1 InteractionArch

一共有三种 Interaction 备选：

1. Default：Returns the pairwise dot product of each sparse feature pair, the dot product of each sparse features with the output of the dense layer, and the dense layer itself (all concatenated).
2. DCN：Returns the output of a Deep Cross Net v2 https://arxiv.org/pdf/2008.13535.pdf with a low rank approximation for the weight matrix. The input and output sizes are the same for this interaction layer (F*D + D).
3. Projection：Return Y*Z and the dense layer itself (all concatenated) where Y is the output of interaction branch 1 and Z is the output of interaction branch 2. Y and Z are of size Bx(F1xD) and Bx(DxF2) respectively for some F1 and F2.

2 Default Interaction

forward：

```
# dense_features.shape = [B, 128]
# sparse_features.shape = [B, 26, 128]
combined_values =
    torch.cat((dense_features.unsqueeze(1), sparse_features),
              dim=1)
# combined_values.shape = [B, 27, 128]

# [B, 27, 27] = [B, 27, 128] <bmm> [B, 128, 27]
interactions =
    torch.bmm(combined_values,
              torch.transpose(combined_values, 1, 2))
# [B, 351]
interactions_flat =
    interactions[:, self.triu_indices[0], self.triu_indices[1]]

# concatenated_dense.shape = [B, 479]
concatenated_dense =
    torch.cat((dense_features, interactions_flat), dim=1)
```

给 forward 形式化的表达。为了方便推导，我们假设 $B = 2$，$embedding\_dim = 4$，$sparse\_feature\_size = 2$：

$$D\ for\ dense[2,4] = \begin{bmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \end{bmatrix}$$

$$S\ for\ sparse[2,2,4] = \begin{bmatrix} [s_{0,0,0}, s_{0,0,1}, s_{0,0,2}, s_{0,0,3}] & [s_{0,1,0}, s_{0,1,1}, s_{0,1,2}, s_{0,1,3}] \\ [s_{1,0,0}, s_{1,0,1}, s_{1,0,2}, s_{1,0,3}] & [s_{1,1,0}, s_{1,1,1}, s_{1,1,2}, s_{1,1,3}] \end{bmatrix}$$

$$C\ for\ Comb[2,3,4] = \begin{bmatrix} [d_{0,0}, d_{0,1}, d_{0,2}, d_{0,3}] & [s_{0,0,0}, s_{0,0,1}, s_{0,0,2}, s_{0,0,3}] & [s_{0,1,0}, s_{0,1,1}, s_{0,1,2}, s_{0,1,3}] \\ [d_{1,0}, d_{1,1}, d_{1,2}, d_{1,3}] & [s_{1,0,0}, s_{1,0,1}, s_{1,0,2}, s_{1,0,3}] & [s_{1,1,0}, s_{1,1,1}, s_{1,1,2}, s_{1,1,3}] \end{bmatrix}$$

$$C^T[2,4,3] = \begin{bmatrix} [d_{0,0}, s_{0,0,0}, s_{0,1,0}] & [d_{0,1}, s_{0,0,1}, s_{0,1,1}] & [d_{0,2}, s_{0,0,2}, s_{0,1,2}] & [d_{0,3}, s_{0,0,3}, s_{0,1,3}] \\ [d_{1,0}, s_{1,0,0}, s_{1,1,0}] & [d_{1,1}, s_{1,0,1}, s_{1,1,1}] & [d_{1,2}, s_{1,0,0}, s_{1,1,2}] & [d_{1,3}, s_{1,0,3}, s_{1,1,3}] \end{bmatrix}$$

$$flat(CC^T)|_{b=\{0,1\}} = \begin{bmatrix} \cdots & F_{b,0} & F_{b,1} \\ F_{b,0} & \cdots & F_{b,2} \\ F_{b,1} & F_{b,2} & \cdots \end{bmatrix}$$

其中：

$$F_{b,0} = d_{b,0}s_{b,0,0} + d_{b,1}s_{b,0,1} + d_{b,2}s_{b,0,2} + d_{b,3}s_{b,0,3}$$
$$F_{b,1} = d_{b,0}s_{b,1,0} + d_{b,1}s_{b,1,1} + d_{b,2}s_{b,1,2} + d_{b,3}s_{b,1,3}$$
$$F_{b,2} = s_{b,0,0}s_{b,1,0} + s_{b,0,1}s_{b,1,1} + s_{b,0,2}s_{b,1,2} + s_{b,0,3}s_{b,1,3}$$

$$Y[2,7] = \begin{bmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} & F_{0,0} & F_{0,1} & F_{0,2} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} & F_{1,0} & F_{1,1} & F_{1,2} \end{bmatrix}$$

backward：已知 $\frac{\partial L}{\partial Y}$，求 $\frac{\partial L}{\partial D}$ 和 $\frac{\partial L}{\partial S}$。

$$\frac{\partial L}{\partial Y} = \begin{bmatrix} \frac{\partial L}{\partial y_{0,0}} & \frac{\partial L}{\partial y_{0,1}} & \frac{\partial L}{\partial y_{0,2}} & \frac{\partial L}{\partial y_{0,3}} & \frac{\partial L}{\partial y_{0,4}} & \frac{\partial L}{\partial y_{0,5}} & \frac{\partial L}{\partial y_{0,6}} \\ \frac{\partial L}{\partial y_{1,0}} & \frac{\partial L}{\partial y_{1,1}} & \frac{\partial L}{\partial y_{1,2}} & \frac{\partial L}{\partial y_{1,3}} & \frac{\partial L}{\partial y_{1,4}} & \frac{\partial L}{\partial y_{1,5}} & \frac{\partial L}{\partial y_{1,6}} \end{bmatrix}, \frac{\partial L}{\partial D} = \begin{bmatrix} \frac{\partial L}{\partial d_{0,0}} & \frac{\partial L}{\partial d_{0,1}} & \frac{\partial L}{\partial d_{0,2}} & \frac{\partial L}{\partial d_{0,3}} \\ \frac{\partial L}{\partial d_{1,0}} & \frac{\partial L}{\partial d_{1,1}} & \frac{\partial L}{\partial d_{1,2}} & \frac{\partial L}{\partial d_{1,3}} \end{bmatrix}$$

以 $\frac{\partial L}{\partial d_{0,0}}$ 为例：

$$\frac{\partial L}{\partial d_{0,0}} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial d_{0,0}} = \frac{\partial L}{\partial Y}\begin{bmatrix} 1 & 0 & 0 & 0 & s_{0,0,0} & s_{0,1,0} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \frac{\partial L}{\partial y_{0,0}} + s_{0,0,0}\frac{\partial L}{\partial y_{0,4}} + s_{0,1,0}\frac{\partial L}{\partial y_{0,5}}$$

类似的，可以计算出所有的项：

$$\frac{\partial L}{\partial d_{i,j}} = \frac{\partial L}{\partial y_{i,j}} + s_{i,0,j}\frac{\partial L}{\partial y_{i,4}} + s_{i,1,j}\frac{\partial L}{\partial y_{i,5}}$$

$$\frac{\partial L}{\partial D} = \frac{\partial L}{\partial Y}|_{[:,0:4]} + \begin{bmatrix} s_{0,0,0}\frac{\partial L}{\partial y_{0,4}} + s_{0,1,0}\frac{\partial L}{\partial y_{0,5}} & s_{0,0,1}\frac{\partial L}{\partial y_{0,4}} + s_{0,1,1}\frac{\partial L}{\partial y_{0,5}} & s_{0,0,2}\frac{\partial L}{\partial y_{0,4}} + s_{0,1,2}\frac{\partial L}{\partial y_{0,5}} & s_{0,0,3}\frac{\partial L}{\partial y_{0,4}} + s_{0,1,3}\frac{\partial L}{\partial y_{0,5}} \\ s_{1,0,0}\frac{\partial L}{\partial y_{1,4}} + s_{1,1,0}\frac{\partial L}{\partial y_{1,5}} & s_{1,0,1}\frac{\partial L}{\partial y_{1,4}} + s_{1,1,1}\frac{\partial L}{\partial y_{1,5}} & s_{1,0,2}\frac{\partial L}{\partial y_{1,4}} + s_{1,1,2}\frac{\partial L}{\partial y_{1,5}} & s_{1,0,3}\frac{\partial L}{\partial y_{1,4}} + s_{1,1,3}\frac{\partial L}{\partial y_{1,5}} \end{bmatrix}$$

$$\frac{\partial L}{\partial D} = \frac{\partial L}{\partial Y}|_{[:,0:4]} + \begin{bmatrix} \frac{\partial L}{\partial y_{0,4}} & \frac{\partial L}{\partial y_{0,5}} \\ \frac{\partial L}{\partial y_{1,4}} & \frac{\partial L}{\partial y_{1,5}} \end{bmatrix} S$$

