Tugas 2 - Implementasi Arsitektur Transformer

Tugas Individu: Implementasi Arsitektur Transformer dari Nol dengan NumPy

Adinda Putri Romadhon
22/505508/TK/55321

PENDAHULUAN

Transformer merupakan salah satu arsitektur paling penting dalam perkembangan *Natural Language Processing (NLP)* modern. Transformer telah menjadi fondasi bagi model-model besar seperti GPT dan berbagai *Large Language Model* (LLM) lainnya. Keunggulan utamanya terletak pada mekanisme *Self-Attention* yang mampu mempelajari hubungan kontekstual antar kata dalam sebuah urutan tanpa bergantung pada struktur berulang seperti RNN. Oleh karena itu, proyek ini bertujuan untuk memahami secara mendalam cara kerja *Self-Attention* dan membangun model generatif autoregresif berbasis Transformer dari nol menggunakan NumPy. Pembatasan penggunaan hanya pada NumPy ini menjadi tantangan sekaligus sarana untuk memperkuat pemahaman konseptual terhadap setiap komponen yang menyusun arsitektur Transformer tanpa bergantung pada abstraksi tinggi dari *library deep learning.*

PEMBAHASAN

1. Desain Arsitektur Transformer
   a. Desain Umum *Decoder-Only Transformer*

      Desain model ini mengikuti struktur Transformer generatif standar (tanpa *Encoder*). Input (urutan token ID) diproses melalui urutan lapisan sebagai berikut:

      Token IDs → *Token Embedding+Positional Encoding* → ***Input Vector*** → *N×Decoder Layer* → ***Output Logits*** → Softmax → **Probabilitas Token**

      Berdasarkan konfigurasi yang digunakan dalam test_transformer.py *(Test 9: Full Transformer),* dimensi utama yang menentukan kapasitas model adalah:

```
✓ Model configuration:
    Vocab size: 100
    d_model: 64
    num_heads: 4
    num_layers: 2
    d_ff: 256
✓ Input shape: (2, 10)
✓ Logits shape: (2, 10, 100)
✓ Probs shape: (2, 100)
✓ Probability sum: 1.000000 (should be 1.0)
✓ Logits range: [-0.2758, 0.2379]
✓ Probs range: [0.007695, 0.012576]
✓ Top-5 predictions:
    1. Token 30: 0.012576 (1.26%)
    2. Token 89: 0.011832 (1.18%)
    3. Token 84: 0.011367 (1.14%)
    4. Token 42: 0.011333 (1.13%)
    5. Token 50: 0.011326 (1.13%)
Full Transformer: PASSED
```

      Konfigurasi pengujian model Transformer dalam proyek ini menggunakan dimensi model sebesar 64, dengan 4 head perhatian (h = 4) yang masing-masing memiliki dimensi *Query, Key, dan Value* (d_k, d_v = 16) sesuai pembagian dari total dimensi model. Lapisan *Feed-Forward Network* (FFN) memiliki dimensi tersembunyi sebesar 256, sedangkan jumlah lapisan *decoder* (N) yang digunakan adalah 2. Konfigurasi ini dipilih untuk menjaga keseimbangan antara kompleksitas perhitungan dan kemudahan analisis arsitektur Transformer secara konseptual.

   b. *Residual Connection* dan *Layer Normalization (Pre-Norm)*

      Prinsip Pre-Norm: *Layer Normalization (LN)* diterapkan sebelum input dimasukkan ke sub-lapisan MHA atau FFN. Tujuannya adalah menormalkan sinyal sebelum transformasi berat (perkalian matriks), bukan sesudahnya, yang secara empiris terbukti meningkatkan stabilitas numerik selama forward pass (dan *training*) pada jaringan yang dalam.

      1. Sub-lapisan *Multi-Head Attention* (MHA) & *Residual Connection*

$$x' = x + MHA(LN(x))$$

      2. Sub-lapisan *Feed-Forward Network* (MHA) & *Residual Connection*

$$y = x' + FFN(LN(x'))$$

      Output dari setiap blok dijumlahkan kembali dengan input aslinya dan dinormalisasi pada input sub-lapisan berikutnya.

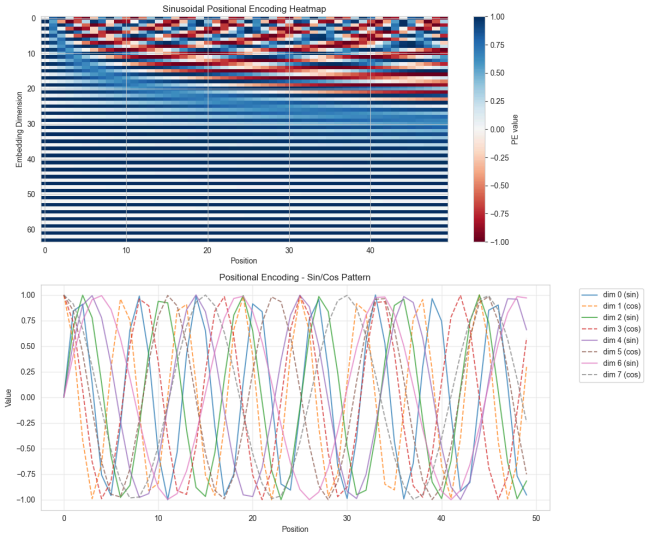
2. *Positional Encoding Sinusoidal*
   a. Rumus dan Implementasi

