Manifold Learning and Artificial Intelligence Lecture 12

Vision Transformer and its Applications

Momiao Xiong, University of Texas School of Public Health

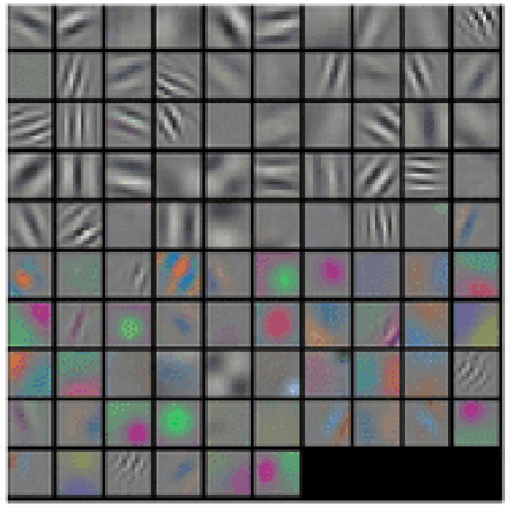
- Time: 10:00 pm, US East Time, 04/15/2023
- 10:00 am, Beijing Time. 04/16/2023

Github Address: https://ai2healthcare.github.io/

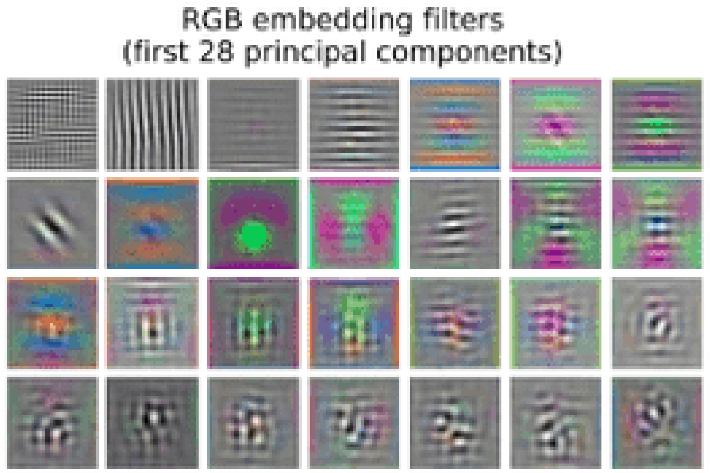
Vision Transformer: What It Is & How It Works [2023 Guide]

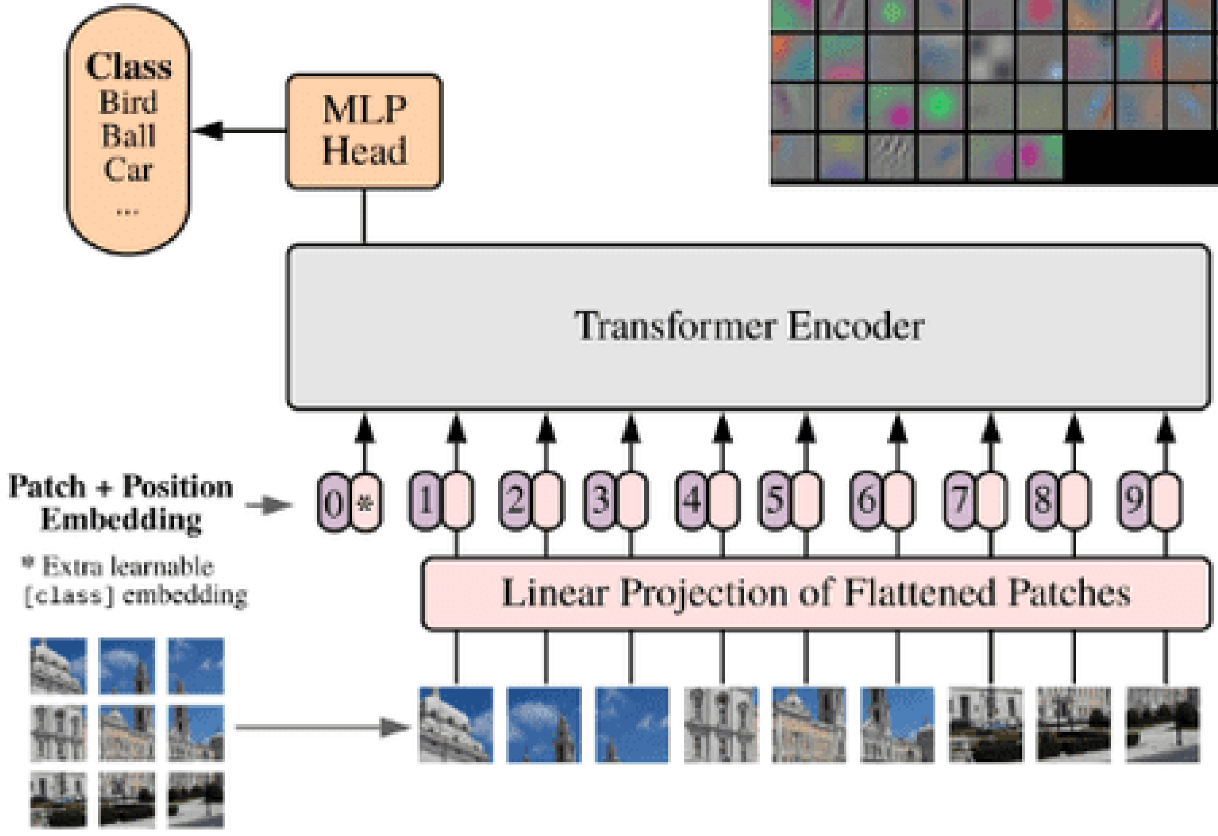
Deval Shah, March 2, 2023

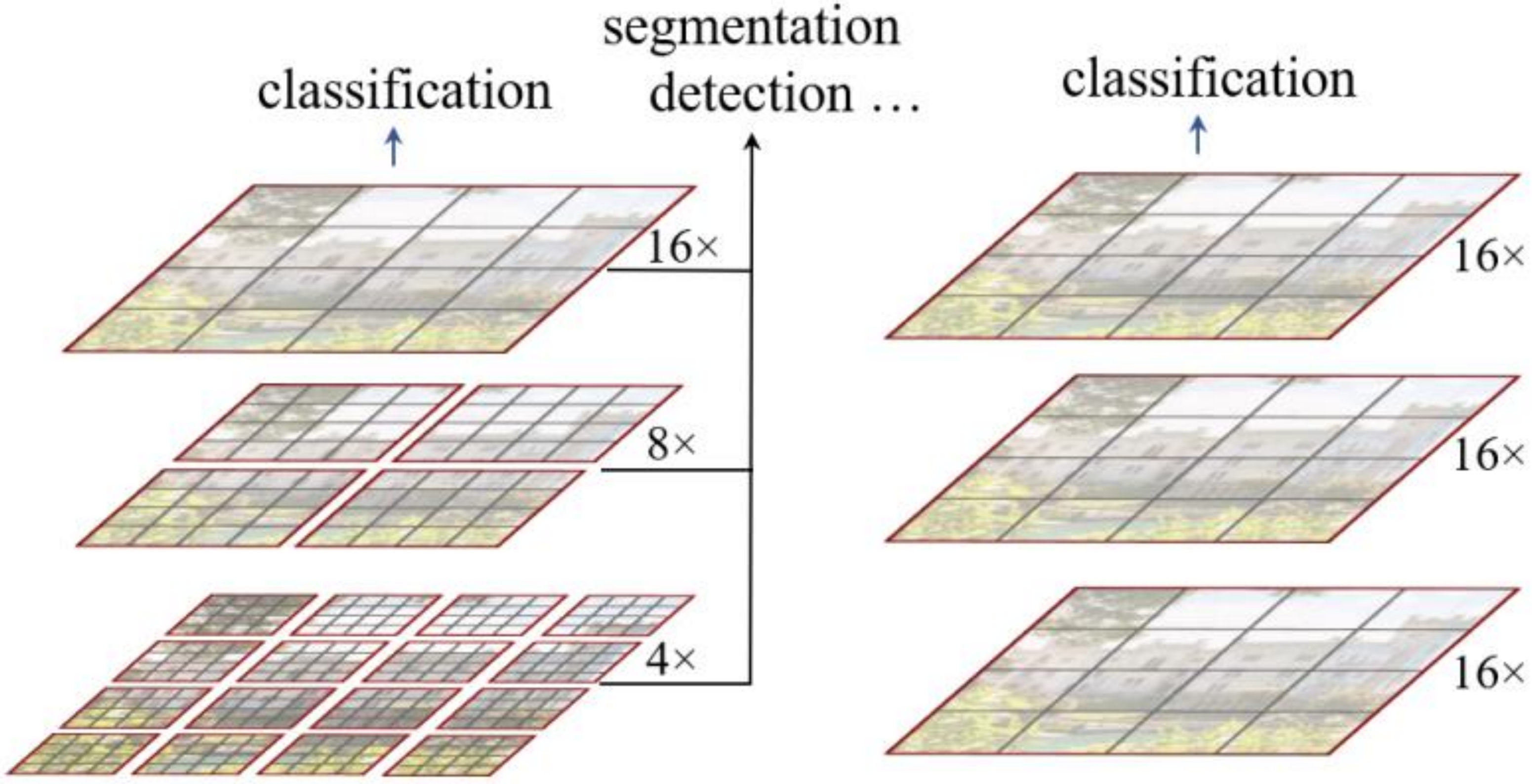
Alexnet 1st conv filters



ViT 1st linear embedding filters



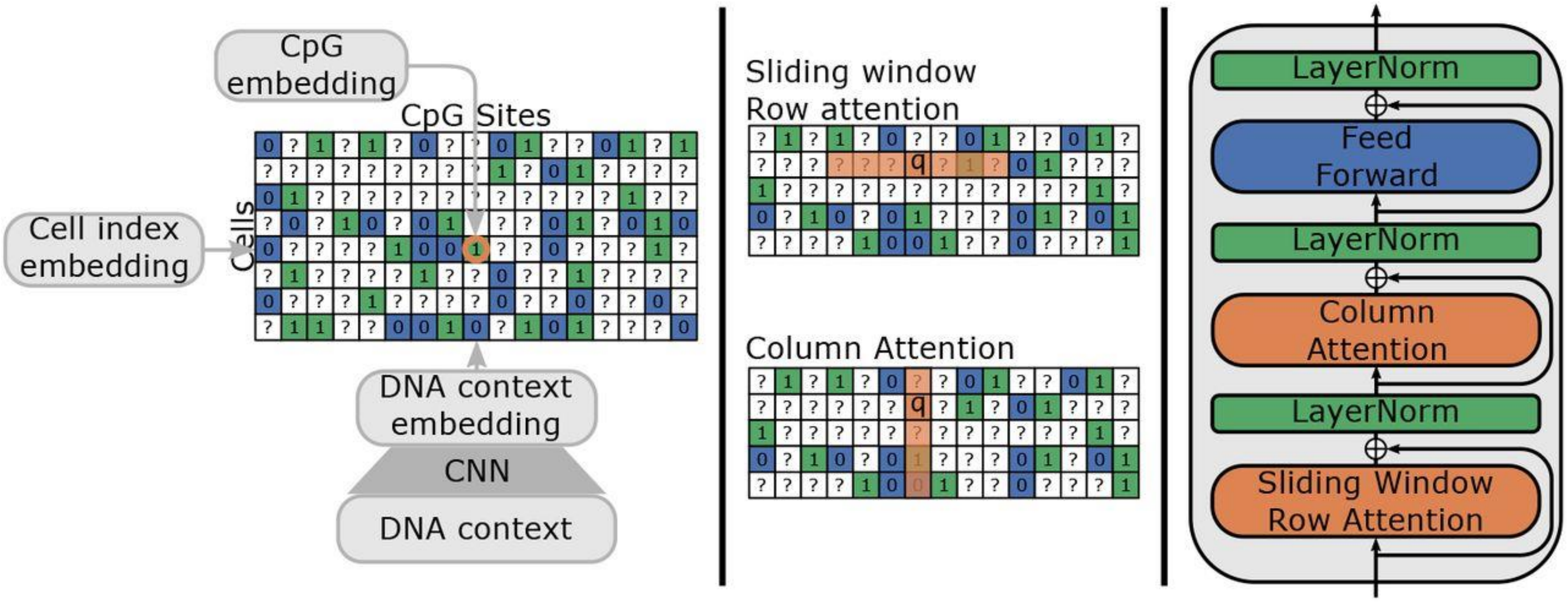

Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0
*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches



