Introduction to Deep Learning (I2DL)

Tutorial 11: NLP & Transformers

Today's Outline

- **Exercise 10 Review**
  - semantic segmentation, and the #1 submission
- **Natural Language Processing**
  - tokenization, datasets, collators, and the transformer
- **Exercise 11 & 12**
  - build a transformer from scratch

Exercise 10 Review

Exercise 10: Leaderboard

Leaderboard

The leaderboard shows for each exercise the highest scoring submission from each user. Only valid submissions are displayed.

Exercise 1 | Exercise 3 | Exercise 4 | Exercise 5 | Exercise 6 | Exercise 7 | Exercise 8 | Exercise 9 | Exercise 10 | Exercise 11 | Exercise 12

#	User	Score
1	u1449	95.95
2	u0541	95.73
3	u0994	95.31
4	u0973	95.01
5	u1503	94.96
6	u0566	94.75
7	u0935	94.54
8	u1223	94.50
9	u1314	94.41
10	u1258	94.28

The #1 Submission: U-Net + Transformer Encoder

- **Architecture:** U-Net
- **Encoder:** mit_b0 (a pretrained **transformer** backbone), weights = imagenet
- **Decoder:** convolutional U-Net decoder with skip connections. (192, 128, 64, 32, 16, 23)
- **Input:** 3×240×240, normalized with imagenet mean and std
- Encoders tried: mobilenet-v2, mobilenetv3, efficientnet-b0 → mit_b0 won

Training Recipe — Exact Settings

- **Loss:** 0.5 · Dice + 0.5 · cross-entropy (ignore_index = −1 for void)
- **Optimizer:** adamw — lr 1e-4 (encoder) / 1e-3 (decoder), weight decay 1e-4
- **Schedule:** cosine annealing to eta_min 1e-6 over 100 epochs, batch size 16
- **Freeze:** encoder frozen for the first 10 epochs, then unfrozen
- **Augment:** h-flip p=0.5, affine (±25°, scale 0.85–1.15), color jitter, grayscale p=0.05
- **Data:** trained on train + val + test pooled together (You should not do this in practice.)

Squeezing Out the Last Points: SWA + TTA

- **Stochastic Weight Averaging:** 10 extra epochs, adamw lr 1e-6 (enc) / 1e-5 (dec), then batch-norm update
- **Test-time augmentation:** average of original + horizontal flip + multiple scales
- **Result:** 95.95 — top of the leaderboard

Natural Language Processing

How Do We Feed Text to a Model?

- **So far, feeding data in was clear:**
  - **Tabular** → load each row as a vector and normalize
  - **Images** → a matrix for convolutions, or a flat vector for linear layers
- **Text** → “what do we do with this?”

Tokenization

- Split text into individual **tokens**
- Tokens can be **words**, **subwords**, **letters**, or **punctuation** — or a mix
- Assign each token an **ID**: “Hi” → 53, “Bye” → 647

Tokenization — Training

- **Simple tokenizer:** split on whitespace, add each new word to a dictionary with an ID
- **Problem:** words never seen during training
- **Fixes:** an unknown-token placeholder, or a smarter **Byte-Pair Encoder**
- → Exercise 11, Notebook 1

Map-Style Datasets

- The datasets so far: **given an index, return that item**
- **Great for** small tables and image-path lists — easy to prefetch, shuffle, parallelize
- **Problem:** huge text files that don't fit in memory, or a live **stream**

Iterable Datasets

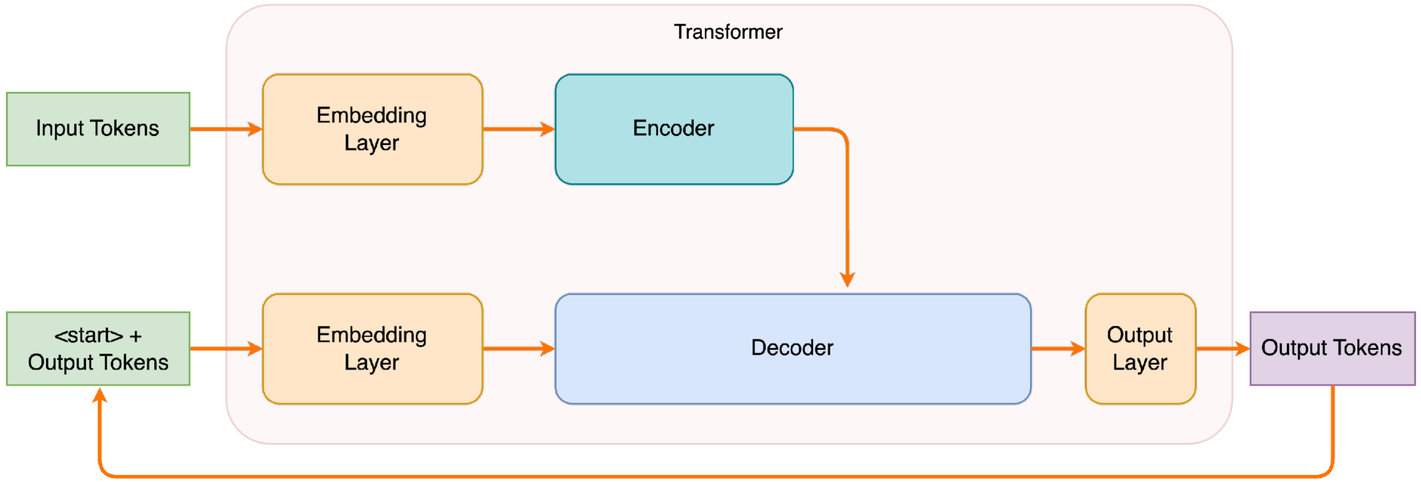
- Do we really need an index? We just want the **next sample**
- Implement **__iter__** instead of __getitem__ — read line by line and **yield** each sample
- **Less memory**, cycles across files; but **no shuffling** and harder to parallelize
- → Exercise 12 (Datasets)

Data Collator

- Batching needs equal dimensions — but **sentences differ in length**
- **Solution:** pad the shorter sentences with a **<pad>** token to match
- [“Hi” “how” “are” “you” “?”] → 5 tokens, [“Good” “and” “you” “?” <pad>] → padded to 5
- This lives in the **data collator**, inside the data loader
- → Exercise 12 (Datasets)

Transformers — High Level

- Token IDs → **embeddings** → the **encoder** processes the input sequence
- The **decoder** starts from a start token, and predicts the next token from the encoder output and what it has produced so far
- The **output layer** gives a distribution over the vocabulary — pick a token, append it, and loop until the end token



Exercise 11 & 12

Exercise 11: Content

- **Notebook 1 — Tokenization**
  - byte-pair encoding; **train your own tokenizer** on a dataset
- **Notebook 2 — Embeddings**
  - turning token IDs into dense **embedding** vectors

Exercise 11: Content and Goal

- **Notebook 3 — Attention**
  - the **attention mechanism**, then **multi-head attention**
- **Notebook 4 — Positional Encoding**
  - giving the model a sense of token **order**
- **Goal:** implement the transformer's core **building blocks** from scratch

Exercise 12: Content and Goal

- **Notebook 1 — Dataset & Collator**
  - an iterable dataset with a padding collator for the translation data
- **Notebook 2 — The Transformer**
  - assemble the encoder and decoder into the full model
- **Goal:** train it to **translate English → German**

Good luck with the exercise and the exam!