# Set Transformers for Lineup Level Models

Aaron Danielson

July 1, 2025

## Overview

This paper describes a general methodology to compress player level information into a lineup embedding which can be used for various tasks. Broadly, player-level features are transformed to create player level embeddings. These may contain the identity of the player, but they can be completely based on attributes of the player. Then, depending on the type of target being modeled, a series of set transformers are applied to interact the features. Then, the resulting embeddings are aggregated using some type of pooling (sum, max, attention pooling) which does not violate the permutation invariance of the lineup. Finally, the lineup embedding is passed to a prediction head (or heads) which are used to focus the attention onto a specific target.

DeepRAPM is a transformer-based model designed to evaluate player impact in basketball possessions. It leverages player embeddings, role-aware attention, and position-indexed biases to model offensive and defensive contributions.

## 1 Input Embeddings

Each possession involves 10 players:

- $p_i$: Player ID
- $r_i \in \{0, 1\}$: Role ID (0 = offense, 1 = defense)
- $s_i \in \{0, \dots, 4\}$: Discrete position index (PG, SG, SF, PF, C)

**Embeddings:**

$$\begin{aligned} \mathbf{e}_i &= \text{PlayerEmbed}(p_i) \in \mathbb{R}^d \\ \mathbf{r}_i &= \text{RoleEmbed}(r_i) \in \mathbb{R}^d \\ \mathbf{z}_i^{(0)} &= [\mathbf{e}_i; \mathbf{r}_i] \in \mathbb{R}^{2d} \end{aligned}$$

## 2 Encoder and Transformer Input

Each embedding $\mathbf{z}_i^{(0)}$ is passed through an MLP block

$$\mathbf{z}_i = \text{MLP}(\mathbf{z}_i^{(0)}) \in \mathbb{R}^d$$

to imbue it with an offensive or defensive interpretation. Then, we stack the embeddings across players to get $\mathbf{Z} \in \mathbb{R}^{10\times d}$. These embeddings will be passed to our set transformer layers.

## 3 Attention Bias via Indexed Lookup

To define an attention mask, consider the categorical index combining the offense/defense role and the position played. Let $c_i$ denote the index

$$c_i = r_i \cdot K + s_i, \quad K = 5$$

such that with a learnable bias tensor:

$$\boldsymbol{B} \in \mathbb{R}^{H\times 10\times 10}$$

The bias between player $i$ and $j$ in head $h$ is

$$\text{Bias}_{h,i,j} = \boldsymbol{B}_{h,c_i,c_j}.$$

This bias term will be added to the attention matrix before a softmax is applied to its rows. A large negative value drives the weights towards zero while a large positive value pushes them towards 1.

## 4 Self-Attention Mechanism

Since the embeddings for the offensive and defensive players are stacked into a single matrix, we can conceive of our set transformer as self-attention (as opposed to cross attention). For a single attention head $h$, the attention score between each player $i$ (query) and $j$ (key) is computed as

$$\alpha_{ij}^{(h)} = \frac{\mathbf{q}_i^{(h)} \cdot \mathbf{k}_j^{(h)}}{\sqrt{d'_h}} + \boldsymbol{B}_{h,c_i,c_j}.$$

where $\mathbf{q}_i^{(h)} = \mathbf{z}_i \mathbf{W}_h^Q$ is the query vector for player $i$ and $\mathbf{k}_j^{(h)} = \mathbf{z}_j \mathbf{W}_h^K$ is the key vector for player $j$. Note that the weight matrices $\mathbf{W}_h^Q \sim d \times d'_h$ and $\mathbf{W}_h^K \sim d \times d'_h$ transform the player embeddings. When $d \neq d'_h$, they alter the dimension of the embeddings. The resulting dimension $d'$ is sometimes called the internal embedding dimension. Note that the internal dimension is split equally across each of the attention heads such that $d'_h = d'/H$.

Intuitively, the query is an item for which we would like to return information and the key is the fingerprint or signature we use to assess the suitability for the query. When two vectors are very similar, their dot product is large. This implies the query attends to that particular key.

The full attention score matrix for head $h$, denoted $\boldsymbol{A}^{(h)} \in \mathbb{R}^{10\times 10}$, is thus:

$$\boldsymbol{A}^{(h)} = \begin{bmatrix} \alpha_{11}^{(h)} & \alpha_{12}^{(h)} & \cdots & \alpha_{1,10}^{(h)} \\ \alpha_{21}^{(h)} & \alpha_{22}^{(h)} & \cdots & \alpha_{2,10}^{(h)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{10,1}^{(h)} & \alpha_{10,2}^{(h)} & \cdots & \alpha_{10,10}^{(h)} \end{bmatrix} = \frac{\mathbf{Q}_h \mathbf{K}_h^\top}{\sqrt{d'_h}} + \boldsymbol{B}_h$$

This structure allows the attention mechanism to consider both similarity in the embedding space and inductive biases based on role-position pairings. The softmax operation is applied *row-wise* across the keys for each query

$$\text{softmax}\left(\boldsymbol{A}^{(h)}\right)_{ij} = \frac{\exp(\alpha_{ij}^{(h)})}{\sum_{j'=1}^{10} \exp(\alpha_{ij'}^{(h)})}.$$