(a) Swin Transformer (ours)

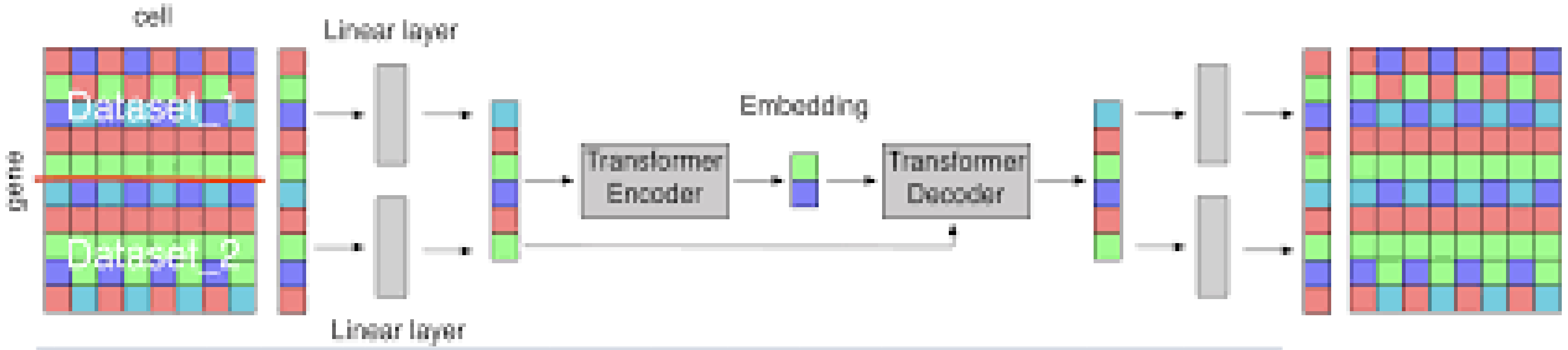
(b) ViT

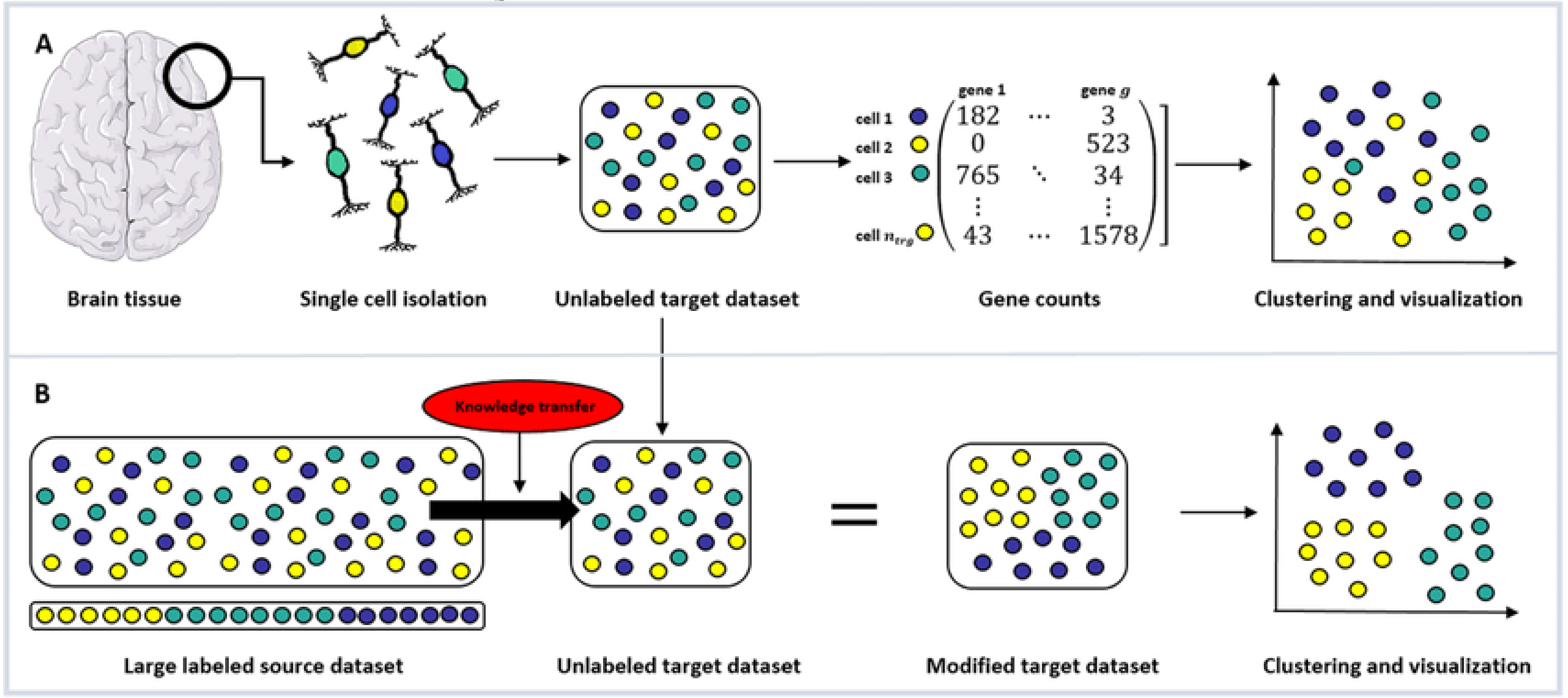
CpG Transfomer



CpG Transformer for imputation of single-cell methylomes, Gaetan DeWaele et al. 2021

