Hand-computable TabPFN-style Toy Forward Pass

1. Raw Input Data

$$X_{\text{train}} = \begin{bmatrix} 1.000000 & 2.000000 & 3.000000 & 4.000000 \\ 5.000000 & 6.000000 & 7.000000 & 8.000000 \end{bmatrix} \in \mathbb{R}^{2\times 4}, \qquad y_{\text{train}} = \begin{bmatrix} 1.000000 \\ 0.000000 \end{bmatrix} \in \mathbb{R}^{2}$$

$$X_{\text{test}} = \begin{bmatrix} 9.000000 & 10.000000 & 11.000000 & 12.000000 \end{bmatrix} \in \mathbb{R}^{1\times 4}$$

$$X_{\text{all}} = \begin{bmatrix} X_{\text{train}} \\ X_{\text{test}} \end{bmatrix} = \begin{bmatrix} 1.000000 & 2.000000 & 3.000000 & 4.000000 \\ 5.000000 & 6.000000 & 7.000000 & 8.000000 \\ 9.000000 & 10.000000 & 11.000000 & 12.000000 \end{bmatrix} \in \mathbb{R}^{3\times 4}$$

2. Feature-group Encoder (tabular $\to$ tokens)

We split features into $G = 2$ groups of size $g = 2$:

$$u_{s,0} = \begin{bmatrix} x_{s1} & x_{s2} \end{bmatrix} \in \mathbb{R}^{1\times 2}, \qquad u_{s,1} = \begin{bmatrix} x_{s3} & x_{s4} \end{bmatrix} \in \mathbb{R}^{1\times 2}.$$

We use your provided encoder matrix (shown as $W_{\text{enc}}^{\top}$ since we multiply row-vectors on the left):

$$W_{\text{enc}}^{\top} = \begin{bmatrix} 1.000000 & 0.500000 & 0.300000 & 0.700000 \\ 0.500000 & 1.000000 & 0.700000 & 0.300000 \end{bmatrix} \in \mathbb{R}^{2\times 4}, \qquad b_{\text{enc}} = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix} \in \mathbb{R}^{4}.$$

Encoding each group to a $D = 4$ vector:

$$e_{s,g} = u_{s,g}\, W_{\text{enc}}^{\top} + b_{\text{enc}} \in \mathbb{R}^{1\times 4}.$$

$$e_{1,0} = \begin{bmatrix} 2.100000 & 2.700000 & 2.000000 & 1.700000 \end{bmatrix}, \qquad e_{1,1} = \begin{bmatrix} 5.100000 & 5.700000 & 4.000000 & 3.700000 \end{bmatrix}$$
$$e_{2,0} = \begin{bmatrix} 8.100000 & 8.700000 & 6.000000 & 5.700000 \end{bmatrix}, \qquad e_{2,1} = \begin{bmatrix} 11.100000 & 11.700000 & 8.000000 & 7.700000 \end{bmatrix}$$
$$e_{3,0} = \begin{bmatrix} 14.100000 & 14.700000 & 10.000000 & 9.700000 \end{bmatrix}, \qquad e_{3,1} = \begin{bmatrix} 17.100000 & 17.700000 & 12.000000 & 11.700000 \end{bmatrix}$$

3. Label Encoder

TabPFN does *not* take y_{test} as input. Instead, the test label is *unknown*. Faithfully, the label encoder takes a 2D input per row:

$$y_s^{\text{in}} = \begin{bmatrix} y_{\text{clean}} \\ \mathbb{1}[\text{NaN}] \end{bmatrix} \in \mathbb{R}^{2},$$

where $y_{\text{clean}} = 0$ when the label is missing (NaN), and the missingness is carried by the indicator. For our toy: $y_{\text{pad}} = [1,\ 0,\ \text{NaN}]$. Thus:

$$Y_{\text{in}} = \begin{bmatrix} 1.000000 & 0.000000 \\ 0.000000 & 0.000000 \\ 0.000000 & 1.000000 \end{bmatrix} \in \mathbb{R}^{3\times 2}.$$

We keep your original "value" row $[1, -1, 0, 0]$ and add a minimal "NaN" row $[0, 0, 1, 1]$:

$$W_y = \begin{bmatrix} 1.000000 & -1.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 1.000000 & 1.000000 \end{bmatrix} \in \mathbb{R}^{2\times 4}, \qquad b_y = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{4}.$$

Label embeddings:

$$L = Y_{\text{in}} W_y + b_y = \begin{bmatrix} 1.000000 & -1.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 1.000000 & 1.000000 \end{bmatrix} \in \mathbb{R}^{3\times 4}.$$

4. Build the Model Input Tensor $E^{(0)} \in \mathbb{R}^{S' \times T_{\mathbf{tok}} \times D}$

For each data row $s \in \{2, 3, 4\}$ (train1, train2, test1), we create a 3-token matrix

$$R_s = \begin{bmatrix} e_{(\text{data row}),0} \\ e_{(\text{data row}),1} \\ L_{(\text{data row})} \end{bmatrix} \in \mathbb{R}^{3\times 4}.$$

We prepend $T = 2$ thinking rows (learned parameters) Θ_0, Θ_1, then add token-slot embedding P_{tok}:

$$E_+^{(0)} = \text{concat}_0(\Theta, R) + P_{\text{tok}} \in \mathbb{R}^{5\times 3\times 4}.$$