Intuitively, row $i$ of the attention matrix is the distribution of player $i$'s attention over the players on the court. The attention can be interpreted as the relevance of the information contained in player $j$'s embedding for a particular task, in this case predicting the number of points scored during a possession. This results in a normalized attention distribution for each query (row), determining how much that player's query vector attends to each of the other players' key vectors. Finally, the attention output for player $i$ is the weighted sum of the values

$$\text{Attn}_i = \sum_{j=1}^{10} \text{softmax}\left(\boldsymbol{A}^{(h)}\right)_{ij} \cdot \mathbf{v}_j^{(h)}.$$

This design ensures that each player's contextual representation is formed based on a weighted combination of all other players' value vectors, guided by role-position biases and query-key similarity.

$$\mathbf{Q}_h = \mathbf{Z}\mathbf{W}_h^Q, \quad \mathbf{K}_h = \mathbf{Z}\mathbf{W}_h^K, \quad \mathbf{V}_h = \mathbf{Z}\mathbf{W}_h^V$$
$$\text{Attn}_h = \text{softmax}\left(\frac{\mathbf{Q}_h\mathbf{K}_h^\top}{\sqrt{d'_h}} + \text{Bias}_h\right)\mathbf{V}_h$$

## 5 Multi-Head Attention Aggregation

Each attention head $h$ computes its own output:

$$\text{Attn}_h = \text{softmax}\left(\frac{\mathbf{Q}_h\mathbf{K}_h^\top}{\sqrt{d}} + \text{Bias}_h\right)\mathbf{V}_h \quad \in \mathbb{R}^{10\times d_h}$$

for $h = 1, \dots, H$.

This formulation ensures that the attention score accounts for both feature similarity (via dot product) and learned relational priors (via indexed biases). These are concatenated across all heads:

$$\text{Concat}(\text{Attn}_1, \dots, \text{Attn}_H) \in \mathbb{R}^{10\times(H\cdot d_h)}$$

Then projected back to the model dimension:

$$\text{MultiHead}(\mathbf{Z}) = \text{Concat}(\text{Attn}_1, \dots, \text{Attn}_H) \cdot \mathbf{W}^O$$

where $\mathbf{W}^O \in \mathbb{R}^{(H\cdot d_h)\times d}$ is the output projection matrix.

This allows the model to jointly attend to information from different representation subspaces at different positions.

## 6 10 × 10 Attention Matrix Structure

Each head operates on a 10 × 10 attention matrix:

$$\text{AttentionMatrix} = \begin{bmatrix} \text{OO}_{5\times 5} & \text{OD}_{5\times 5} \\ \text{DO}_{5\times 5} & \text{DD}_{5\times 5} \end{bmatrix}$$

Where:

- OO: Offense to Offense (team synergy)
- OD: Offense to Defense (targeting matchups)
- DO: Defense to Offense (matchups / reactions)
- DD: Defense to Defense (defensive coordination)

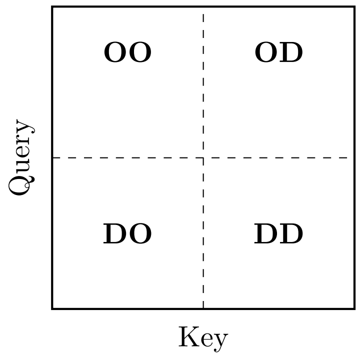

Figure 1: Structure of the 10 × 10 attention matrix in DeepRAPM.

## 7 Transformer Layer Computation

$$\begin{aligned} \mathbf{Z}_1 &= \text{LayerNorm}(\mathbf{Z}) \\ \mathbf{Z}_2 &= \mathbf{Z} + \text{MultiHead}(\mathbf{Z}_1) \\ \mathbf{Z}_3 &= \text{LayerNorm}(\mathbf{Z}_2) \\ \mathbf{Z}_4 &= \mathbf{Z}_2 + \text{MLP}(\mathbf{Z}_3) \end{aligned}$$

## 8 Pooling and Output

The 10 transformed player embeddings are pooled:

$$\mathbf{z}_{\text{lineup}} = \frac{1}{10} \sum_{i=1}^{10} \mathbf{z}_i$$

Final prediction:

$$\hat{y} = \text{MLP}_{\text{head}}(\mathbf{z}_{\text{lineup}})$$

## 9 Loss

We support both:

- Cross-entropy for multiclass scoring outcomes (0–4 points)
- Gaussian NLL for probabilistic regression

$$\mathcal{L}_{\text{CE}} = -\sum_{c=1}^{5} y_c \log \hat{y}_c \quad \mathcal{L}_{\text{NLL}} = \frac{1}{2} \log \hat{\sigma}^2 + \frac{(y - \hat{\mu})^2}{2\hat{\sigma}^2}$$

## Conclusion

This formulation allows DeepRAPM to learn interaction dynamics between players based on their roles, positions, and lineup context ? providing fine-grained interpretability and generalization to unseen lineups.