Developing MicroXL, a TensorFlow Recommendation Model

for the 2025 NUS Datathon

Shane Bharathan, Gan Xinyee, Ng Jing Xuan, Evie Low

1 Appendix

Input Shapes

- var_1 **ID:**
$$var_1\texttt{\_id} \in \mathbb{R}^B \quad \text{(a vector of } B \text{ integers)}$$
- var_2 **ID:**
$$var_2\texttt{\_id} \in \mathbb{R}^B \quad \text{(a vector of } B \text{ integers)}$$
- var_1 **Metadata Features:**
$$var_1\texttt{\_features} \in \mathbb{R}^{B \times F_c}$$
- var_2 **Metadata Features:**
$$var_2\texttt{\_features} \in \mathbb{R}^{B \times F_p}$$

var_1 Branch Processing

Step 2.1: Embedding for var_1 ID

- **Operation:** The var_1 ID is passed through an embedding layer.
- **Embedding Details:** The weight matrix is
$$E_c \in \mathbb{R}^{\text{num\_}var_1\text{s} \times (d/2)},$$
where for example if $d = 64$, then the output dimension is 32.
- **Input/Output Shapes:**
$$\text{Input: } (B,) \quad \Longrightarrow \quad e_c \in \mathbb{R}^{B \times 32}.$$

Step 2.2: Dense Transformation for var_1 Metadata

- **Operation:** The var_1 metadata features $x_c \in \mathbb{R}^{F_c}$ are passed through a Dense layer with ReLU activation.
- **Dense Layer Details:** The transformation is given by:
$$f_c(x_c) = \text{ReLU}(W_c\, x_c + b_c),$$
where
$$W_c \in \mathbb{R}^{d \times F_c}, \quad b_c \in \mathbb{R}^d.$$
For $d = 64$, this layer produces 64 neurons.
- **Input/Output Shapes:**
$$\text{Input: } var_1\texttt{\_features} \in \mathbb{R}^{B \times F_c} \quad \Longrightarrow \quad f_c(x_c) \in \mathbb{R}^{B \times 64}.$$

Step 2.3: Concatenation on the var_1 Side

- **Operation:** Concatenate the var_1 ID embedding e_c and the transformed metadata $f_c(x_c)$ along the feature dimension.
- **Shape Calculation:** The embedding contributes 32 dimensions and the dense output contributes 64 dimensions, giving:
$$32 + 64 = 96 \text{ dimensions.}$$
- **Output Shape:**
$$u_c = \texttt{concat}(e_c, f_c(x_c)) \in \mathbb{R}^{B \times 96}.$$

Step 2.4: Final Projection for var_1

- **Operation:** The concatenated var_1 vector u_c is passed through a final Dense layer with ReLU activation to project it into a d-dimensional space.
- **Projection Details:** The projection is given by:

$$z_c = \text{ReLU}(W_f\, u_c + b_f),$$

where

$$W_f \in \mathbb{R}^{d\times 96}, \quad b_f \in \mathbb{R}^d.$$

For $d = 64$, this maps the 96-dimensional input to 64 neurons.

- **Input/Output Shapes:**

$$\text{Input: } u_c \in \mathbb{R}^{B\times 96} \quad \Longrightarrow \quad z_c \in \mathbb{R}^{B\times 64}.$$

var_2 Branch Processing

Step 3.1: Embedding for var_2 ID

- **Operation:** The var_2 ID is passed through its embedding layer.
- **Embedding Details:** The weight matrix is

$$E_p \in \mathbb{R}^{\text{num\_}var_2\times (d/2)},$$

which for $d = 64$ gives an output dimension of 32.

- **Input/Output Shapes:**

$$\text{Input: } (B,) \quad \Longrightarrow \quad e_p \in \mathbb{R}^{B\times 32}.$$

Step 3.2: Dense Transformation for var_2 Metadata

- **Operation:** The var_2 metadata features $x_p \in \mathbb{R}^{F_p}$ are passed through a Dense layer with ReLU activation.
- **Dense Layer Details:** The transformation is:

$$f_p(x_p) = \text{ReLU}(W_p\, x_p + b_p),$$

where

$$W_p \in \mathbb{R}^{d\times F_p}, \quad b_p \in \mathbb{R}^d.$$

For $d = 64$, this yields 64 neurons.

