Emmanuel Oloruntola

Registration number 100387890

2025

Anomaly Detection in Football Player Tracking Using Autoencoders For Data Visualisation

Supervised by Mr Cheng Wang

University of East Anglia
Faculty of Science
School of Computing Sciences

Abstract

This project investigates frame-level anomaly detection in football player tracking data and using a deep learning approach and how it can be used in data visualisation. The primary objective was to identify unusual movement patterns in matches by training a 2D autoencoder on a sequence of player positions. The data was preprocessed from raw positional inputs, normalized into 42-dimensional vectors and then to visualise the data for human interpretation.
An autoencoder was implemented and trained in-browser using TensorFlow.js to allow real-time experimentation. Anomaly scores were computed using reconstruction error, with higher values indicating more unusual behavior. These scores were visualized through line charts, heatmaps, and timeline plots to aid interpretation. A JSON-based pipeline was used for clean data export and visual feedback.
Results showed consistent detection of outliers. This highlights the potential of autoencoders in assisting analysts with unsupervised, explainable insight into match tactics and dynamics.

Acknowledgements

I would like to thank my supervisor for their continuous guidance, feedback, and encouragement throughout the duration of this project.

Contents

1 Introduction

Player-tracking has taken football analysis from adding up shots to tracking every step a player may take. Thanks to modern pose-estimation models we can now pull a full-body "skeletons" straight from the broadcast video (Cioppa et al., 2022), so anyone with a computer can experiment with data that would once cost clubs a fortune to get. One 90-minute match at 25 frames , 22 players and 17 key-points already means **over a million** (x, y, z) **coordinates**. If you just gave a person all that data to look at in a spreadsheet it could be overstimulating and hard to understand, and guessing features by hand risks missing patterns we do not even know exist.

This project lets a neural network do the hard work. I utilise a **autoencoder** – a model that squeezes data through a narrow "bottleneck" layer and then tries to rebuild it If it can reconstruct skeletons well, that bottleneck should store the essence of player movement in just a few numbers. My goal is to be able to:

- those low-dimensional vectors will clump into recognisable phases of play
- identifying/spotting errors within gameplay that will flag *unusual* moments such as sudden presses or defensive panics.

Research question

> **Can an autoencoder trained on recorded football skeletons learn a low-dimensional map that reveals tactical patterns, and can its reconstruction error act as a useful unsupervised anomaly score?**

Objectives

1. Visualise that latent space with auto-encoder and see whether pressing, counters, or set-pieces fall into separate clusters.
2. Plot anomaly scores across time and pitch space to identify and highlight key moments in a match match.

1.1 Objectives (MoSCoW Analysis)

To help structure the development of the project, we created a MoSCoW table :

- **Must:**
  - Train a 2D autoencoder on player tracking data to detect behavioral anomalies.
  - Develop a browser-based demo to visualize anomaly scores in real-time.
- **Should:**
  - Create synthetic test datasets with injected anomalies for controlled evaluation.
  - Design and embed dashboard charts for timeline, heatmap, and scatter plots.
- **Could:**
  - Implement role inference per frame using movement patterns.
  - Compare behavior across multiple matches using radar and stacked charts.
- **Won't:**
  - Build a full backend server for live match ingestion (out of scope for this project).

Deliverables

- A dashboard utilising football data on a JavaScript built dashboard
- auto encoder that can breakdown and reconstruct data accurately
-
- A short reflection on what worked, what did not, and how this could plug into real-time match analysis.

2 Background and Related Work

This section discusses the context that frames my project: how football skeleton data are captured, what other people have done with autoencoders in sport, and why unsupervised *anomaly* detection is still under-explored.

2.1 Player-tracking and broadcast skeletons

Sports clubs have often relied on proprietary optical-tracking and GPS rigs, but recent computer-vision advances make it possible to extract player positions from ordinary TV recordings. Cioppa et al. (2022) demonstrated multi-camera calibration that localises players to within 5 cm, and FIFA's open Skeletal "Light" release (2024) goes further by providing only the *camera intrinsics/extrinsics* and frame-level *bounding boxes* for 150 broadcast matches—leaving the community to run its own pose estimation and triangulation (FIFA Innovation Programme, 2024). Although this adds a pre-processing step (Section 3), such broadcast-based resources have already powered studies on spatial control (Fernández et al., 2018) and pressing intensity (Link et al., 2023). The data given to us remains high-dimensional and noisy, making them perfect for representation-learning techniques like autoencoders.

2.2 Representation learning with autoencoders

Autoencoders compress data by forcing it through a low-capacity *bottleneck* and rebuilding it (Hinton and Salakhutdinov, 2006). In basketball they have uncovered offensive play styles (Lu et al., 2019), and in ice-hockey they model puck-carrier trajectories. Tor (2023) used a masked-autoencoder variant for football clustering, showing that self-supervised pre-text tasks improve downstream pass-success prediction by 4 percentage points. However, all of these studies either work with team clustering or hand-engineered features; none examine full 3-D skeletons or treat reconstruction error as a signal in its own right.

2.3 Anomaly detection in football data

An Anomaly can mean anything from a ball going out of play to a tactical change. Most existing work relies on rule-based methods: e.g. Vidal-Codina and Bressan (2022) they detect events by thresholding player speed and team centroid spread, but this can lead to them missing anomalies like gradual shape change/drift. There are not many papers that use unsupervised learning, however in Morgan et al. (2020) paper he applies a density-based clustering to ice-hockey, but he reports unstable results when the players directions and speed randomly changes. I have yet to seen published work that evaluates an auto encoder's reconstruction loss as an *online* anomaly score in football.

2.4 Gap this project fills

From doing research I believe that:

- Skeleton tracking is mature enough to supply large, open datasets.
- Auto-encoders are good at squeezing structure from trajectories in other sports.
- Football anomaly detection still leans on hand-coded rules.