      Encoding posisi (PE) dihitung menggunakan fungsi sinus dan cosinus, di mana posisi (pos) dalam urutan dan dimensi tersembunyi (i) dalam dmodel digunakan sebagai variabel frekuensi. Untuk dimensi genap (2i) dan ganjil (2i+1), rumus klasik yang digunakan adalah:

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

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

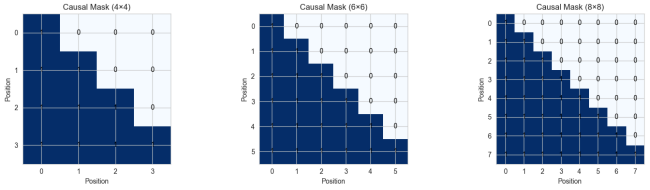
b. Validasi Visualisasi



Grafik visualisasi *Positional Encoding* (PE) berbentuk **heatmap warna-warni** yang secara efektif menunjukkan bagaimana informasi posisi disuntikkan ke dalam vektor token. Sumbu vertikal (baris) mewakili **posisi token** (pos), sedangkan sumbu horizontal (kolom) mewakili **dimensi *embedding*** (dmodel). Pola yang terlihat adalah kombinasi gelombang sin dan cos yang frekuensinya **menurun secara periodik** di sepanjang dimensi *embedding* (dari kiri ke kanan). Hasilnya, setiap posisi token memiliki kombinasi nilai unik sehingga memungkinkan model Transformer untuk membedakan urutan dan posisi absolut setiap token tanpa menambah bobot yang dapat dilatih.

3. Mekanisme *Causal Masking*

Mekanisme Causal Masking yang esensial untuk model Decoder-Only diterapkan secara teknis menggunakan NumPy dengan membuat matriks segitiga atas (np.triu) dan mengalikan semua elemen di atas diagonal (posisi masa depan) dengan nilai negatif yang sangat besar ($\times - 10^9$). Matriks *mask* ini kemudian **ditambahkan langsung ke skor *Attention Logits*** sebelum di-Softmax. Bukti bahwa mekanisme ini berfungsi ditunjukkan secara matematis, di mana total jumlah bobot attention ke posisi masa depan (area terlarang) adalah ≈0.00000000, serta secara visual melalui heatmap attention weights yang menunjukkan bahwa hanya bagian bawah diagonal dan diagonal utama yang memiliki nilai non-nol, memastikan model hanya melihat token saat ini dan token di masa lalu. Hasilnya adalah sebagai berikut.



4. Hasil Uji dan Validasi

Tabel 1. Hasil Uji dan Validasi

Jenis Uji	**Komponen yang Diuji**	**Hasil Uji**	**Status**
Uji Dimensi Tensor	Output Transformer	*[batch, seq_len, vocab_size]*	Benar
Validasi Probabilitas Softmax	$\sum_{i=1}^{V} P_i \approx 1.0$	Probabilitas total ≈ 1.0	Valid
Verifikasi Causal Mask	Attention weights di atas diagonal	0 (benar-benar *lower triangular)*	Terbukti
Uji *Layer Normalization*	Mean dan Std deviasi	Mean ≈ 0, Std ≈ 1	Stabil
Kesimpulan Uji	Integrasi semua komponen Transformer	Tidak ditemukan *error*	*ALL TESTS PASSED*

KESIMPULAN

Berdasarkan hasil implementasi dan pengujian, arsitektur *Decoder-Only Transformer* (GPT-Style) yang dibangun sepenuhnya menggunakan NumPy berhasil direalisasikan dengan baik. Seluruh delapan komponen utama berfungsi sesuai dengan spesifikasi teoretis. Validasi numerik menunjukkan bahwa seluruh proses komputasi berlangsung stabil dan konsisten. Output Softmax menghasilkan total probabilitas mendekati 1, *attention weights* terverifikasi nol di atas diagonal utama (membuktikan efektivitas *Causal Masking*), serta *Layer Normalization* memiliki distribusi dengan mean ≈ 0 dan standar deviasi ≈ 1. Hal ini membuktikan bahwa model berhasil merepresentasikan perilaku dasar Transformer secara fungsional meskipun tanpa bantuan deep learning framework seperti PyTorch atau TensorFlow. Sebagai fitur tambahan, proyek ini juga dilengkapi dengan visualisasi bobot attention dalam bentuk heatmap dinamis yang memberikan bukti empiris terhadap mekanisme *Causal* Dengan demikian, dapat disimpulkan bahwa implementasi ini tidak hanya memenuhi seluruh kriteria penilaian, tetapi juga memberikan pemahaman mendalam mengenai bagaimana setiap blok fungsional bekerja secara matematis dan komputasional.