- **Input/Output Shapes:**

$$\text{Input: } var_2\texttt{\_features} \in \mathbb{R}^{B\times F_p} \quad \Longrightarrow \quad f_p(x_p) \in \mathbb{R}^{B\times 64}.$$

Step 3.3: Concatenation on the var_2 Side

- **Operation:** Concatenate the var_2 ID embedding e_p and the transformed metadata $f_p(x_p)$.
- **Shape Calculation:** The embedding contributes 32 dimensions and the dense output contributes 64 dimensions:

$$32 + 64 = 96 \text{ dimensions.}$$

- **Output Shape:**

$$u_p = \texttt{concat}(e_p, f_p(x_p)) \in \mathbb{R}^{B\times 96}.$$

Step 3.4: Final Projection for var_2

- **Operation:** The concatenated var_2 vector u_p is passed through the same final Dense layer (shared) with ReLU activation.
- **Input/Output Shapes:**

$$\text{Input: } u_p \in \mathbb{R}^{B\times 96} \quad \Longrightarrow \quad z_p \in \mathbb{R}^{B\times 64}.$$

Interaction and Output

Step 4.1: Element-wise Multiplication

- **Operation:** Perform element-wise multiplication between the projected var_1 and var_2 vectors.
- **Input Shapes:** Both z_c and z_p are in $\mathbb{R}^{B\times 64}$.
- **Output:**

$$m = z_c \odot z_p \in \mathbb{R}^{B\times 64}.$$

Step 4.2: Dot Product (Reduction over Features)

- **Operation:** Sum the elements of m over the feature dimension to compute the dot product:

$$s = \sum_{i=1}^{64} m_i.$$

- **Output Shape:**

$$s \in \mathbb{R}^{B\times 1}.$$

Step 4.3: Sigmoid Activation

- **Operation:** Apply the sigmoid function to s to obtain a probability:

$$\hat{y} = \sigma(s) \in \mathbb{R}^{B\times 1}.$$

Summary of the Mathematical Flow

Embedding Lookups:

$$e_c = E_c(c) \in \mathbb{R}^{\frac{d}{2}},$$
$$e_p = E_p(p) \in \mathbb{R}^{\frac{d}{2}}.$$

Feature Transformations:

$$f_c(x_c) = \mathrm{ReLU}\big(W_c\, x_c + b_c\big) \in \mathbb{R}^d,$$
$$f_p(x_p) = \mathrm{ReLU}\big(W_p\, x_p + b_p\big) \in \mathbb{R}^d.$$

Concatenation:

$$u_c = \begin{pmatrix} e_c \\ f_c(x_c) \end{pmatrix} \in \mathbb{R}^{\frac{3d}{2}},$$
$$u_p = \begin{pmatrix} e_p \\ f_p(x_p) \end{pmatrix} \in \mathbb{R}^{\frac{3d}{2}}.$$

Final Projection:

$$z_c = \mathrm{ReLU}\big(W_f\, u_c + b_f\big) \in \mathbb{R}^d,$$
$$z_p = \mathrm{ReLU}\big(W_f\, u_p + b_f\big) \in \mathbb{R}^d.$$

Interaction & Output:

$$\hat{y} = \sigma\Big(z_c^\top z_p\Big) \in (0, 1).$$

This formulation shows how the model jointly leverages learned ID embeddings and transformed metadata to produce a score via a dot product in a common latent space. Each component (embeddings, dense layers, and the final projection) plays a role in learning a representation that makes the dot product

$$z_c^\top z_p$$

indicative of the likelihood of a match between a var_1 and a var_2.

For further clarity, below is a sample implementation with real values in the next section.

Synthetic Example of the Model Transformation

Assumptions:

- Embedding size: $d = 4 \Rightarrow$ ID embeddings are of size 2.
- Batch size: $B = 3$.
- Unique var_1s: 5; unique var_2: 7.
- var_1 and var_2 metadata features are 3-dimensional.