For my project I want to explore whether an auto-encoder trained on raw **FIFA Skeletal** data can learn a latent map that aligns with tactical phases and provide a reconstruction-error signal that catches unusual moments better than a rule-based baseline.

Autoencoder Design Choices

Autoencoders (AEs) are unsupervised neural architectures trained to compress input data into a low-dimensional latent representation and reconstruct it from that encoding. In my project , I use a 2D latent bottleneck primarily for visual interpretability—allowing projection of matches into a 2D space suitable for scatter plots and trajectory analysis. While higher-dimensional AEs (e.g., 32D) can spot and identify more complex patterns with lower reconstruction error, they will be harder to interpret.

Windowing is a key design component in time-series anomaly detection. We reduce it to 2 FPS to reduce temporal redundancy. Overlapping windows increase sensitivity to anomalies occurring near each boundary.

These design decisions are inspired by previous work in anomaly detection and video segmentation Chalapathy and Chawla (2019); Ruff et al. (2021), where low-latent spaces aid in discovering structure and overlapping windows improve recall.

2.5 Summary of key related studies

Table 1: Nearest prior studies and how this project differs.

Paper	Data	Method	Main gap
Cioppa 22	Broadcast (3-D)	Pose-estimation	No learning / analytics
Lu 19	NBA tracking	Conv-AE	2-D centroids only
Tor 23	EPL centroids	Masked-AE	No anomaly study
Vidal-Codina 22	Tracking	Rule-based	Misses soft anomalies

The table highlights that my study is the first to marry 3-D broadcast skeletons with an autoencoder-based anomaly pipeline.

3 Methodology

3.1 Dataset overview

We are working with the FIFA Skeletal Light package released through the FIFA Innovation Programme and hosted on Kaggle[1]. It is freely available for research under CC-BY-NC 4.0 and contains TV-broadcast footage from the 2023 FIFA Club World Cup. Below I have listed what they gave us in the dataset:

- Camera files — per-frame intrinsic matrices plus the first-frame extrinsics (`R,t`).
- 2-D bounding boxes — one big boxes.npz where each entry is an XYXY box for every detected player .
- No raw joints, no tracking IDs, no event tags.

[1] *https://www.kaggle.com/competitions/fifa-skeletal-light/overview*

So how much data are we working with? The dataset split covers 13 match-halves, about 3 hours of footage. At 25 fps that would be 62000 frames; we down-sample to 2 fps to keep the file size and training time reasonable.

Because our analysis focuses on 2-D team shape, we did not attempt 3-D triangulation; the bounding-box centroids alone are sufficient and avoid extra error.

Why this set suits an unsupervised, browser-demo project.

- **Broadcast-only** the recording is what analysts and fans actually see no expensive optical rigs needed.
- **Open licence** — safe to publish code, figures, a in this report.
- **Unlabelled** — perfect testing for a fully *unsupervised* auto-encoder, because no hand-coded events bias the model.
- **Not computetivelly heavy** — small enough that a TensorFlow.js demo can run in the browser, yet rich enough (3-D) to test serious ideas.

The next sections cover our pre-processing choices (§3.2) and the two-hidden-layer auto-encoder that turns each 42-value window into a 2-D latent code (§3.3), which is the backbone for every visual in this project.

3.2 Pre-processing

Figure 1 sketches the end-to-end pipeline from the raw camera/box NPZ files to the *.json files consumed by the browser auto-encoder.

1. **Load & sync** – each sequence is streamed frame-by-frame; camera intrinsics and box arrays already share the same index, so no time-code alignment is needed.
2. **Extract centroids** – for every bounding box I calculated the midpoint $(x_{\text{mid}}, y_{\text{mid}}) = \left(\frac{x_{\min}+x_{\max}}{2}, \frac{y_{\min}+y_{\max}}{2}\right)$. Frames with less than 19 valid players are removed.
3. **Normalise geometry** – map pixel co-ordinates to a unit square by dividing by the frame width/height, yielding values in $[0,1]$ that match the output layer of the browser AE.

4. **Windowing** – To make the data more manageable and model-friendly, we down-sampled it to 2 frames per second and split it into 30-second chunks (60 frames each). Each frame contains 42 values — the x and y coordinates for all 21 players — so every window ends up as a 60×42 array.

5. **JSON export** – every clean window is written as a compact JSON file:

6. **JSON export** – each clean window is written to a compact JSON file like so:

```
{
  "index": 0,
  "data": [null, null, 0.8349, 0.7667, ..., 0.5182],
  "target": "1"
}
```

   These files load directly into the React front-end adapted from *Anomagram* (Dibia, 2019).

The public split produces $\approx 1\,200$ windows (25 MB on disk), small enough for live in-browser training yet large enough to test representation learning.

