Detailed Architecture Plan: Cross-Attention with Squeeze-and-Excitation Layers for Tabular Data

Contents

1 Introduction

In this document, we present a comprehensive architecture for efficiently processing tabular data using advanced neural network components. The architecture integrates **Cross-Attention (CA)** and **Squeeze-and-Excitation (SE)** layers, along with several optimization techniques to balance computational efficiency and performance. The aim is to capture complex feature interactions inherent in tabular data while maintaining a computationally efficient model.

2 Notation and Preliminaries

Let us define the notation and mathematical preliminaries used throughout the document.

- Let n be the number of numerical features and k be the number of categorical features.
- Numerical features are denoted by $\mathbf{x} \in \mathbb{R}^n$.
- Categorical features are represented by $\mathbf{c} = [c_1, c_2, \ldots, c_k]$, where $c_i \in \{1, 2, \ldots, V_i\}$ and V_i is the cardinality of the i-th categorical feature.
- The embedding dimension is denoted by d.
- The reduction ratio in SE layers is denoted by r.

3 Input Embedding Layer

The first step involves transforming raw input features into a unified embedding space suitable for subsequent processing.

3.1 Categorical Feature Embedding

Each categorical feature c_i is embedded into a d-dimensional vector space using an embedding matrix:

$$\mathbf{e}_{\text{cat},i} = \mathbf{W}_{\text{cat},i}[c_i], \tag{1}$$

where:

- $\mathbf{W}_{\text{cat},i} \in \mathbb{R}^{V_i \times d}$ is the embedding matrix for the i-th categorical feature.
- c_i is the index into the embedding matrix for the i-th feature.

Shared Embedding Space Optimization:

For categorical features with similar cardinality, we can share the embedding matrices to reduce memory usage:

$$\mathbf{e}_{\text{cat},i} = \mathbf{W}_{\text{cat}}[c_i], \tag{2}$$

where $\mathbf{W}_{\text{cat}} \in \mathbb{R}^{V_{\max} \times d}$ and $V_{\max} = \max_i V_i$.

3.2 Numerical Feature Embedding

Each numerical feature x_j is projected into the embedding space via a linear transformation:

$$\mathbf{e}_{\text{num},j} = \mathbf{W}_{\text{num},j} x_j + b_{\text{num},j}, \tag{3}$$

where:

- $\mathbf{W}_{\text{num},j} \in \mathbb{R}^{1 \times d}$ is the weight vector for the j-th numerical feature.
- $b_{\text{num},j} \in \mathbb{R}^d$ is the bias vector.

Alternatively, we can embed all numerical features jointly:

$$\mathbf{e}_{\text{num}} = \mathbf{W}_{\text{num}} \mathbf{x} + \mathbf{b}_{\text{num}}, \tag{4}$$

with $\mathbf{W}_{\text{num}} \in \mathbb{R}^{n \times d}$ and $\mathbf{b}_{\text{num}} \in \mathbb{R}^d$.

Normalization:

Before embedding, numerical features are normalized to have zero mean and unit variance:

$$\tilde{x}_j = \frac{x_j - \mu_j}{\sigma_j}, \tag{5}$$

where μ_j and σ_j are the mean and standard deviation of the j-th feature across the training set.

3.3 Hybrid Embedding

The final embedding matrix $\mathbf{E} \in \mathbb{R}^{(k+n) \times d}$ is constructed by concatenating the embeddings:

$$\mathbf{E} = [\mathbf{e}_{\text{cat},1}; \ldots; \mathbf{e}_{\text{cat},k}; \mathbf{e}_{\text{num},1}; \ldots; \mathbf{e}_{\text{num},n}], \tag{6}$$

where ; denotes vertical concatenation.

4 Cross-Attention Layer (CA)

The cross-attention mechanism allows for interaction between categorical and numerical features.

4.1 Formulation

We define the queries, keys, and values as:

$$\mathbf{Q} = \mathbf{E}_{\text{cat}} \mathbf{W}_Q, \quad \mathbf{E}_{\text{cat}} \in \mathbb{R}^{k \times d}, \tag{7}$$

$$\mathbf{K} = \mathbf{E}_{\text{num}} \mathbf{W}_K, \quad \mathbf{E}_{\text{num}} \in \mathbb{R}^{n \times d}, \tag{8}$$

$$\mathbf{V} = \mathbf{E}_{\text{num}} \mathbf{W}_V, \tag{9}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_k}$ are projection matrices, and d_k is the dimension of the keys and queries. The cross-attention output is computed as:

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{10}$$

with $\mathbf{A} \in \mathbb{R}^{k \times d_k}$.