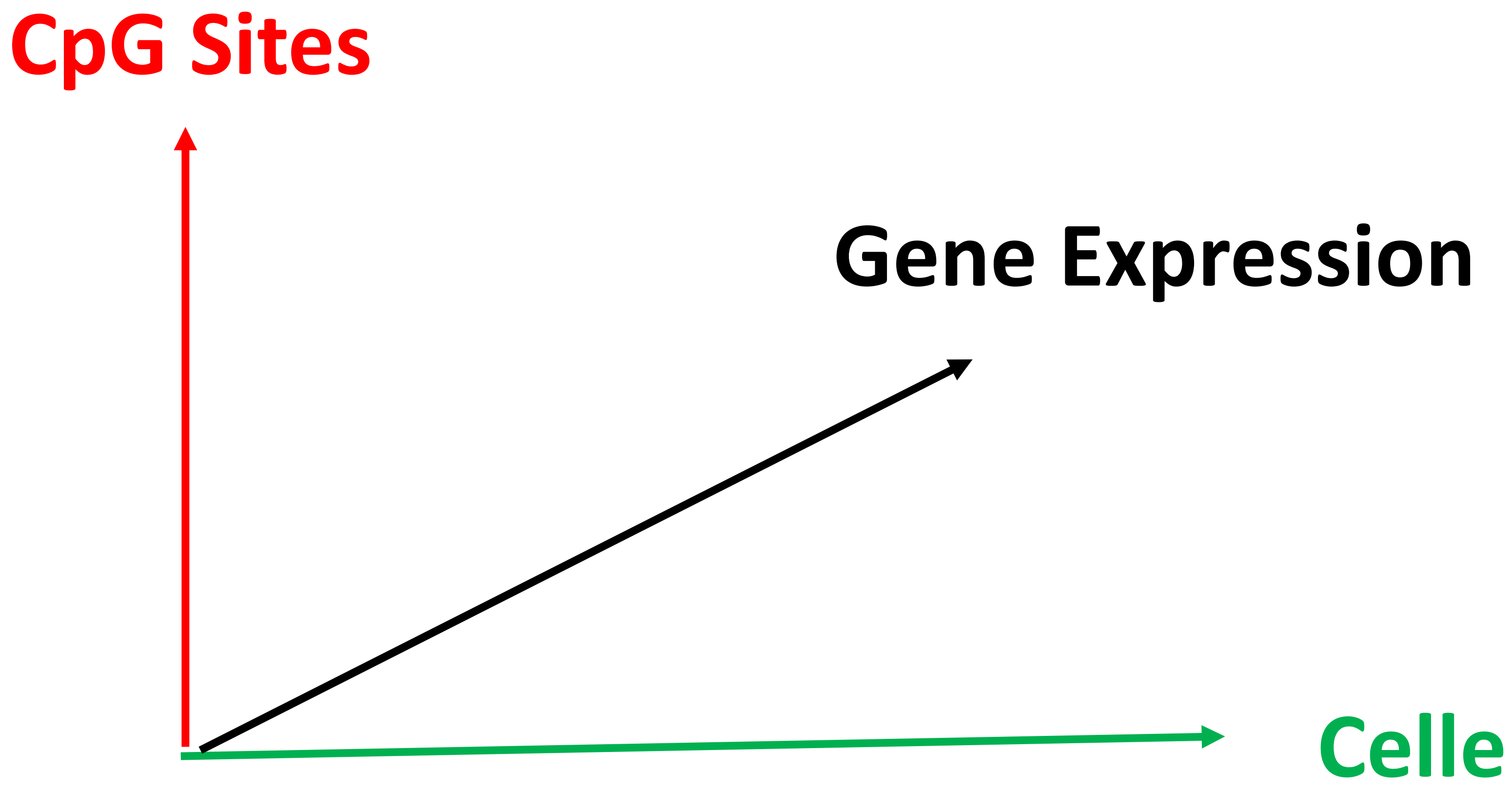
Gene Expression Transformer



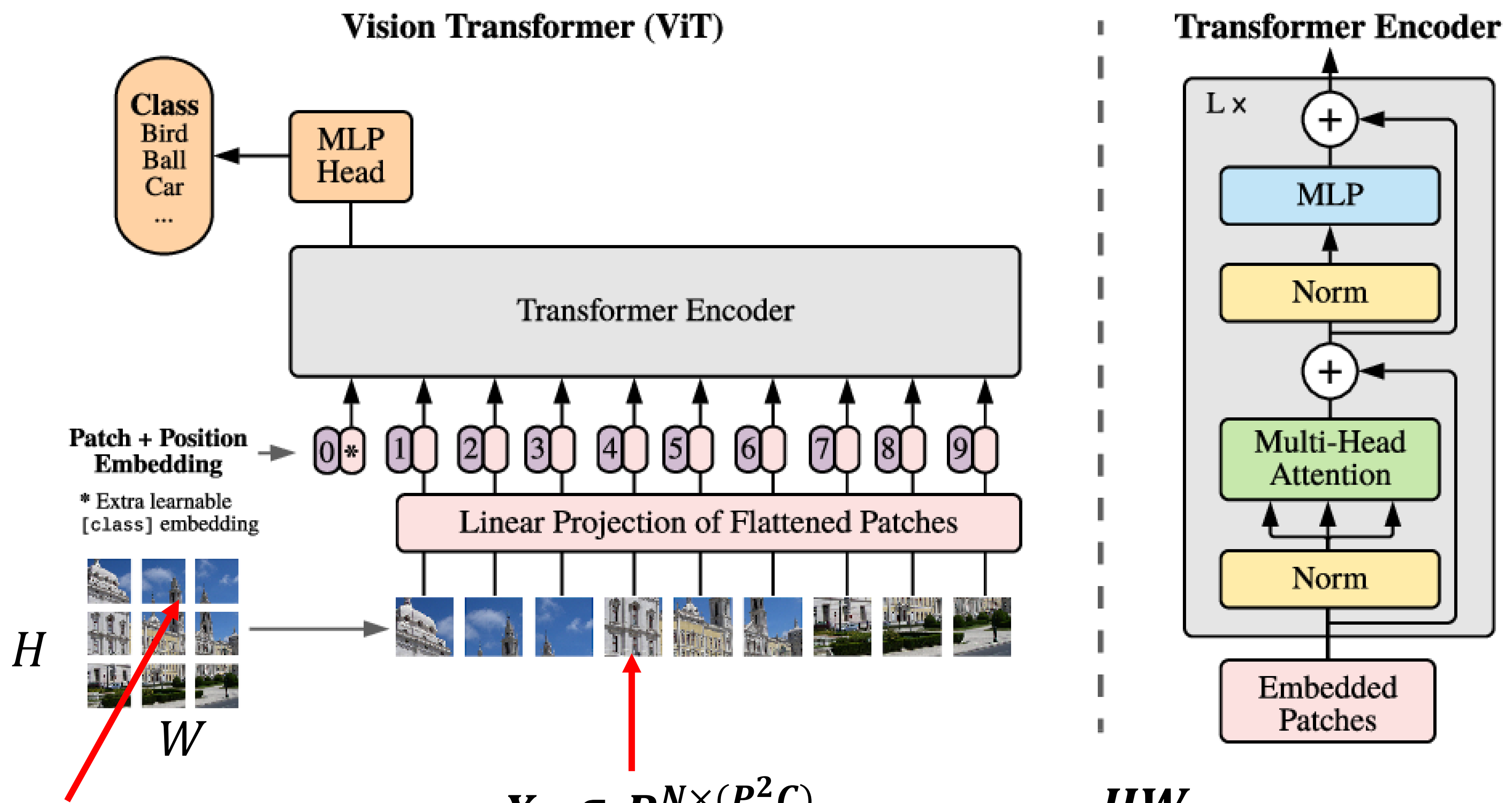


MULTITASK LEARNING FOR TRANSFORMERS WITH APPLICATION TO LARGE-SCALE SINGLE-CELL TRANSCRIPTOMES

Minxing Pang (2021)


CpG Sites
Gene Expression
Celle

Split an image into patches



$X \in R^{H \times W \times C}$

$X_p \in R^{N \times (P^2 C)}$

C: Number of Channels

$N = \frac{HW}{P^2}, (P, P)$: *size of Patche*

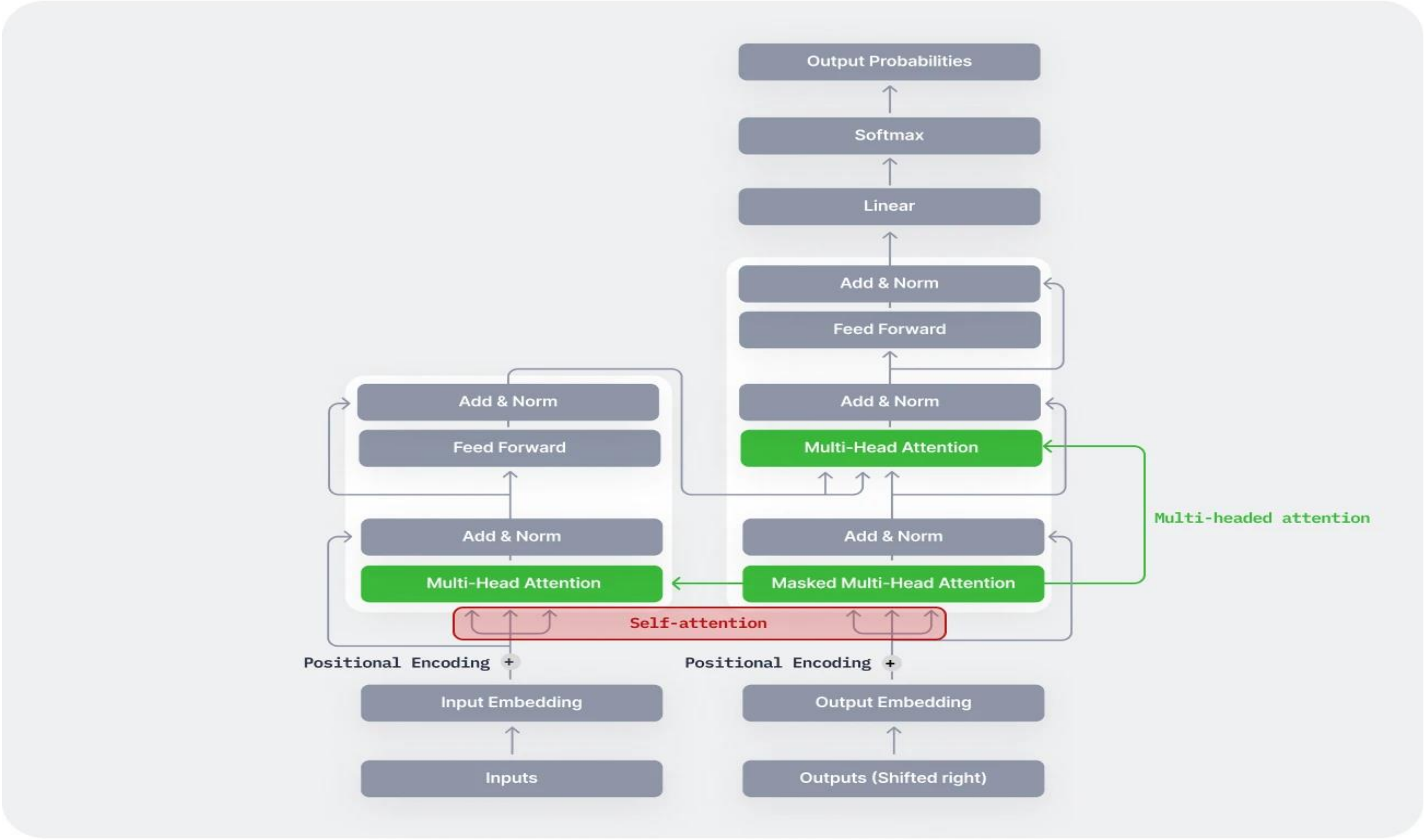
There are multiple blocks in the ViT encoder, and each block consists of three major processing elements:

- Layer Norm
- Multi-head Attention Network (MSA)
- Multi-Layer Perceptrons (MLP)

Layer Norm keeps the training process on track and lets the model adapt to the variations among the training images.

Multi-head Attention Network (MSA) is a network responsible for generating attention maps from the given embedded visual tokens. These attention maps help the network focus on the most critical regions in the image, such as object(s). The concept of attention maps is the same as that found in the traditional computer vision literature (e.g., saliency maps and alpha-matting).

MLP is a two-layer classification network with GELU (Gaussian Error Linear Unit) at the end. The final MLP block also called the MLP head, is used as an output of the transformer. An application of softmax on this output can provide classification labels (i.e., if the application is Image Classification).


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Multi-headed attention
Self-attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (Shifted right)

flatten the patches and map to D dimensions with a trainable linear projection

$$\boldsymbol{Z_0 = \left[X_{class};\ X_p^1 E; \dots, X_p^N E\right] + E_{pos}} \qquad \boldsymbol{E \in R^{(P^2.C)\times D}, E_{pos} \in R^{(N+1)\times D}}$$

$$\boldsymbol{Z_0 \in R^{(N+1)\times D}} \qquad \boldsymbol{X_p^i \in R^{P^2 C}}$$

- **Layer normalization (LN)**

$$Y = LN(Z_{l-1}) = \begin{bmatrix} LN(Z_{l-1}^0) \\ \vdots \\ LN(Z_{l-1}^N) \end{bmatrix}, \mathrm{LN}\left(Z_{l-1}^i\right) = \begin{bmatrix} LN(Z_{l-1}^{i,1}) \\ \vdots \\ LN(Z_{l-1}^{i,D}) \end{bmatrix}$$

Layer normalization (LN)

$$\mu_i = \frac{1}{D}\sum_{k=1}^{D} Z_{l-1}^{i,k}, \sigma_i^2 = \frac{1}{D}\sum_{k=1}^{D}\left(z_{l-1}^{i,k} - \mu_i\right)^2$$

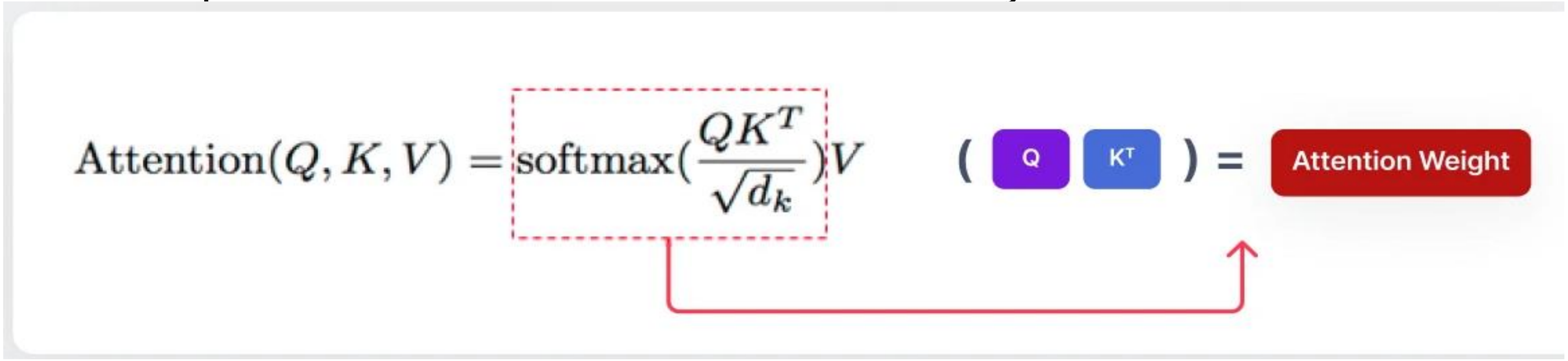
$$\hat{z}_{l-1}^{i,k} = \frac{z_{l-1}^{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}}$$

$$Z_{l-1}^{i} = \begin{bmatrix} z_{l-1}^{i,1} \\ \vdots \\ z_{l1}^{i,D} \end{bmatrix}, \mathrm{Y} = \begin{bmatrix} y^0 \\ \vdots \\ y^N \end{bmatrix}$$

$$\boldsymbol{Y = LN(Z_{l-1}) = \begin{bmatrix} LN(z_{l-1}^{0}) \\ \vdots \\ LN(z_{l-1}^{N}) \end{bmatrix}}$$

$$y_i = LN_{\gamma,\beta}\left(Z_{l-1}^{i}\right) = \gamma \hat{Z}_{l-1}^{i} + \beta, i = 0, 1, \ldots, N$$

The attention mechanism used in the Transformer uses three variables: Q (Query), K(Key), and V (Value). Simply put, it calculates the attention weight of a Query token (token: something like a word) and a Key token and multiplies the Value associated with each Key. In short, it calculates the association (attention weight) between the Query token and the Key token and multiplies the Value associated with each Key.



