Universität Stuttgart
Institut für Maschinelle Sprachverarbeitung

Yixuan Xiao

Introduction to Coding Sessions

About Myself

- Studied Computer Science/AI (Beijing Institute of Technology; Uni Edinburgh)
- Competitive programmer (ICPC)
- Senior Algorithm Engineer - High Performance Computing (Baidu)
- Studied Computational Linguistics (Uni Stuttgart)
- Doing my PhD in Speech Processing – Deepfake Detection

Background

1

Programming experience (General Definition)

- Learner
  - Knowledge:
    - Know some syntax and grammar
    - can read the code, but not sure how to write
  - Problem solving:
    - Can write scripts or a single function;
    - If one’s told the exact steps of a sequential task, can code it

Programming experience (General Definition)

- Basic
  - Knowledge:
    - Know more syntax and grammars, and how to code
    - Can understand more complex code
  - Problem solving:
    - Can complete simple tasks independently
    - Might need to check the documentation from time to time

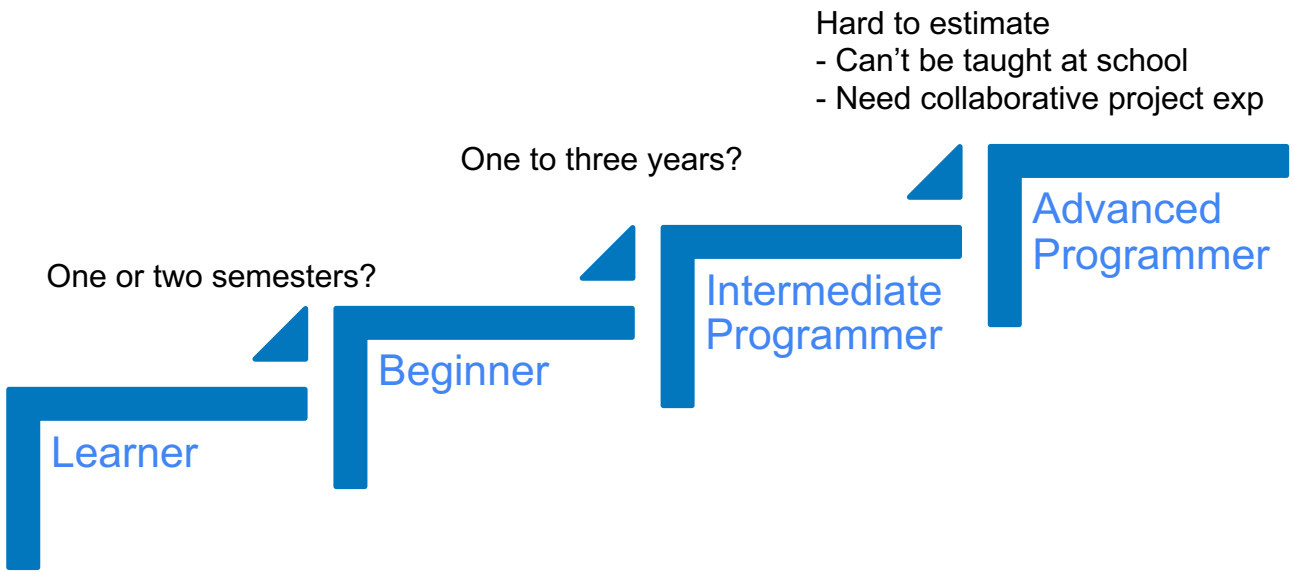
Programming experience (General Definition)

- Intermediate programmer
  - Knowledge:
    - Very familiar with syntax and grammar
    - Know more algorithms, data structures
    - Code is more organized and modularized
  - Problem-solving:
    - Given a medium-sized question, can think about the solutions independently
    - Can break down the solutions into smaller tasks and implement them
    - When encountered problem, can effectively use documentation/search engine to find solution

Programming experience (General Definition)

- Advanced programmer (Architect)
  - Knowledge:
    - Very familiar with syntax and grammar
    - Very familiar with many algorithms, data structures
    - Can write efficient code
    - Code is more organized, modularized, maintainable, and scalable
  - Problem-solving:
    - Given a complex programmer, can design and architect large systems from scratch
    - The system can be used to integrate more programmers
    - Also know who can be responsible for which part

How long does it need to level up?