Resulting slices $E_+^{(0)}[s]$:

$$E_+^{(0)}[0] = \begin{bmatrix} 0.11 & 0.12 & 0.13 & 0.14 \\ 0.21 & 0.22 & 0.23 & 0.24 \\ 0.31 & 0.32 & 0.33 & 0.34 \end{bmatrix}, \qquad E_+^{(0)}[1] = \begin{bmatrix} 0.15 & 0.16 & 0.17 & 0.18 \\ 0.25 & 0.26 & 0.27 & 0.28 \\ 0.35 & 0.36 & 0.37 & 0.38 \end{bmatrix}$$

$$E_+^{(0)}[2] = \begin{bmatrix} 2.2 & 2.8 & 2.1 & 1.8 \\ 5.3 & 5.9 & 4.2 & 3.9 \\ 1.3 & -0.7 & 0.3 & 0.3 \end{bmatrix}, \qquad E_+^{(0)}[3] = \begin{bmatrix} 8.2 & 8.8 & 6.1 & 5.8 \\ 11.3 & 11.9 & 8.2 & 7.9 \\ 0.3 & 0.3 & 0.3 & 0.3 \end{bmatrix}$$

$$E_+^{(0)}[4] = \begin{bmatrix} 14.2 & 14.8 & 10.1 & 9.8 \\ 17.3 & 17.9 & 12.2 & 11.9 \\ 0.3 & 0.3 & 1.3 & 1.3 \end{bmatrix}$$

5. Transformer Block: Column-Attention (within items)

Standard attention ($D = 4$, single head) applied independently to each 3×4 matrix.

$$W_Q = \text{diag}(0.1, 0.2, 0.1, 0.2), \quad W_K = \text{diag}(0.1, 0.1, 0.1, 0.1), \quad W_V = I_4.$$

Output $X'_s = X_s + \text{softmax}(\frac{QK^\top}{2})V$. Stacking results gives $E^{(1)}$.

Example Row $s = 4$ (Test):

$$X'_4 = \begin{bmatrix} 30.542494 & 31.742523 & 21.653397 & 21.053352 \\ 33.757339 & 34.957359 & 23.866688 & 23.266659 \\ 11.688443 & 11.997224 & 9.338827 & 9.029969 \end{bmatrix}$$

6. LayerNorm 1

Standard token-wise LayerNorm: $E_{\text{LN}}^{(1)} = \text{LN}(E^{(1)})$. Example Slice ($s = 4$, Test):

$$E_{\text{LN}}^{(1)}[4] = \begin{bmatrix} 0.873795 & 1.116945 & -0.934583 & -1.056158 \\ 0.886211 & 1.106212 & -0.941211 & -1.051212 \\ 0.809413 & 1.170145 & -0.899594 & -1.079960 \end{bmatrix}$$

7. Row-Attention (across items, per token slot)

We perform attention across the 5 rows, separately for token slots $t = 0, 1, 2$. **Test Leakage Prevention:** Keys/Values are restricted to rows 0..3 (Thinking + Train). Queries use all rows.

Focus: Label Token Slot ($t = 2$) We compute $(X^{(t)})' = X^{(t)} + \text{softmax}(\frac{QK^\top}{2})V$. Resulting matrix for $t = 2$ (Label Token Slot) written back into $E^{(2)}$:

$$(X^{(2)})' = \begin{bmatrix} -1.345414 & -0.557608 & 0.365306 & 1.537717 \\ -1.345413 & -0.557608 & 0.365307 & 1.537718 \\ 1.743781 & -1.053738 & -0.382356 & -0.307686 \\ 0.777352 & 1.067473 & -0.968615 & -0.875887 \\ 0.792784 & 1.055540 & -0.977013 & -0.870686 \end{bmatrix} \quad \text{(Last row is Test)}$$

8. LayerNorm 2

$E^{(2)}_{\text{LN}} = \text{LN}(E^{(2)})$. Example Slice for Test Row ($s = 4$):

$$E^{(2)}_{\text{LN}}[4] = \begin{bmatrix} 0.526944 & 1.379527 & -0.983962 & -0.922509 \\ 0.572410 & 1.348058 & -1.034239 & -0.886230 \\ 0.861658 & 1.126422 & -1.072915 & -0.915165 \end{bmatrix}$$

9. MLP + LayerNorm 3

$E^{(3)} = E^{(2)}_{\text{LN}} + \text{GELU}(E^{(2)}_{\text{LN}})$, followed by LN. Final model state $E^{(3)}_{\text{LN}}$ for Test Row ($s = 4$):

$$E^{(3)}_{\text{LN}}[4] = \begin{bmatrix} 0.362668 & 1.475764 & -0.937410 & -0.901022 \\ 0.419818 & 1.443537 & -0.975789 & -0.887566 \\ \mathbf{0.808145} & \mathbf{1.173889} & \mathbf{-1.039184} & \mathbf{-0.942849} \end{bmatrix}$$

10. Readout to Test Logits

We extract the **Label Token** from the **Test Row**:

$$h_{\text{test}} = E^{(3)}_{\text{LN}}[4, 2, :] = \begin{bmatrix} 0.808145 & 1.173889 & -1.039184 & -0.942849 \end{bmatrix}.$$

Using a minimal classifier head (W_{out} selects first 2 dims):

$$\text{logits} = h_{\text{test}} W_{\text{out}} = \begin{bmatrix} 0.808145 & 1.173889 \end{bmatrix}.$$

Prediction: Class 1 (since $1.17 > 0.81$).