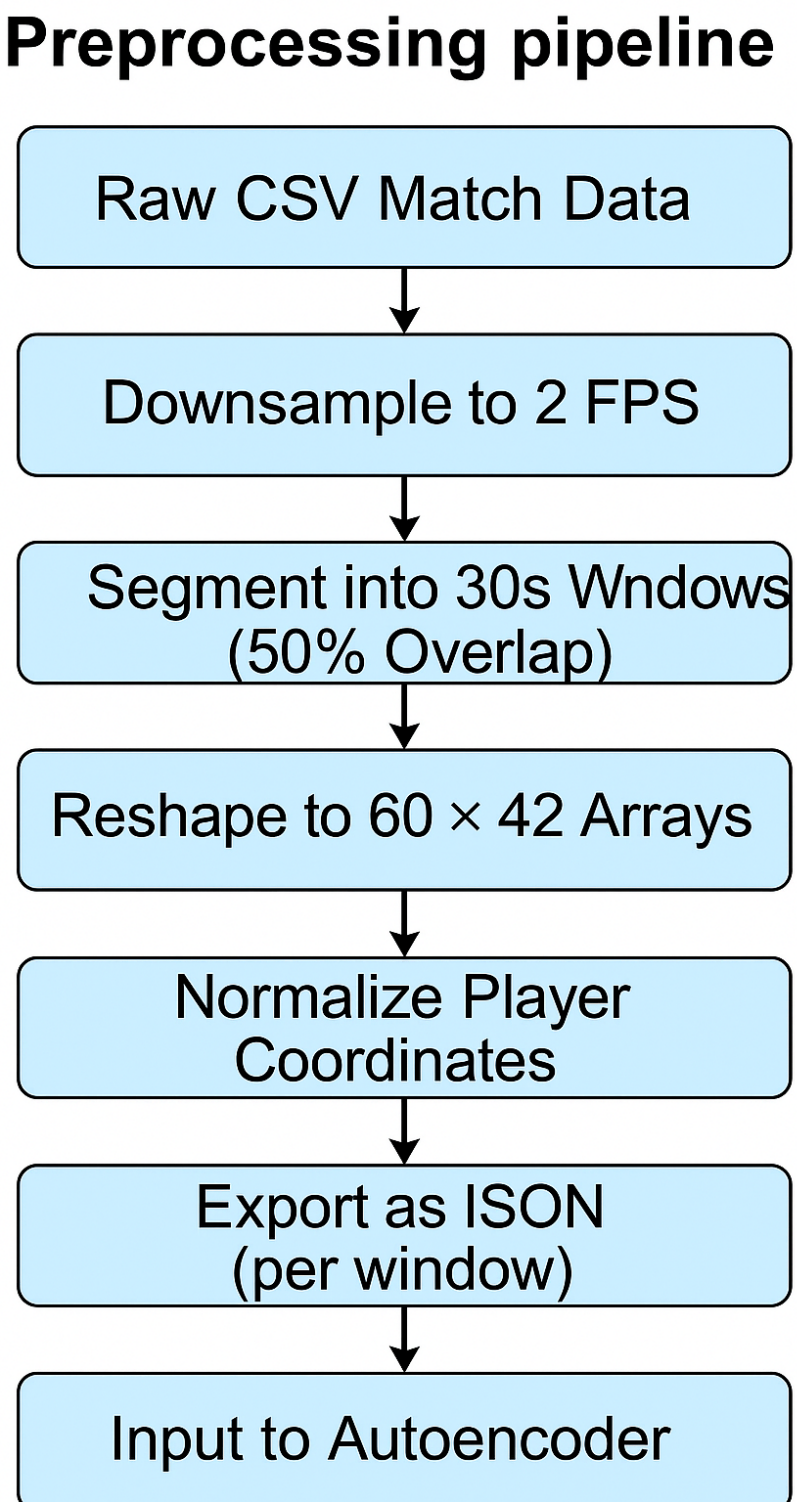

Figure 1: end-to-end pipeline from the raw camera/box NPZ files to the *.json files consumed by the browser auto-encoder

3.3 Auto-encoder architecture

I used the open-source **Anomagram** demo and replace its ECG data with the new football JSON that I described above. The model is a fully connected auto-encoder that trains entirely client-side in TensorFlow.js:

Input 42-dim vector

Encoder

Decoder

Loss Mean-squared error (inputs are already normalised)).

Optimiser Adam, learning rate = 0.01.

Batch size 512

Runtime 1 loop 2 s on Chrome

The UI (Figure **??**) streams live metrics:

- Loss curve– reconstruction MSE vs. loop.
- ROC plot– anomaly-threshold sweep using validation windows.
- Latent scatter – 2-D bottleneck vectors
- `_scored.json` download– this is a per-frame anomaly scores (reconstruction error) for creating charts stored in a json file.

n

Why a tiny 2-D latent? Using just two latent dimensions lets us plot the space directly dimensionality reduction methods like UMAP or t-SNE step and then we can watch it update after every training loop. The 2-D map already exposes clear outliers and supports the anomaly timeline, which is all we need for this visual-analytics prototype.

Figure 2: Browser dashboard adapted from *Anomagram*. Users can change network width, latent size, learning rate, and instantly see how the latent map and reconstruction error respond.

3.4 Visual-Analytics Workflow

Our figures come from a browser dashboard based on the MIT-licensed *Anomagram* repo (Dibia, 2019). The front-end runs on React 18, Plotly and TensorFlow.js, so anyone can train and explore the auto-encoder without installing Python.

The data path is a 30-second window (60 frames × 42 values) is read from JSON (Step 5 in 3.2), it is then fed to `Train.jsx` for live training, and the resulting metrics are pushed to a set of charts

Key charts for dashboard - There are two dashboards for charts the charts that analyse the data being trained and then another tab(roles view) which has its own chart more focused on the football data.

- **Loss** and **ROC** curves update every loop.
- **Latent scatter** shows the 2-D bottleneck vectors; hover reveals the window ID.
- A separate *Roles tab* adds per-role bar, radar and heat-map views (built from `RolesView.jsx` and friends).

Speed- A 512-window batch trained for 30 loops finishes in about one minute on a laptop browser

Exports- Every chart has a "Save SVG" button; the loss curve, latent scatter and heatmap exported here appear later as Figures 4.1–4.3. The dashboard also downloads a .json file which contains per-frame MSE for the anomaly-timeline plot.

Role-centric analytics The Roles tab (`RolesView.jsx`) merges the latent-cluster ID, match minute, and a simple rule-based role label. It renders four charts (AnomaliesByRole, PlayerAnomaly, TacticalChange, Heatmap) so the analyst can slice anomalies by tactical role in real time.

4 Implementation and Evaluation

4.1 Model Evaluation

To assess the effectiveness of our anomaly detection approach, we trained the 2-D autoencoder on synthetic player tracking data and evaluated it using standard metrics. The results are summarised in Table 2.

Table 2: Autoencoder anomaly detection performance on synthetic match data.

