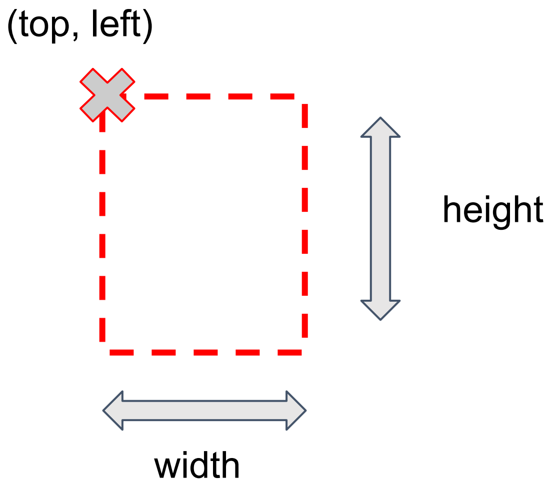
- Tips: **do not shuffle the test set**. Predictions should be made sequentially, according to their indexed order in **test_metadata.pkl**

```
digit_detection/
├── data
│   ├── README.md
│   ├── SVHN
│   │   ├── test
│   │   │   ├── 1.png
│   │   │   ├── 1.png
│   │   │   ├── 3.png
│   │   │   └── ...
│   │   ├── test_metadata.pkl
│   │   ├── train
│   │   │   ├── 1.png
│   │   │   ├── 2.png
│   │   │   ├── 3.png
│   │   │   └── ...
│   │   └── train_metadata.pkl
```

Data preparation (done during block 1)

- Data provided directly by the Mila team.
- Based on the original data found in SVHN
- Only use the data provided by Mila as we keep the test set blind. **Do not pick up the official test set available online. We reserve the right to reject a model if we have valid reasons to believe that the test set was used to build/refine the model.**
- We have formatted the original bounding box data so that it can easily be imported in python through pickle files
- Follow the instructions of Goodfellow et al. 2013 for the data preprocessing and data loader construction

Data preparation




```
with open('train_metadata.pkl', 'rb') as f:
    train_metadata = pickle.load(f)
```

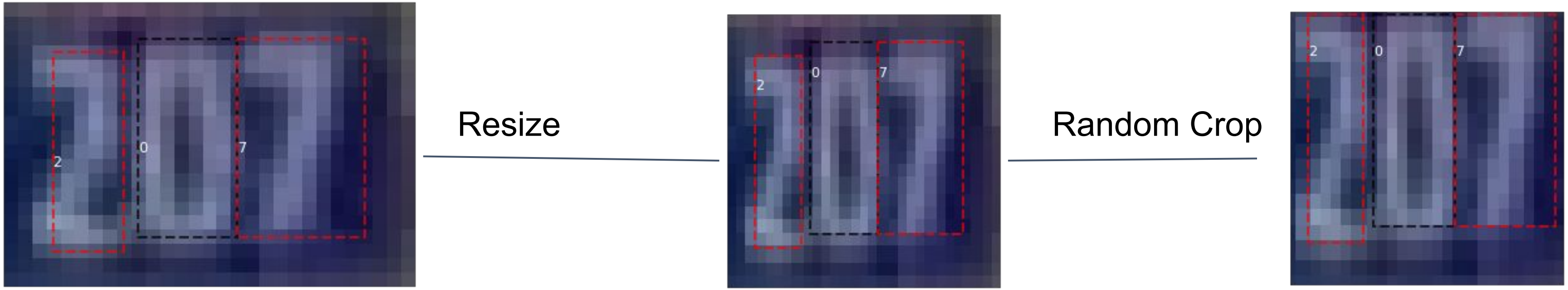
```
In [33]: train_metadata[17065]

Out[33]: {'filename': '17066.png',
          'metadata': {'height': [14, 14, 14],
           'label': [2, 0, 7],
           'left': [56, 62, 69],
           'top': [17, 16, 16],
           'width': [5, 7, 9]}}
```

