UNC CHARLOTTE

The WILLIAM STATES LEE COLLEGE *of* ENGINEERING

Introduction to ML
Lecture 12: Transformers for Computer Vision

Hamed Tabkhi
Department of Electrical and Computer Engineering,
University of North Carolina Charlotte (UNCC)
htabkhiv@uncc.edu

Motivation

- The Transformer architecture was initially proposed for sequence-to-sequence learning, with a focus on machine translation.
- Subsequently, Transformers emerged as the model of choice in various natural language processing tasks (Brown *et al.*, 2020, Devlin *et al.*, 2018, Radford *et al.*, 2018, Radford *et al.*, 2019, Raffel *et al.*, 2020).
- However, in the field of computer vision the dominant architecture has remained the CNN.
- Naturally, researchers started to wonder if it might be possible to do better by adapting Transformer models to image data.
- This question sparked immense interest in the computer vision community.

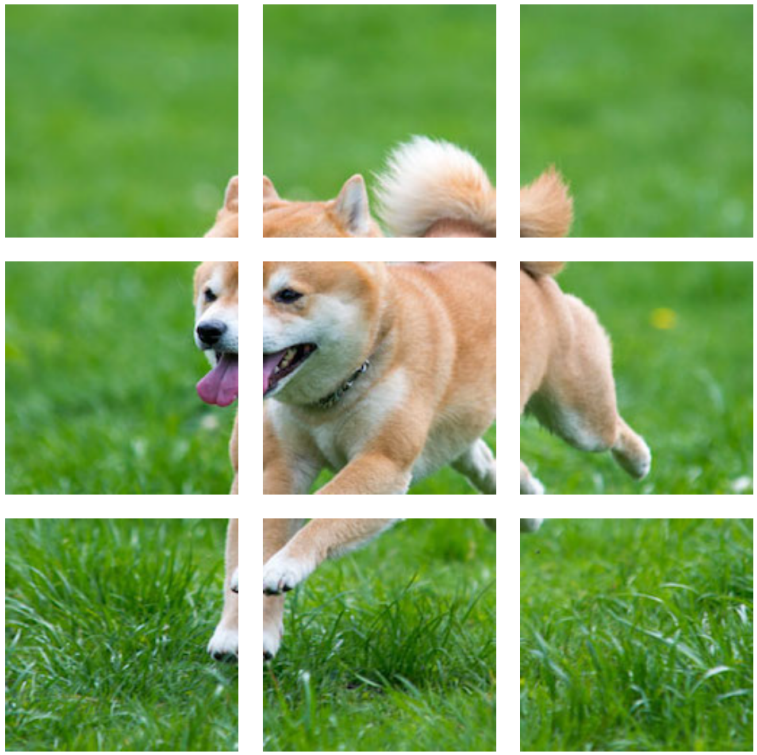
Motivation

- Recently, Ramachandran *et al.* (2019) proposed a scheme for replacing convolution with self-attention.
- However, its use of specialized patterns in attention makes it hard to scale up models on hardware accelerators.
- Then, Cordonnier *et al.* (2020) theoretically proved that self-attention can learn to behave similarly to convolution.

- In a more recent approach, without specific constraints on patch size, *vision Transformers* (ViTs) extract patches from images and feed them into a Transformer encoder to obtain a global representation, which will finally be transformed for classification (Dosovitskiy *et al.*, 2021).

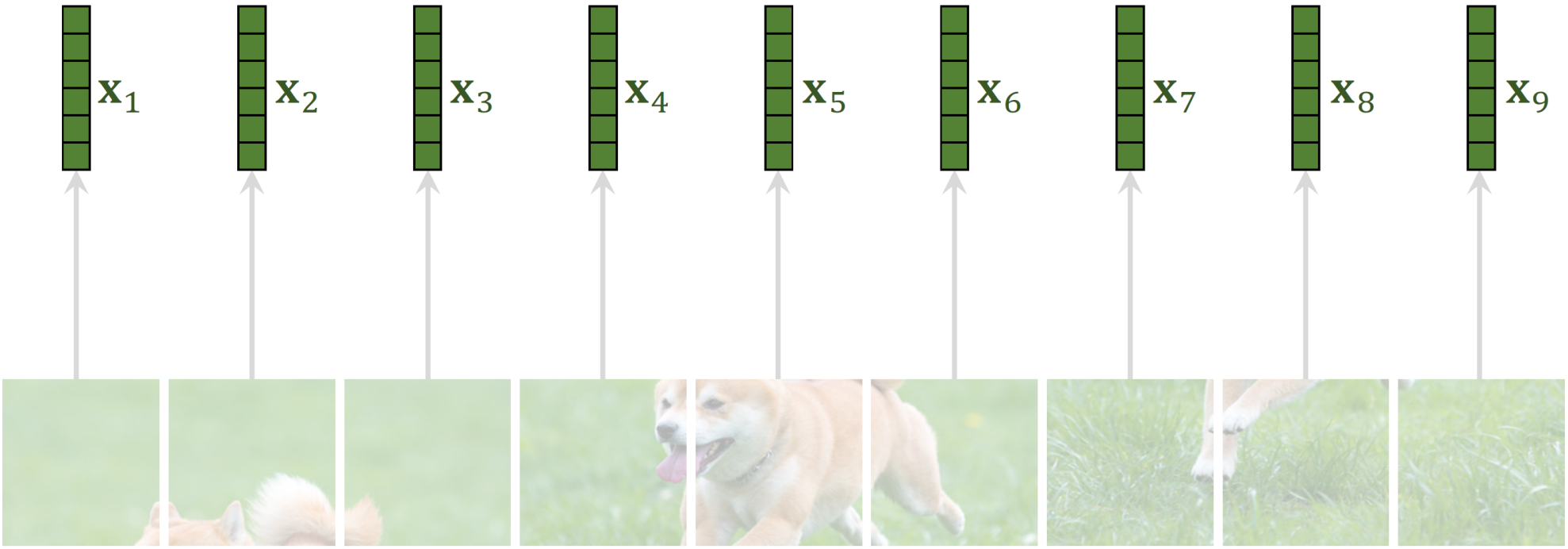
- **Notably, Transformers show better scalability than CNNs: when training larger models on larger datasets, vision Transformers outperform ResNets by a significant margin.**
- Similar to the landscape of network architecture design in natural language processing, Transformers also became a gamechanger in computer vision.