Whare are expected (and what can be achieved)

Feelings	Tools	Application	Mechanism
• Be more confident in coding • Feel more comfortable with coding	• PyTorch • Training Pipeline • Linux	• Prepare better for future courses • Learn how to solve an actual task	• Know low-level implementation • Know more than calling the API • Build from scratch

Whare are expected (and what can be achieved)

Feelings	Tools	Application	Mechanism
• Be more confident in coding • Feel more comfortable with coding	• PyTorch • Training Pipeline • Linux	• Prepare better for future courses • Learn how to solve an actual task independently	• Know low-level implementation • Know more than calling the API • Build from scratch
Need more practice yourself.	Sure.	Depends on your background.	Will be difficult.

About the underlying mechanism

Will it be covered ➔ Some will be.
But is it the course expected outcome? ➔ No.

Why?

- Could be difficult. Optimized code can be counter-intuitive to learners, need to know more data structure and algorithms
    - Might exceed the expected self-study time for learners. (200/15 = **13.3 hrs/week**)

AI Coding: The good, the bad and the ugly

The good

- Studying: Explain the code for you.
- Productivity: Help you code faster.
    - When you forget some grammar, just type what you need and ask AI to complete that.
- Solutions: Fast prototyping. Test your idea at an early stage.

The bad

- Studying: more errors when encounter out-of-domain data
    - A new tool that undergoes rapid development → Documentation change frequently
    - A less-seen programming language → C + Embedded systems
    - A less-seen problem: a very specific task that require a lot of context
- Productivity: Help you code faster. → Also generate bugs faster.
- Solutions: Fast prototyping (Could be a broken prototype, especially when you don’t understand the logic fully.)

The ugly?

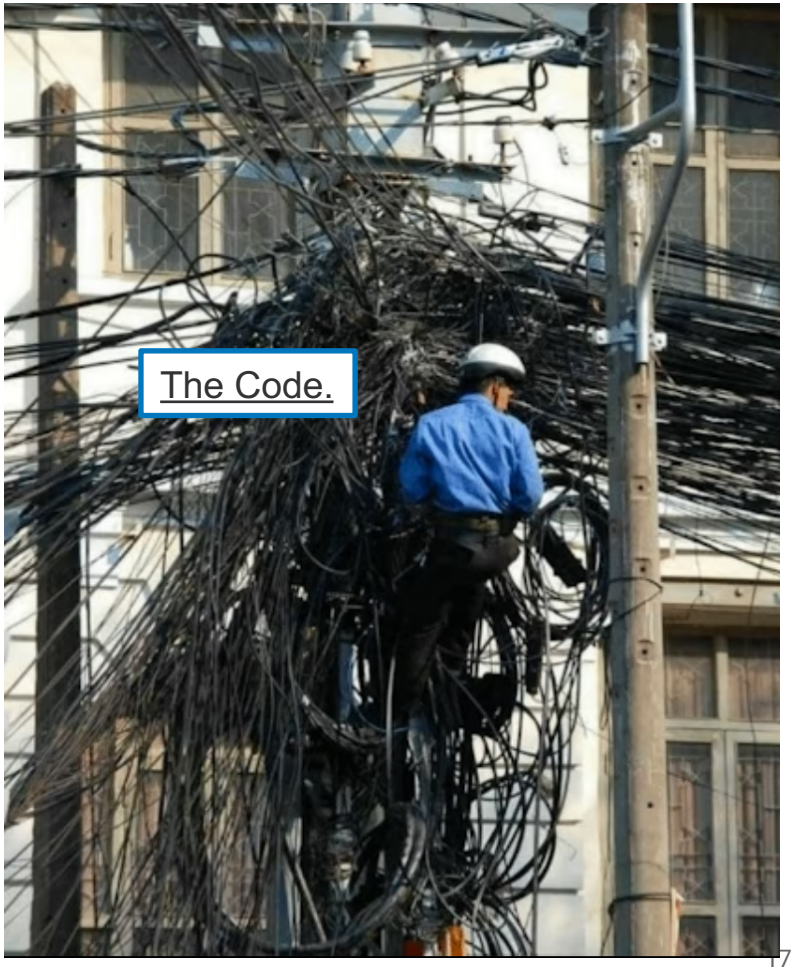
- A change of mindset: From developing code to preventing your AI agent from making mistakes.
    - You write more unit tests and AI does the actual development
- A change of skill set. Solution-oriented → Problem-oriented
    - the ability to search for information and how to verify → the ability to describe the problem and the expected outcome

What’s the point of learning coding?

- We want to be a good manager. It means
  - You understand your business: what’s the problem and the key points of the solutions
  - You know how to break down the task: design patterns, code modularization, etc
  - You know how to assign the task: evaluate the complexity of the solution and the capacity of AI
  - **You can back your team up: if the AI code fails, you know how to fix!**

DON’T DO

- Generate code by AI → Fix the bugs by AI → More bugs → Fix the bugs by AI → More bugs → ….
- “Hey, could you take a look at this error? I (and AI) couldn’t fix it.”
    - EMOTIONAL DAMAGE
- Only use AI when you can still understand what’s going on.