Data preparation

- Data processing pipeline as explained in Goodfellow et al. 2013

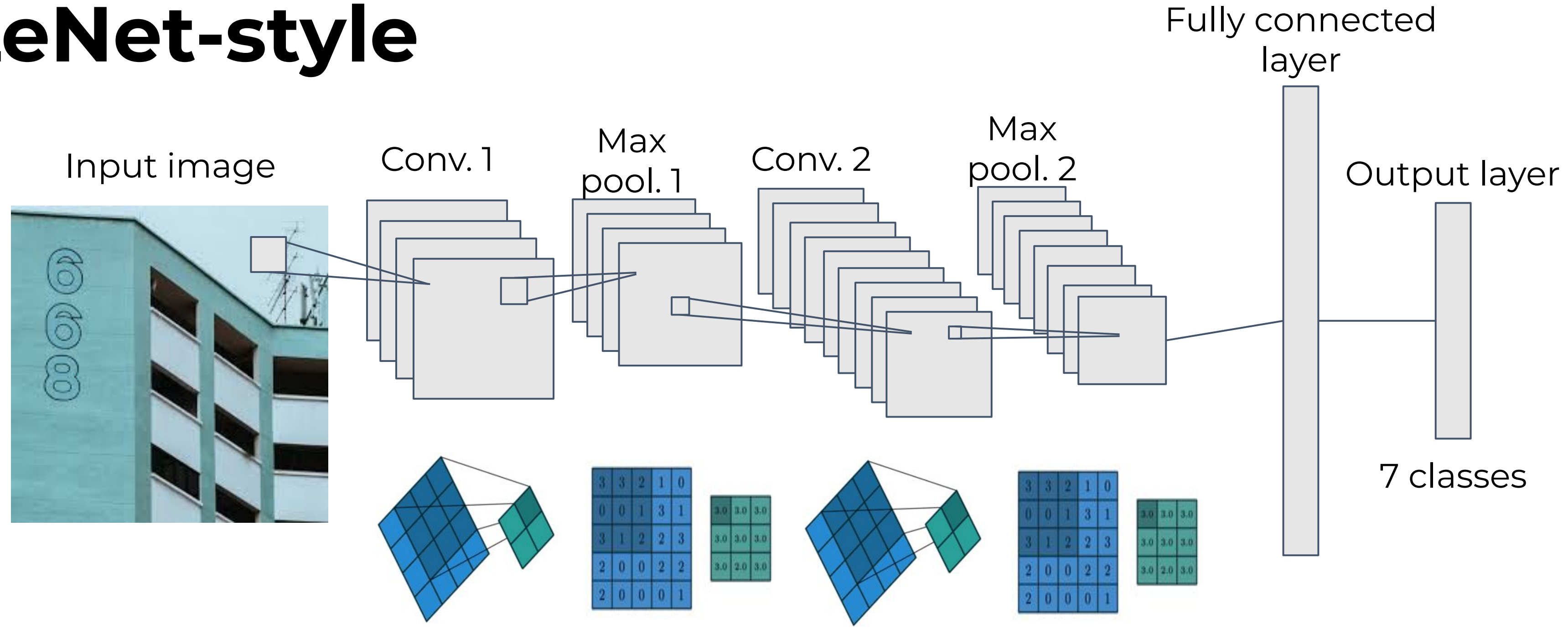
- Notice how the bounding box coordinates will change after cropping and resizing