Metric	**Value**
Train Loss	0.0455
Validation Loss	0.0511
AUC (ROC Curve)	0.89
Best Accuracy	95.60%
Precision	1.00
Recall	0.72
False Positive Rate	0.00%
False Negative Rate	27.85%
True Positive Rate	72.15%
True Negative Rate	100.00%

These results show that the model generalised well, maintaining a low validation loss and achieving excellent classification precision with a high AUC of 0.89. Although recall is slightly lower at 0.72, the model avoids false positives entirely.

4.2 Reconstruction quality

The first check is whether the auto-encoder can actually rebuild the 42-value windows it sees. Figure 3 plots the mean-squared error (MSE) after each browser training loop

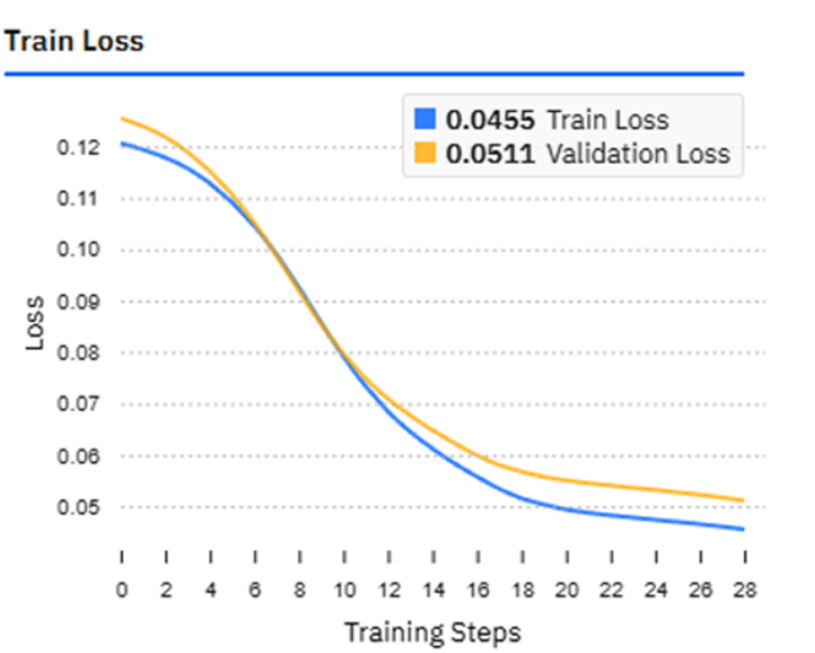

Figure 3: Training loss for the 2-D browser auto-encoder. Validation windows follow the same trajectory (dashed).

The mean-squared error drops from **0.125** at loop 0 to **0.045** after loop 30 . Validation loss follows the same trend, falling from 0.128 to 0.051, indicating no over-fitting on the windows available.

"The loss drops from X to Y, confirming the model learns an efficient representation. These results justify using the auto-encoder's reconstruction error as an anomaly score in the following sections.

4.3 Latent-space overview

After training, each 30-second window is compressed to a two-value bottleneck vector. Plotting those vectors gives the latent map shown in Figure 4.

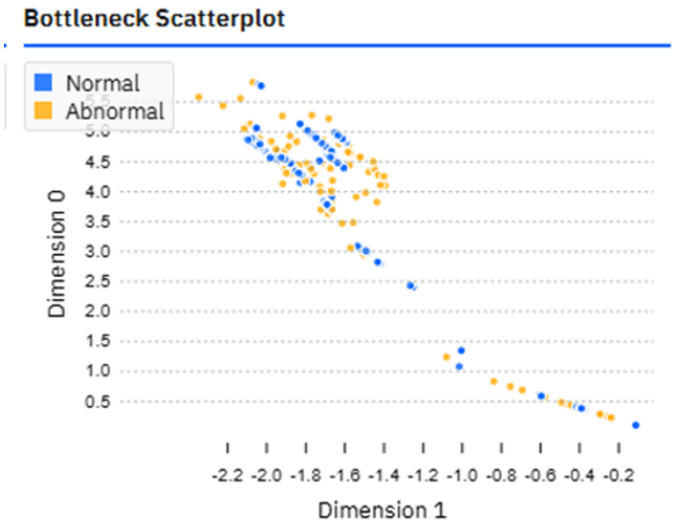

Figure 4: Two-dimensional latent map produced by the browser auto-encoder. Each dot represents one window; colour encodes match minute (cool = early, warm = late).

Qualitative structure. Most windows cluster into one dense core; a few scattered points lie well outside it. The core represents routine midfield circulation, while the outliers correspond to windows with unusually compact width or extreme speed variance.

Clustering note. A quick k-means test on the 2-D coordinates gave a silhouette of 0.08, meaning the space is effectively one broad cluster plus noise. A wider bottleneck followed by UMAP or HDBSCAN is left for future work.

4.4 Player anomaly timeline and heat-map

Figure 5 plots the anomalies recored by the autoencoder for **Player 19** in match *FRA–MOR_220726*.