Vision Transformers (ViTs)



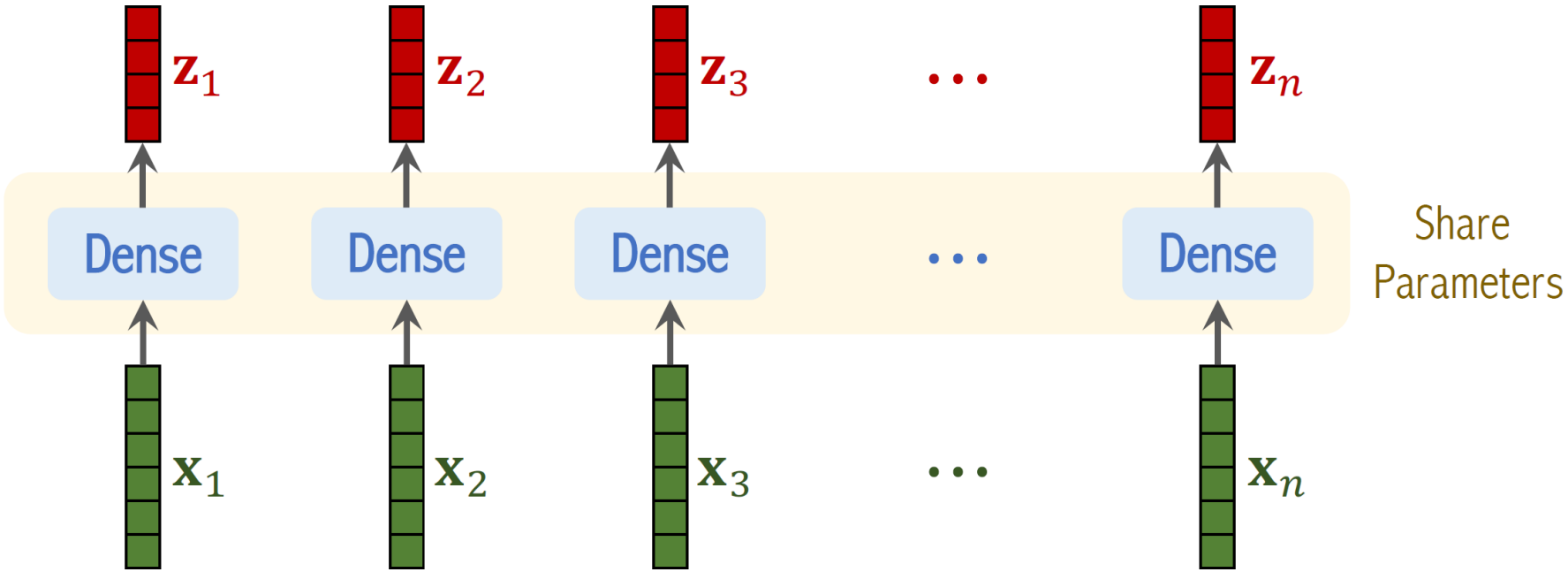
Vision Transformers (ViTs)

If the patches are $d_1 \times d_2 \times d_3$ tensors, then the vectors are $d_1 d_2 d_3 \times 1$.



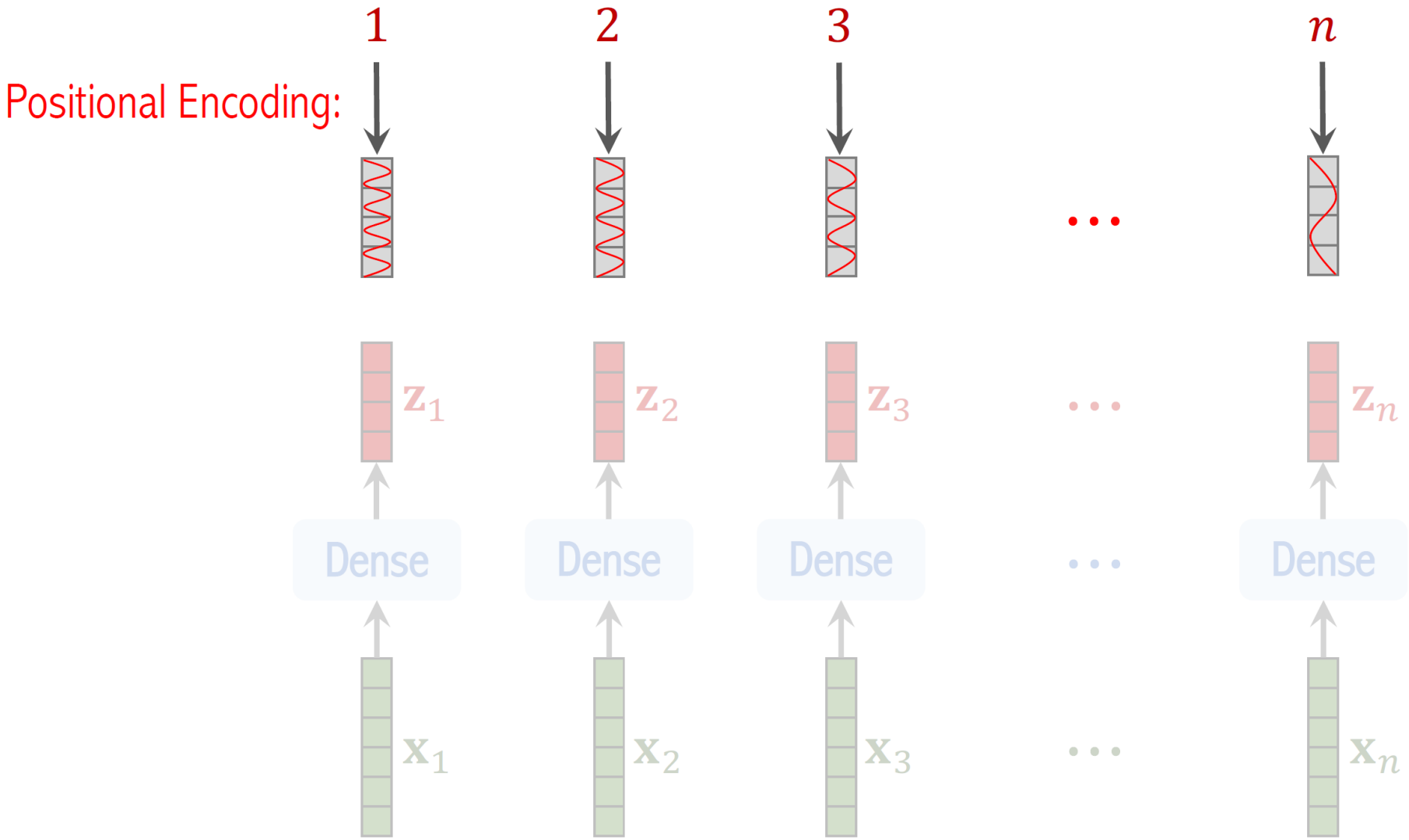
UNC CHARLOTTE

Vision Transformers (ViTs)



