Laporan Implementasi Transformer dari Nol dengan NumPy

Implementasi Decoder-Only Transformer (GPT-style)

1 Desain Arsitektur

Implementasi ini menggunakan arsitektur *decoder-only Transformer* (GPT-style) yang terdiri dari beberapa komponen utama yang disusun secara modular:

1.1 Komponen Utama

1. **Token Embedding**: Mengubah token input menjadi vektor dense dimensi d_{model}. Embedding di-scale dengan $\sqrt{d_{model}}$ untuk stabilitas numerik.
2. **Positional Encoding**: Menggunakan sinusoidal encoding untuk memberikan informasi posisi token dalam sequence.
3. **Multi-Head Attention**: Membagi attention menjadi beberapa head untuk menangkap berbagai aspek relasi antar token.
4. **Feed-Forward Network**: Terdiri dari 2 lapisan linear dengan aktivasi GELU sebagai non-linearitas.
5. **Residual Connection + Layer Normalization**: Menggunakan pre-norm architecture dimana LayerNorm diterapkan sebelum attention dan FFN.
6. **Output Layer**: Proyeksi akhir ke ukuran vocabulary dan softmax untuk prediksi token berikutnya.

1.2 Alur Forward Pass

$x \rightarrow$ Embedding $\rightarrow$ Pos. Enc. $\rightarrow$ Transformer Blocks $\rightarrow$ LayerNorm $\rightarrow$ Output Projection $\rightarrow$ Softmax

2 Pemilihan Positional Encoding

Dipilih **sinusoidal positional encoding** dengan pertimbangan:

- **Deterministik**: Tidak memerlukan pembelajaran parameter tambahan, cocok untuk implementasi from scratch.
- **Generalisasi panjang**: Dapat menangani sequence dengan panjang yang belum pernah dilihat saat training.

- **Matematis sederhana**: Menggunakan fungsi sinus dan cosinus dengan frekuensi berbeda:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

3 Causal Masking

Causal mask diimplementasikan untuk mencegah token mengakses informasi dari posisi masa depan:

- Menggunakan upper triangular matrix dengan nilai 1 di posisi yang harus di-mask
- Saat attention, posisi yang di-mask diberi nilai -10^9 sebelum softmax
- Setelah softmax, attention weight untuk posisi masa depan menjadi ≈ 0

Contoh mask untuk sequence length 5:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4 Bukti Uji Sederhana

4.1 Pengujian Dimensi Tensor

```
Token embedding: (2, 10, 64)
Positional encoding: (2, 10, 64)
Multi-head attention: (2, 10, 64)
Full model logits: (2, 10, 100)
Next token probs: (2, 100)
```

Dengan konfigurasi: batch_size=2, seq_len=10, d_model=64, vocab_size=100

4.2 Verifikasi Softmax

Probabilitas token berikutnya dijamin sum to 1:

```
assert np.allclose(np.sum(probs, axis=-1), 1.0)
 Probability sum check passed
```

4.3 Verifikasi Causal Mask

Causal mask berhasil dihasilkan dengan bentuk upper triangular, mencegah attention ke token masa depan. Token pada posisi i hanya dapat attend ke posisi $\leq i$.

5 Kesimpulan

Implementasi Transformer decoder-only berhasil dibuat menggunakan NumPy tanpa library deep learning. Semua komponen (embedding, positional encoding, attention, FFN, layer norm, causal mask) berfungsi dengan benar sesuai verifikasi dimensi dan pengujian matematis.

Repository: `https://github.com/iZcy/transformer-numpy`