Exercise Sheet 11: Transformers

Due on 05.07.2024, 10:00

Stefan Baumann (stefan.baumann@lmu.de)

Important Notes:

1. **Email**: Frequently check your email address registered for Moodle. All notifications regarding the course will be sent via Moodle.
2. **Due date**: The exercise sheets will usually be uploaded 1 week in advance of the due date, which is indicated at the top of the exercise sheet.
3. **Submission**: Put your code and your report (a single PDF file) inside a single ZIP file. Both PDF file and ZIP file should contain your surname and your matriculation number (e.g. *Surname-MatriculationNumber.zip*). You may use Jupyter[1] for exporting your PDF, but you still have to hand in a .py file for us to test your code. **Submissions that fail to follow the naming convention will not be graded.**

General Information:

All programming exercises must be completed in Python. The proposed solution for the exercises will be compiled in Python 3 (3.8 or above). You may use standard Python libraries or Anaconda (open source distribution for Python) to complete your exercises. We will be using PyTorch[2] as our main deep learning framework. This exercise will *optionally* also use einops[3].

If you have any problems or questions about the exercise, you are welcome to use the student forum on the lecture Moodle page, as most of the time other students might have similar question. For technical issues about the course (for example, in case you cannot upload the solution to Moodle) you can write an email to the person responsible for the exercise (indicated at the top of the exercise sheet).

[1] https://jupyter.org/
[2] https://pytorch.org/
[3] https://einops.rocks/

Task 1: Self-Attention **(6P)**

In this task, you will implement the self-attention mechanism commonly used in Transformer models, yourself. In the Jupyter notebook `attention.ipynb` accompanying this PDF, you will be provided with the skeleton of a class `SelfAttention2d` that performs self-attention on a 2d sequence of tokens (as in, e.g., a Vision Transformer).
As in [Vaswani et al., 2017], attention is defined as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{1}$$

which is then extended into multi-head attention as

$$\begin{aligned}\text{MultiHeadAttention}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V).\end{aligned} \tag{2}$$

In 2d self-attention, every token will attend to every other token in the whole 2d sequence.
The following steps are required:

- Add the correct missing modules in the module's `__init__()` method. (1P)
- Implement the multi-head self-attention forward pass in the `forward()` method. (3+2P)
  - Correct implementation in the edge case of `num_heads=1` (3P)
  - Correct implementation of multi-head attention (2P)

It is expected that the students implement the attention logic themselves. No points will be awarded for using pre-existing attention mechanism implementations such as `torch.nn.functional.scaled_dot_product_attention()`. The notebook contains unit tests against PyTorch-native layers for both the single-head and multi-head case. These may not be changed. See the notebook for further instructions.

Task 2: Vision Transformer **(14P)**

In this task, you will implement a Vision Transformer model yourself. In the Jupyter notebook `vision_transformer.ipynb` accompanying this PDF, you will be provided with the skeleton of a set of classes that you can complete to build a working Vision Transformer yourself. Training code is provided and does not need to be modified. The hyperparameters set in the respective classes and provided in the training code are sufficient to successfully train a ViT that outperforms the

performance targets set below. Refer to [Dosovitskiy et al., 2021] for details of the architecture.

The following steps are required:

- Implement the `FeedForwardBlock` class. (1P)
- Implement the `SelfAttentionTransformerBlock` class. (2P)
- Implement the `VisionTransformer` class. (2+2+1P)
  - Add the correct missing parts in the module's `__init__()` method. (2P)
  - Implement the `patchify()` method. (2P)
  - Implement the `forward()` method. (1P)
- Train a CIFAR10 classification model using your `VisionTransformer` implementation and the provided training code (do not modify the training code). Any correct vision transformer implementation should be able to reach more than 70% classification accuracy. The intermediate steps are for cases where you have an error in the previous tasks, but still obtained a partially working ViT. Workarounds such as using a CNN or a pre-implemented Transformer instead will not receive any points. (2+2+2P)
  - Reach at least 25% validation accuracy. (2P)
  - Reach at least 50% validation accuracy. (2P)
  - Reach at least 70% validation accuracy. (2P)

Important Note: Submit exactly one ZIP file via Moodle before the deadline. The ZIP file should contain your executable code and your report in PDF format. Make sure that it runs on different operating systems and use relative paths. Non-trivial sections of your code should be explained with short comments, and variables should have self-explanatory names. The PDF file should contain your written code, all figures, explanations and answers to questions. Make sure that plots have informative axis labels, legends, and captions. Missing plots will result in point reduction.

References

[Dosovitskiy et al., 2021] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*. iii

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. ii