UNC CHARLOTTE

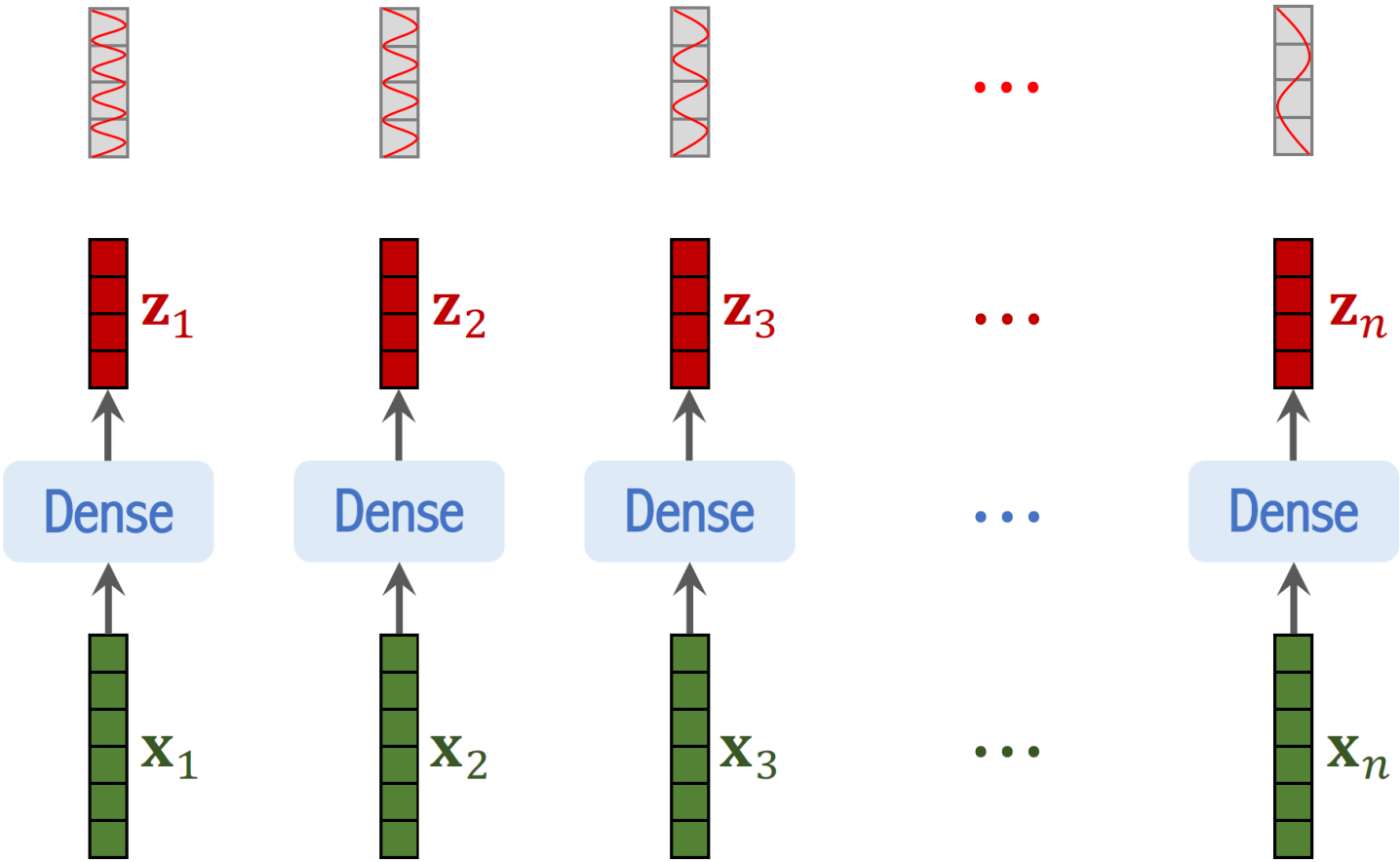
Vision Transformers (ViTs)



UNC CHARLOTTE

Vision Transformers (ViTs)

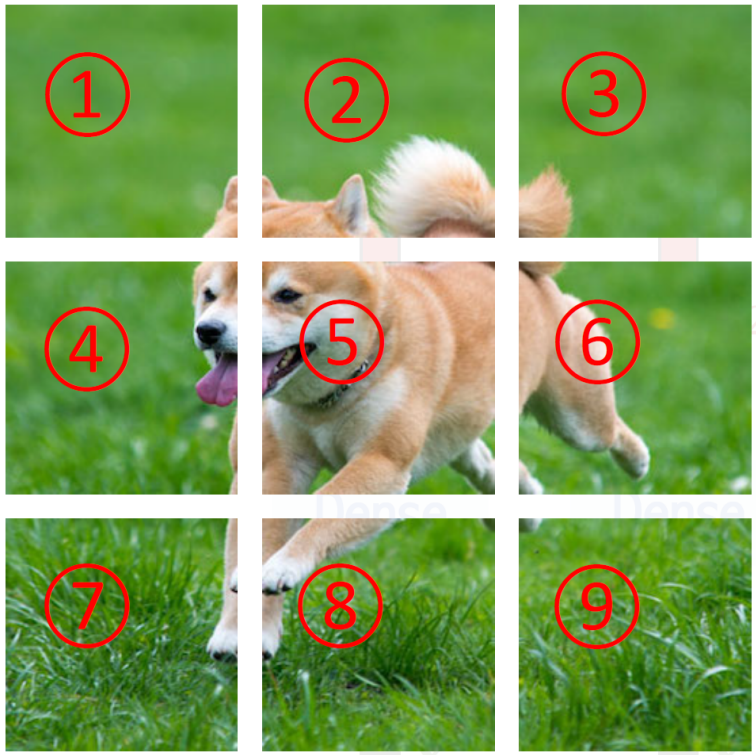
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$.

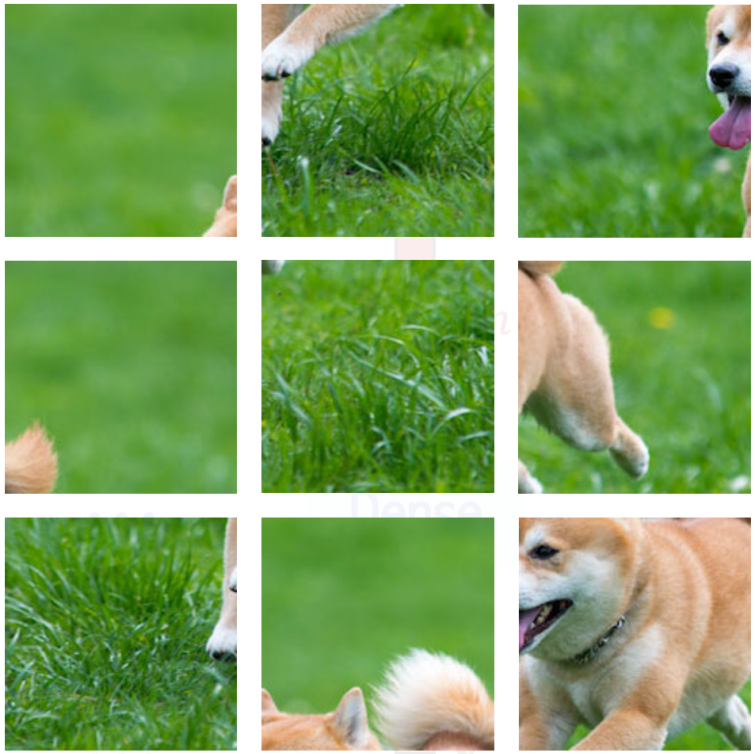


UNC CHARLOTTE

Vision Transformers (ViTs)

