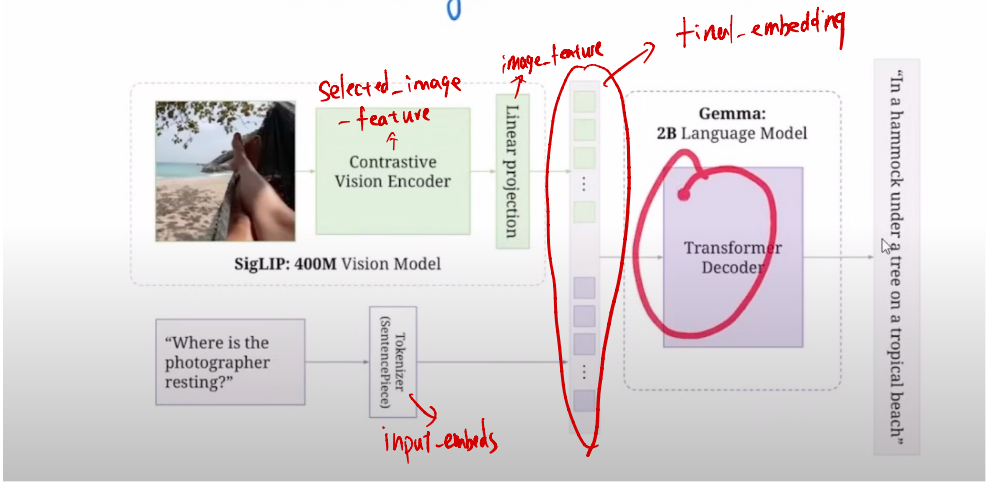
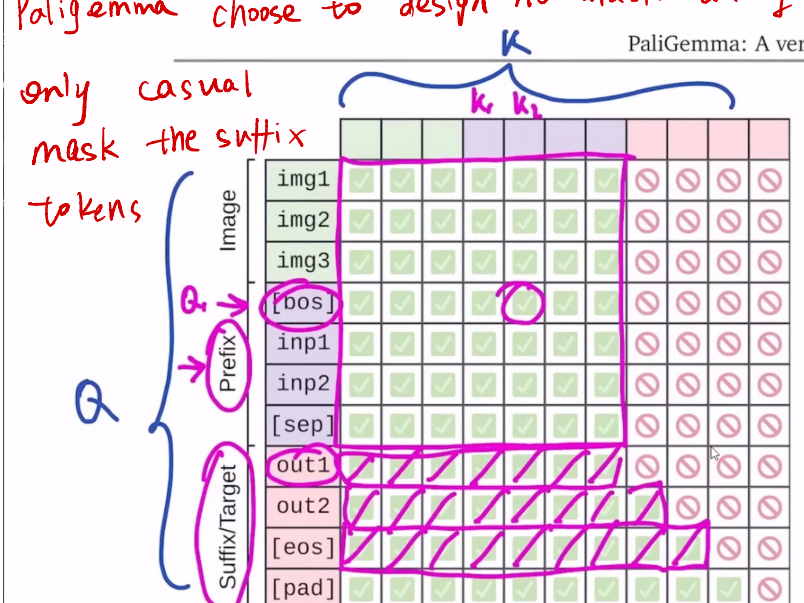
Coding a vision language model from scratch!