4.2 Multi-Head Cross-Attention

To capture different representation subspaces, we employ h attention heads:

$$\mathbf{A}_i = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^\top}{\sqrt{d_k/h}}\right)\mathbf{V}_i, \quad i = 1, \ldots, h, \tag{11}$$

where $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i$ are the projections for the i-th head.

The outputs from all heads are concatenated:

$$\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2; \ldots; \mathbf{A}_h], \tag{12}$$

and projected back to the original dimension:

$$\mathbf{A} = \mathbf{A}\mathbf{W}_O, \tag{13}$$

where $\mathbf{W}_O \in \mathbb{R}^{hd_k \times d}$.

4.3 Optimization Techniques

Blockwise Cross-Attention:

To reduce computational complexity, features are divided into blocks:

$$\mathbf{E}_{\text{cat}} = [\mathbf{E}_{\text{cat}}^{(1)}, \mathbf{E}_{\text{cat}}^{(2)}, \ldots], \quad \mathbf{E}_{\text{num}} = [\mathbf{E}_{\text{num}}^{(1)}, \mathbf{E}_{\text{num}}^{(2)}, \ldots], \tag{14}$$

and attention is computed within blocks.

Complexity Analysis:

Standard attention complexity is $\mathcal{O}((k+n)^2 d)$; blockwise attention reduces it to $\mathcal{O}(B^2 d)$, where B is the block size.

5 Squeeze-and-Excitation (SE) Layer

The SE layer adaptively recalibrates feature maps by modeling channel-wise relationships.

5.1 Squeeze Step

Compute the channel-wise global descriptor $\mathbf{z} \in \mathbb{R}^d$:

$$\mathbf{z} = \frac{1}{k+n}\sum_{i=1}^{k+n}\mathbf{E}_i, \tag{15}$$

where $\mathbf{E}_i \in \mathbb{R}^d$ is the i-th feature embedding.

5.2 Excitation Step

Pass $\mathbf{z}$ through a bottleneck consisting of two fully connected layers:

$$\mathbf{s} = \sigma\left(\mathbf{W}_2 \cdot \delta\left(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1\right) + \mathbf{b}_2\right), \tag{16}$$

where:

- $\mathbf{W}_1 \in \mathbb{R}^{d \times \frac{d}{r}}$, $\mathbf{b}_1 \in \mathbb{R}^{\frac{d}{r}}$.
- $\mathbf{W}_2 \in \mathbb{R}^{\frac{d}{r} \times d}$, $\mathbf{b}_2 \in \mathbb{R}^d$.
- $\delta(\cdot)$ is the ReLU activation function.
- $\sigma(\cdot)$ is the sigmoid activation function.

5.3 Reweighting

Reweight the embeddings:

$$\mathbf{E}'_i = \mathbf{E}_i \odot \mathbf{s}, \tag{17}$$

for $i = 1, \ldots, k+n$, where $\odot$ denotes element-wise multiplication.

6 Lightweight Feedforward Layer

Process the reweighted embeddings through a feedforward network.

6.1 Feedforward Network with Residual Connection

The feedforward operation is defined as:

$$\mathbf{F}(\mathbf{E}') = \mathbf{E}' + \phi\left(\mathbf{W}_f \mathbf{E}' + \mathbf{b}_f\right), \tag{18}$$

where:

- $\mathbf{W}_f \in \mathbb{R}^{d \times d'}$, $\mathbf{b}_f \in \mathbb{R}^{d'}$.
- $\phi(\cdot)$ is an activation function (e.g., ReLU).
- The residual connection $\mathbf{E}'$ ensures gradient flow.

6.2 Bottleneck Structure

To reduce computational cost, we use a bottleneck:

$$\mathbf{F}(\mathbf{E}') = \mathbf{E}' + \mathbf{W}_f^{(2)} \phi\left(\mathbf{W}_f^{(1)} \mathbf{E}' + \mathbf{b}_f^{(1)}\right) + \mathbf{b}_f^{(2)}, \tag{19}$$

where:

- $\mathbf{W}_f^{(1)} \in \mathbb{R}^{d \times d_b}$, $\mathbf{W}_f^{(2)} \in \mathbb{R}^{d_b \times d}$.
- $d_b < d$ is the bottleneck dimension.

7 Output Layer

The processed features are fed into the output layer for prediction.

7.1 Classification or Regression Head

For classification, the output probabilities are computed using:

$$\hat{y} = \text{softmax}\left(\mathbf{W}_{\text{out}} \mathbf{F}(\mathbf{E}') + \mathbf{b}_{\text{out}}\right), \tag{20}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{C \times d}$, C is the number of classes.
For regression:

$$\hat{y} = \mathbf{W}_{\text{out}} \mathbf{F}(\mathbf{E}') + \mathbf{b}_{\text{out}}, \tag{21}$$

with $\hat{y} \in \mathbb{R}$.