注意中间这一项的 shape 是 $[2,1,2]$，左乘 S 之后得到一个 $shape = [2,1,4]$，需要把中间的这个维度 $squeeze$ 掉再和第一项相加。因为 Interaction 是两个输入，因此还需要计算关于另一个输入的导数：

$$\frac{\partial L}{\partial S} = \begin{bmatrix} [\frac{\partial L}{\partial s_{0,0,0}}, \frac{\partial L}{\partial s_{0,0,1}}, \frac{\partial L}{\partial s_{0,0,2}}, \frac{\partial L}{\partial s_{0,0,3}}] & [\frac{\partial L}{\partial s_{0,1,0}}, \frac{\partial L}{\partial s_{0,1,1}}, \frac{\partial L}{\partial s_{0,1,2}}, \frac{\partial L}{\partial s_{0,1,3}}] \\ [\frac{\partial L}{\partial s_{1,0,0}}, \frac{\partial L}{\partial s_{1,0,1}}, \frac{\partial L}{\partial s_{1,0,2}}, \frac{\partial L}{\partial s_{1,0,3}}] & [\frac{\partial L}{\partial s_{1,1,0}}, \frac{\partial L}{\partial s_{1,1,1}}, \frac{\partial L}{\partial s_{1,1,2}}, \frac{\partial L}{\partial s_{1,1,3}}] \end{bmatrix}$$

以第一项为例：

$$\frac{\partial L}{\partial s_{0,0,0}} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial s_{0,0,0}} = \frac{\partial L}{\partial Y}\begin{bmatrix} 0 & 0 & 0 & 0 & d_{0,0} & 0 & s_{0,1,0} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = d_{0,0}\frac{\partial L}{\partial y_{0,4}} + s_{0,1,0}\frac{\partial L}{\partial y_{0,6}}$$

由于 S 比较长，省略一些中间过程：

$$\frac{\partial L}{\partial S} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

其中：

$$A = \begin{bmatrix} \frac{\partial L}{\partial y_{0,4}} & \frac{\partial L}{\partial y_{0,6}} \end{bmatrix}\begin{bmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ s_{0,1,0} & s_{0,1,1} & s_{0,1,2} & s_{0,1,3} \end{bmatrix}, B = \begin{bmatrix} \frac{\partial L}{\partial y_{0,5}} & \frac{\partial L}{\partial y_{0,6}} \end{bmatrix}\begin{bmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ s_{0,0,0} & s_{0,0,1} & s_{0,0,2} & s_{0,0,3} \end{bmatrix}$$
$$C = \begin{bmatrix} \frac{\partial L}{\partial y_{1,4}} & \frac{\partial L}{\partial y_{1,6}} \end{bmatrix}\begin{bmatrix} d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ s_{1,1,0} & s_{1,1,1} & s_{1,1,2} & s_{1,1,3} \end{bmatrix}, D = \begin{bmatrix} \frac{\partial L}{\partial y_{1,5}} & \frac{\partial L}{\partial y_{1,6}} \end{bmatrix}\begin{bmatrix} d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ s_{1,0,0} & s_{1,0,1} & s_{1,0,2} & s_{1,0,3} \end{bmatrix}$$

整理得：

3 DCN Interaction

forward：

```
# dense_features.shape = [B, 128]
# sparse_features.shape = [B, 26, 128]
# combined_values.shape = [B, 27, 128]
combined_values =
    torch.cat((dense_features.unsqueeze(1), sparse_features),
              dim=1)

# size B X (F*D + D)
concatenated_dense = crossnet(combined_values.reshape([B, -1]))
```

LowRankCrossNet（当前参考代码使用的是这个）

1. 初始化
   - $N = in\_features = (num\_sparse\_features + 1) * embedding\_dim$
   - num_layers
   - $K = low\_rank$
   - 初始化 num_layers 个 w, v, b
   - $w.shape = [in\_features, low\_rank]$
   - $v.shape = [low\_rank, in\_features]$
   - $b.shape = [in\_features, 1]$
2. 计算

$$x_{\underset{[B,N,1]}{l+1}} = \hat{x}_{\underset{[B,N,1]}{}} \otimes \left[\underbrace{w_{\underset{[N,K]}{l}} * \left(\underbrace{v_{\underset{[K,N]}{l}} * x_{\underset{[B,N,1]}{l}}}_{[B,K,1]} \right)}_{[B,N,1]} + b_{\underset{[N,1]}{l}} \right] + x_{\underset{[B,N,1]}{l}}$$