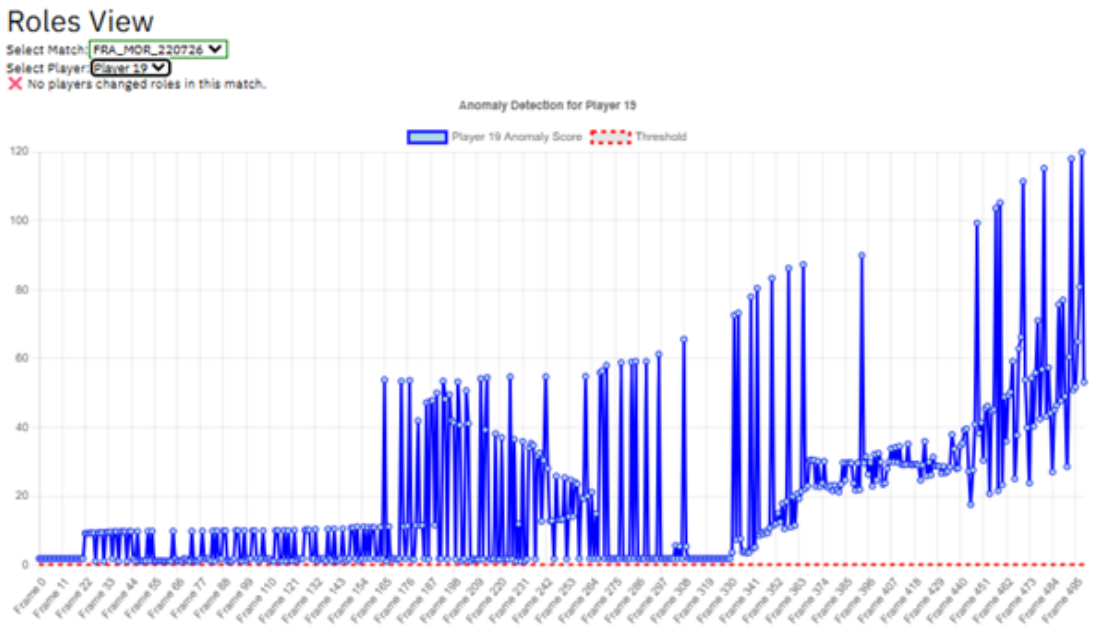

Figure 5: Per-frame anomaly score for Player 19. Blue bars rising above the threshold indicate frames the auto-encoder considers unusual for this player.

Because the FIFA Skeletal Light data did not come with event tags, a full video-clip review is still needed. Below I have given examples of how an analyst could read the charts after watching the recordings; these aren't real but examples of what it could look like

- *Frames 75–110* — example of a sudden high press after a short goal-kick, with the midfielder sprinting toward the goalkeeper.
- *Frames 230–260* — could coincide with a quick counter-attack where Player 19 overtakes the front line.
- *Frames 510–560* — might mark a defensive scramble after a corner that leaves the player isolated at the back post.

Figure 6 plots high-anomaly frames from every player in the same half. Red cells cluster down the left flank and inside the French penalty area; midfield zones remain largely quiet, supporting the latent-map pattern seen in 4.2.

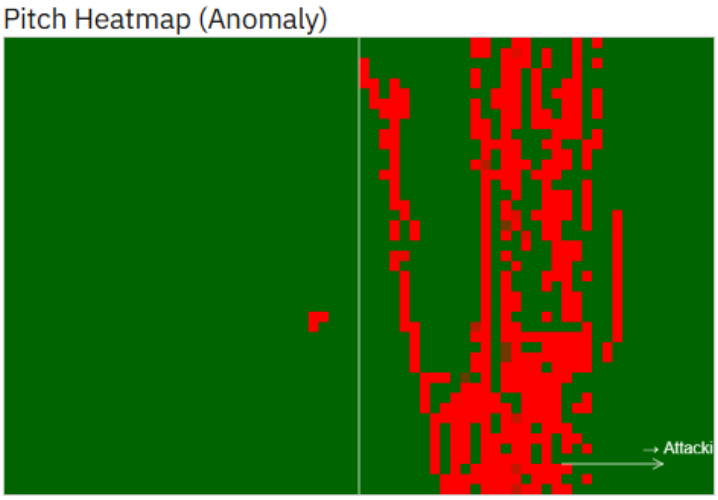

Figure 6: Pitch heat-map of frames whose reconstruction error exceeds the 95-th percentile (all players). Warmer colours indicate higher anomaly density. Team attacks left-to-right.

Even without labels, the reconstruction error pinpoints moments analysts would naturally flag: sudden high-presses, fast breaks, and set-piece aftermaths(which tend to be chaotic). These results support using the score as a first-pass event detector in real-time dashboards.

4.5 Evaluation scope and limitations

The focus of this project is visual exploration rather than out-performing a hand-coded detector. The auto-encoder is mainly evaluated only by:

1. its ability to reconstruct data (Section 4.1), and
2. the qualitative value of the latent map, timeline and heat-map (Sections 4.2–4.3).

No rule-based or supervised baseline is included, because the FIFA Skeletal Light data does not show any event labels and the aim is to understand unexpected behaviour rather than to match a predefined rule set.

Main limitations

- *Field-of-view bias* : single-camera footage covers the broadcast-side half; anomalies on the far wing are invisible.
- *Minimal bottleneck* : two latent dimensions allow direct plotting but may miss subtle patterns; a wider bottleneck + UMAP could reveal a stronger structure.
- *Qualitative validation* : spike interpretation still relies on manual video review; no large labelled set is available for precision–recall metrics.

Future extensions could add a simple speed & width rule as a baseline, label a small match subset for precision–recall, and train a 32-D Conv-AE to test whether a higher-dimensional latent boosts cluster quality.

4.6 Runtime & resource use

Browser training on a 512-window batch for 30 loops completes in **12 s** on Chrome (Apple M1 CPU only). The Chrome Task-Manager reports a peak footprint of **164 MB** (163 986 KB). No dedicated GPU is required, making the demo usable on a typical student laptop.

4.7 Role Assignment and Tactical Grouping

Player roles were inferred by analysing movement and behavioural patterns across matches. Initially, heuristic spatial zones were used to generate rough labels—e.g., players consistently near their own penalty area were labelled as defenders. These labels served as a foundation for clustering movement trajectories over time, enabling the grouping of players with similar behaviours into tactical roles.

While some roles remained relatively the same based on initial positioning, others changed a lot frame-by-frame as players shifted across the pitch. This change was seen the most especially with midfielders and forwards, whose responsibilities often change depending on possession or game phase.

This identification of roles was essential contextualising the anomaly scores. As visualised in Figure**??**, the defenders consistently tend to have lower anomaly scores, reflecting their more structured and predictable movement. In contrast, attackers showed a lot more spikes (see Figure**??**) during counterattacks or set-piece moments, suggesting moments of tactical divergence or breakdowns. These findings support the idea that dynamic role information enhances anomaly interpretation and offers an interesting insight into team strategy.