Paligemma choose to design no mask among image_features and prefix tokens

only casual mask the suffix tokens

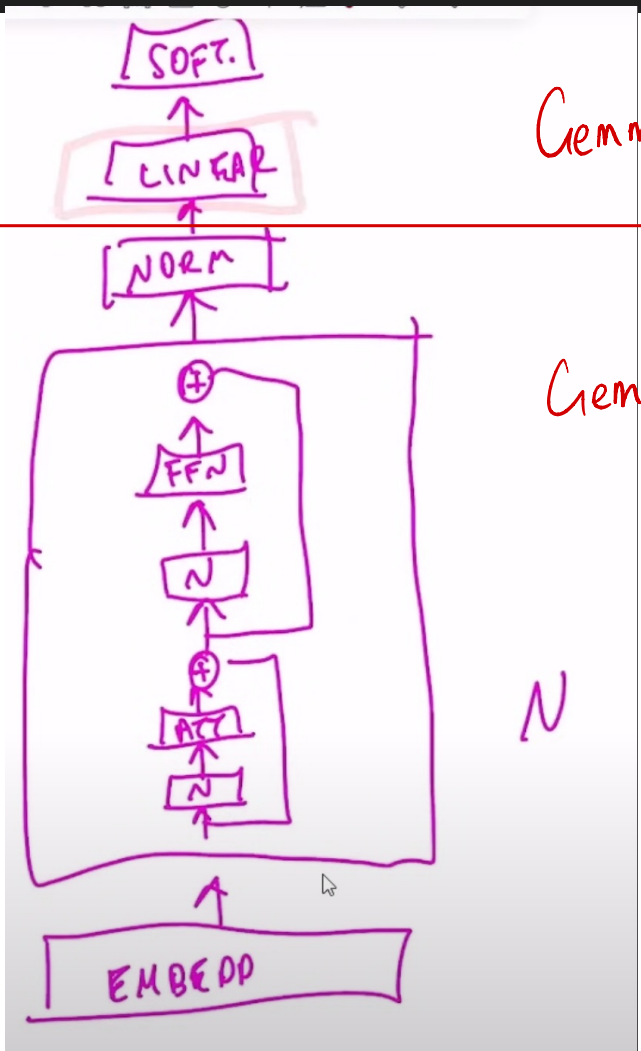
PaliGemma: A versatile 3B VLM for t



which we straightfo special lo marks the token, it We also t merged (the prefix der to ma model, w whole in In this w at the tas their repr

```python
#### CREATE THE ATTENTION MASK ####
dtype, device = inputs_embeds.dtype, inputs_embeds.device
min_dtype = torch.finfo(dtype).min
q_len = inputs_embeds.shape[1]

if kv_cache is None or kv_cache.num_items() == 0:
    # Do not mask any token, because we're in the prefill phase
    # This only works when we have no padding, we need to mask <pad> with min_dtype if there is padding
    causal_mask = torch.full(
        (batch_size, q_len, q_len), fill_value=0, dtype=dtype, device=device
    ) # fill_value=0: PaliGemma lm is designed in a way that all the image tokens
      # and prefix tokens can attend to each other to bettter understand the task
else:
    # Since we are generating tokens, the query must be one single token
    assert q_len == 1
    kv_len = kv_cache.num_items() + q_len
    # Also in this case we don't need to mask anything, since each query should be able to attend all previous tokens.
    # This only works when we have no padding
    causal_mask = torch.full(
        (batch_size, q_len, kv_len), fill_value=0, dtype=dtype, device=device
    )
```



Gemma for CausaLLM

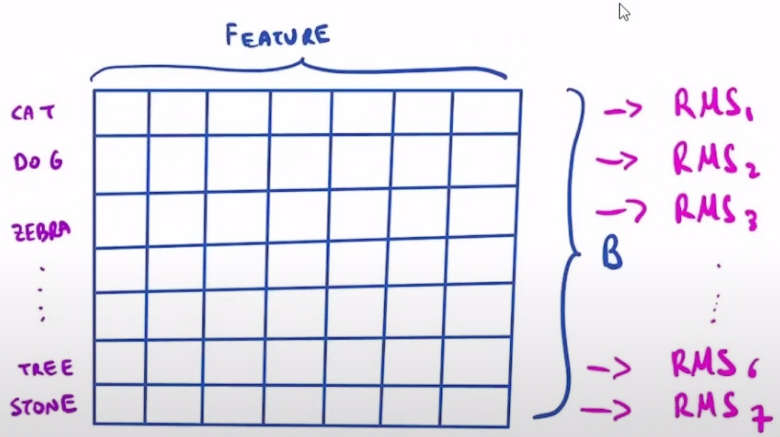
Gemma Model.

```
GemmaModel
├── embed_tokens (词向量)
├── transformer layers (多层 decoder block)
└── norm → hidden_states  [B, L, H]
        ↓
GemmaForCausalLM
└── lm_head: Linear(hidden_size → vocab_size)
        ↓
  logits (预测下一个 token)
```