   其中 $\hat{x} = input.unsqueeze(2), x_0 = \hat{x}, l \in [0, num\_layers-1]$。$\otimes$ 代表 element-wise multiplication，$*$ 代表 matrix multiplication。

3. 最终输出 shape 为 [B, in_features]，因此在使用这个 Interaction 的时候，OverArch 的第一层就是 $[in\_features, 1024]$。

CrossNet（这个是论文新提出的方法）

1. 初始化
   - $N = in\_features = (num\_sparse\_features + 1) * embedding\_dim$
   - $num\_layers = wait\ for\ input$
   - 初始化 num_layers 个 w, v, b
   - $w.shape = [in\_features, in\_features]$
   - $b.shape = [in\_features, 1]$
2. 计算

$$x_{\underset{[B,N,1]}{l+1}} = \hat{x}_{\underset{[B,N,1]}{}} \otimes \left[\underbrace{w_{\underset{[N,N]}{l}} * x_{\underset{[B,N,1]}{l}}}_{[B,N,1]} + b_{\underset{[N,1]}{l}} \right] + x_{\underset{[B,N,1]}{l}}$$

   其中 $\hat{x} = input.unsqueeze(2), x_0 = \hat{x}, l \in [0, num\_layers-1]$。$\otimes$ 代表 element-wise multiplication，$*$ 代表 matrix multiplication。

3. 最终输出 shape 为 [B, in_features]，因此在使用这个 Interaction 的时候，OverArch 的第一层就是 $[in\_features, 1024]$。

VectorCrossNet

1. 初始化
   - $N = in\_features = (num\_sparse\_features + 1) * embedding\_dim$
   - $num\_layers = wait\ for\ input$

- 初始化 num_layers 个 w, v, b
- $w.shape = [in\_features, 1]$
- $b.shape = [in\_features, 1]$

2. 计算

$$\underset{[B,N,1]}{x_{l+1}} = \underset{[B,N,1]}{\hat{x}} * \left(\underbrace{\underset{[N,1]}{w_l} \bullet \underset{[B,N,1]}{x_l}}_{[B,1,1]} \right) + \underset{[N,1]}{b_l} + \underset{[B,N,1]}{x_l}$$

其中 $\hat{x} = input.unsqueeze(2)$，$x_0 = \hat{x}$，$l \in [0, num\_layers - 1]$。$\bullet$ 代表 tensordot，$*$ 代表 matrix multiplication。

LowRankMixtureCrossNet

1. 初始化

- $N = in\_features = (num\_sparse\_features + 1) * embedding\_dim$
- num_layers
- $K = num\_experts$
- $L = low\_rank$
- $activation = for\ example\ relu$
- 初始化 num_layers 个 u, v, b, c
- $u.shape = [num\_experts, in\_features, low\_rank]$
- $v.shape = [num\_experts, low\_rank, in\_features]$
- $b.shape = [in\_features, 1]$
- $c.shape = [num\_experts, low\_rank, low\_rank]$
- $num\_experts$ 个线性层 $g = Linear(N, 1, bias = False)$，对所有的 LowRankMixtureCrossNet 层共用。

2. 计算

- $\hat{x} = input.unsqueeze(2), [B, N, 1]$
- $x_l[B, N]$ 分别经过 $num\_experts$ 个线性层，每层输出一个 $[B, 1]$，stack 到一起组成 $gating_l[B, K, 1]$
- 用 $f(x)$ 表示激活函数，计算

$$expert_l[e] = \hat{x} * \left[u_l[e] * f\left(c_l[e] * f\left(v_l[e] * x_l \right) \right) + b_l \right], e \in [0, num\_experts - 1]$$

- 此时 $expert_l[e].shape = [B, N, 1]$，squeeze 掉最后一维之后 $expert_l[e].shape = [B, N]$。
- 把所有的 $expert_l[e]$ 叠起来，$experts_l.shape = [B, N, K]$:

$$experts_l = satck([expert_l[e]\ e\ from\ 0\ to\ num\_experts], 2)$$

- 再计算

$$x_{l+1} = experts_l * softmax(gating_l) + x_l$$

- 以上这是一层 LowRankMixtureCrossNet，每层都是一样的，首尾相接，最后输出的是一个 $[B, N, 1]$，后续处理与其他的 CrossNet 类似。