Summary

What the course aims to teach

- How to process data?
- How to use PyTorch to train a model?
- Prepare you for future courses, e.g., TeamLab, Master's Thesis

What to do during self-study time

- Practice so you feel comfortable with coding
- Read the documentation and other materials to practice (since they have example codes)

Session Preview

Could change based on the learning progress

2

Basics in PyTorch I&II

- (Code) Tensor operations
- (Theory) Training pipeline: the relation among (Dataset, DataLoader, Model, Loss)
- (Code) How to prepare a dataset (with Pandas)
- (Theory) A Training Loop

PyTorch – Neural Nets

- (Theory) Parameter Tensors and Normal Tensors
  - Computation Graph
- (Code) Build a NN
  - API
  - (Optional) Low-level implementation
- (Code) Recap: A training loop

PyTorch – RNN/LSTM

- (Code) Build a one layer RNN/LSTM
  - API
  - (Opt) Low-level implementation
- (Code) Masking
- (Code) Train a simple LSTM

PyTorch – Seq2Seq & Attention

- (Code) Different attention score calculation
- (Code) Seq2Seq implementations
- (Code) Train a simple seq2seq model

PyTorch – Tricks

- To be determined

PyTorch – Transformation

- (Code) simple QKV implementation
- (Code) API
- (Code) stacking layers
- (Code) Train a simple one-layer transformation

PyTorch – Fine-tuning Large Models

- To be determined

Now Let’s Code

3

Training Pipeline

4

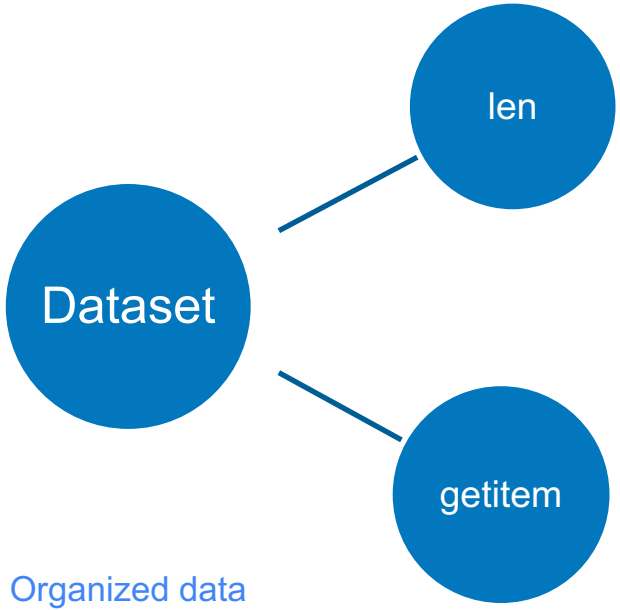
Four important modules:

Dataset	•Represent a dataset •Process a single sample
DataLoader	•Group samples into batches •Handle how to pad samples
Model	•The core algorithm •Input: batch •Output: logits, scores, ...
Loss	•The loss function •Input: **label**, model's output

Dataset

Raw data

- A folder having many files
- A csv file



Organized data
`torch.Dataset`

- total number of samples
- fetch one sample from the dataset

Dataset - Outcome

Indix	List[Sample]
0	Audio path, Label: 1
1	Audio path, Label: 0
2	Audio path, Label: 0
3	Audio path, Label: 1
.....	

