3.4.1 Root Mean Square Layer Normalization

The original Transformer implementation of [Vaswani et al., 2017] uses layer normalization [Ba et al., 2016] to normalize activations. Following Touvron et al. [2023], we will use root mean square layer normalization (RMSNorm; Zhang and Sennrich, 2019, equation 4) for layer normalization. Given a vector $a \in \mathbb{R}^{d_{\text{model}}}$ of activations, RMSNorm will rescale each activation a_i as follows:

$$\text{RMSNorm}(a_i) = \frac{a_i}{\text{RMS}(a)} g_i, \tag{1}$$

where $\text{RMS}(a) = \sqrt{\frac{1}{d_{\text{model}}} \sum_{i=1}^{d_{\text{model}}} a_i^2 + \varepsilon}$. Here, g_i is a learnable "gain" parameter (there are `d_model` such parameters total), typically initialized to 1 and ε is a hyperparameter that is often fixed at 1e-5.

Problem (`rmsnorm`): Root Mean Square Layer Normalization (1 point)

Deliverable: Implement RMSNorm as a `torch.nn.Module`. To test your implementation against our provided test, you will first need to implement the test adapter at `[adapters.run_rmsnorm]`. Then, run `pytest -k test_rmsnorm` to test your implementation.

3.4.2 Position-Wise Feed-Forward Network

As originally described in section 3.3 of Vaswani et al. [2017], the Transformer feed-forward network consists of two linear transformations with a ReLU activate between them. The dimensionality of the inner feed-forward layer is typically 4x the input dimensionality.

Following the GPT and GPT-2 architecture Radford et al. [2018], we will use the GELU activation function [Hendrycks and Gimpel, 2016] instead of the ReLU activation function:

$$\text{GELU}(x) = x \cdot \frac{1}{2} \left[1 + \text{erf}(x/\sqrt{2}) \right] \tag{2}$$

In addition, following recent models like PaLM [Chowdhery et al., 2022] and LLaMA [Touvron et al., 2023], we omit the biases in the feed-forward network linear transformations. Thus, our feed-forward network is defined as follows:

$$\text{FFN}(x) = \text{GELU}(xW_1)W_2, \tag{3}$$

where $x \in \mathbb{R}^{d_{\text{model}}}$, $W_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$, and $W_2 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$, with $d_{\text{ff}} = 4d_{\text{model}}$.

Problem (`positionwise_feedforward`): Implement the position-wise feed-forward network (2 points)

(a) **Deliverable**: Implement the GELU activation function. To test your implementation against our provided tests, you will need to implement the test adapter at `[adapters.run_gelu]`. Then, run `pytest -k test_gelu` to test your implementation.

(b) **Deliverable**: Implement the position-wise feed-forward network. To test your implementation, implement the test adapter at `[adapters.run_positionwise_feedforward]`. Then, run `pytest -k test_positionwise_feedforward` to test your implementation.