RMS Normalization.

learnable parameter, each of a feature → g_i

$$\bar{a}_i = \frac{a_i}{\text{RMS}(\mathbf{a})} g_i, \quad \text{where } \text{RMS}(\mathbf{a}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} a_i^2}.$$



In this case each item is treated independently.

Grouped Query Attention:

Bottleneck of computation in GPU:

copy matrix from HBM to local memory for cores to access

Idea: Share the same key head for multiple queries head. Deduce the cost of copying memory

Rotary Pos_ Embedding

Relative position embedding

In this paper, we introduce a novel method, namely Rotary Position Embedding(RoPE), to leverage the positional information into the learning process of PLMS. Specifically, RoPE encodes the absolute position with a rotation matrix and meanwhile incorporates the explicit relative position dependency in self-attention formulation. Note that the proposed RoPE is prioritized over the existing methods through valuable properties, including the sequence length flexibility, decaying inter-token dependency with increasing relative distances, and the capability of equipping the linear self-attention with relative position encoding. Experimental results on various long text classification benchmark datasets show that the enhanced transformer with rotary position embedding, namely RoFormer, can give better performance compared to baseline alternatives and thus demonstrates the efficacy of the proposed RoPE.

Incorporate relative information with the rotation of context representations.

Transformer-based language modeling usually leverages the position information of individual tokens through a self-attention mechanism. As can be observed in Equation (2), $\boldsymbol{q}_m^\intercal \boldsymbol{k}_n$ typically enables knowledge conveyance between tokens at different positions. In order to incorporate relative position information, we require the inner product of query $\boldsymbol{q}_m$ and key $\boldsymbol{k}_n$ to be formulated by a function g, which takes only the word embeddings $\boldsymbol{x}_m$, $\boldsymbol{x}_n$, and their relative position $m-n$ as input variables. In other words, we hope that the inner product encodes position information only in the relative form:

$$\langle f_q(\boldsymbol{x}_m, m), f_k(\boldsymbol{x}_n, n)\rangle = g(\boldsymbol{x}_m, \boldsymbol{x}_n, m-n). \tag{11}$$

The ultimate goal is to find an equivalent encoding mechanism to solve the functions $f_q(\boldsymbol{x}_m, m)$ and $f_k(\boldsymbol{x}_n, n)$ to conform the aforementioned relation.

3.2.1 A 2D case

We begin with a simple case with a dimension $d = 2$. Under these settings, we make use of the geometric property of vectors on a 2D plane and its complex form to prove (refer Section **(3.4.1)** for more details) that a solution to our formulation Equation (11) is:

$$\begin{aligned} f_q(\boldsymbol{x}_m, m) &= (\boldsymbol{W}_q \boldsymbol{x}_m) e^{im\theta} \\ f_k(\boldsymbol{x}_n, n) &= (\boldsymbol{W}_k \boldsymbol{x}_n) e^{in\theta} \\ g(\boldsymbol{x}_m, \boldsymbol{x}_n, m-n) &= \mathrm{Re}[(\boldsymbol{W}_q \boldsymbol{x}_m)(\boldsymbol{W}_k \boldsymbol{x}_n)^* e^{i(m-n)\theta}] \end{aligned} \tag{12}$$

where $\mathrm{Re}[\cdot]$ is the real part of a complex number and $(\boldsymbol{W}_k \boldsymbol{x}_n)^*$ represents the conjugate complex number of $(\boldsymbol{W}_k \boldsymbol{x}_n)$. $\theta \in \mathbb{R}$ is a preset non-zero constant. We can further write $f_{\{q,k\}}$ in a multiplication matrix:

