EECS 182 Deep Neural Networks
Fall 2026 Gireeja Ranade

Discussion 12

1. Relative Positional Embedding

Year	Model Name	Type of Positional Embedding (PE)
2018, 2019, 2020	GPT-1, 2, and 3	Learned absolute PE
2021	Gopher	Relative PE (Transformer-XL)
2022	PaLM	RoPE
2023, 2024	LLaMA, LLaMA-2, and LLaMA-3	RoPE
2023	Grok-1	RoPE
2024	DeepSeek-V2, V3	decoupled RoPE
2025	LLaMA-4	iRoPE (interleaving RoPE and NoPE)
2025	Qwen-3	RoPE
2025	Gemma 3	RoPE

Table 1: Types of positional embeddings used in large language models over the years.

(a) **Why do we need positional encoding? Describe a situation where word order information is necessary for the task performed.**

(b) **How does relative positional embedding work? How is it different from the absolute positional embedding?**

 All Rights Reserved. This may not be publicly shared without explicit permission.

(c) Consider a list of 2-dimensional vectors $\left[\left(x_1^{(1)}, x_1^{(2)}\right), \left(x_2^{(1)}, x_2^{(2)}\right), \left(x_3^{(1)}, x_3^{(2)}\right), \ldots\right]$. Now for any choice of parameter ω, the corresponding RoPE encoding is

$$\mathrm{RoPE}\left(x_t^{(1)}, x_t^{(2)}, t\right) = \begin{pmatrix} \cos t\omega & -\sin t\omega \\ \sin t\omega & \cos t\omega \end{pmatrix} \begin{pmatrix} x_t^{(1)} \\ x_t^{(2)} \end{pmatrix} = \begin{pmatrix} x_t^{(1)} \cos t\omega - x_t^{(2)} \sin t\omega \\ x_t^{(2)} \cos t\omega + x_t^{(1)} \sin t\omega \end{pmatrix}$$

Equivalently, if we write the 2-dimensional vectors as complex numbers $z_t := x_t^{(1)} + jx_t^{(2)}$, then RoPE multiplies z_t by the unit complex phase $e^{jt\omega}$, where ω is a fixed real frequency:

$$\mathrm{RoPE}\left(z_t, t\right) = e^{jt\omega} z_t$$

The positional dependence of the dot product is only on relative location:

$$\mathrm{RoPE}(x, m)^T \, \mathrm{RoPE}(y, n) = \mathrm{RoPE}(x, m+k)^T \, \mathrm{RoPE}(y, n+k)$$

for any integer k. This is obvious if we look at RoPE in the complex number form.

Extend RoPE to work for $2n$-dimensional vectors.

(d) **Does adding RoPE change the number of learnable parameters in the model? If so, how many new parameters are added (compared to a similar model with no positional encoding)?**

 All Rights Reserved. This may not be publicly shared without explicit permission.

2. Vision Transformer

A Vision Transformer (ViT) uses a Transformer encoder to classify images. It treats image patches as tokens, much as a text Transformer treats words or subwords as tokens. We consider the original supervised image-classification architecture:

1. **Split the image into patches.** Divide the image into non-overlapping $P \times P$ pixel patches. ViT/16 variants use $P = 16$.
2. **Form patch embeddings.** For an image with C channels ($C = 3$ for RGB), flatten each patch into P^2C numbers. Apply the same learned linear projection to every patch to obtain a D-dimensional token embedding.
3. **Add a class token and positional information.** Prepend a learned classification (CLS) token and add learned absolute positional embeddings to the patch tokens and the CLS token.
4. **Encode and classify.** Pass the sequence through Transformer encoder blocks containing multi-head self-attention and feed-forward networks. Feed the final CLS representation to a classification head to predict the image's label.

The encoder uses full, unmasked self-attention. The questions below explore this classifier and how its architecture or training objective could be adapted to other tasks.

Below is a diagram of ViT for supervised image classification.

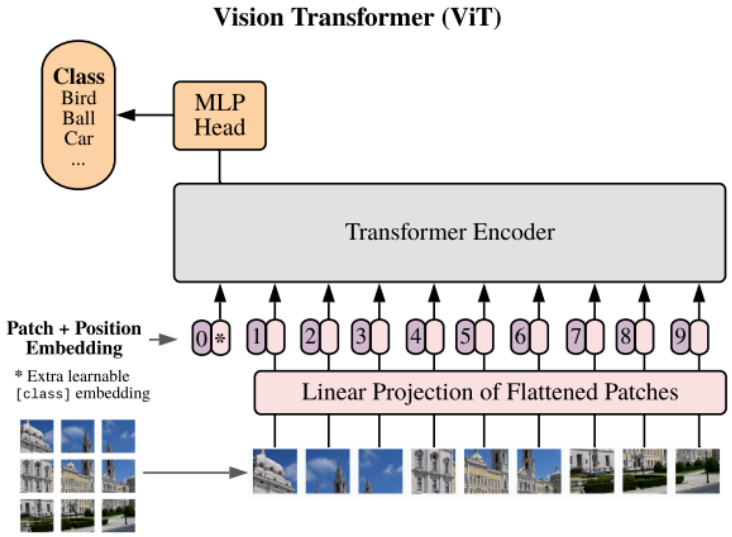


Figure 1: Vision Transformer (Link to Google blog)

(a) Does it matter which order you flatten the sequence of patches?

 All Rights Reserved. This may not be publicly shared without explicit permission.

(b) What is the complexity of the vision transformer attention operation? Assume you have an image of size $H \times W$ and patches of size $P \times P$. Only consider the time of the attention operation, not the time to produce queries, keys, and values. Queries, keys, and values each have dimension D.

(c) What is the receptive field of one sequence item after the first layer of the transformer? How does this compare to a CNN, and what are the pros and cons of this?

(d) If we forgot to include positional encodings, could the model learn anything at all? State one task where a model could perform well without positional encoding, and one task where it would do poorly.

(e) If you wanted to add a few conv layers into this architecture, how would you incorporate them?

(f) How would you use this architecture to do GPT-style autoregressive generation of images?

(g) So far the vision transformer training procedure has been fully-supervised. Yet we know that most of the available data are unlabelled. How can we do BERT-style self-supervised **representation** learning with vision transformers?

- *Hint 1*: Think about how BERT is trained. How should the input image be modified? What should be the target?
- *Hint 2*: ViT in this question only has an encoder. For masked-patch training, add a patch-prediction head or decoder. What must it predict?

Contributors: Matteo Guarrera, Mert Cemri, Sizhe Chen, Gireeja Ranade, Kevin Li, Anant Sahai, Olivia Watkins, Jerome Quenum, Saagar Sanghavi, Yuxi Liu, Qiyang Li.

 All Rights Reserved. This may not be publicly shared without explicit permission.