$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$
(Q K^T) =
Attention Weight

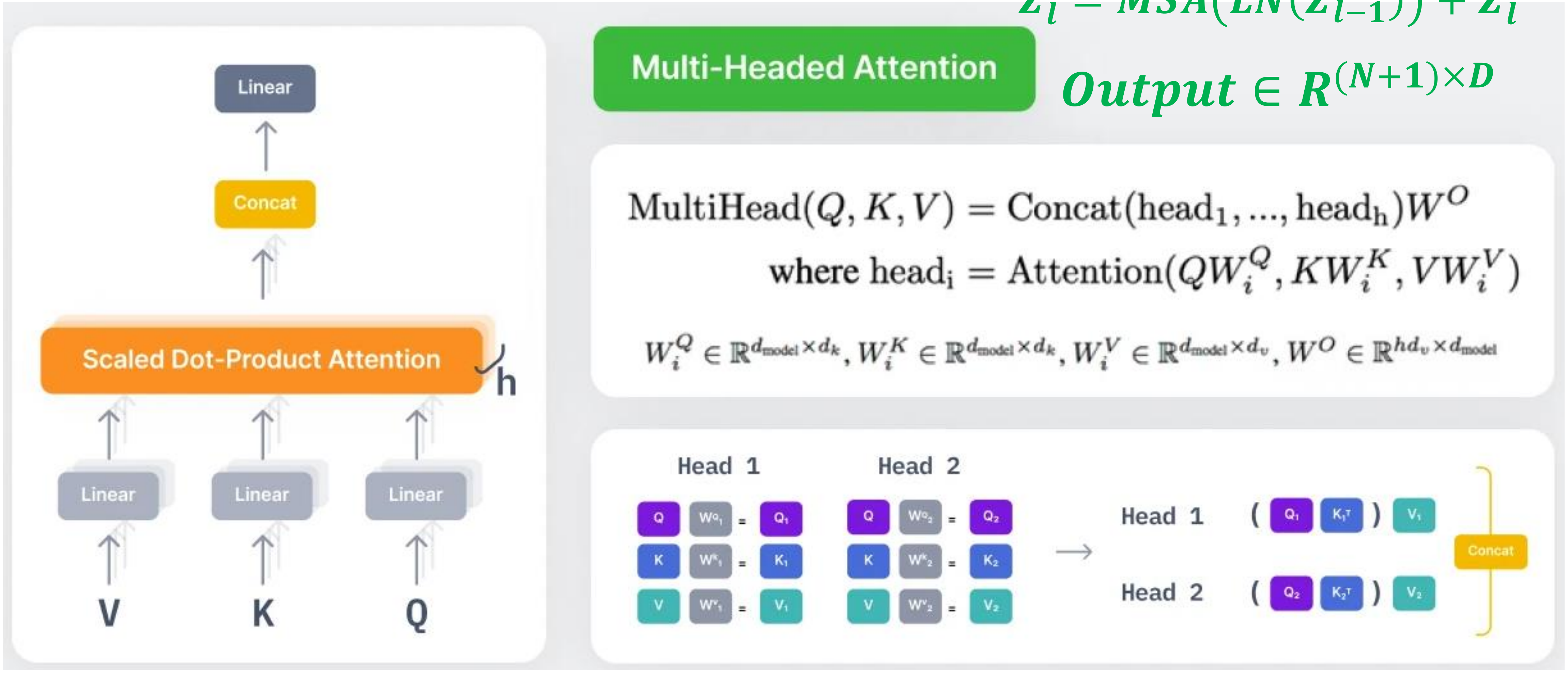
$$[Q,K,V] = ZW_{QKV}, Z \in R^{(N+1)\times D}, W_{QKV} \in R^{D\times 3D_h}$$

$$A = softmax\left(\frac{QK^T}{\sqrt{D_h}}\right), A \in R^{(N+1)\times(N+1)}, \qquad SA(Z) = AV \in R^{(N+1)\times D_h}$$

Defining the Q, K, and V calculation as a single head, the multi-head attention mechanism is defined as follows.

$$Z'_l = MSA(LN(Z_{l-1})) + Z'_l$$

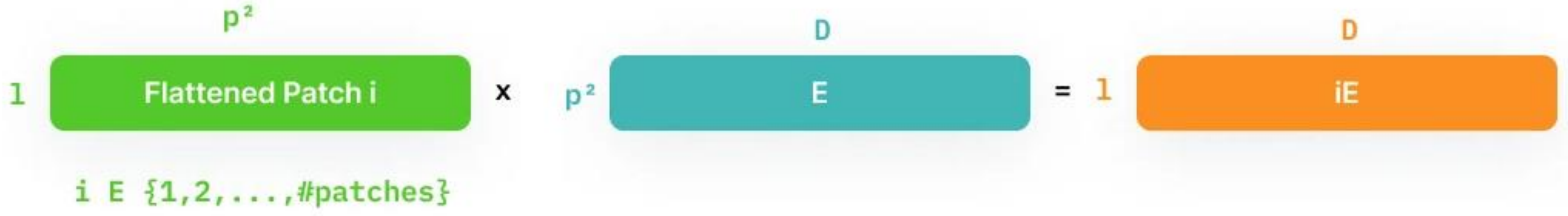
$$Output \in R^{(N+1)\times D}$$

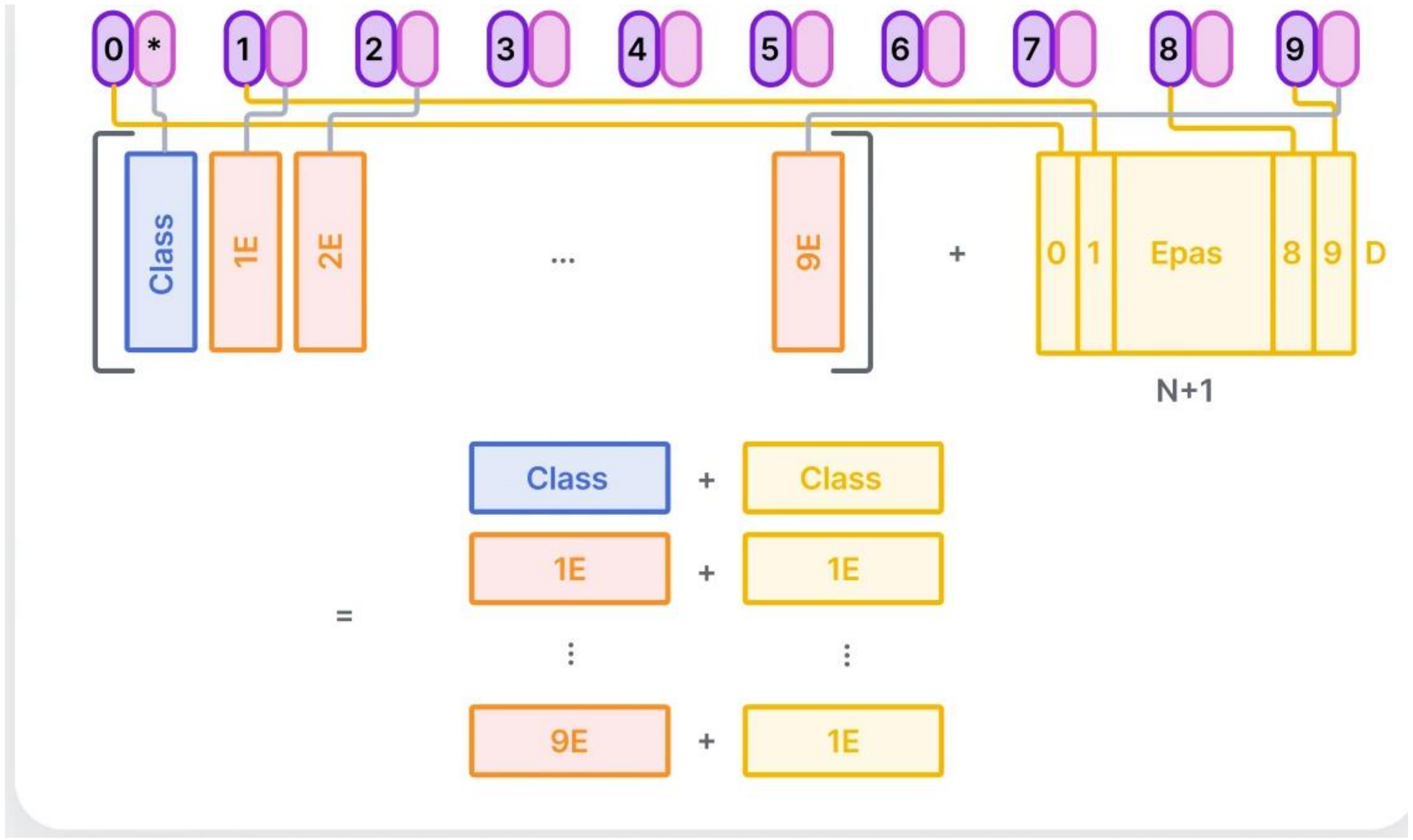


$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1},\ldots,\mathrm{head_h})W^O$$

$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$W_i^Q \in \mathbb{R}^{d_{\text{model}}\times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}}\times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}}\times d_v}, W^O \in \mathbb{R}^{hd_v\times d_{\text{model}}}$$





D is the fixed latent vector size

MLP Layer

$$Z_l = MLP\left(LN(Z_l')\right) + Z_l', l = 1, \dots, L$$

$$MLP\left(LN(Z_l')\right) = W_2\sigma\left(W_1 LN(Z_1') + b_1\right) + b_2$$

$$W_1, W_2 \in R^{(N+1)\times(N+1)}$$

$$\sigma = GELU(x) = xP(X \leq x) = \mathrm{x}\Phi(x) \approx 0.5x(1 + \tanh(\sqrt{\frac{2}{\pi}}(x + 0.0447x^3))$$

$GELU: Gaussian\ Error\ Linear\ Unit$

$$Y = LN(Z_L^0) \in R^D$$

$$\boldsymbol{N = \frac{HW}{P^2}}$$

Copy from ChatGPT

The MLP (Multi-Layer Perceptron) layer is one of the key components of the Transformer, which is a type of neural network architecture used for natural language processing tasks, such as machine translation, text classification, and language modeling. The Transformer was introduced in the paper "Attention is All You Need" by Vaswani et al. in 2017, and has become a popular model in the field of deep learning due to its ability to effectively model long-range dependencies in sequences.