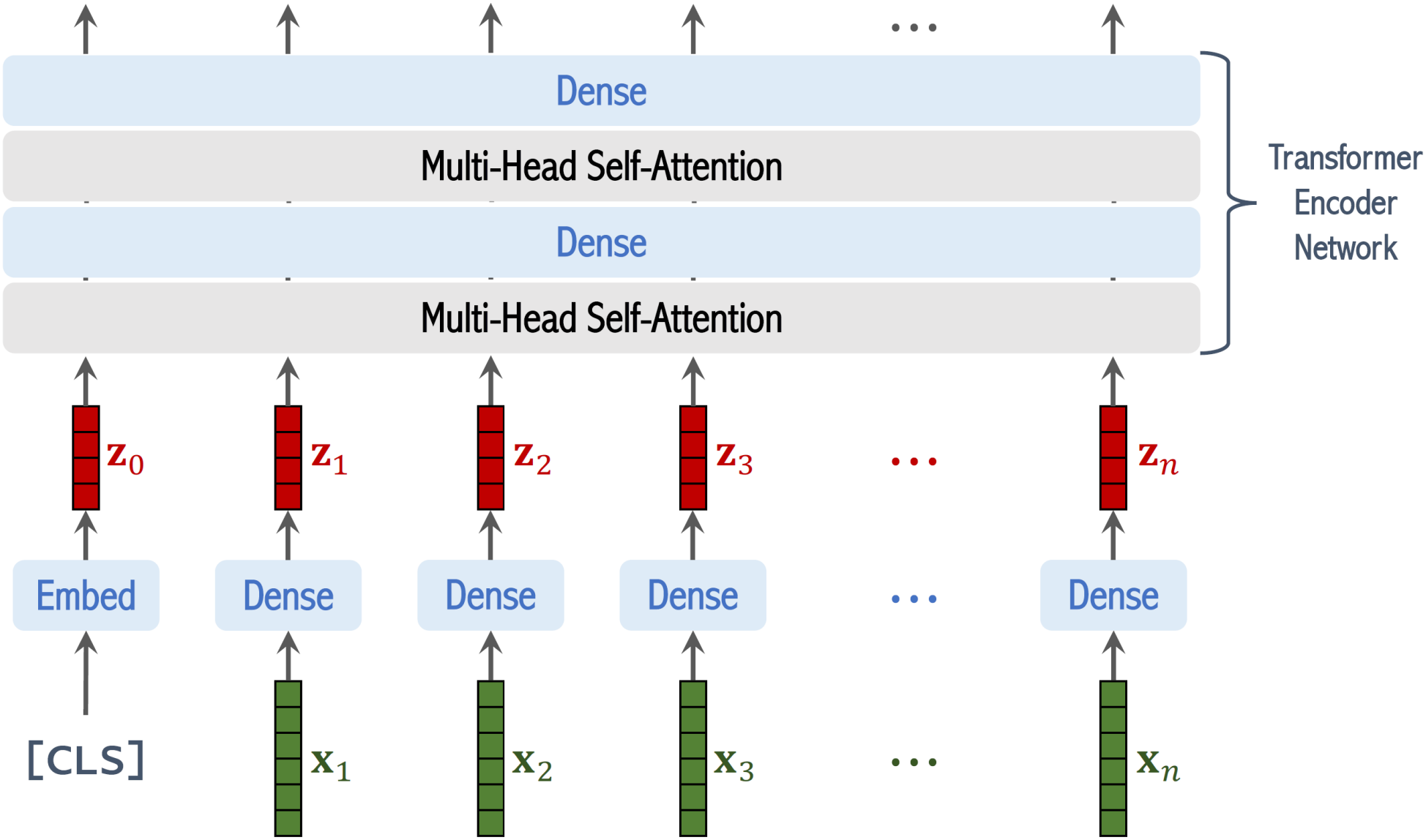
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)