1. var_1 **Branch**

(a) var_1 **Embedding Matrix** $\boldsymbol{E_c}$ **(shape** 5×2**):**

$$E_c = \begin{pmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \\ 0.5 & 0.6 \\ 0.7 & 0.8 \\ 0.9 & 1.0 \end{pmatrix}.$$

Suppose our batch has var_1 IDs (using 1-indexing):

$$\text{IDs} = \begin{pmatrix} 2 \\ 4 \\ 1 \end{pmatrix}.$$

Thus, the corresponding var_1 embeddings are:

$$E_c^{\text{batch}} = \begin{pmatrix} 0.3 & 0.4 \\ 0.7 & 0.8 \\ 0.1 & 0.2 \end{pmatrix}.$$

(b) var_1 **Metadata Features:** Assume the raw features (each row is one sample) are

$$X_c = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}.$$

(c) var_1 **Dense Transformation:** We use a weight matrix W_c (shape 4×3) and zero bias:

$$W_c = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad b_c = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Then, for each sample x_c, the transformation is:

$$f_c(x_c) = \text{ReLU}\big(W_c\, x_c + b_c\big).$$

This gives, for each sample:

$$f_c\big((1,2,3)^\top\big) = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 1+2+3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 6 \end{pmatrix},$$

$$f_c\big((4,5,6)^\top\big) = \begin{pmatrix} 4 \\ 5 \\ 6 \\ 4+5+6 \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \\ 6 \\ 15 \end{pmatrix},$$

$$f_c\big((7,8,9)^\top\big) = \begin{pmatrix} 7 \\ 8 \\ 9 \\ 7+8+9 \end{pmatrix} = \begin{pmatrix} 7 \\ 8 \\ 9 \\ 24 \end{pmatrix}.$$

Stacking these results, we have:

$$F_c = \begin{pmatrix} 1 & 2 & 3 & 6 \\ 4 & 5 & 6 & 15 \\ 7 & 8 & 9 & 24 \end{pmatrix}.$$

(d) Concatenation: For each example, we concatenate the 2-dimensional embedding with the 4-dimensional dense output to obtain a 6-dimensional vector:

$$u_c = \begin{pmatrix} \text{embedding} \\ \text{dense output} \end{pmatrix}.$$

Thus, for our batch:

$$U_c = \begin{pmatrix} 0.3 & 0.4 & 1 & 2 & 3 & 6 \\ 0.7 & 0.8 & 4 & 5 & 6 & 15 \\ 0.1 & 0.2 & 7 & 8 & 9 & 24 \end{pmatrix}.$$

(e) Final Projection (var_1 Side): We use a final dense layer with weight matrix W_f (shape 4×6) and zero bias:

$$W_f = \begin{pmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix}.$$

Then, $z_c = \mathrm{ReLU}(W_f\, u_c)$. Computing for each example:

Example 1: $u_c^{(1)} = (0.3,\, 0.4,\, 1,\, 2,\, 3,\, 6)^\top$

$$\begin{aligned} &\text{Row 1:} && 0.5(0.3) + 0.5(0.4) && = 0.15 + 0.20 = 0.35, \\ &\text{Row 2:} && 0.5(1) + 0.5(2) && = 0.5 + 1.0 = 1.5, \\ &\text{Row 3:} && 0.5(3) + 0.5(6) && = 1.5 + 3.0 = 4.5, \\ &\text{Row 4:} && 0.25(0.3 + 0.4 + 1 + 2 + 3 + 6) && = 0.25(12.7) = 3.175. \end{aligned}$$

Thus,

$$z_c^{(1)} = \begin{pmatrix} 0.35 \\ 1.5 \\ 4.5 \\ 3.175 \end{pmatrix}.$$

Example 2: $u_c^{(2)} = (0.7,\, 0.8,\, 4,\, 5,\, 6,\, 15)^\top$

$$\begin{aligned} &\text{Row 1:} && 0.5(0.7) + 0.5(0.8) && = 0.35 + 0.40 = 0.75, \\ &\text{Row 2:} && 0.5(4) + 0.5(5) && = 2 + 2.5 = 4.5, \\ &\text{Row 3:} && 0.5(6) + 0.5(15) && = 3 + 7.5 = 10.5, \\ &\text{Row 4:} && 0.25(0.7 + 0.8 + 4 + 5 + 6 + 15) && = 0.25(31.5) = 7.875. \end{aligned}$$

