CE7455: Deep Learning for Natural Language Processing

Assignment # 3

Assignment Logistics

- **Submission Due:** 5 April 2022 (before 6:30 pm)

- **Submission:** Send it to ce7455.assignments@gmail.com with the subject "A3-YourStudentID".
- This is an **individually-graded assignment**. Although you can discuss with your peers, you have to submit your own assignment (following the university guideline).
- Softcopy submission: A AI6127-A3-StudentID.zip file containing the following files and folder should be submitted: (i) Report.PDF, (ii) Readme.txt, (ii) SourceCode.
  - (i) Report.PDF should contain the written part.
  - (ii) Readme.txt should include instructions to run the code and explanations of sample output obtained from your code.
  - (iii) SourceCode folder should contain all your source code. The libraries should NOT be included in the softcopy submission to minimize the file size. You can also submit a **Jupyter notebook.**

1 Question One [50 marks]

The Transformer [12] has been the dominant encoder/decoder architecture for many NLP tasks in recent years. With the access to larger GPU/TPU memory and its distributed computational capability, researchers have trained transformers with millions [9, 4, 8, 3], billions [2, 1, 10, 7] and trillions [5] of parameters. Some work [13, 6, 14] has been done on scaling the self/cross attention further. However, the basic structure of the model widely remains same. A taxonomy of the different modifications of the transformer architecture is shown in Figure 2 for those interested.

In this assignment, we will dive deep into the original transformer architecture proposed in [12]. In general, the transformer utilizes self/cross-attention with multiple heads and combines them via a regular feed-forward network. In the original base transformer, the basic hyperparameters are,

- Number of layers, $N = 6$
- Feature dimension for each token, $d_{model} = 512$

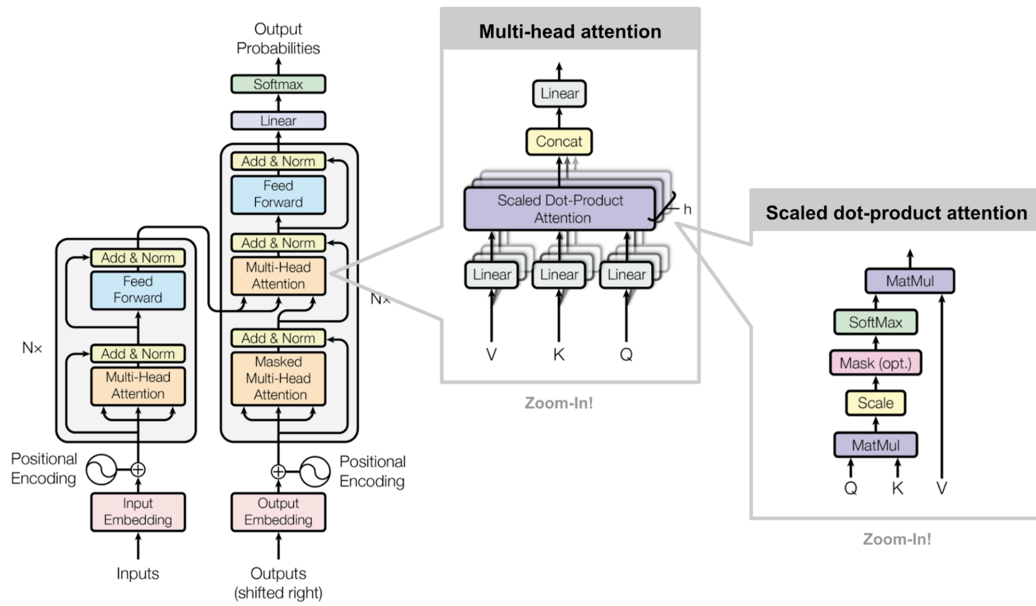

Figure 1: Transformer Architecture [12]

- Number of heads, $h = 8$
- Key and value representation size, $d_k = \frac{d_{model}}{h} = 64$
- Hidden representation size of the feed-forward layer, $ffn\_dim = 1024$

Answer the following questions:

(a) Assume you have only one layer ($N = 1$) in the transformer.

- Calculate the total number of parameters in the transformer encoder layer based on the remaining default hyper-parameters mentioned above.
- Now increase the number of layers one by one up to 12 and show how the number of parameters increases compared to the number of layers N.
- For each of the previous calculations, increase the token representation size of d_{model} from 512 to 1024 and 2048, and show how the number of parameters increases compared to the number of layers.
- Discuss how the total number of parameters changes with respect to the hyper-parameter values based on your experiments.
- Implementation guide : You can use *torch.nn.TransformerEncoderLayer* to implement the transformer layer and calculate the number of parameters.

(b) Do you agree/disagree with the following statement (provide necessary arguments to support your stand):

"Multi-head attention works as an ensemble of heads in the transformer architecture."

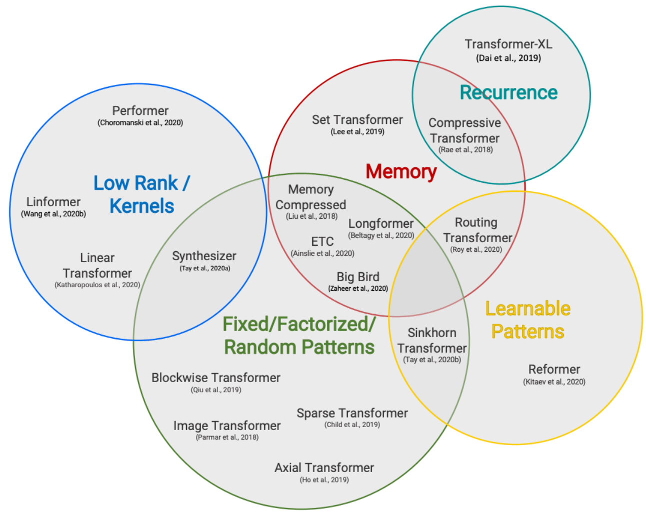

Figure 2: Taxonomy of Efficient Transformer Architectures. [11]

(c) How is the decoder training and decoder inference different in the transformer architecture, compared to recurrent models like LSTM/GRU?

(d) How does the transformer architecture capture sequence information despite having no sequential probability calculation like RNN?

(e) Follow the example code here. You will find a *TransformerModel* class in the *model.py* file which implements the *TransformerEncoder* class from the *torch.nn* library. You can also use the *Huggingface* library to train a language model. For training a language model with *Huggingface*, follow the tutorial here. We recommend writing the transformer encoder or decoder from scratch following the tutorial from Alexander Rush and integrating it into the project. Apart from that you may also look into popular library implementations (fairseq, OpenNMT-py, huggingface) and borrow different modules from there to implement your Transformer architecture.

- Train the word language model with **wikitext** (data is given in the repository). Train the model for 10 epochs. Select the best model based on development set perplexity. Report the perplexity of the test set.
- You may perform hyper-parameter search on the model based on the hyper-parameters (N, d_{model}, h, d_k, ffn_dim) discussed above in the assignment.
- What happens when you do not perform scaling (no $\sqrt{d_k}$ in the softmax in equation 1 in [12]) in the attention head? Report the effect of scaling on perplexity.

Note: If you are using a third party library for implementing the *Transformer* model (*i.e. Huggingface*), be careful about the model parameters. You may have to handle the parameters differently in different libraries.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[3] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[5] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021.

[6] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020.

[7] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[9] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[11] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey, 2020.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[13] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020.

[14] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2021.