The MLP layer in the Transformer is used to process the input embeddings of each token in the sequence independently and non-linearly. It consists of two feed-forward layers with a ReLU activation function applied in between. The MLP layer is applied independently to each token in the sequence, which allows for parallel processing of the tokens, making it highly efficient for handling long sequences.

The input to the MLP layer is a tensor of shape (batch_size, seq_length, d_model), where batch_size is the number of input sequences in a batch, seq_length is the length of the input sequences, and d_model is the dimensionality of the input embeddings. The output of the MLP layer is also a tensor of the same shape (batch_size, seq_length, d_model).

The MLP layer can be mathematically described as follows:
First, a linear transformation is applied to the input tensor, which involves multiplying the input tensor by a learnable weight matrix and adding a learnable bias vector. This operation can be expressed as:
linear_1_output = input_tensor * W1 + b1

where W1 is the weight matrix and b1 is the bias vector for the first linear transformation.
Next, a ReLU activation function is applied element-wise to the linear_1_output, which introduces non-linearity to the model:
relu_output = ReLU(linear_1_output)

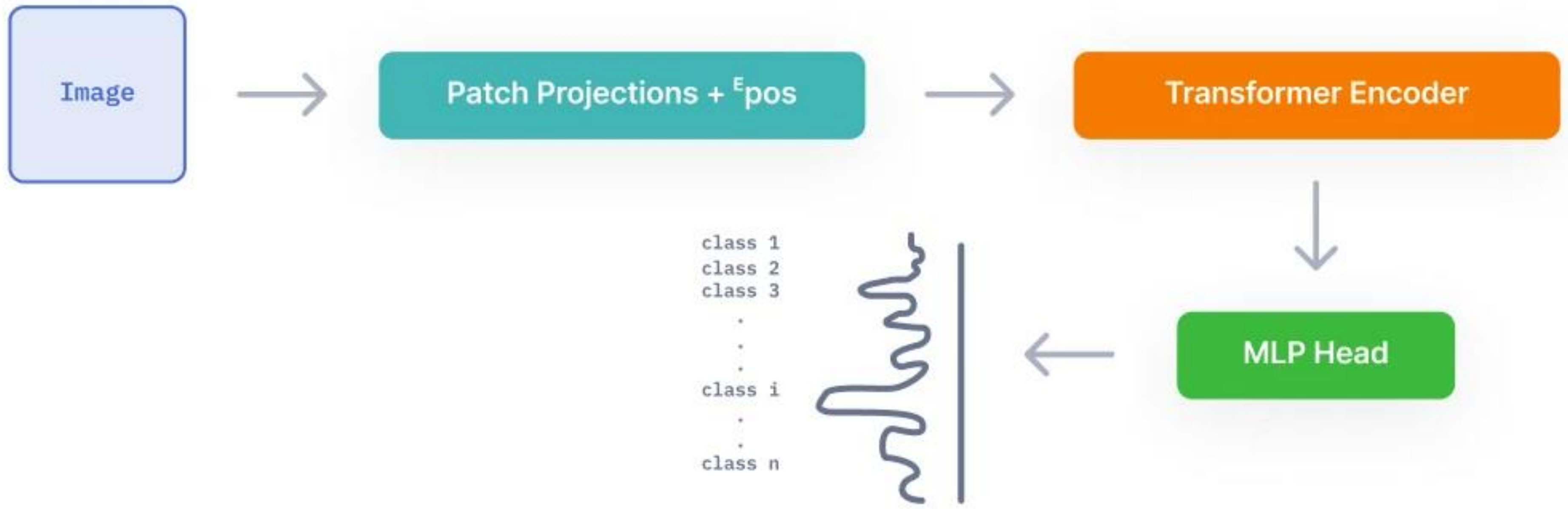
Finally, another linear transformation is applied to the output of the ReLU activation, which involves multiplying the output of the ReLU activation by another learnable weight matrix and adding another learnable bias vector: linear_2_output = relu_output * W2 + b2

where W2 is the weight matrix and b2 is the bias vector for the second linear transformation.

```
import torch
import torch.nn as nn
class MLP(nn.Module):
    def __init__(self, d_model, d_ff):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(d_model, d_ff)
        self.fc2 = nn.Linear(d_ff, d_model)
        self.relu = nn.ReLU()

    def forward(self, x):
        """
        Args:
            x: Input tensor of shape (batch_size, seq_length, d_model)
        Returns:
            mlp_output: Output tensor of MLP layer of shape (batch_size, seq_length, d_model)
        """
        # Linear transformation 1
        linear1_output = self.fc1(x)
        relu_output = self.relu(linear1_output)
        # Linear transformation 2
        linear2_output = self.fc2(relu_output)
        # Additive residual connection
        mlp_output = x + linear2_output
        return mlp_output
```

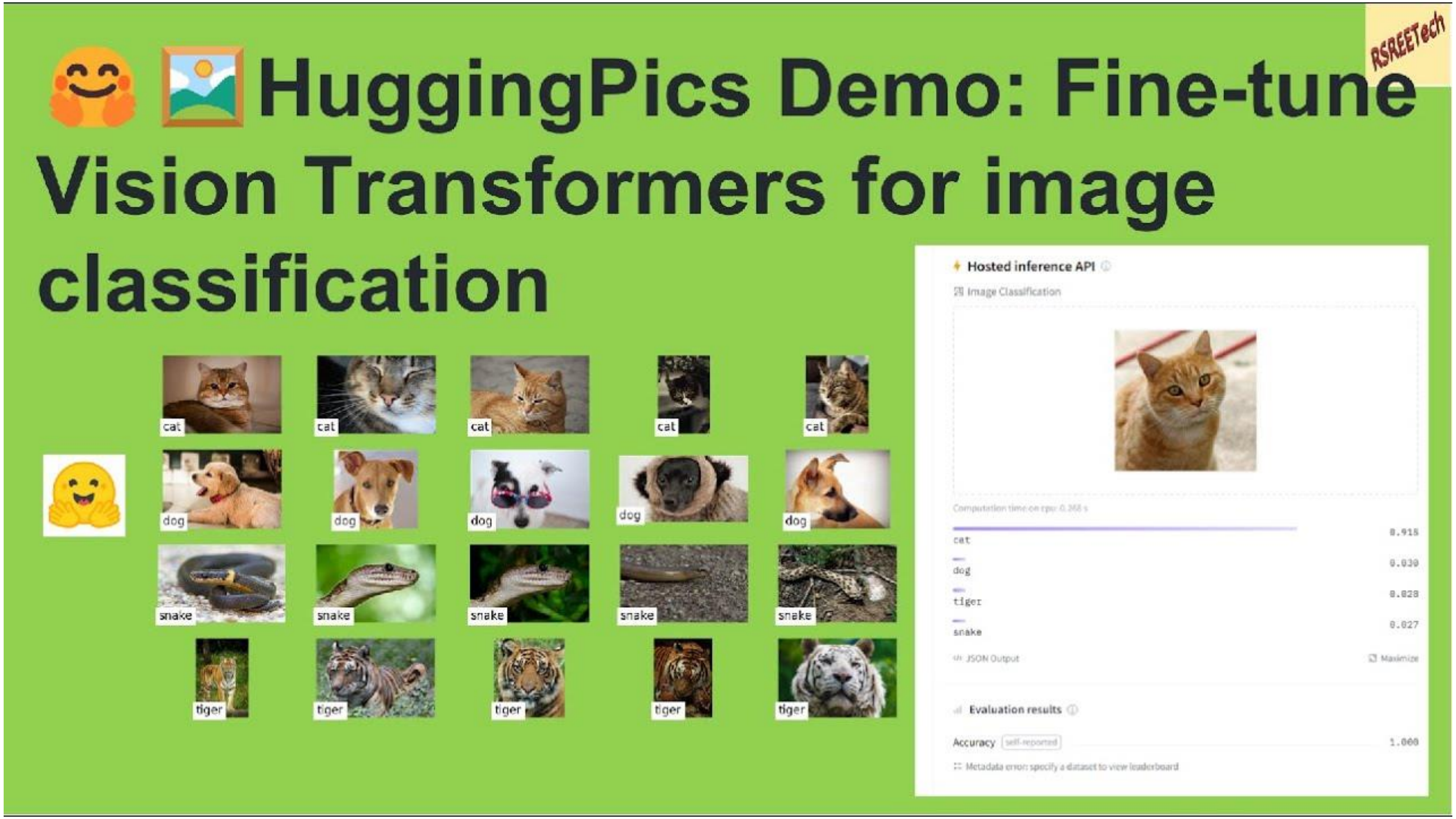
Altogether, these patch projections and positional embeddings form a larger matrix that'll soon be put through the Transformer Encoder.



The MLP Head inputs the Transformer outputs related to the special [class] embedding and ignores the other outputs.

Six applications of Vision Transformers

Image Classification



HuggingPics Demo: Fine-tune Vision Transformers for image classification

Hosted inference API

Image Classification

Computation time on cpu: 0.368 s

```
cat      0.915
dog      0.030
tiger    0.028
snake    0.027
```