4 Projection Interaction

简单描述就是来自 SparseArch 和 DenseArch 的 feature 先拼在一起，然后分别过两个 mlp，然后再 bmm，再 concatenate。forward:

```
# combined_values.shape = [B, 27, 128]
# dense_features.shape = [B, 128], D=128
combined_values = torch.cat(
    (dense_features.unsqueeze(1), sparse_features), dim=1)

interaction_branch1_out = self.interaction_branch1(
    torch.reshape(combined_values, (B, -1)))  # [B, 256]

```

```
interaction_branch2_out = self.interaction_branch2(
    torch.reshape(combined_values, (B, -1))) # [B, 512]

interactions = torch.bmm(
    interaction_branch1_out.reshape([B, -1, D]),  # [B, 2, 128]
    interaction_branch2_out.reshape([B, D, -1]),  # [B, 128, 4]
)

# interactions_flat.shape = [B, 8]
interactions_flat = torch.reshape(interactions, (B, -1))
# concatenated_dense.shape = [B, 136]
concatenated_dense = torch.cat((dense_features, interactions_flat),
                               dim=1)
```

一些关于初始化参数的细节

- $num\_sparse\_features = 26$。
- $interaction\_branch1 = DenseArch\{-?-\}$，需要给定超参来指定，例如 $D_1, D_2, ..., D$。
- $interaction\_branch2 = DenseArch\{-?-\}$，需要给定超参来指定，例如 $E_1, E_2, ..., E$。
- 这两个 DenseArch 最后一层的输出 size$D$ 和 E 必须都是 $embedding\_dim$ 的整数倍，但 D 和 E 可以不同。
- 设 $projected\_dim\_X = interaction\_branchX\_layer\_sizes[-1]//embedding\_dim$，则使用 ProjectionInteraction 是的 OverArch 的第一层的输入 size 就是 $embedding\_dim + projected\_dim\_1 * projected\_dim\_2 = 128 + 2 * 4 = 136$
- $interaction\_branch$ 的具体结构和层数是超参控制的。

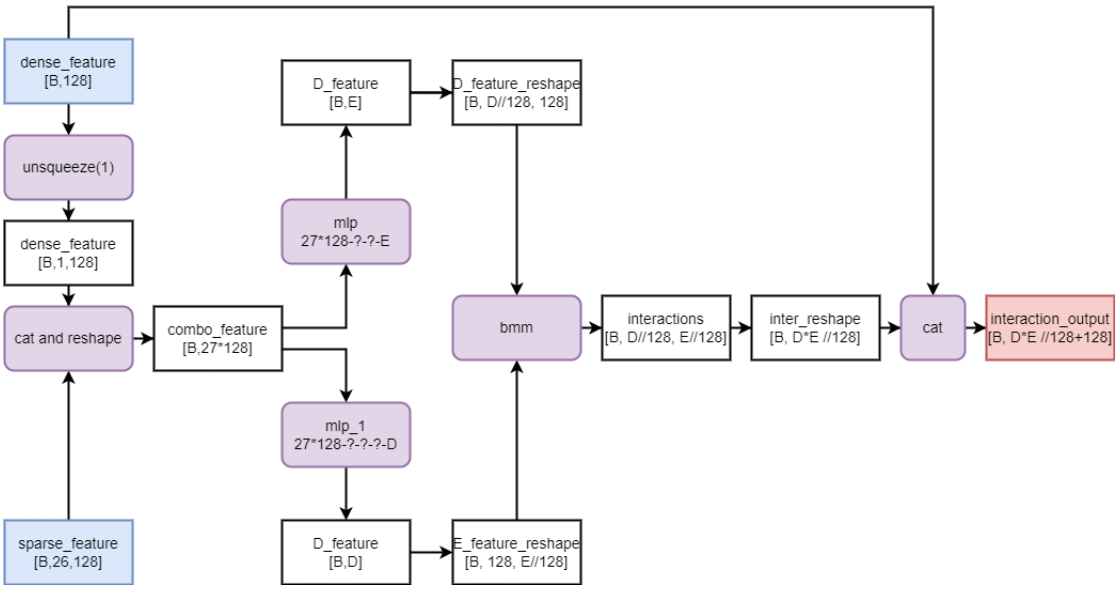

Figure 1: Projection Interaction