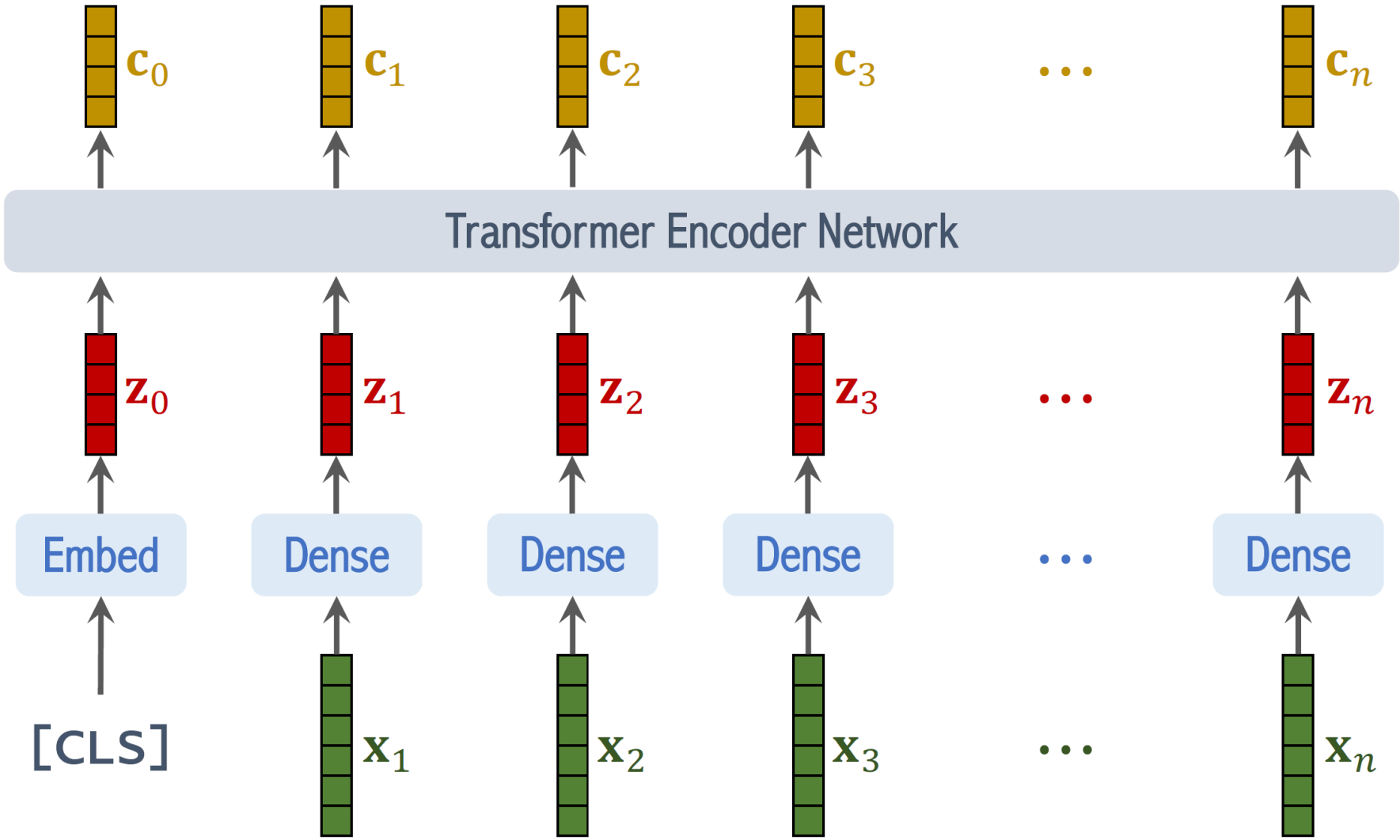
UNC CHARLOTTE

Vision Transformers (ViTs)



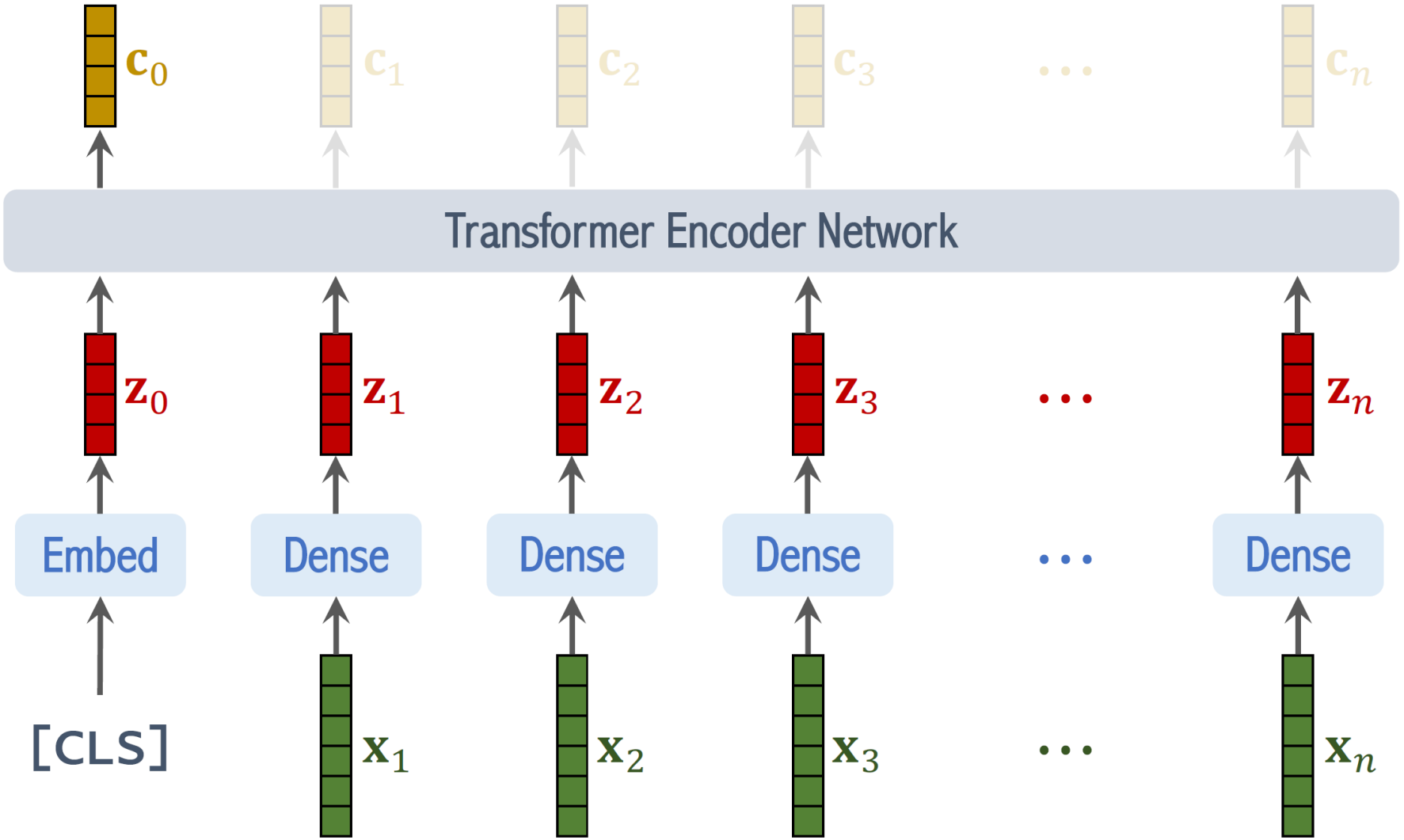
UNC CHARLOTTE

Vision Transformers (ViTs)



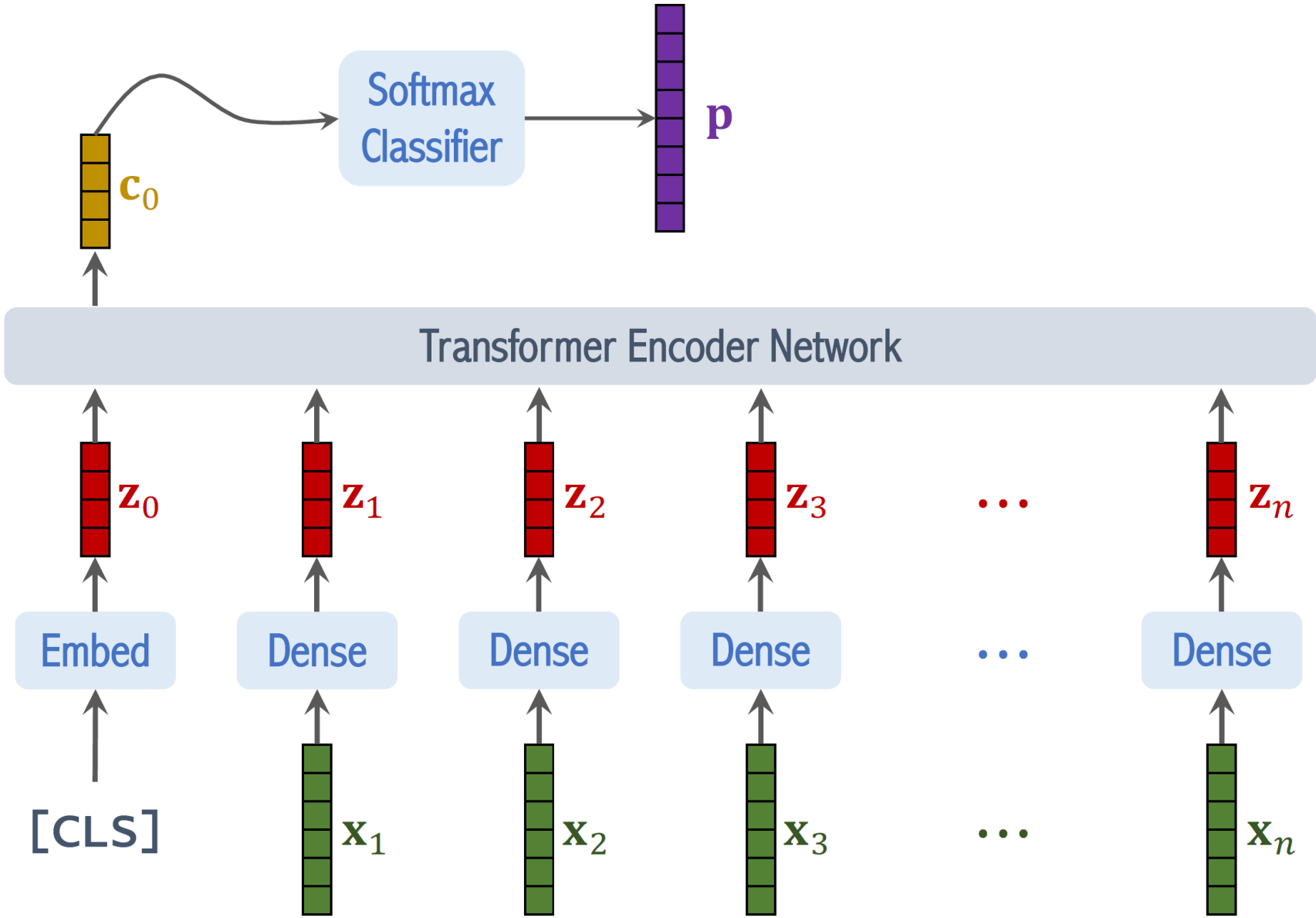
UNC CHARLOTTE

Vision Transformers (ViTs)



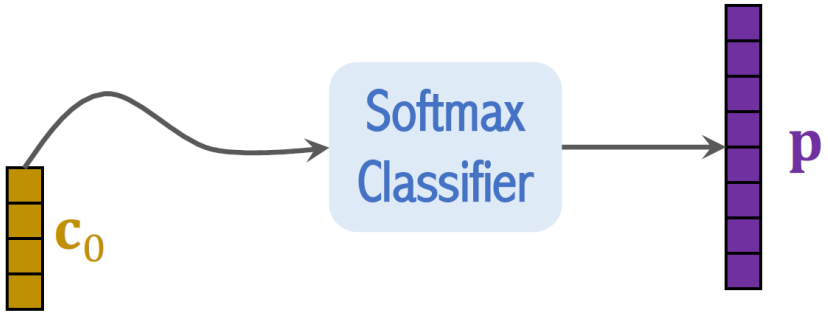
UNC CHARLOTTE

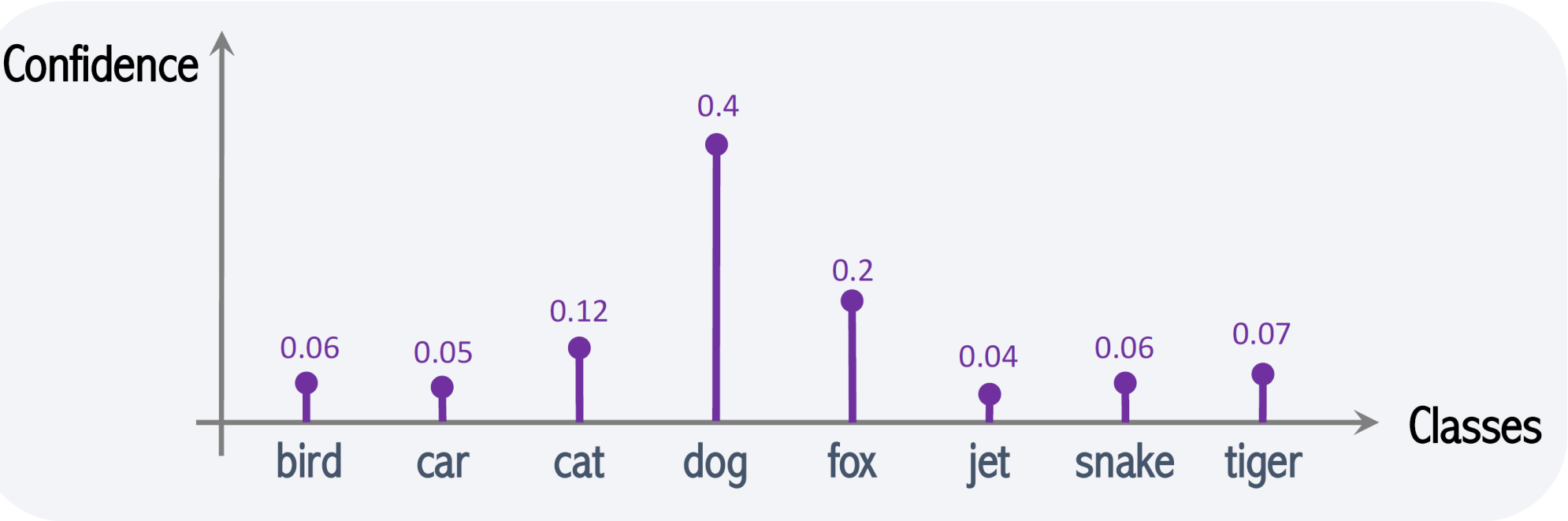
Vision Transformers (ViTs)



