Wednesday, 9 July 2025

Vision Transformer With Any Resolution (ViTar)

Complete Understanding

Vision Transformer with Any Resolution

- New transformer-based architecture that can handle images of different sizes without requiring resizing or cropping.
- Images of Satellite with different resolution and sizes, in this traditional ViTs fit these images into a fixed shape, which may technically function. So ViTAR addresses this as :-

1. Support input of any resolution.
2. Use an **Fuzzy Positional Encoding** method to understand where each patch is, even if the image size changes.
3. Robust performance on real-world data.

Real World Applications

1. **Remote sensing :** Process satellite images at various resolutions without compromising information quality.
2. **Medical Imaging :** Analyse CT scans, MRIs or X-rays in their original resolution.
3. **Surveillance System :** Handle different frame sizes from various camera feeds.
4. **Document Analysis :** Manage scanned document with different formats and aspect ratios.

How is ViTAR trained ?

- ViTAR is trained using datasets with varying image sizes.
- During training, it learns how to generalise across resolutions.
- Ideal for real-world datasets, but other ViTs only rely on neat and curated , fixed size dataset like ImageNet.

Pros & Cons Of ViTAR

- **Pros -** Works with images of any size - no need for resizing or cropping.
    - Maintains spatial details in high-resolution or irregular images.
    - Ideal for real-world problems like satellite and medical imaging.
    - Fuzzy positional encoding improves flexibility.
    - Reduces preprocessing steps.
- **Cons -** May require more training data to learn robust patterns across sales.
    - Slightly complex architecture as ViT.
    - Newer - limited availability.
    - Computationally Intensive on training.

Structure of ViTAR

ViTAR's pipeline

1. Patch Embedder with Any-Resolution Flexibility

- Input image - whether 200 X 300 or 4K- split into fixed - size patches (16X16).
- No. Of patches is variable not the patch size.

2. Linear Projection

- Each patch is flattened & passed through a small linear layer, turn raw pixels into patch embedding vector.

3. Fuzzy Positional Encoding

- To keep track of where each patch came from, ViTAR adds a learned, interpolated positional vector that scales smoothly with resolution.
- Help model to understand spatial layout even when image dimension change.

4. Transformer Encoder Blocks

- Stacks of Multi-Head Self Attention (MHSA) + MLP layers process all patch embeddings in parallel, sharing information across the entire image.
- LayerNorm & residual connections keep training stable.

5. Adaptive Aggregation

- For classification, a special [CLS] token or global average pooling gathers a single image representation.
- For dense segmentation & detection tasks, patch outputs are reshaped back to spatial grids.

6. Task-Specific Head

- Simple linear layer (classification), a small decoder (segmentation) or a detection head attaches here.

Positional Encoding in Vanilla ViT

Positional Encodings

- Classical Transformer sine-cosine formula build positional embeddings for 14 X 14 grid as :-
- 14 X 14 = 196 patches (224 / 16 = 14)
- Standard Formula :-

$$\mathrm{PE}(p,\ 2d) = \sin\left(p/10000^{\frac{2d}{D}}\right)$$

$$\mathrm{PE}(p,\ 2d+1) = \cos\left(p/10000^{\frac{2d}{D}}\right)$$

- **Why low frequency channels change smoothly across the grid while high frequency channels oscillate rapidly(capture fine spatial detail) ?**
- The Positional formula is :-

$$\theta_{p,d} = \frac{p}{10000^{\frac{d}{D}}}$$

- P = patch (token) index
- d = channel (embedding dimension)
- D = total embedding width
- As d gets larger , denominator 10000^(d/D) grows exponentially, making the angle and therefore the sin/cosine value change more slowly from one to next patch.
- **Low index channel** (0, 1) -> oscillate quickly -> high frequency
- **High Index Channel** (510, 511, ..) -> vary slowly -> low frequency.