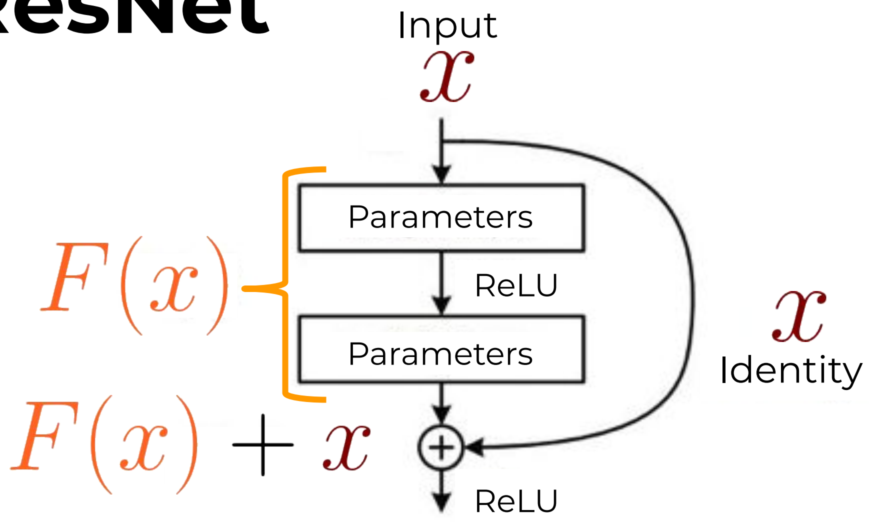
Data preparation - To keep in mind

- 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
    - …

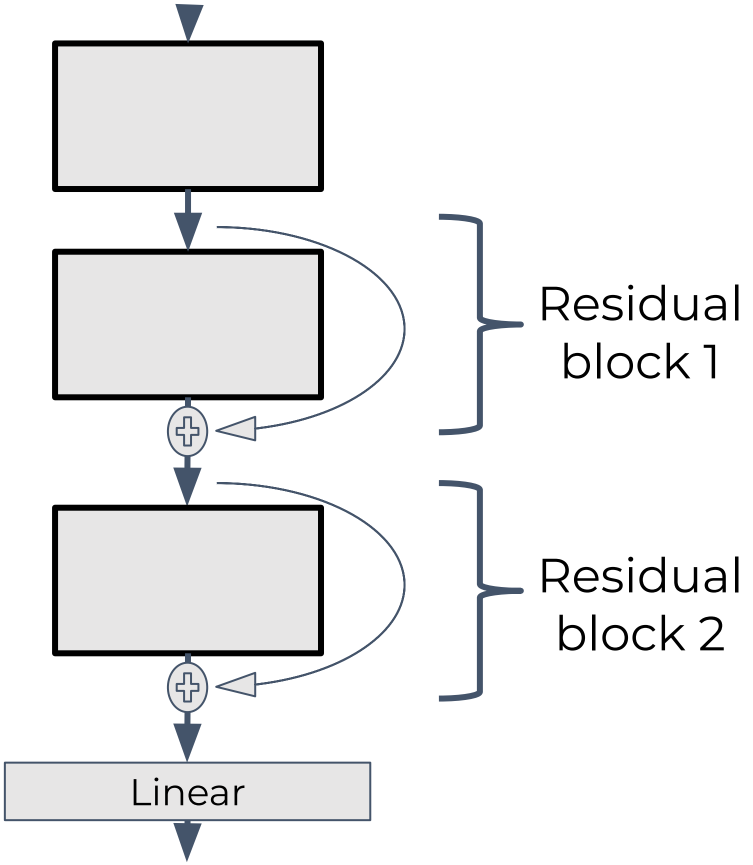
Model example: LeNet-style



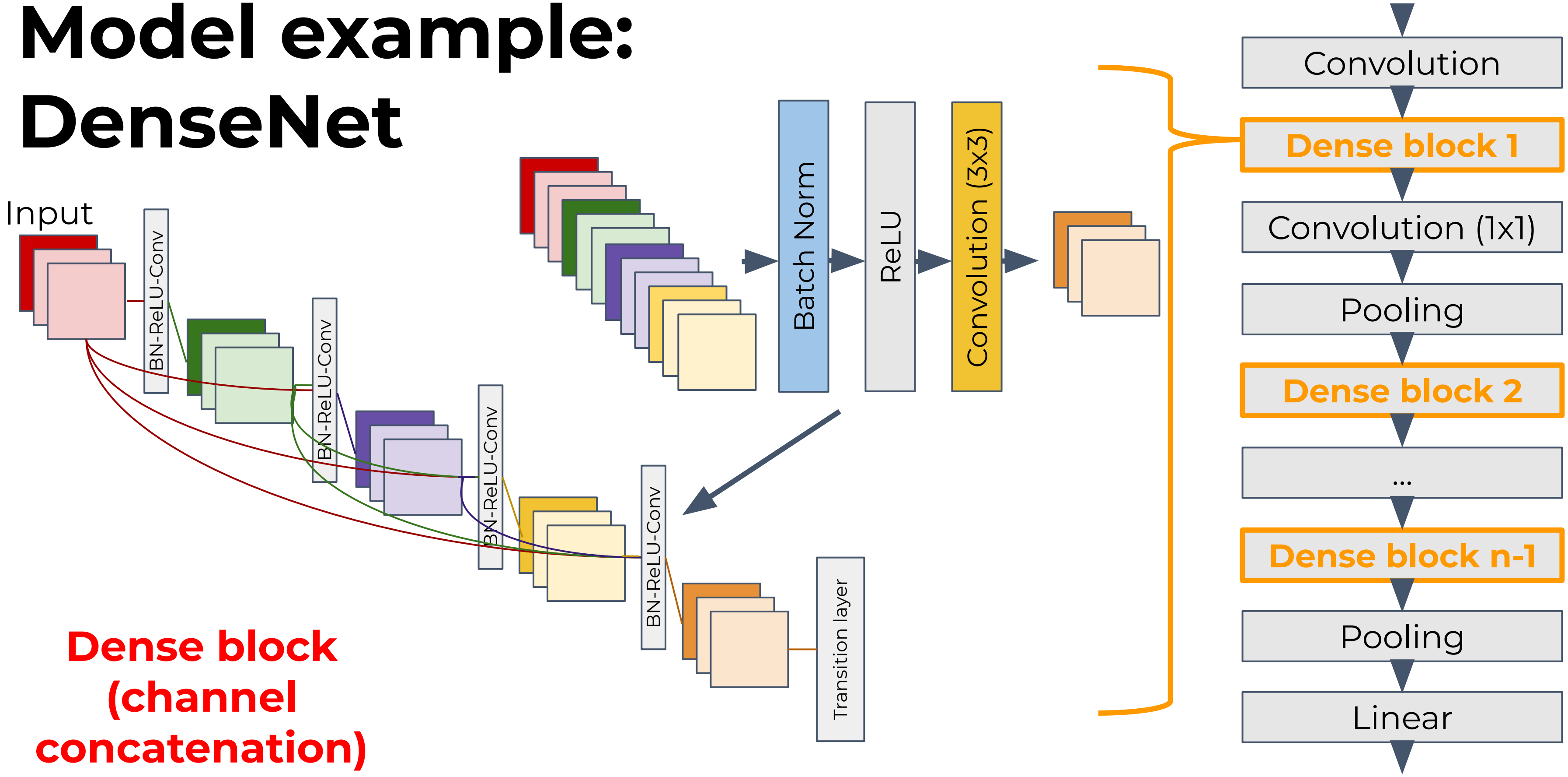
Model example: ResNet



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)
```

Official evaluation metric

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

- Sequence transcription accuracy (for block 2)
  - # correct sequences / total # of sequences

Informative evaluation metrics

- Accuracy, precision, recall, … per class
- etc.

Examples of other 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

Examples of loss functions

- Binary cross entropy = $-clog(p) - (1-c)log(1-p)$

Where

  - c = gold truth label (0 or 1)
  - p = predicted probability for label 1

- Cross entropy = $-\sum_j c_{i,j} log(p_{i,j})$

Where

  - $c_{i,j}$ = gold truth for example i / class j
  - $p_{i,j}$ = predicted probability for example i / class j

Scores

Accuracy

```
import torch

_, predicted = torch.max(outputs.data, 1)
total = labels.size(0)
correct = torch.sum(predicted == labels.data)
accuracy  = 100 * correct / total
```

F1-score

```
from sklearn.metrics import (
     accuracy_score, f1_score)

accuracy = accuracy_score(labels, predicted)
f1 = f1_score(labels, predicted,
average='macro')
```

Loss function

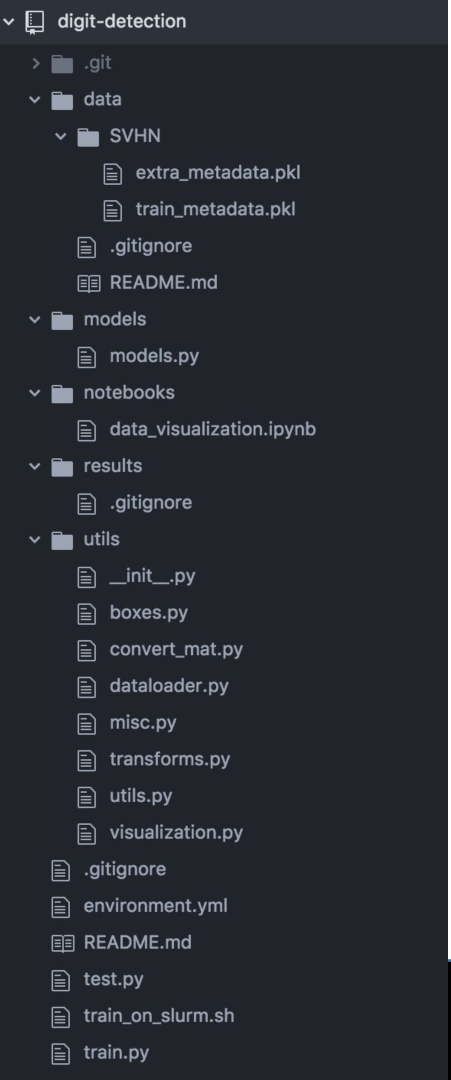
Binary cross entropy

```
import torch.nn as nn

criterion = nn.BCELoss()
loss = criterion(outputs, labels)
running_loss = loss.data[0]
```

Cross entropy

```
criterion = nn.CrossEntropyLoss()
```



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

- Training diagram for a model instance

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.

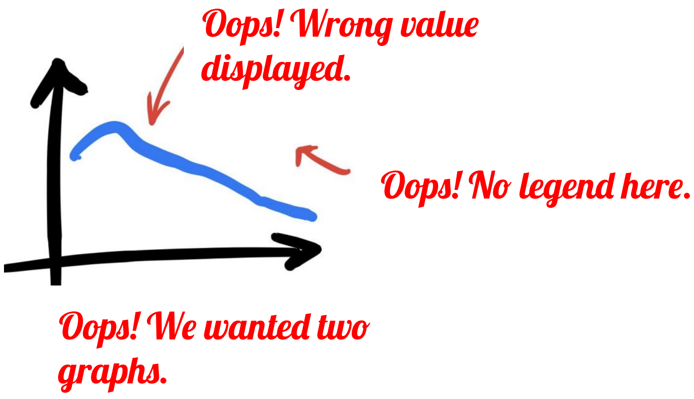
Implementation of a deep learning model

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

- Sometimes we even want to use (or start from) a pre-trained model. Search for "pretrained model" or "pretrained model weights".

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.