Thus,

$$z_c^{(2)} = \begin{pmatrix} 0.75 \\ 4.5 \\ 10.5 \\ 7.875 \end{pmatrix}.$$

Example 3: $u_c^{(3)} = (0.1,\ 0.2,\ 7,\ 8,\ 9,\ 24)^\top$

$$\begin{aligned}
&\text{Row 1:} && 0.5(0.1) + 0.5(0.2) && = 0.05 + 0.10 = 0.15,\\
&\text{Row 2:} && 0.5(7) + 0.5(8) && = 3.5 + 4 = 7.5,\\
&\text{Row 3:} && 0.5(9) + 0.5(24) && = 4.5 + 12 = 16.5,\\
&\text{Row 4:} && 0.25(0.1 + 0.2 + 7 + 8 + 9 + 24) && = 0.25(48.3) = 12.075.
\end{aligned}$$

Thus,

$$z_c^{(3)} = \begin{pmatrix} 0.15 \\ 7.5 \\ 16.5 \\ 12.075 \end{pmatrix}.$$

Stacking these, the var_1 final representation is:

$$Z_c = \begin{pmatrix} 0.35 & 1.5 & 4.5 & 3.175 \\ 0.75 & 4.5 & 10.5 & 7.875 \\ 0.15 & 7.5 & 16.5 & 12.075 \end{pmatrix}.$$

2. var_2 **Branch**

(a) var_2 **Embedding Matrix** $\boldsymbol{E_p}$ **(shape** 7×2**):**

$$E_p = \begin{pmatrix} 0.1 & 0.3 \\ 0.2 & 0.4 \\ 0.3 & 0.5 \\ 0.4 & 0.6 \\ 0.5 & 0.7 \\ 0.6 & 0.8 \\ 0.7 & 0.9 \end{pmatrix}.$$

Suppose our batch has var_2 IDs:

$$\text{IDs} = \begin{pmatrix} 3 \\ 7 \\ 5 \end{pmatrix}.$$

Thus, the corresponding var_2 embeddings are:

$$E_p^{\text{batch}} = \begin{pmatrix} 0.3 & 0.5 \\ 0.7 & 0.9 \\ 0.5 & 0.7 \end{pmatrix}.$$

(b) var_2 **Metadata Features:** Assume the features are

$$X_p = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

(c) var_2 **Dense Transformation:** Using the same dense layer as for var_1s,

$$W_p = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad b_p = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

we get:

$$f_p\big((1,0,1)^\top\big) = \begin{pmatrix}1\\0\\1\\2\end{pmatrix},$$

$$f_p\big((0,1,0)^\top\big) = \begin{pmatrix}0\\1\\0\\1\end{pmatrix},$$

$$f_p\big((1,1,1)^\top\big) = \begin{pmatrix}1\\1\\1\\3\end{pmatrix}.$$

Thus,

$$F_p = \begin{pmatrix}1 & 0 & 1 & 2\\ 0 & 1 & 0 & 1\\ 1 & 1 & 1 & 3\end{pmatrix}.$$

(d) Concatenation: The concatenated var_2 vectors are:

$$U_p = \begin{pmatrix}0.3 & 0.5 & 1 & 0 & 1 & 2\\ 0.7 & 0.9 & 0 & 1 & 0 & 1\\ 0.5 & 0.7 & 1 & 1 & 1 & 3\end{pmatrix}.$$

(e) Final Projection (var_2 Side): Again, using the same W_f (and zero bias), we compute:

Example 1: $u_p^{(1)} = (0.3,\ 0.5,\ 1,\ 0,\ 1,\ 2)^\top$

$$\begin{aligned}
&\text{Row 1:} && 0.5(0.3) + 0.5(0.5) && = 0.15 + 0.25 = 0.4,\\
&\text{Row 2:} && 0.5(1) + 0.5(0) && = 0.5 + 0 = 0.5,\\
&\text{Row 3:} && 0.5(1) + 0.5(2) && = 0.5 + 1 = 1.5,\\
&\text{Row 4:} && 0.25(0.3 + 0.5 + 1 + 0 + 1 + 2) && = 0.25(4.8) = 1.2.
\end{aligned}$$