UNC CHARLOTTE

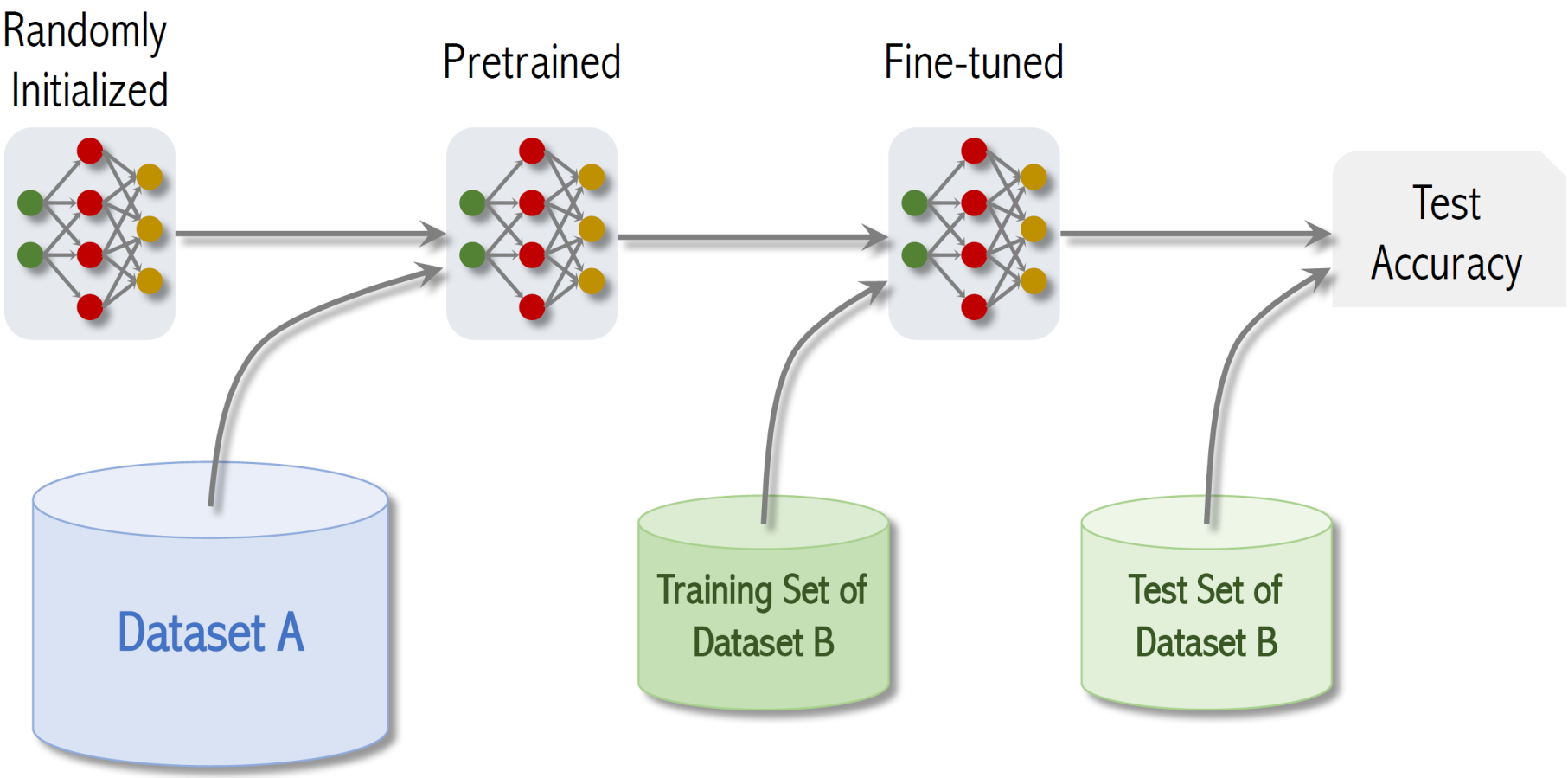
Vision Transformers (ViTs)





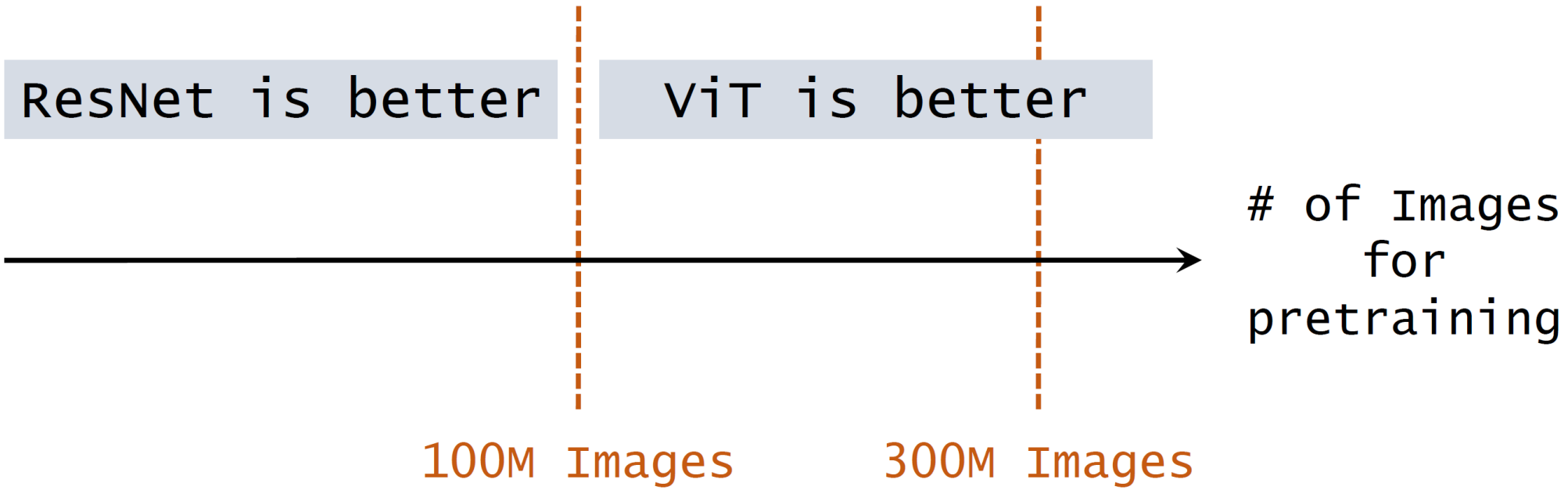
UNC CHARLOTTE

Vision Transformers (ViTs)



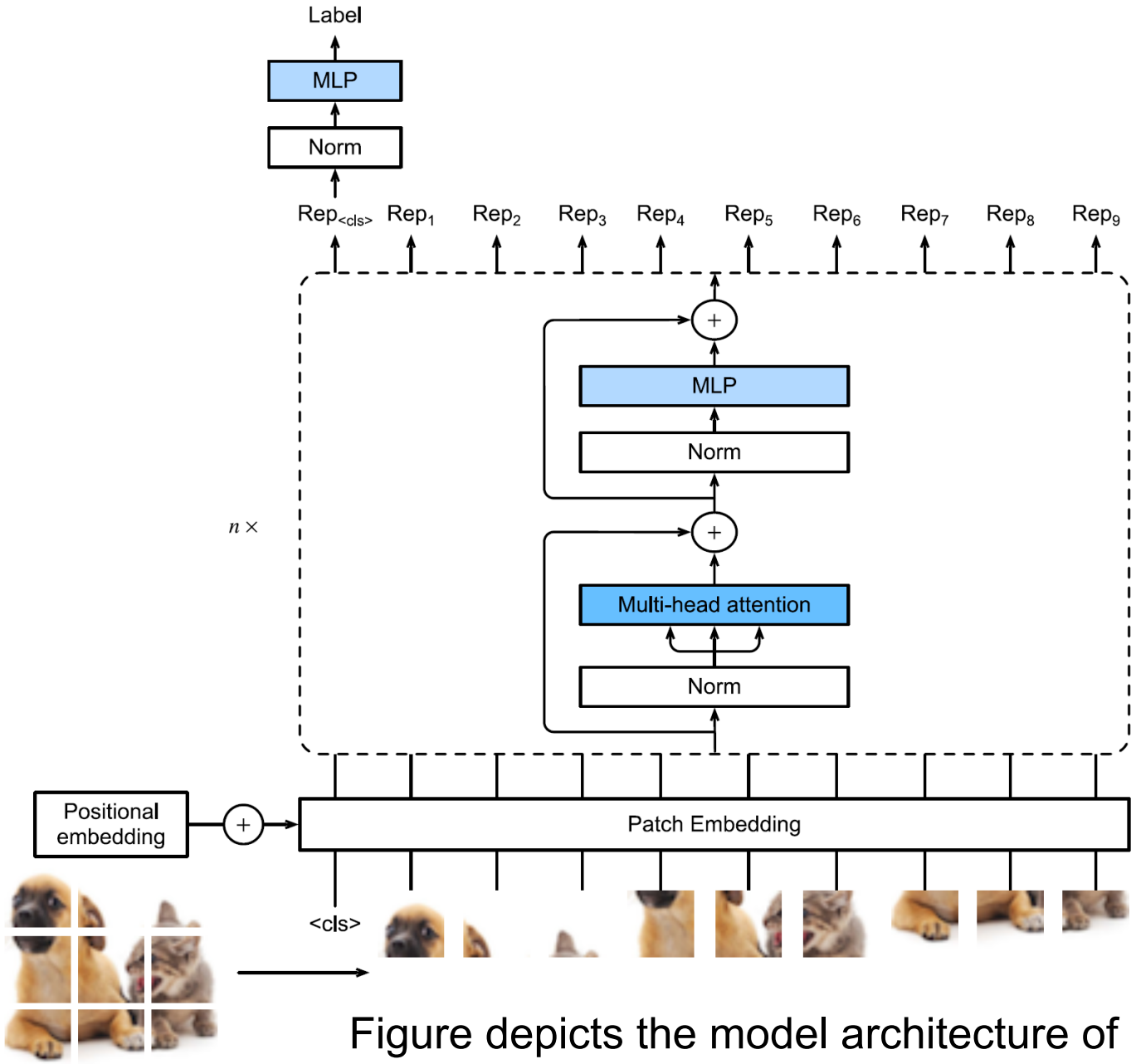
UNC CHARLOTTE

Vision Transformers (ViTs)



UNC CHARLOTTE

Model



- In this example, an image is split into 9 patches.
- A special <cls> token and the 9 flattened image patches are transformed via patch embedding
- N Transformer encoder blocks into 10 representations, respectively.
- The <cls> representation is further transformed into the output label.

Figure depicts the model architecture of vision Transformers.

UNC CHARLOTTE

Model

- This architecture consists of a stem that patchifies images, a body based on the multi-layer Transformer encoder, and a head that transforms the global representation into the output label.
- Consider an input image with height h, width w, and c channels.
  - Specifying the patch height and width both as p, the image is split into a sequence of $m = hw/p_2$ patches, where each patch is flattened to a vector of length cp_2.
  - In this way, image patches can be treated similarly to tokens in text sequences by Transformer encoders.
  - A special “<cls>” (class) token and the m flattened image patches are linearly projected into a sequence of $m + 1$ vectors, summed with learnable positional embeddings.
  - The multi-layer Transformer encoder transforms$m$+1 input vectors into the same amount of output vector representations of the same length.
- It works exactly the same way as the original Transformer encoder, only differing in the position of normalization.
- Since the “<cls>” token attends to all the image patches via self-attention, its representation from the Transformer encoder output will be further transformed into the output label.

Patch Embedding

- To implement a vision Transformer, let's start with patch embedding.
- Splitting an image into patches and linearly projecting these flattened patches can be simplified as a single convolution operation, where both the kernel size and the stride size are set to the patch size.

Vision Transformer Encoder

- The MLP of the vision Transformer encoder is slightly different from the position-wise FFN of the original Transformer encoder.
- First, here the activation function uses the Gaussian error linear unit (GELU), which can be considered as a smoother version of the ReLU (Hendrycks and Gimpel, 2016).
- Second, dropout is applied to the output of each fully connected layer in the MLP for regularization.

Vision Transformer Encoder

- The vision Transformer encoder block implementation just follows the pre-normalization design, where normalization is applied right *before* multi-head attention or the MLP.
- In contrast to post-normalization ("add & norm"), where normalization is placed right *after* residual connections, pre-normalization leads to more effective or efficient training for Transformers (Baevski and Auli, 2018, Wang *et al.*, 2019, Xiong *et al.*, 2020).

Putting It All Together

- The forward pass of vision Transformers below is straightforward.
- First, input images are fed into an PatchEmbedding instance, whose output is concatenated with the “<cls>” token embedding.
- They are summed with learnable positional embeddings before dropout.
- Then the output is fed into the Transformer encoder that stacks num_blks instances of the ViTBlock class.
- Finally, the representation of the “<cls>” token is projected by the network head.

Summary and Discussion

- You may notice that for small datasets like Fashion-MNIST, our implemented vision Transformer does not outperform the ResNet.
- Similar observations can be made even on the ImageNet dataset (1.2 million images). This is because Transformers *lack* those useful principles in convolution, such as translation invariance and locality.
- However, the picture changes when training larger models on larger datasets (e.g., 300 million images),
- where vision Transformers outperform ResNets by a large margin in image classification,
- demonstrating intrinsic superiority of Transformers in scalability (Dosovitskiy *et al.* , 2021).
- The introduction of vision Transformers has changed the landscape of network design for modeling image data. They were soon shown effective on the ImageNet dataset with dataefficient training strategies of DeiT (Touvron *et al.*, 2021).
- However, quadratic complexity of self-attention makes the Transformer architecture less suitable for higher resolution images.
- Towards a general-purpose backbone network in computer vision, Swin Transformers addressed the quadratic computational complexity with respect to image size and added back convolution-like priors, extending the applicability of Transformers to a range of computer vision tasks beyond image classification with state-of-the-art results (Liu *et al.*, 2021).