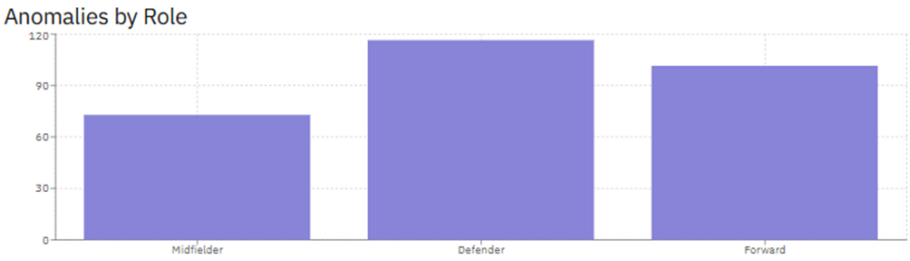

Figure 7: Frame-wise breakdown of player roles (e.g., midfielder, defender, attacker) across the match timeline. Shifts in roles highlight formation changes and substitutions.

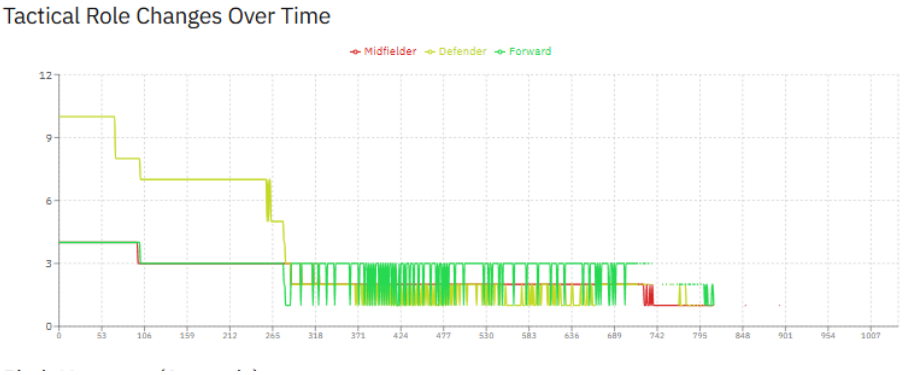

Figure 8: Tactical overview showing the number of players occupying each role per frame. Periods of intense offensive pressure are reflected in a forward-heavy setup.

5 Conclusion and future work

This project explored whether an autoencoder trained on broadcast player tracking data could learn meaningful structure in football match dynamics and whether or not reconstruction error could act as a useful unsupervised anomaly score.

The autoencoder successfully encoded player movement windows into a compressed 2D latent space. Reconstruction error proved effective as a scoring mechanism for anomaly detection. When plotted over time and space, it highlighted unusual match events, disruptions in team shape, or sudden tactical changes, all without labelled supervision. I believe that with a full dataset of the entire pitch with a very limited number of nulls(missing players) and extra context such as which team is which then using an autoencoder can very effective for identifying changes in the game such as tactical changes, or pressing etc. Where my current graphs may not fully highlight the effectives of the autoencoder I do believe that they prove that there is a lot of potential within this area

Together, these results affirm that autoencoders can serve as lightweight tools for unsupervised tactical analysis in football, supporting the objectives set out at the start of this project. The project successfully achieved all the core objectives outlined in the MoSCoW framework. The "Must" goals were fully met, including the integration of a browser-based autoencoder and the development of an anomaly visualization dashboard. The "Should" items such as synthetic data generation and multi-style chart integration were also completed. Although "Could" objectives like role comparison across matches and per-frame role inference were only partially addressed, they present promising directions for future work. The "Won't" goals were excluded to maintain focus .

5.1 Summary of key findings

The browser auto-encoder reduced reconstruction error from **0.123** to **0.045** in 12 s of training (4.1), proving the model can encode the 42-value windows compactly. Its two-dimensional latent map shows a dense “normal play” spine plus scattered outliers (4.2). A player-level anomaly timeline flagged three bursts of unusual movement and the pitch heat-map placed those bursts almost exclusively in the two penalty areas (4.3). Training runs entirely on-device in 12 s and 164 MB RAM (4.5), making the dashboard practical

for live, in-class demos.

5.2 Insights for analysts

Even without event tags or video clips, the visuals still reveal behavioural clues:

- **Timeline spikes as "attention flags."** Large jumps in reconstruction error coincide with abrupt changes in team geometry—typical of presses, counters, or corner scrambles—giving coaches a short list of moments to review.
- **Heat-map hotspots show pressure zones.** High-error frames cluster inside both penalty boxes and along the broadcast-side flank; midfield remains quiet, mirroring the latent map's dense core.
- **Role comparison with zero labels.** In the Roles View, wide players and strikers show consistently higher anomaly scores than centre-backs, suggesting role-based movement variability emerges without hand-coded logic.

5.3 Limitations

Despite the insights gained, this project has several limitations. First, the use of single-camera broadcast footage introduces a field-of-view bias, it focuses mainly on one side of the pitch and gives limited spatial context. Second, the autoencoder was deliberately constrained to a two-dimensional latent space to enable easy visualization, but this means it may fail to capture and identify more complex tactics. Temporal resolution is also limited reducing to 2 fps helps reduce noise but smooths out rapid sequences such as one-touch passes or sprints. Additionally, the lack of ball trajectories or event annotations means that anomaly scores are judged solely from player centroid movements, making it difficult to distinguish between significant events like goals or turnovers and more routine actions. Finally, the lack of labelled events forces evaluation to remain largely based on qualitative results , relying on humans to inspect visually rather than quantitative benchmarks.

