Generative AI recreating football game

1. List of different approaches of how a such game can be re-created :

Player Tracking: Use player tracking data, often obtained from GPS or computer vision systems (video analysis). This data provides detailed information about player positions, velocities, and movements on the field.

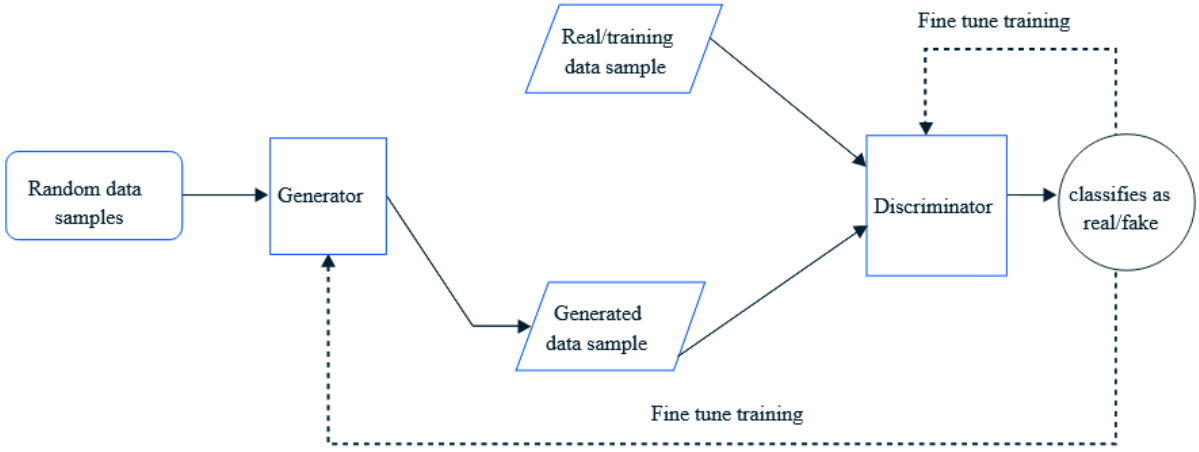
Drawing numbers from a distrubtion : Drawing numbers from a distribution is a popular technique for generating synthetic data, which involves sampling numbers from a distribution to create a dataset that follows a curve based on real-world data.

Machine Learning and AI: Machine learning and AI techniques can be applied to predict player actions, game outcomes, and even generate realistic game scenarios.

2. Model used :

1. Model presentation :

In our case, we have tabular data, the first idea is to use the GANs (Generative Adversarial Networks) models. GANs treat the learning process as a contest between two distinct networks: a generator network and a discriminator network that attempts to classify the samples between those from the real world and the fakes generated by the model. The generator adjusts the parameters of its model at each training iteration in order to generate more convincing examples to fool the discriminator.



Since we are dealing with tabular Data, we choose to use the CTGAN model. This model is designed to deal with challenges posed by tabular datasets. CTGAN attends to the specific characteristics of both numeric and categorical features and incorporates those characteristics into the generator model (in our case «label » : categorical, « norm » : numerical).

Numerical features :

Continuous characteristics in tabular data are frequently non-Gaussian, in contrast to image data, where pixel values typically follow a Gaussian-like distribution. Additionally, they frequently adhere to multimodal distributions, in which probability distributions contain more than one mode.

The CTGAN uses mode-specific normalization to capture these behaviors. Each value in a continuous feature is represented by a one-hot vector denoting its sampling mode and a scalar that reflects the value normalized in accordance with that mode using a VGM (Variational Gaussian Mixture) model.

Categorical features :

The "real" categorical values are explicitly encoded in a one-hot-vector while the generator outputs probability distributions across all feasible categorical values. These are easily distinguishable by the discriminator by comparing the distribution sparseness between real and synthetic data.

Model limitations

CTGAN has proven to be a powerful architecture for tabular data, it still has some limitations : Optimizing hyperparameters for datasets with different characteristics, Handling high-cardinality features…

2. Methodology :

The idea is to generate data by training a model for each type of action and then defining three styles {attacking, defending, balanced} of play in which each action has a specific weight. Using these weights and the duration of the matches, we will have the number of actions to generate for each type of action. All the actions will be shuffled to create a game and avoid suspicious patterns: {shot, shot, shot}.

- ### Preprocessing :

We create a data frame for each type of action. Each action will have a fixed gait length; we make sure that the gait length falls within a predefined range. Actions that do not have the maximal length will be imputed by random numbers drawn from the normal distribution of the same action. Finally, the gaits are divided into columns (one column per acceleration value).

The "no action" action is deleted because we have a single line for this action and so it is impossible to train a model for this action.

- ### Model generation

To generate a match, we've created a function that takes as parameters the duration of the match (between 0 and 120 minutes) and the style of match (attacking, defending, balanced). This function returns a dataframe with the same format as the two training files. This dataframe is then saved as a json file.

Bibliographie :

https://sdv.dev/SDV/user_guides/single_table/ctgan.html : CTGAN documentation