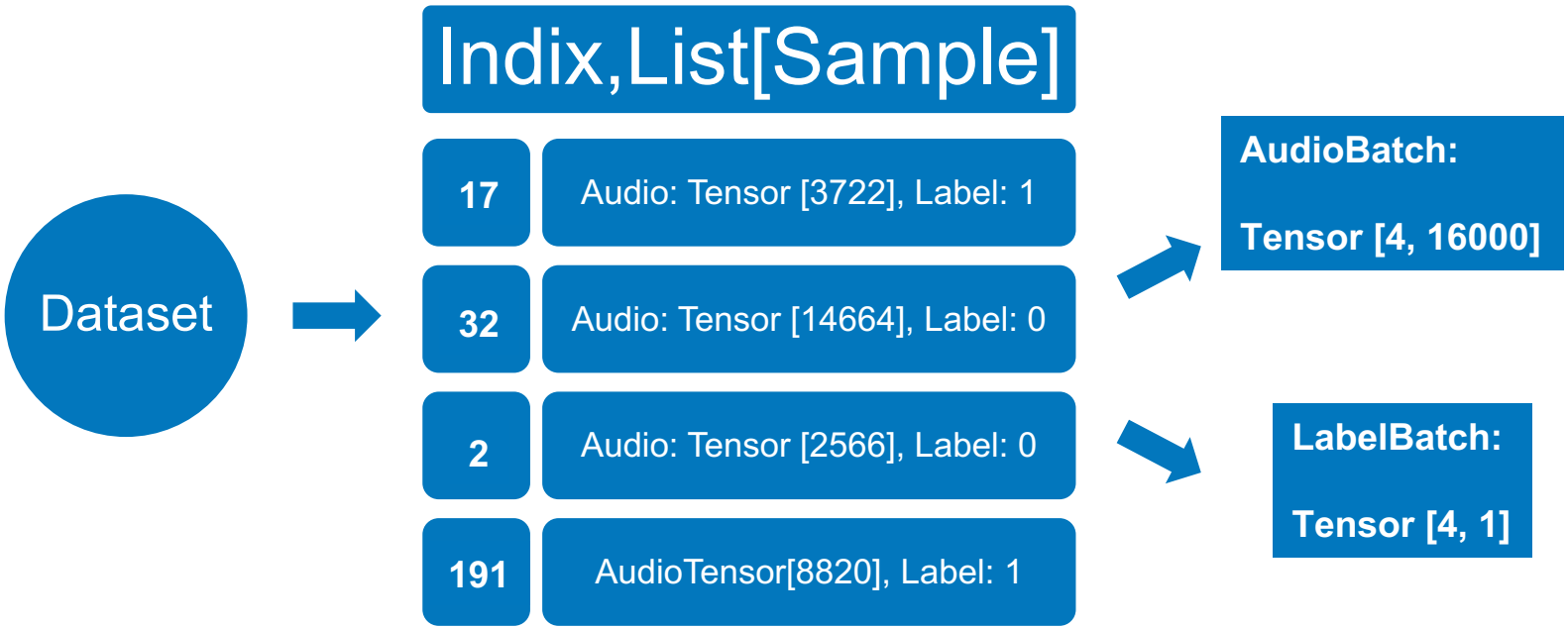
- `len()`: return the total amount of samples
- `__getitem(index)__`: given the index, process the data.

- E.g.,

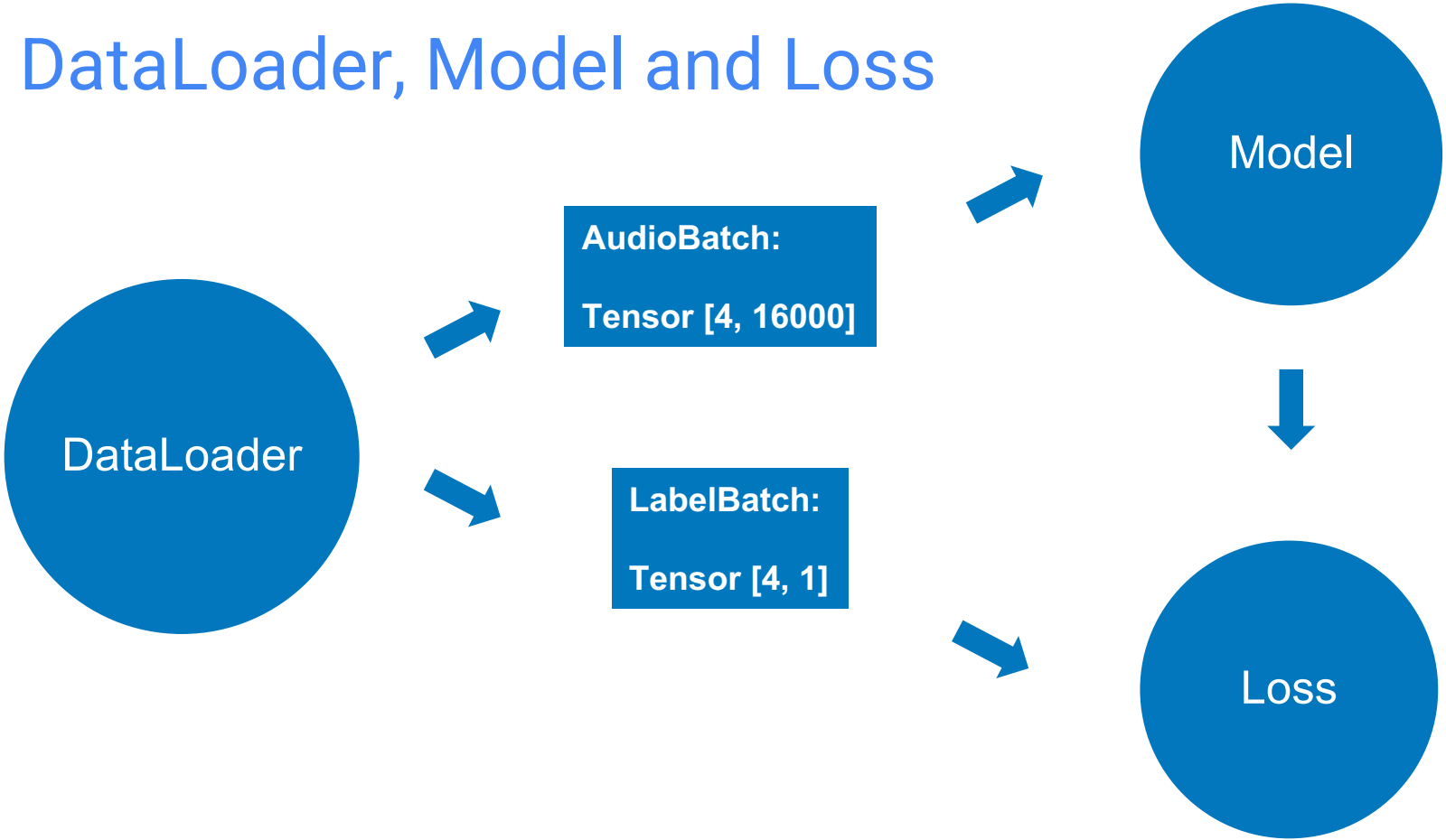
AudioTensor[8820], Label: 1

DataLoader: collate_fn

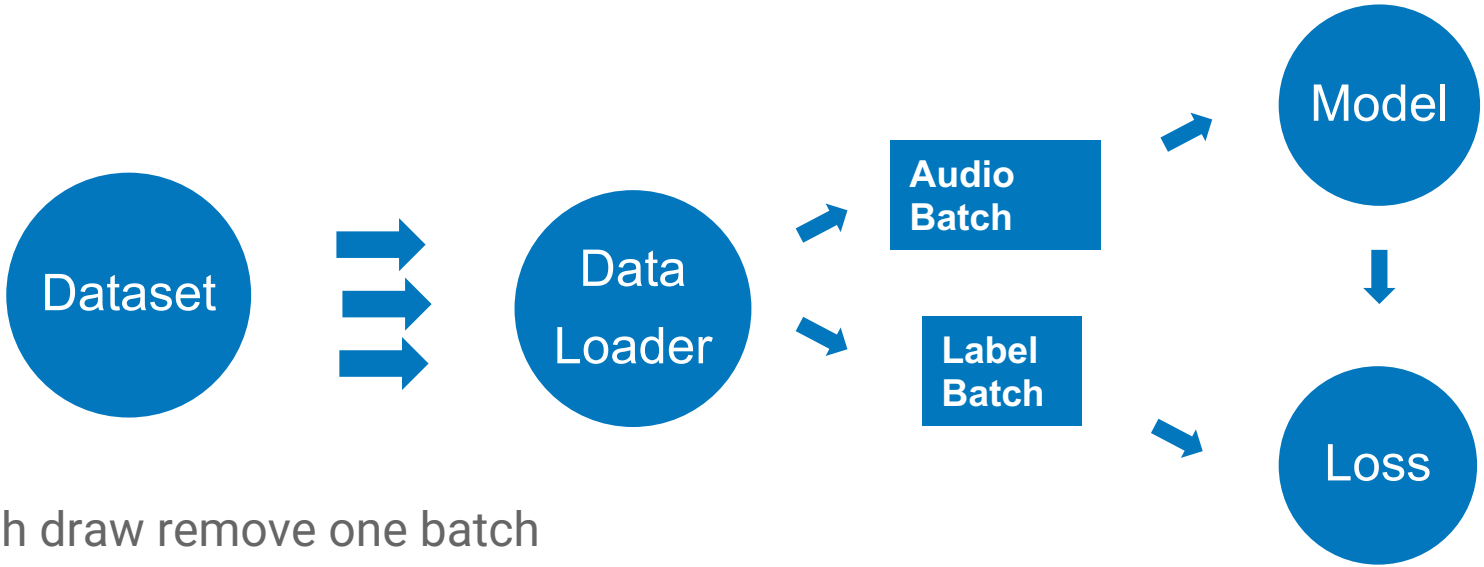
Collate_fn(batches) -> List[Tensor]



DataLoader, Model and Loss



A training loop



- Each draw remove one batch
- Keep pooling batches till all batches run out