8 Efficient Training and Optimization Techniques

8.1 Layer Normalization

Layer normalization is applied to stabilize the activations:

$$\text{LayerNorm}(\mathbf{h}) = \frac{\mathbf{h} - \mu}{\sigma} \odot \gamma + \beta, \tag{22}$$

where:

- $\mu = \frac{1}{d}\sum_{i=1}^{d} h_i$, $\sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(h_i - \mu)^2}$.
- $\gamma, \beta \in \mathbb{R}^d$ are learnable parameters.

8.2 Mixed Precision Training

Utilize lower-precision floating-point formats (e.g., FP16) for weights and activations to reduce memory usage and increase computational speed.

8.3 Gradient Clipping

To prevent exploding gradients, gradients are clipped:

$$\nabla\theta \leftarrow \nabla\theta \cdot \min\left(1, \frac{\tau}{\|\nabla\theta\|}\right), \tag{23}$$

where τ is the clipping threshold.

8.4 Learning Rate Scheduling

Implement cosine annealing for the learning rate η:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{\pi t}{T}\right)\right), \tag{24}$$

where t is the current epoch, and T is the total number of epochs.

9 Pruning and Quantization

9.1 Structured Pruning

Remove entire neurons or filters based on their importance scores s_i:

$$s_i = \|\mathbf{w}_i\|_2, \tag{25}$$

where $\mathbf{w}_i$ is the weight vector associated with the i-th neuron.

9.2 Quantization

Reduce precision of weights and activations:

$$w_q = \text{round}\left(\frac{w}{\Delta}\right)\Delta, \tag{26}$$

where Δ is the quantization step size.

10 Sparse Attention and Feature Selection

10.1 Sparse Cross-Attention

Introduce a sparsity mask $\mathbf{M} \in \{0, -\infty\}^{k \times n}$:

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}, \tag{27}$$

where $\mathbf{M}_{ij} = -\infty$ if the i-th query should not attend to the j-th key.

10.2 Feature Selection via Principal Component Analysis (PCA)

Reduce dimensionality of numerical features:

$$\mathbf{x}_{\text{PCA}} = \mathbf{U}^\top(\mathbf{x} - \mu), \tag{28}$$

where $\mathbf{U}$ contains the top p eigenvectors of the covariance matrix, and μ is the mean vector.

11 Regularization and Generalization

11.1 Dropout

Randomly zero out elements of the embeddings:

$$\tilde{\mathbf{E}} = \mathbf{E} \odot \mathbf{D}, \tag{29}$$

where $\mathbf{D} \sim \text{Bernoulli}(p)$, and p is the keep probability.

11.2 Early Stopping

Monitor validation loss L_{val} and stop training when:

$$L_{\text{val}}^{(t)} > L_{\text{val}}^{(t-\delta t)}, \tag{30}$$

for a specified patience period δt.

11.3 Weight Decay

Add L2 regularization term to the loss function L:

$$L_{\text{total}} = L + \lambda \|\theta\|_2^2, \tag{31}$$

where λ is the weight decay coefficient.

12 Complete Architecture Flow

1. **Input Embedding Layer**:
   - Transform categorical and numerical features into embeddings $\mathbf{E}$.
2. **Cross-Attention Layer**:
   - Compute cross-attention between categorical queries and numerical keys/values.
3. **Squeeze-and-Excitation Layer**:
   - Generate channel-wise weights $\mathbf{s}$ and reweight embeddings.

4. **Lightweight Feedforward Layer**:
    - Process embeddings through feedforward network with residual connections.
5. **Output Layer**:
    - Produce final predictions $\hat{y}$.
6. **Optimization and Regularization**:
    - Apply techniques like LayerNorm, dropout, and weight decay.

13 Key Benefits of This Architecture

1. **Efficiency**: Cross-attention reduces computational complexity from $\mathcal{O}((k+n)^2)$ to $\mathcal{O}(kn)$.
2. **Feature Importance**: SE layers emphasize important features through learned scaling factors.
3. **Training Efficiency**: Techniques like mixed precision training and gradient clipping improve convergence speed.
4. **Regularization**: Methods like dropout and weight decay prevent overfitting.
5. **Sparse Attention**: Focusing on important feature interactions reduces unnecessary computations.

14 Conclusion

The proposed architecture effectively balances computational efficiency and performance for tabular data processing. By integrating cross-attention mechanisms and squeeze-and-excitation layers, the model captures intricate feature interactions while remaining scalable. The detailed mathematical formulations provide a solid foundation for implementation and further research.