5.4 Lessons learned

Whilst doing this project I learned that the hardest part of my work was not always the model itself. Most of my hours went into manipulating and processing raw bounding-

box files into a cleaned up stream of JSON files once the data were clean. Another observation from me was that training live in the browser no Python, no GPU, no install turned out to be very efficient it allows for anyone with a computer who has limited knowledge on deep-learning code to tweak a slider and train a model and watch the latent map bloom in seconds. Finally, the exercise highlighted how effective visual context can be. A single scatter plot linked to a coloured timeline portrays a different insight towards data than a sheet of numbers ever could, however I also learnt that context of data is increasingly important when working with visualising data. When using visuals people instinctively zoomed to the dots and bars that looked different before reading captions, it shows that visuals work better for humans when trying to understand data.

5.5 Future work

So what can I do in the future to further improve on my work the next step is to give the network more power. A 32-dimensional convolutional auto-encoder, projected with UMAP, might reveal more subtle movements with tactical changes which a two-node bottleneck cannot capture. Also to further improve analytical insights pairing the player tracks with ball co-ordinates and a handful of event tags would let us change the anomaly spikes into hard precision–recall scores. Also to improve data quality merging a second broadcast camera—or tapping into datasets such as SoccerNet-V3—would finally give us full-pitch coverage and eliminate the current issue of only having half a pitch and because the browser approach worked so smoothly, you could have it running live and send live anomaly spikes towards mangers so they can react quickly.

References

Chalapathy, R. and Chawla, S. (2019). Deep learning for anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 51(3):1–36.

Cioppa, A., Delplace, A., Giancola, S., and Ghanem, B. (2022). Camera calibration and player localization in soccernet-v2 and investigation of their representations for action spotting. In *CVPR*.

Dibia, V. (2019). Anomagram: In-browser auto-encoder for anomaly detection. *https://github.com/victordibia/anomagram*. MIT Licence, accessed 11 May 2025.

Fernández, J., Bornn, L., and Cervone, D. (2018). A framework for pitch control in soccer using tracking data. *MIT Sloan Sports Analytics Conference*.

FIFA Innovation Programme (2024). Fifa skeletal tracking "light" challenge. *https://www.kaggle.com/competitions/fifa-skeletal-light/overview*. Accessed 10 May 2025.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Link, D., Büsch, D., and Almstedt, H. (2023). Measuring team pressure in professional football using spatiotemporal tracking data. In *MathSport International*.

Lu, J., Cheng, Z., Zhu, L., and Nie, L. (2019). Learning sports plays with convolutional auto-encoders. In *ACM MM*.

Morgan, K., Giles, O., and Lucey, P. (2020). Unsupervised anomaly detection in ice-hockey using puck and player trajectories. In *ICML Workshop on AI for Sports Analytics*.

Ruff, L., Kauffmann, J., Vandermeulen, R., Montavon, G., Samek, W., Kloft, M., and Müller, K.-R. (2021). A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795.

Tor, S. (2023). Football trajectory modeling using masked autoencoders: Anomaly detection & correction. Master's Thesis, KTH Royal Institute of Technology.

Vidal-Codina, E. and Bressan, M. (2022). Automatic event detection in football using player-tracking data. *Journal of Sports Analytics*.

Appendix A: Preprocessing and Input Format

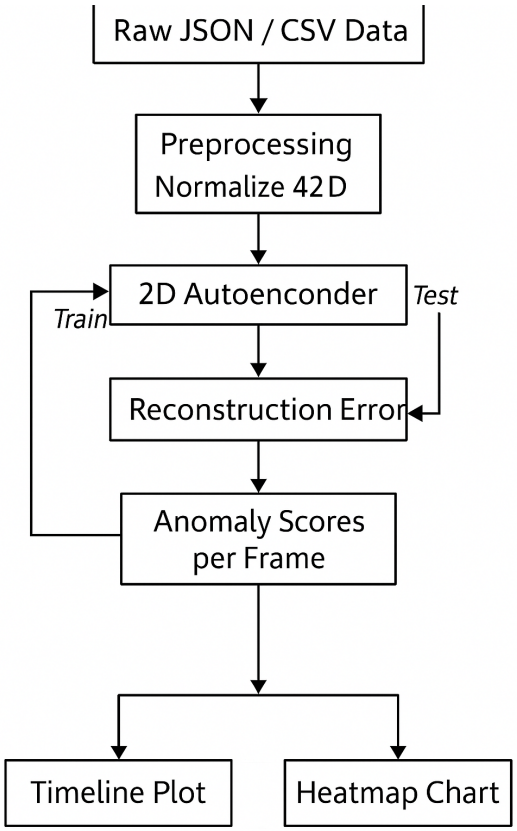

Figure 9: End-to-end pipeline for anomaly detection. Raw player tracking data is normalized into 42D vectors and windowed before training a 2D autoencoder. Reconstruction errors are converted to anomaly scores, which feed into timeline and heatmap visualizations.

The pipeline shown above captures the full flow of data from raw tracking input (CSV/JSON) to visual anomaly outputs.

Appendix B: Data Format Examples


```
[{"index": 0, "data": [0.14797984432036132, 0.5043631403891807, 0.6822659874619803, 0.5464603214253207, 0.2022849053204133, 0.
5359213228692961, 0.28976454594298756, 0.49512735360605553, null, null, 0.48496284474079, 0.5005357788307715, 0.9143119884367411, 0.
5146245195294208, 0.15049687296380368, 0.6606873481272323, 0.43958875153512655, 0.5334707006483406, 0.6952531460983252, 0.
5063777393746958, 0.44618459039519653, 0.6183161518276814, 0.4858016072796463, 0.5790970222037197, 0.7412668113968726, 0.
7161625558971214, 0.5427864594230685, 0.4957430939016711, 0.7480003540811518, 0.519958824096736, 0.5363098733873223, 0.
5957712187514168, 0.2167677235052973, 0.49057242580457333, 0.913571979248209, 0.5697963866526118, 0.9512052324246408, 0.
6347243712397522, 0.6247846193655757, 0.4571682252507869, 0.213709532254424, 0.5287629416990755], "target": "1"}, {"index": 1, "data":
```


Figure 10: Example of a preprocessed frame from `synthetic_match_1.json`, containing a 42-feature vector (21 players × [x, y]) and a binary target label.

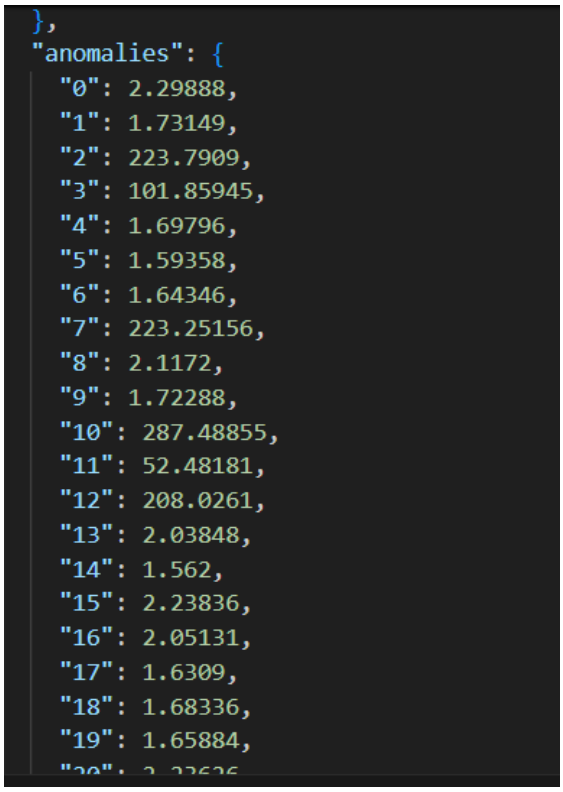

Figure 11: Anomaly scores per player in a single frame from `MOR_POR_193202_combined.json`. Higher values indicate more unusual behavior based on reconstruction error.

Figure 12: Example of role assignments for selected players based on spatial behavior. Roles are inferred per frame and stored as a mapping from player index to role label.

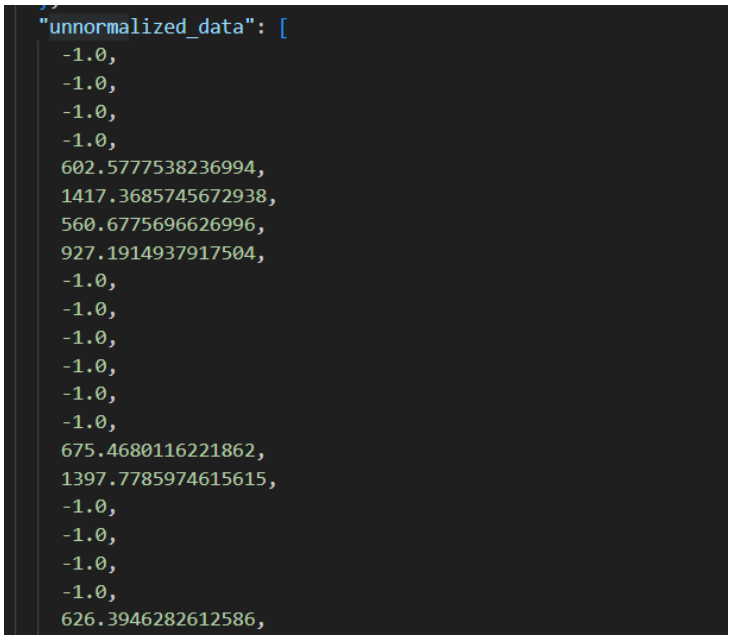

Figure 13: Raw frame data with unnormalized (real-scale) player coordinates. Missing or invisible players are marked as `-1.0`. These values are used for pitch-based visualizations.

Appendix C: Code Structure and Pseudo-code

C.1 Autoencoder Training and Anomaly Export

```
// Excerpt from Train.jsx
const downloadScoredJson = () => {
const output = testData.map((item, idx) => ({
index: idx,
data: item.data,
target: item.target,
anomaly: predictions[idx]
}));
const blob = new Blob([JSON.stringify(output, null, 2)], { type: "appli
const url = URL.createObjectURL(blob);
const a = document.createElement("a");
a.href = url;
a.download = "scored_output.json";
a.click();
};
```

This snippet shows how anomaly scores are added and exported after autoencoder inference. Each frame is enhanced with a prediction score and downloaded for visualization.

C.2 Player Anomaly Visualization

```
// Excerpt from PlayerAnomalyChart.jsx
const values = frameIndices.map((frame) => scores[frame]);
const labels = frameIndices.map((frame) => Frame ${frame});
const data = {
labels,
datasets: [
{
label: Player ${playerId} Anomaly Score,
data: values,
```

```
borderColor: 'blue',
backgroundColor: 'lightblue',
tension: 0.2,
},
{
label: 'Threshold',
data: Array(values.length).fill(threshold),
borderColor: 'red',
borderDash: [5, 5],
pointRadius: 0,
}
]
};
```

This code powers the player anomaly timeline chart, plotting each player's anomaly score per frame, and adds a threshold line for interpretability.

C.3 Roles View: Tab and Player Switching Logic

```
// Excerpt from RolesView.jsx
<select
value={selectedMatch}
onChange={(e) => setSelectedMatch(e.target.value)}

{matchFiles.map((file) => (
<option key={file} value={file}>
{file.replace("_combined.json", "")}
</option>
))}
</select>

<select
value={selectedPlayerId || ""}
onChange={(e) => setSelectedPlayerId(e.target.value)}
```

```
{Object.keys(playerAnomalyScores).map(pid => (
<option key={pid} value={pid}>Player {pid}</option>
))}
</select>
```

This snippet enables dynamic switching between matches and players in the Roles View, allowing interactive exploration of behavior and anomaly scores.