JSON Output

Maximize

Evaluation results

Accuracy self-reported 1.000

Metadata error: specify a dataset to view leaderboard

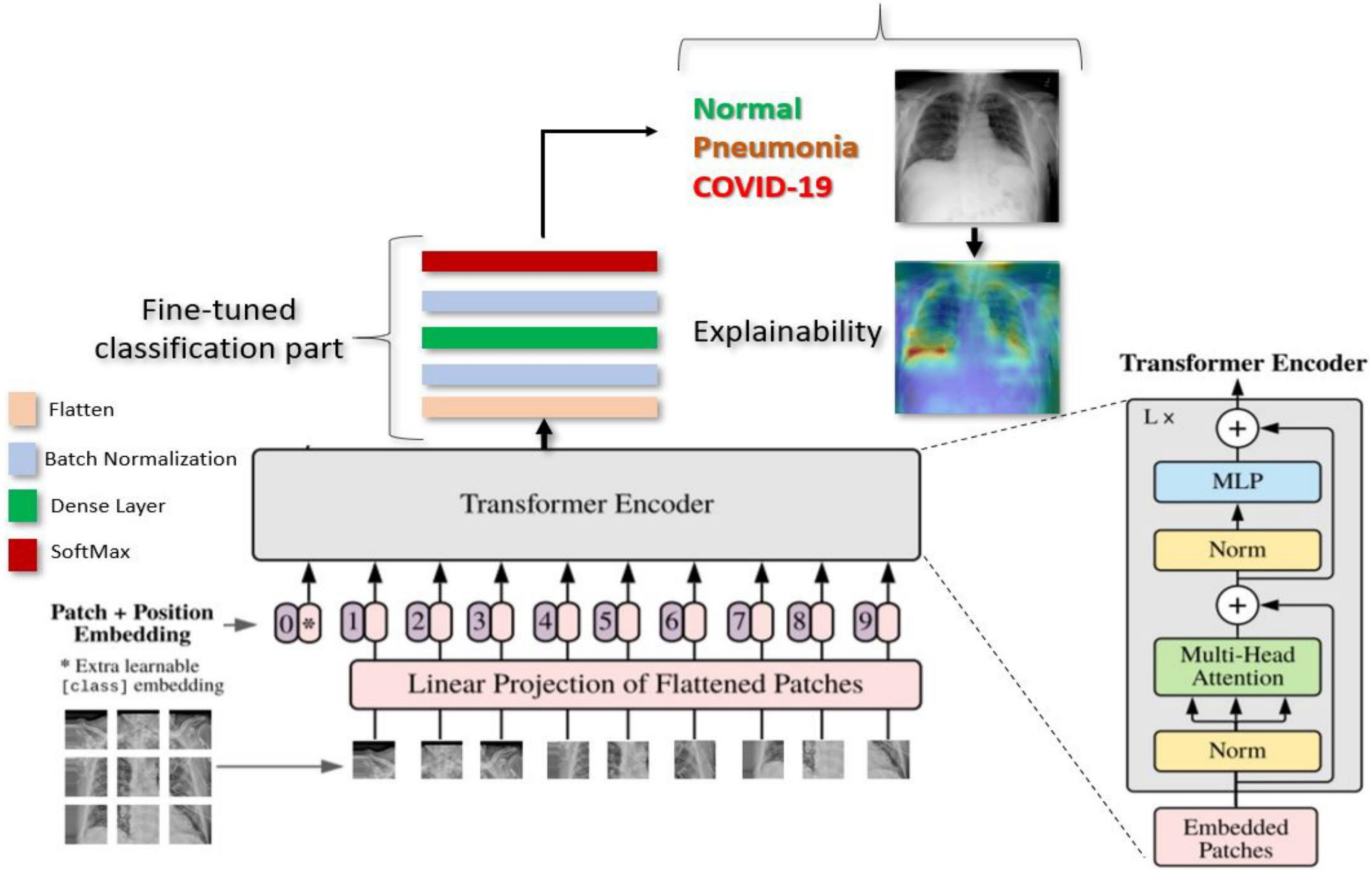

Normal
Pneumonia
COVID-19
Fine-tuned classification part
Explainability
Flatten
Batch Normalization
Dense Layer
SoftMax
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
Linear Projection of Flattened Patches
0 1 2 3 4 5 6 7 8 9
Transformer Encoder
L ×
MLP
Norm
Multi-Head Attention
Norm
Embedded Patches

Image captioning

A person on a dirt bike jumping in the air.

A cat sitting on the edge of a sink.

A person on a motorcycle is racing on a track.

Image segmentation

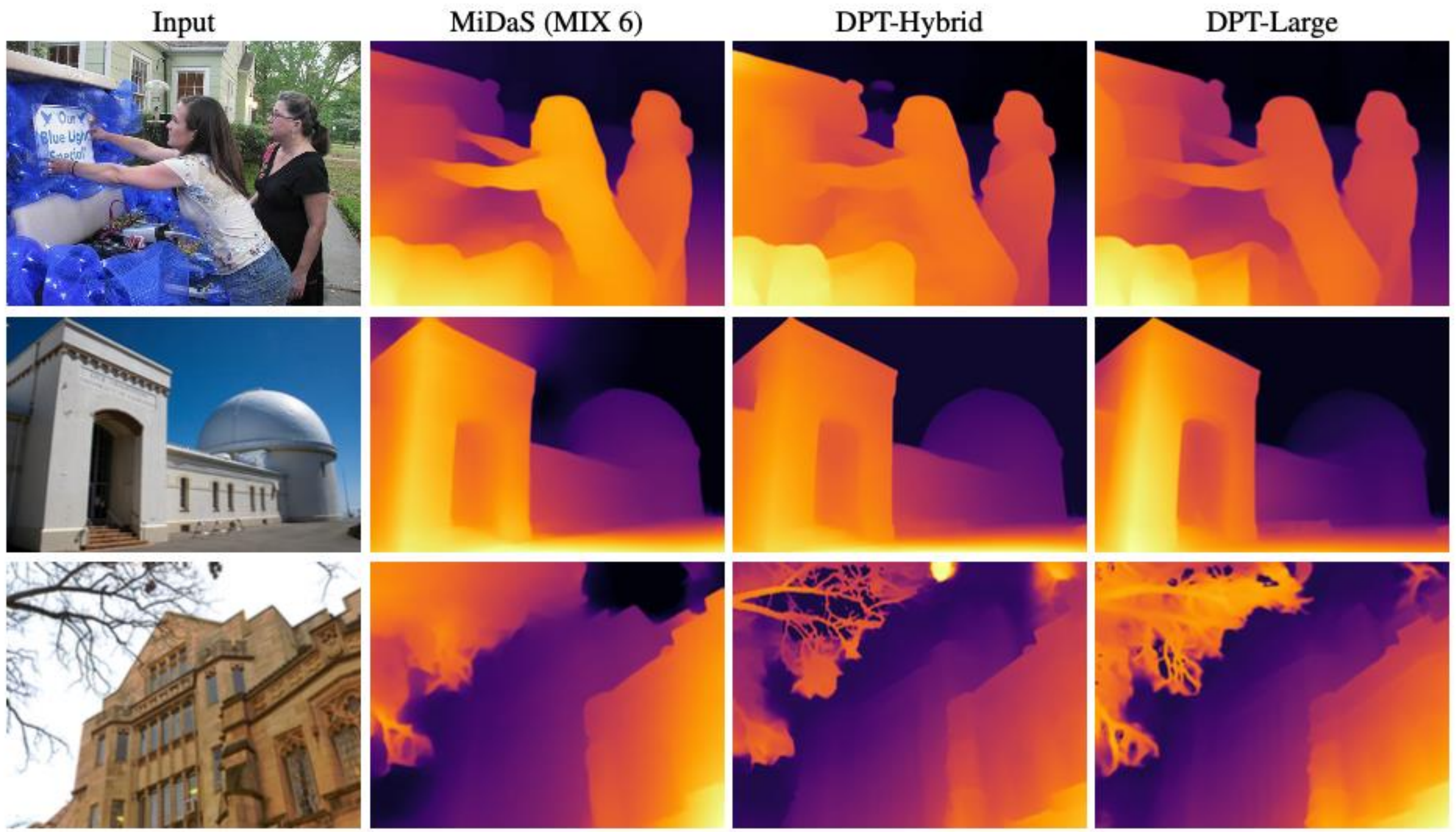
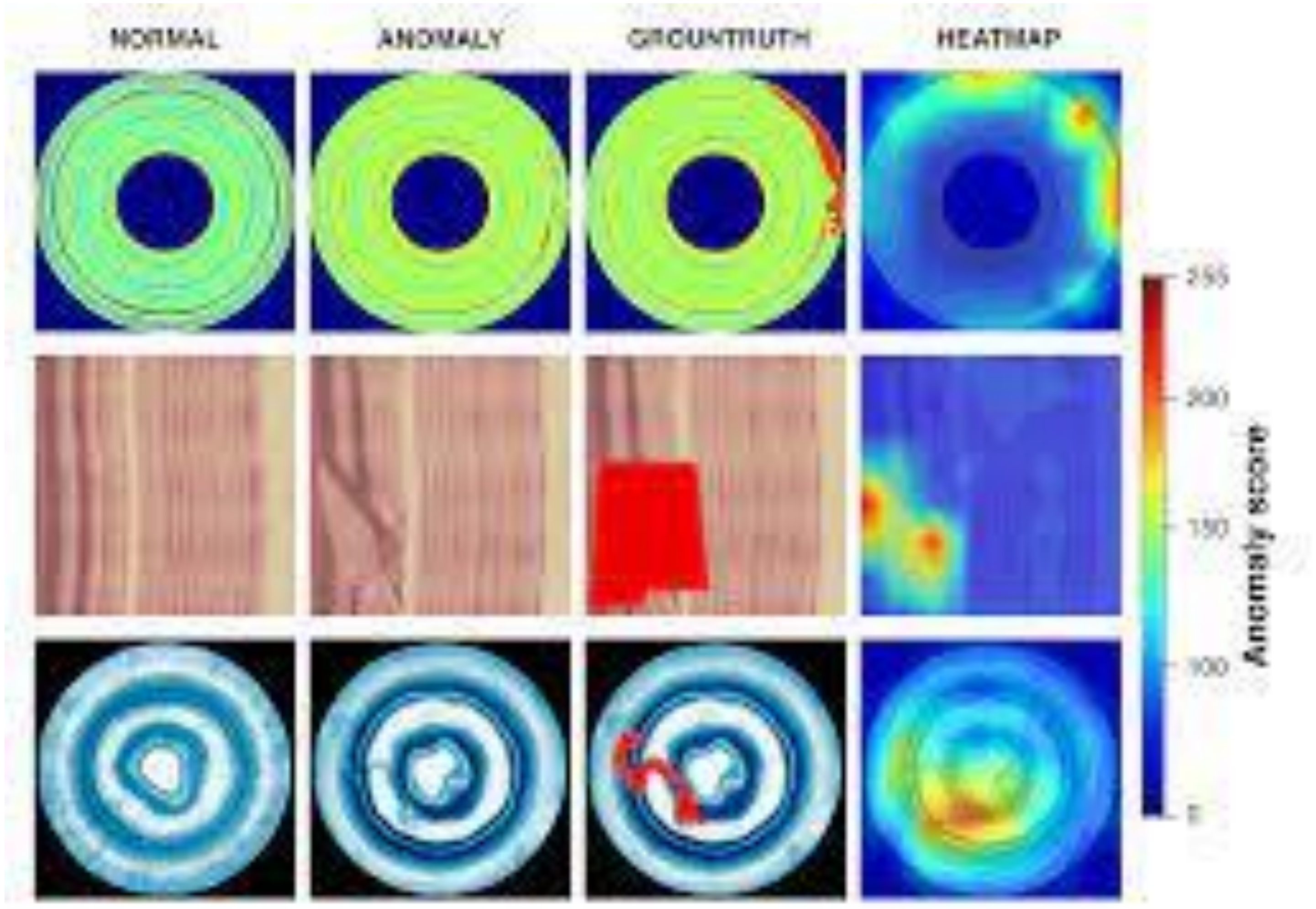


Figure 2. Sample results for monocular depth estimation. Compared to the fully-convolutional network used by MiDaS, DPT shows better global coherence (e.g., sky, second row) and finer-grained details (e.g., tree branches, last row).

