Player Performance Prediction using PyTorch

This document details a PyTorch-based deep learning model for predicting player performance scores and injury risks, leveraging physiological, performance, and psychological metrics gathered over five weeks. The model uses LSTM layers to process time-series data, enabling accurate and insightful predictions for sports analytics.

 **by Majid Alamdari**

Project Overview: Predicting Performance and Injury Risk

This project focuses on creating a robust system that predicts two critical outputs: player performance scores and injury risk. The model leverages a combination of physiological, performance, and psychological metrics to achieve these predictions. By using deep learning techniques, specifically LSTM layers, the model can learn complex patterns and temporal dependencies within the data.

The ultimate goal is to provide sports analysts and coaches with valuable insights that can inform training strategies, optimize player performance, and minimize the risk of injuries. This proactive approach can significantly enhance player well-being and team performance.

Input Features: Comprehensive Data Collection

The model utilizes a rich set of input features categorized into three main types:

- Physiological Metrics: Includes data such as **heart rate**, **oxygen saturation**, and other vital signs that reflect the player's physical condition.
- Performance Metrics: Encompasses variables like **speed**, **agility**, **endurance**, and other quantifiable measures of athletic ability.
- Psychological and Environmental Factors: Covers aspects such as **stress levels**, **fatigue**, and other psychological and environmental influences on performance.

These data are collected over a period of five weeks to capture trends and changes in player condition. The combination of these diverse inputs allows the model to provide a holistic assessment of player readiness and potential.

Output Predictions: Performance Score and Injury Risk

The model produces two distinct outputs, each serving a critical purpose:

- Performance Score: A **continuous numerical value** that represents the overall performance level of the player. This score is derived using regression techniques, providing a quantitative measure of athletic ability.
- Injury Risk: A **binary classification** indicating the likelihood of a player sustaining an injury. This output is crucial for proactive injury prevention and management.

By providing both a performance score and an injury risk assessment, the model offers a comprehensive overview of player status, enabling data-driven decision-making in sports management.

Deep Learning Architecture: LSTM-Based Time-Series Analysis

The core of the prediction model is its deep learning architecture, which utilizes LSTM layers to effectively handle time-series data. Separate LSTM layers are employed for each input type (physiological, performance, and psychological metrics) to capture the unique temporal patterns within each dataset.

Following the LSTM layers, the extracted features are combined and fed into a fully connected neural network. This network processes the integrated information to produce the final predictions for performance score and injury risk. The architecture is designed to capture complex interactions and dependencies between different input variables, providing accurate and nuanced predictions.

Training with PyTorch: Loss Functions and Optimization

The model is trained using the PyTorch deep learning framework, which provides the flexibility and tools needed to implement and optimize the complex architecture. The training process involves the following key components:

- Mean Squared Error (MSE) Loss: Used to measure the difference between the predicted and actual performance scores, guiding the model to improve its regression accuracy.
- Binary Cross-Entropy Loss (BCE): Applied to assess the accuracy of the injury risk classification, enabling the model to distinguish between high-risk and low-risk scenarios.
- Adam Optimizer: Employed to efficiently update the model's weights during training, converging towards optimal parameters that minimize the loss functions.

Through rigorous training and optimization, the model learns to make accurate predictions based on the input data, ensuring reliability and effectiveness in real-world applications.

Installation and Setup: Getting Started

To begin using this project, follow these steps:

1. Clone the repository from GitHub: