Assignment 4

Deadline: 20.01.2025

By the deadline, you are required to submit the following:

Report. Prepare a single PDF file that presents clear and concise evidence of your completion of each of the listed tasks. **Use the overleaf template available on iCorsi**. **Be sure to write a report of no more than 6 pages**.

Source code. A **jupyter notebook with output using the template available on iCorsi**, performing each of the specified tasks. If a task is accomplished in the code but not documented in the report, it will be considered incomplete. Therefore, make sure to report all your work in the submitted report thoroughly. **For this assignment, python files will not accepted**.

General hints. The question marked with * is more challenging, and we recommend possibly leaving it as the last question to solve. To obtain the maximum grade, not only should all exercises be completed correctly, but the plots must also be clear, have a legend, and have labels on the axes and the text should be written in clear English. For saving plots in high quality, consider using the command `matplotlib.pyplot.savefig`. **Clarity of plots/text and code will account for 5 points each**. Best practices for coding include commenting on each function and commenting on the code using the standard rules for Python.

Submission rules: As already discussed in class, we will stick to the following rules.

- Code either not written in Python or not using PyTorch receives a grade of 0.
- Submission after the deadline will not accepted
- If plagiarism is suspected, TAs and I will thoroughly investigate the situation, and we will summon the student for a face-to-face clarification regarding certain answers they provided. In case of plagiarism, a score reduction will be applied to all the people involved, depending on their level of involvement.
- If extensive usage of AI tools is detected, we will summon the student for a face-to-face clarification regarding certain answers they provided. If the answers are not adequately supported with in-person answers, we will proceed to apply a penalty to the evaluation, ranging from 10% to 100%.

- GPU usage is recommended for this assignment

Heuristic for TSP Using Transformers

The Traveling Salesman Problem (TSP) is a classic optimization problem in computer science and mathematics. It involves finding the shortest possible route (tour) for a salesman to visit a given set of cities exactly once and return to the starting point. Despite its simple formulation, TSP is computationally challenging and classified as NP-hard, making it an excellent candidate for heuristic and approximation methods.

In this assignment, you will develop a heuristic solution for TSP using transformer models. Originally designed for natural language processing, transformers have demonstrated a certain versatility in diverse tasks.

Your task includes understanding, coding, evaluating, and critiquing your heuristic model. This involves encoding the problem into a format suitable for transformers, training the model, and analyzing its performance compared to traditional approaches.

Dataset (20 pts)

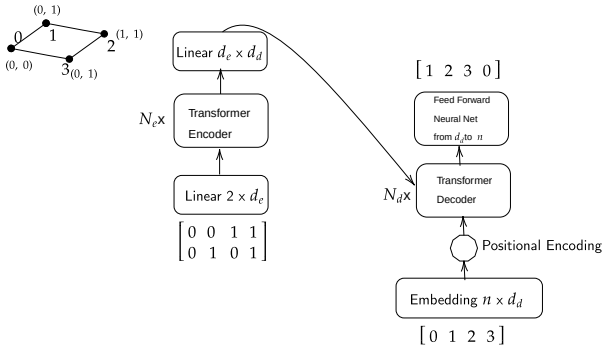
For this assignment, we will use data generated in [1], which contains *Euclidean TSP instances*. The dataset is created by randomly generating points using a uniform distribution in the unit square and deriving a graph from these points. The data is pre-processed and divided into training, validation, and test sets, with a fixed number of nodes $n = 20$ for simplicity. Additionally, a smaller dataset, `dummy`, is provided for preliminary testing. We recommend starting with this dataset to validate your pipeline before scaling up. You will find all these datasets in pickle format, in the folder `data`

- (1 pt) Load the `dummy` dataset, get a single data item and explain its Python type.
- (5 pts) Describe the edge attributes `tour` and `weight`, as well as the node attribute `pos`.
- (10 pts) Implement a dataset class. Focus on the `__getitem__` method to return:
  - X: A tensor of node coordinates with size 20×2.
  - y: A tour starting from 0 and ending with 0.
- (4 pts) Create `Dataset` objects for training, validation, and testing, along with their respective `Dataloader`.

Model (35 pts)

You will implement a transformer-based architecture for this task. As discussed in previous exercises, the original transformer architecture [2] can be adapted for different tasks.

- (15 pts) Implement the following architecture:



To be even more clear, just note that here the small example has $n = 4$.

- (10 pts) Explain where and why masking is used.
- (5 pts) Explain why positional encoding may be omitted in the encoder.
- (5 pts) Explain why positional encoding is necessary in the decoder.

Training (25 pts)

Training transformers is challenging, so our goal is to train the model for 10 minutes (or any reasonable time budget you want to invest) without overfitting.

Standard Training (10 pts)

Train your model using a standard pipeline. Ensure you shift y appropriately as shown in the model diagram. Report the training and validation loss, demonstrating no overfitting. As always, choose on how many steps you want to report your loss.

Training with Gradient Accumulation (15 pts)

Train your model using gradient accumulation. Decide on the number of accumulation steps and comment on the differences compared to standard training. Report the training and validation loss, ensuring no overfitting. As always, choose on how many steps you want to report your loss.

Testing (15 pts)

Test your model on the test set and compare it with two baselines:

- `random_tour`: Samples a random tour.
- `greedy_algorithm`: Starts from a node and selects the closest unvisited node iteratively.

Both baseline functions are provided in the template. Calculate the optimality gap for each instance using:

$$\text{gap} = \frac{\text{Approx} - \text{OPT}}{\text{OPT}}$$

The `compute_gap` function is pre-implemented.

- (10 pts) Explain the function `transformer_tsp`, which uses the model you trained to predict a tour. You should notice that it's close to the greedy sampling strategy we used in NLP task. Which is the main difference?
- (5 pts) Plot four boxplots showing the gap distribution for random tours, the greedy algorithm, and the model. Test both the model trained with gradient accumulation and with standard training.

Critique (5 pts)

Critique the limitations of the model, focusing on:

- Model architecture.
- Dataset size and diversity.
- Generalizability to larger graphs (e.g., 30 or 100 nodes) or graphs without coordinates.

You may also discuss:

- Model size.
- Hyperparameter choices.
- Any other relevant factors.

References

[1] Chaitanya K Joshi et al. "Learning the travelling salesperson problem requires rethinking generalization". In: *arXiv preprint arXiv:2006.07054* (2020).

[2] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).