Thus,

$$z_p^{(1)} = \begin{pmatrix}0.4\\0.5\\1.5\\1.2\end{pmatrix}.$$

Example 2: $u_p^{(2)} = (0.7,\ 0.9,\ 0,\ 1,\ 0,\ 1)^\top$

$$\begin{aligned}
&\text{Row 1:} && 0.5(0.7) + 0.5(0.9) && = 0.35 + 0.45 = 0.8,\\
&\text{Row 2:} && 0.5(0) + 0.5(1) && = 0 + 0.5 = 0.5,\\
&\text{Row 3:} && 0.5(0) + 0.5(1) && = 0 + 0.5 = 0.5,\\
&\text{Row 4:} && 0.25(0.7 + 0.9 + 0 + 1 + 0 + 1) && = 0.25(3.6) = 0.9.
\end{aligned}$$

Thus,

$$z_p^{(2)} = \begin{pmatrix}0.8\\0.5\\0.5\\0.9\end{pmatrix}.$$

Example 3: $u_p^{(3)} = (0.5,\ 0.7,\ 1,\ 1,\ 1,\ 3)^\top$

$$\begin{aligned}
&\text{Row 1:} && 0.5(0.5) + 0.5(0.7) && = 0.25 + 0.35 = 0.6,\\
&\text{Row 2:} && 0.5(1) + 0.5(1) && = 0.5 + 0.5 = 1,\\
&\text{Row 3:} && 0.5(1) + 0.5(3) && = 0.5 + 1.5 = 2,\\
&\text{Row 4:} && 0.25(0.5 + 0.7 + 1 + 1 + 1 + 3) && = 0.25(7.2) = 1.8.
\end{aligned}$$

Thus,

$$z_p^{(3)} = \begin{pmatrix} 0.6 \\ 1 \\ 2 \\ 1.8 \end{pmatrix}.$$

Stacking these, we have:

$$Z_p = \begin{pmatrix} 0.4 & 0.5 & 1.5 & 1.2 \\ 0.8 & 0.5 & 0.5 & 0.9 \\ 0.6 & 1 & 2 & 1.8 \end{pmatrix}.$$

3. Interaction & Final Output

For each example i we compute the dot product:

$$s_i = z_c^{(i)} \cdot z_p^{(i)}.$$

Example 1:

$$s_1 = 0.35 \cdot 0.4 + 1.5 \cdot 0.5 + 4.5 \cdot 1.5 + 3.175 \cdot 1.2 \approx 0.14 + 0.75 + 6.75 + 3.81 = 11.45.$$

Example 2:

$$s_2 = 0.75 \cdot 0.8 + 4.5 \cdot 0.5 + 10.5 \cdot 0.5 + 7.875 \cdot 0.9 \approx 0.6 + 2.25 + 5.25 + 7.0875 = 15.1875.$$

Example 3:

$$s_3 = 0.15 \cdot 0.6 + 7.5 \cdot 1 + 16.5 \cdot 2 + 12.075 \cdot 1.8 \approx 0.09 + 7.5 + 33 + 21.735 = 62.325.$$

Finally, applying the sigmoid function:

$$\hat{y}_i = \sigma(s_i) = \frac{1}{1 + e^{-s_i}},$$

we have, for these large positive s_i, approximately:

$$\hat{y} \approx \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Summary: The overall transformation for our batch is as follows:

$$\begin{aligned} var_1 \text{ branch:} \quad & E_c^{\text{batch}} \in \mathbb{R}^{3\times 2}, \quad F_c \in \mathbb{R}^{3\times 4}, \quad U_c \in \mathbb{R}^{3\times 6}, \quad Z_c \in \mathbb{R}^{3\times 4}, \\ var_2 \text{ branch:} \quad & E_p^{\text{batch}} \in \mathbb{R}^{3\times 2}, \quad F_p \in \mathbb{R}^{3\times 4}, \quad U_p \in \mathbb{R}^{3\times 6}, \quad Z_p \in \mathbb{R}^{3\times 4}, \end{aligned}$$

and the final output is obtained by computing the row-wise dot product:

$$\hat{y}_i = \sigma\left(z_c^{(i)} \cdot z_p^{(i)}\right), \quad i = 1, 2, 3.$$

In our synthetic example the dot var_2 are large and hence the sigmoid yields outputs very close to 1.