DPT (DensePredictionTransformers)

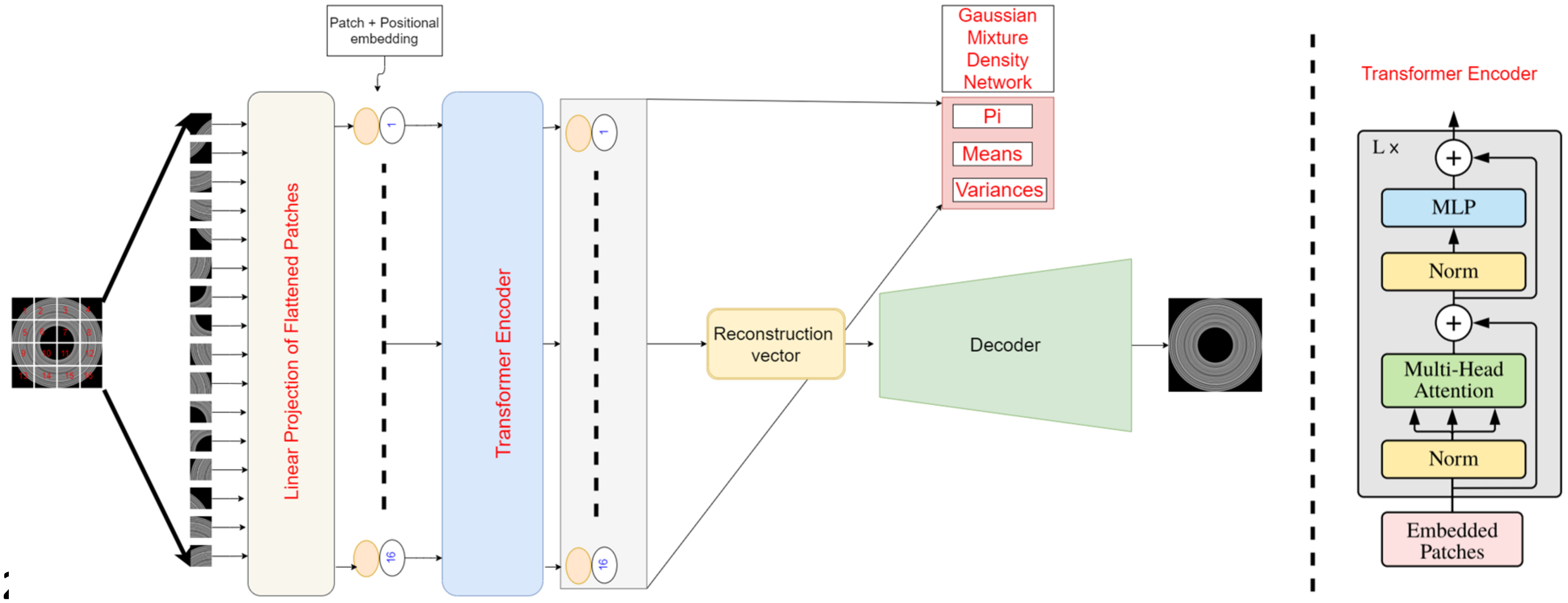
Anomaly detection

VT-ADL: A Vision Transformer Network for Image Anomaly Detection and Localization



Code: https://github.com/pankajmishra000/VT-ADL

An overview of the model



$X_p \in R^{N\times(P\times P\times C)}$. $N = \dfrac{HW}{P^2}$, (P, P): *patch size*, D: *dimension of embedding of a Patch*

Patch 1 , X_p^1 **Patch 2 , X_p^2** … **Patch N , X_p^N**

$X_p^i \in R^{P\times P\times\times C}$

Transformer Encoder

- The input patches are first mapped to the embedding space and are augmented with positional information

$$\boldsymbol{Z_0} = \left[\boldsymbol{X_{class}};\ \boldsymbol{X_p^1 E}; \dots, \boldsymbol{X_p^N E}\right] + \boldsymbol{E_{pos}} \qquad (1)$$

$$\boldsymbol{E} \in \boldsymbol{R}^{(P^2.C)\times D}, \boldsymbol{E_{pos}} \in \boldsymbol{R}^{(N+1)\times D}$$

- Then passed to a Multi-headed Self-Attention block

Layer normalization (LN)

$$Y = LN(Z_{l-1}) = \begin{bmatrix} LN(Z_{l-1}^{0}) \\ \vdots \\ LN(Z_{l-1}^{N}) \end{bmatrix}, \mathrm{LN}\left(Z_{l-1}^{i}\right) = \begin{bmatrix} LN(Z_{l-1}^{i,1}) \\ \vdots \\ LN(Z_{l-1}^{i,D}) \end{bmatrix}$$

Layer normalization (LN)

$$\mu_i = \frac{1}{D}\sum_{k=1}^{D} Z_{l-1}^{i,k}, \sigma_i^2 = \frac{1}{D}\sum_{k=1}^{D}\left(z_{l-1}^{i,k} - \mu_i\right)^2$$

$$\hat{z}_{l-1}^{i,k} = \frac{z_{l-1}^{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}}, k = 1, \dots, D$$

$$Z_{l-1}^{i} = \begin{bmatrix} z_{l-1}^{i,1} \\ \vdots \\ z_{l1}^{i,D} \end{bmatrix}, \mathrm{Y} = \begin{bmatrix} y^0 \\ \vdots \\ y^N \end{bmatrix}$$

$$y_i = LN_{\gamma,\beta}\left(Z_{l-1}^{i}\right) = \gamma \hat{Z}_{l-1}^{i} + \beta, i = 0, 1, \dots, N$$

$$\boldsymbol{Y = LN(Z_{l-1}) = \begin{bmatrix} LN(z_{l-1}^{0}) \\ \vdots \\ LN(z_{l-1}^{N}) \end{bmatrix}}$$

Multi-headed Self-Attention (MSA) block

$$Z_l' = MSA\left(LN(Z_{l-1})\right) + Z_{l-1}, l = 1, \ldots, L$$

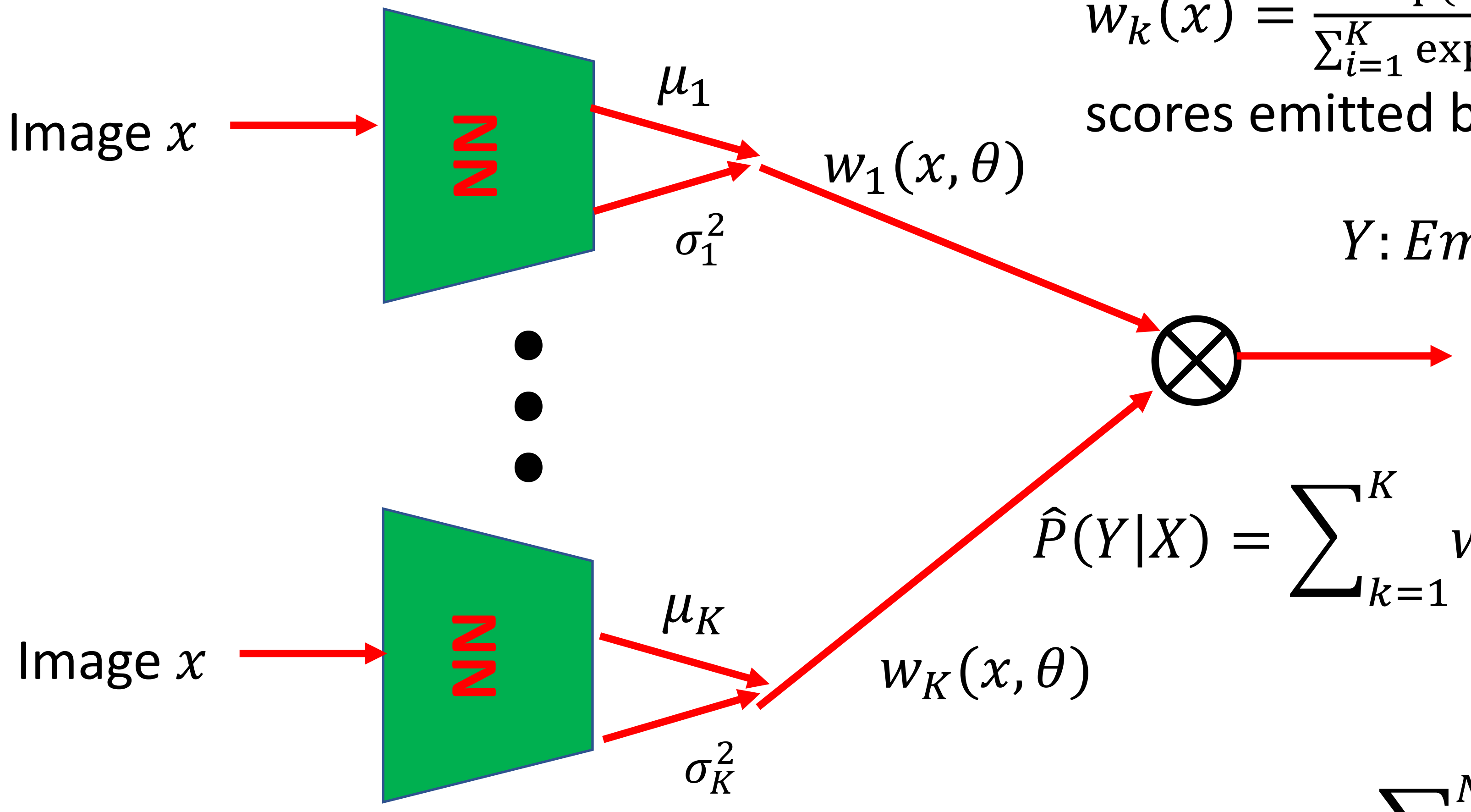
MLP

$$Z_l = MLP\left(LN\left(Z_l'\right)\right) + Z_l', l = 1, \ldots, L$$

Transformer Decoder

The decoder is used to decode the reconstruction vector back to the original image shape. It maps $R^{512} \rightarrow R^{H \times W \times C}$. In our experiments with the MVTec and BTAD dataset, we used 5 transposed convolutional layers, with batch normalization and ReLU in-between, except for the last layer, we use tanh as the final nonlinearity

Gaussian Mixture Density Network



$w_k(x) = \frac{\exp(a_k^w(x))}{\sum_{i=1}^{K}\exp(a_i^w(x))}$, $a_k^w(x) \in R$: the logit scores emitted by the neural networks.

$Y: Embedding$

$$\hat{P}(Y|X) = \sum_{k=1}^{K} w_k(x;\theta) N(Y|\mu_k(x;\theta), \sigma_k^2(x,\theta))$$

$$LL = -\sum_{n=1}^{N} \log P_\theta(Y_n|X_n)$$

OBJECTIVE AND LOSSES

- **Training the network has two objectives**:

on one side we want the decoder output to resemble the network input, as in **reconstruction-based anomaly detection**. This forces the encoder to catch features that are relevant to describe the normal data.

On the other side, the goal is to train the **Gaussian mixture density network** to model the **manifold** where the encoded features of normal images reside.

- **Mean Squared Error (MSE):**

$$MSE = \frac{1}{HW}\left\|X - \hat{X}\right\|_F^2 \xrightarrow{\hspace{2em}} \text{the Frobenius norm}$$

- **The Structural Similarity Index (SSIM)**

$$SSIM(X,Y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

- **Hence, the final objective function to minimize is the weighted addition of the above three losses.**

$$\boldsymbol{L(X) = -LL + \lambda_1 MSE\left(X,\hat{X}\right) + \lambda_2 SSIM\left(X,\hat{X}\right), \lambda_1 = 5, \lambda_2 = 0.5}$$

Action recognition

Interesting paper by the Google Research team where they use a pure-transformer based models for video classification, drawing upon the recent success of such models in image classification. The model extracts spatiotemporal tokens from the input video, which are then encoded by a series of transformer layers.

ViViT: A Video Vision Transformer, Anurag Arnab et al. 2021

Autonomous driving

Segment Everything Everywhere All at Once
Xueyan Zou et al. April 13, 2023

The success of Large Language Models (LLMs) such as ChatGPT [38] have shown the importance of modern AI models in interacting with humans, and have provided a glimpse of AGI [3]. The ability to interact with humans requires a user-friendly interface that can take as many types of human inputs as possible and generate responses that humans can easily understand.

In this work, we advocate a universal interface for segmenting everything everywhere with multi-modal prompts. To achieve this goal, we propose a new prompting scheme that has four important properties, versatility, compositionality, interactivity, and semantic-awareness.

