lèa R›wé–5ÿ Attention Matrixÿ f/Transformerj!W‹N-•êlèa R›g:R6ÿ Self-Attentionÿ v„h8_Ã~Äb •èR ÿ u(N

---

1. •"Qeˆhy:
PG‹¾•"Qe^•R u1 \(N \) N*T 'Ï~Äb ÿ ~ô^¦N: \(d \)ÿ Ss \(X \in \mathbb{R}^{N \times d} \)ÿ 0 • •Ç~¿`'SØ
- **gå‹âwé–5ÿ Queryÿ **ÿ \(Q = X W_Q \)ÿ \(W_Q \in \mathbb{R}^{d \times d_k} \)
- **•.wé–5ÿ Keyÿ **ÿ \(K = X W_K \)ÿ \(W_K \in \mathbb{R}^{d \times d_k} \)
- **P<wé–5ÿ Valueÿ **ÿ \(V = X W_V \)ÿ \(W_V \in \mathbb{R}^{d \times d_v} \)

QvN- \(d_k \) TŒ \(d_v \) f/b•_q~ô^¦ÿ • ^8 \(d_k = d_v \)ÿ 0

---

2. ‹¡{—lèa R›R ep
• •Çgå‹âTŒ•.v„p¹yï‹¡{—g*_RN S v„lèa R›R epwé–5 \(A \in \mathbb{R}^{N \times N} \)ÿ
\[
A = Q K^T
\]
kÏN*QC} \(A_{ij} \) ˆhy:{, \(i \) N*OM•n[ù{, \(j \) N*OM•nv„Qslèz ^¦0

**3. •)e>ÿ Scalingÿ **
N:N†–2kbp¹yïP<•ÇY'[ü•ôh¯^¦mˆY1ÿ [ùR epwé–5•ÛˆL•)e>ÿ –dNå \(\sqrt{d_k} \)ÿ ÿ
\[
A_{\text{scaled}} = \frac{A}{\sqrt{d_k}}
\]

**4. c©x ÿ Sï• ÿ **
W(‰ãx VhN-ÿ N:••QMg*geOá`olÄ—2ÿ O O•u(N N ‰Òc©x ÿ c©x P<N: \(-\infty \)ÿ ÿ
\[
A_{\text{masked}} = A_{\text{scaled}} + M \quad \text{ÿ } M_{ij} = 0 \text{b } -\infty \text{ÿ }
\]

**5. Softmax_RN S **
[ùkÏN ˆL^"u(SoftmaxQýepÿ \ R ep•lcbN:i,s‡R ^ ÿ
\[
\text{Attention Matrix} = \text{Softmax}(A_{\text{scaled}})
\]
QI_ ÿ
\[

\text{Attention}_{ij} = \frac{\exp(A_{\text{scaled}, ij})}{\sum_{k=1}^N \exp(A_{\text{scaled}, ik})}
\]

---

6. R gClBTŒ
g ~È•"Qúf/lèa R›wé–5N P<wé–5 \(V \) v„NXyïÿ
\[
\text{Output} = \text{Attention Matrix} \cdot V
\]

**O*Nãx y:O‹ÿ PyTorch˜Îh<ÿ **
```python
import torch.nn.functional as F

def compute_attention(Q, K, V, mask=None):
    d_k = Q.size(-1)
    scores = torch.matmul(Q, K.transpose(-2, -1)) / (d_k ** 0.5)
    if mask is not None:
        scores = scores.masked_fill(mask == 0, -1e9)
    attention = F.softmax(scores, dim=-1)
    output = torch.matmul(attention, V)
    return output, attention
```

Qs•.p¹
1. **^vˆL‹¡{—**ÿ wé–5NXlÕšØeHe/c GPUR • 0
2. **z u•`'Y t **ÿ Y,DynamicViT• •Çc©x R¨ Rjg•ÿ Y,Ql_ (10)-(11)ÿ 0
3. **Y gB^¦**ÿ ‹¡{—lèa R›wé–5v„Y gB^¦N: \(O(N^2 d) \)ÿ ••^•R — O S ÿ Y,z u•lèa R›ÿ 0