$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W_{\{q,k\}}^{(11)} & W_{\{q,k\}}^{(12)} \\ W_{\{q,k\}}^{(21)} & W_{\{q,k\}}^{(22)} \end{pmatrix} \begin{pmatrix} x_m^{(1)} \\ x_m^{(2)} \end{pmatrix} \tag{13}$$

where $(x_m^{(1)}, x_m^{(2)})$ is $\boldsymbol{x}_m$ expressed in the 2D coordinates. Similarly, g can be viewed as a matrix and thus enables the solution of formulation in Section **(3.1)** under the 2D case. Specifically, incorporating the relative position embedding is straightforward: simply rotate the affine-transformed word embedding vector by amount of angle multiples of its position index and thus interprets the intuition behind *Rotary Position Embedding*.

3.2.2 General form

In order to generalize our results in 2D to any $\boldsymbol{x}_i \in \mathbb{R}^d$ where d is even, we divide the d-dimension space into $d/2$ sub-spaces and combine them in the merit of the linearity of the inner product, turning $f_{\{q,k\}}$ into:

$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}_{\Theta,m}^d \boldsymbol{W}_{\{q,k\}} \boldsymbol{x}_m \tag{14}$$

where

$$\boldsymbol{R}_{\Theta,m}^d = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix} \tag{15}$$

is the rotary matrix with pre-defined parameters $\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, ..., d/2]\}$. A graphic illustration of RoPE is shown in Figure **(1)**. Applying our RoPE to self-attention in Equation (2), we obtain:

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = (\boldsymbol{R}_{\Theta,m}^d \boldsymbol{W}_q \boldsymbol{x}_m)^\intercal (\boldsymbol{R}_{\Theta,n}^d \boldsymbol{W}_k \boldsymbol{x}_n) = \boldsymbol{x}^\intercal \boldsymbol{W}_q^\intercal R_{\Theta,n-m}^d \boldsymbol{W}_k \boldsymbol{x}_n \tag{16}$$

where $\boldsymbol{R}_{\Theta,n-m}^d = (\boldsymbol{R}_{\Theta,m}^d)^\intercal \boldsymbol{R}_{\Theta,n}^d$. Note that $\boldsymbol{R}_\Theta^d$ is an orthogonal matrix, which ensures stability during the process of encoding position information. In addition, due to the sparsity of R_Θ^d, applying matrix multiplication directly as in Equation (16) is not computationally efficient; we provide another realization in theoretical explanation.

3.4.2 Computational efficient realization of rotary matrix multiplication

Taking the advantage of the sparsity of $\boldsymbol{R}^d_{\Theta,m}$ in Equation (15), a more computational efficient realization of a multiplication of R^d_Θ and $\boldsymbol{x} \in \mathbb{R}^d$ is:

element-wise product

$$\boldsymbol{R}^d_{\Theta,m}\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{d-1} \\ x_d \end{pmatrix} \otimes \begin{pmatrix} \cos m\theta_1 \\ \cos m\theta_1 \\ \cos m\theta_2 \\ \cos m\theta_2 \\ \vdots \\ \cos m\theta_{d/2} \\ \cos m\theta_{d/2} \end{pmatrix} + \begin{pmatrix} -x_2 \\ x_1 \\ -x_4 \\ x_3 \\ \vdots \\ -x_d \\ x_{d-1} \end{pmatrix} \otimes \begin{pmatrix} \sin m\theta_1 \\ \sin m\theta_1 \\ \sin m\theta_2 \\ \sin m\theta_2 \\ \vdots \\ \sin m\theta_{d/2} \\ \sin m\theta_{d/2} \end{pmatrix} \tag{34}$$