总之，ChatGPT一上场就基本取得成功，旗开得胜，这也是概率论的胜利，贝叶斯的胜利？

- Formula for Bayes' Theorem

$$P(\underset{\text{Next word}}{A}|\underset{\text{Set of previous words}}{B}) = \frac{P(A)P(B|A)}{P(B)}$$

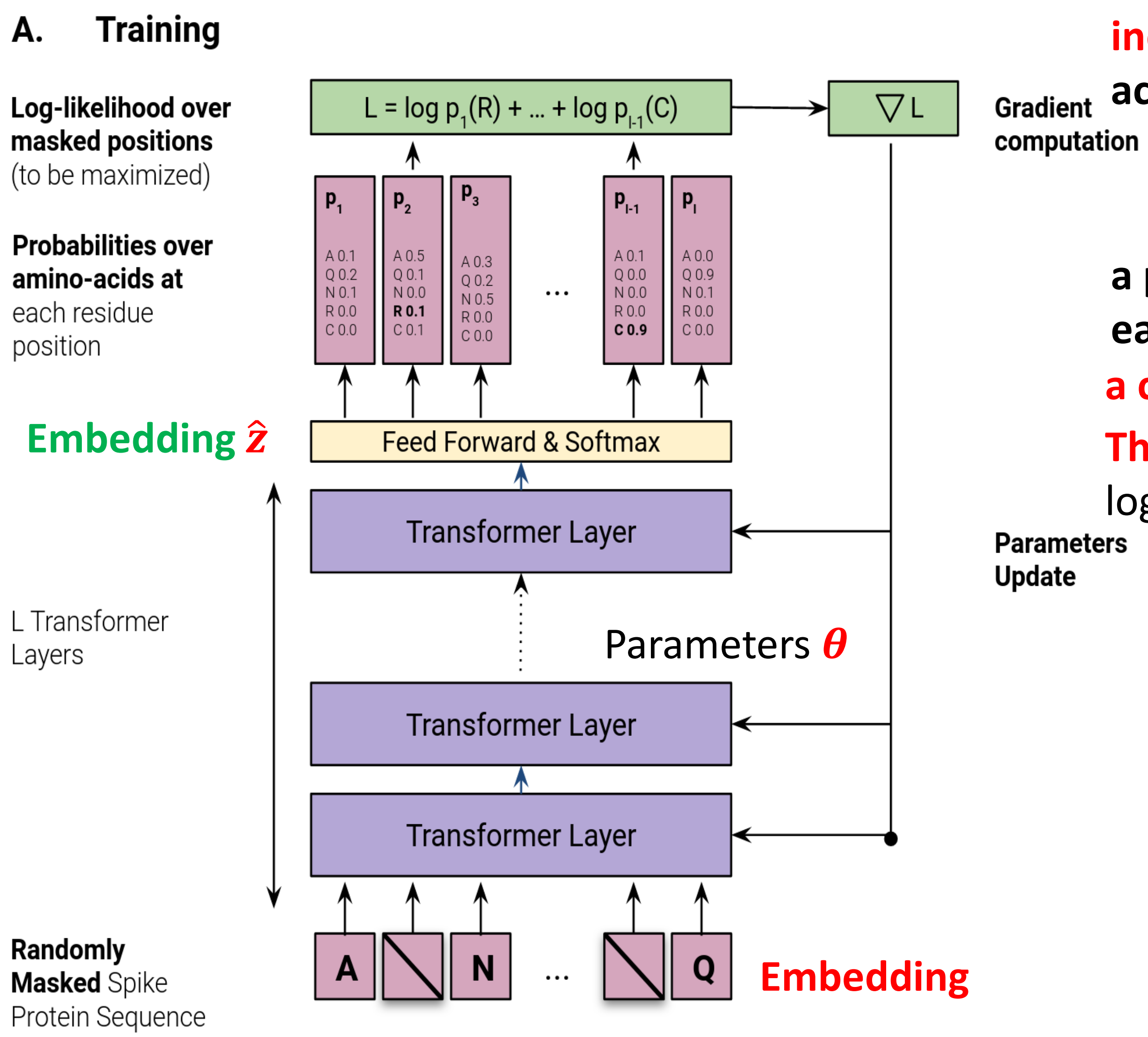
$$\tilde{\boldsymbol{h}}_j^i = \boldsymbol{TransDec}^M(\boldsymbol{w}_{0:j-1}^i, \tilde{\boldsymbol{h}}_i)$$

$$\boldsymbol{P}\left(\boldsymbol{w}_j^i \middle| \boldsymbol{w}_{0:j-1}^i, \tilde{\boldsymbol{D}}\right) = \boldsymbol{softmax}\,(\boldsymbol{W}_{vocab}\boldsymbol{h}_j^i)$$

By transformer and definition of Probability

Training

Token: class, Mask, 20 amino acids


A. Training
Log-likelihood over masked positions (to be maximized)
L = log p_1(R) + ... + log p_{l-1}(C)
∇L
Gradient computation
Probabilities over amino-acids at each residue position
p_1 A 0.1 Q 0.2 N 0.1 R 0.0 C 0.0
p_2 A 0.5 Q 0.1 N 0.0 R 0.1 C 0.1
p_3 A 0.3 Q 0.2 N 0.5 R 0.0 C 0.0
p_{l-1} A 0.1 Q 0.0 N 0.0 R 0.0 C 0.9
p_l A 0.0 Q 0.9 N 0.1 R 0.0 C 0.0
Embedding ẑ
Feed Forward & Softmax
Transformer Layer
Parameters θ
Parameters Update
L Transformer Layers
Transformer Layer
Transformer Layer
Randomly Masked Spike Protein Sequence
A N ... Q
Embedding

Each amino acid is first tokenized, i.e. mapped to their index in the vocabulary containing the natural amino acids and then projected to an embedding space.

UniRef100 dataset, Fine tune the GISAID :

a positional encoding must be added to the embedding of each token

a corrupted sequence $\hat{x}$

The training objective corresponds to the negative log-likelihood of the true sequence at the corrupted positions.

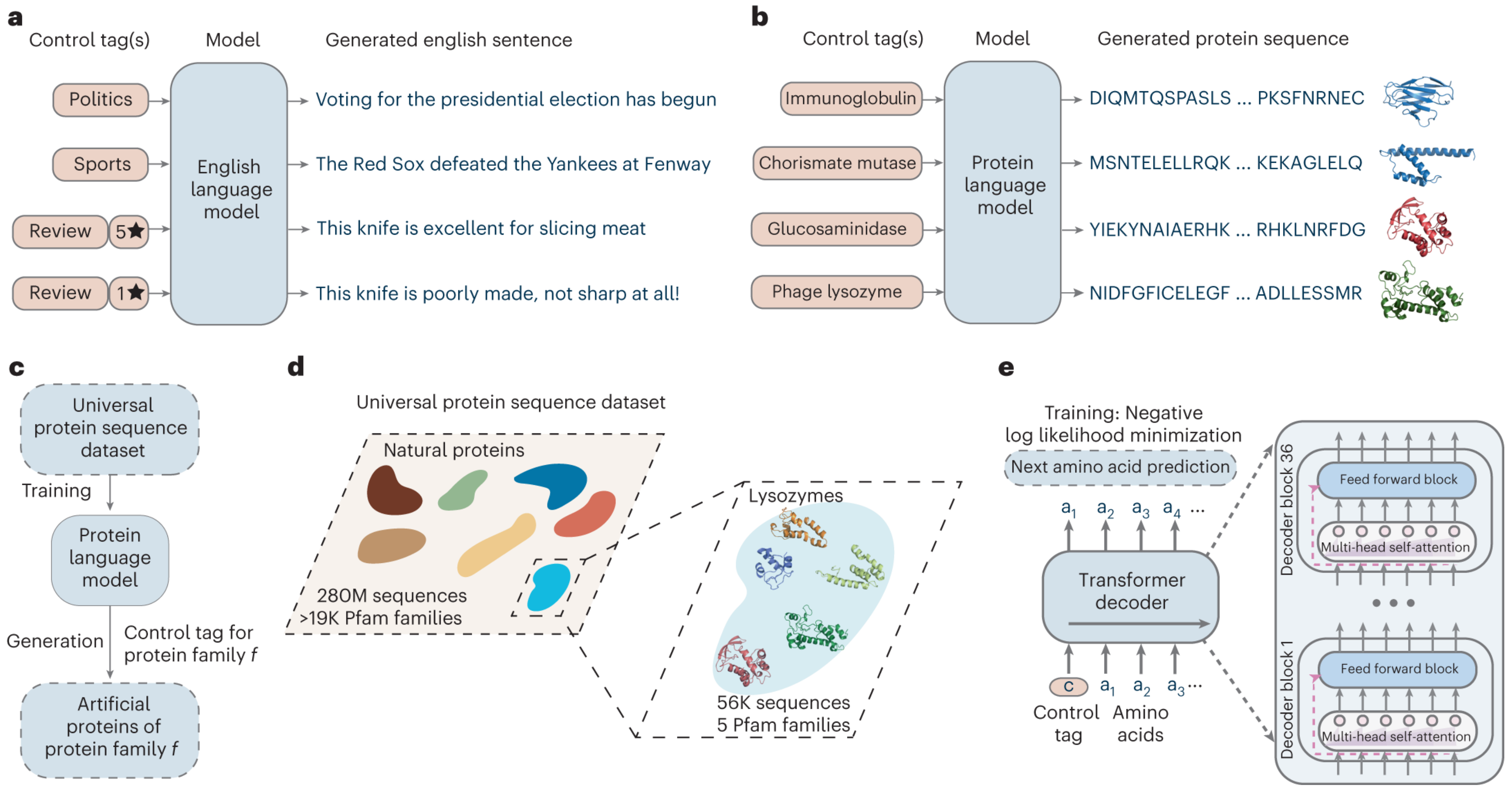
$$L(\hat{x}|x) = -\sum_{i \in M} \log P(x_i|\hat{x})$$

$$P(x_i|\hat{x}) = softmax\,(W\hat{z}_i + b)$$

$$\max_{\theta} \log P(x_i|\hat{x}, \theta)$$

GPT is one of LLM

Large language models generate functional protein sequences across diverse families nature biotechnology Ali Madani et al. 2023


a
Control tag(s)
Model
Generated english sentence
Politics
Sports
Review
5★
Review
1★
English language model
Voting for the presidential election has begun
The Red Sox defeated the Yankees at Fenway
This knife is excellent for slicing meat
This knife is poorly made, not sharp at all!
b
Control tag(s)
Model
Generated protein sequence
Immunoglobulin
Chorismate mutase
Glucosaminidase
Phage lysozyme
Protein language model
DIQMTQSPASLS ... PKSFNRNEC
MSNTELELLRQK ... KEKAGLELQ
YIEKYNAIAERHK ... RHKLNRFDG
NIDFGFICELEGF ... ADLLESSMR
c
Universal protein sequence dataset
Training
Protein language model
Generation
Control tag for protein family f
Artificial proteins of protein family f
d
Universal protein sequence dataset
Natural proteins
280M sequences
>19K Pfam families
Lysozymes
56K sequences
5 Pfam families
e
Training: Negative log likelihood minimization
Next amino acid prediction
a1 a2 a3 a4 ...
Transformer decoder
C a1 a2 a3 ...
Control tag
Amino acids
Decoder block 36
Feed forward block
Multi-head self-attention
Decoder block 1
Feed forward block
Multi-head self-attention