Calculate positional embeddings in Vanilla ViT

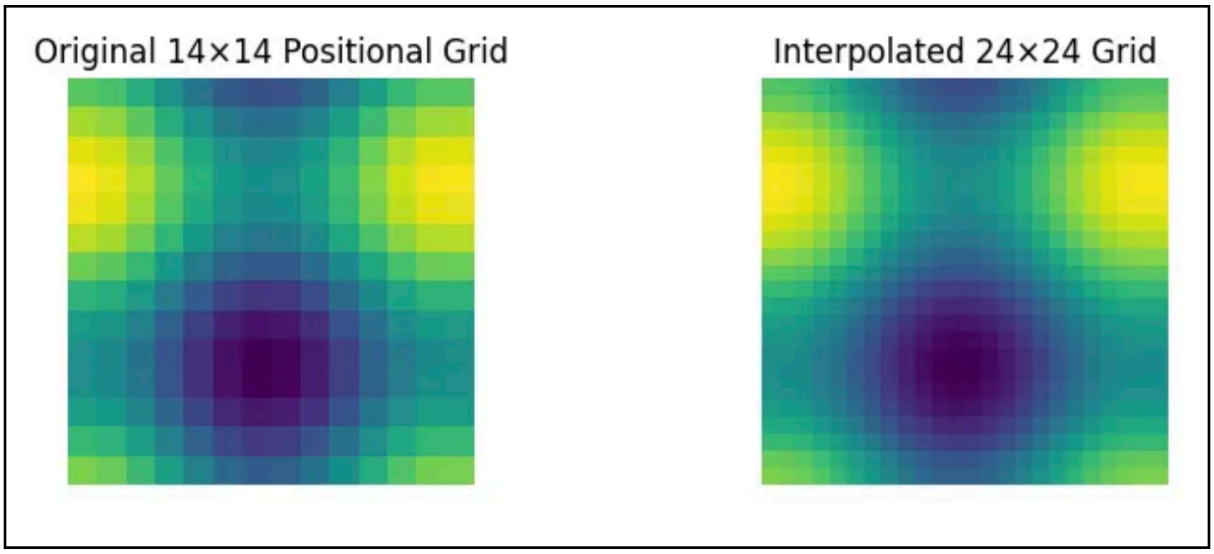
- How classic ViT transforms 2D patch coordinates into learnable vectors and then how these vectors combined with patch features.

1. **Indexing Patches :** ViT flattens a 14 X 14 grid into 196 long sequence.
2. **Parameter table :** Single tensor pos_embed stores a 768-D vector for the [CLS] token plus one vector per patch.
3. **Simple addition :** Model literally adds the right row of pos_embed to each patch projected pixels, giving the transformer a sense of spatial order before self-attention.

Fuzzy Positional Encoding - ViTAR

- ViTAR introduces a flexible approach by replacing that grid lookup table with fuzzy positional encoding.
- Classic ViT stores one row for each patch in a 224 X 224 image.
- The model has no rows for extra patches if given a wider or taller picture, so we have to crop the image first for the ViT.
- ViTAR swaps that grid spreadsheet for a **stretchy rubber sheet** of numbers.
- It is not a fixed list of rows.

- It's a continuous mathematical surface so if the image grows from 14 X 14 to 24 X 24 patches, ViTAR just samples new points on the same surface by using smooth bicubic interpolation.
- Instead of **one hot position**, **ViTAR** learns a **continuous function** that can be **interpolated** by any **patch** grid.
- **resample_abs_pos_embed** -> performs bicubic interpolation on the 2-D positional grid , like resizing the image but in embedding space.



- Low frequency channel remains smooth while the high frequency channels display stripes.
- No new information is created, pattern stretches allowing ViTAR to preserve positional meaning to a higher resolution.

Techniques that ViTAR brings

1. **Patch-Embedder with Variable Patch count -** Keeps the Patch count constant (16 X 16 px) but lets the number of patches grow or shrink with input and output -> Prevent spatial distortion that may arise with rescaling of image. Preserves high-frequency in large images and avoids info. loss in small ones. -> Convolutional stride uses stride = patch size, so kernel size is fixed but output feature map scales with the input. Positional info. is injected after patch extraction (via fuzzy / absolute-relative hybrid embeddings) so model never assumes a fixed grid.

2. **Adaptive Aggression head -** produce size-agnostic image representation for classification or size-consistent grid for dense tasks. -> Enables a single checkpoint to serve both 224 X 224 benchmark. Eliminates the need to keep multiple heads. -> For classification : Uses a flexible CLS token whose query attends to all patch token. Optionally blends with global average pooling across patches. For segmentation / detection : reshape the seq. back into [H X P] * [W X P] token grids then applies decoder - header that expects the adaptive grid directly.
3. **Resolution - aware training augmentation -** Feeds each training image at random scales (and aspect ratio), so the model learns invariance across a broad resolution spectrum. -> Forces the attention layers to generalise positional relationships rather than memorise a single grid. Improves sample efficiency by turning one high resolution photo into many scale variants. -> Typical pipeline , maintain aspect ratio , then pad to next multiple of each size. Mixes standard colour/ crop augmentation with scale jitter every epoch. Reduces **resolution shift** error validation , accuracy drops far less when tested on sizes unseen at train time.

- ViTAR -> advancement in Vision transformers for applications involves diverse or high resolution image data.
- It brings flexibility, preserve details and avoid clunky preprocessing steps.