Predictive Safety and Dynamic Risk Estimation in Human-Robot Collaboration

Francesco Sicilia s354909, Alessio Sorrentino s353528, Alessandro Tommasi s353532
Politecnico di Torino

Abstract. Ensuring operator safety in industrial Human-Robot Collaboration (HRC) requires not only the real-time detection of hazardous proximity but also the proactive anticipation of collision risks. This paper presents a two-phase system for dynamic risk assessment using a modified version of the CHICO dataset, which includes 20 subjects performing realistic industrial actions with annotated collision scenarios. The first phase, a Multilayer Perceptron (MLP) classifies instantaneous risk classification (safe, near-collision, collision) based on single-frame joint distances. In the second phase, by encoding motion history from 10 past frames, we employ an LSTM-based encoder-decoder to predicts the evolution of risk over a 1-second future horizon (25 frames). The system incorporates root-relative normalization for spatial invariance and adheres to the dataset split protocol from arXiv:2208.07308. This work advances HRC safety by providing a real-time capable framework, validated on diverse industrial actions and offers a foundation for future extensions in multi-modal risk prediction, allowing smoother preventive interventions.

1 Introduction

Human–robot collaboration (HRC) in industrial environments increases productivity and flexibility, but it also introduces safety-critical scenarios where humans and robots share the same workspace. In these settings, risk estimation must be both accurate and timely, because safety interventions (e.g., slowing down, stopping, re-planning) require advance warning rather than purely reactive detection. Using 3D joint positions instead of processing a series of RGB frames is particularly attractive for real-time HRC because it is lightweight, privacy-preserving compared to RGB, and provides a compact representation of human motion that can be processed at high frequency.

1.1 From risk detection to risk prediction

A key limitation of purely single-frame risk assessment is that it cannot explicitly encode motion dynamics. Even if a model correctly identifies "near-collision" or "collision" at the current instant, it may still be insufficient in practice: a safe and responsive HRC system should estimate how risk will evolve in the next moments, enabling proactive mitigation. This motivates moving from static classification to temporal forecasting.

According to Human–Robot Collaboration, dynamic risk is often characterized using Time-to-Collision (TTC), which estimates the remaining time before impact under the assumption of constant relative velocity. While TTC provides an intuitive temporal interpretation of risk, we deliberately did not include it explicitly in our risk formulation. First, TTC is highly sensitive to noise in velocity estimation, which is critical when using 3D skeletal data. Second, it relies on simplified kinematic assumptions (linear motion and constant speed) that rarely hold in realistic HRC scenarios, where human motion is inherently non-linear and discontinuous. Finally, TTC becomes unstable or undefined when the relative velocity approaches zero. For these reasons, we adopt a distance-based instantaneous risk definition, and rely on the LSTM to implicitly learn motion dynamics and temporal risk evolution, making an explicit TTC computation redundant.

2 Data preparation

2.1 CHICO Dataset

The CHICO dataset (Collaborative Human–Industrial Cooperation) provides motion data captured in industrial human–robot collaboration scenarios. The raw dataset is organized as a collection of temporal sequences of actions.
Each sequence is stored as a serialized file (*.pkl*) and consists of an ordered list of frames. At each frame, the data includes:

- Human skeleton joints: a set of 3D joint coordinates representing the human body posture. In the data provided, the human skeleton is represented by 15 joints, each described by Cartesian coordinates *(x, y, z)*.
- Robot joints: a set of 3D joint coordinates describing the robot configuration at the same time instant. The robot is represented by 9 joints, also encoded as *(x, y, z)* coordinates.

Formally, each frame in the original dataset has the structure:

$$frame_t = [H_t, R_t]$$

where $H_t \in \mathbb{R}^{9\times3}$ denotes the human joints and $R_t \in \mathbb{R}^{15\times3}$ denotes the robot joints at time t.

For the purposes of model development and evaluation, the dataset is partitioned into training, validation, and test sets using a subject-independent split.

2.2 Preprocessing

The original CHICO dataset does not provide explicit frame-level risk labels. Therefore, in this work, risk annotations are automatically derived from the raw 3D skeleton data using a distance-based geometric criterion.

For each frame, the minimum Euclidean distance between any human joint and any robot joint is computed. Given the human joint set H_t and the robot joint set H_t,the pairwise distances are defined as:

$$d_{(i,j)}(t) = ||\, H_{t,i} - R_{t,j}\, ||_2$$

where $i = 1, \dots, 15$ indexes human joints and $j = 1, \dots, 9$ indexes robot joints.

The minimum distance at frame t is then:

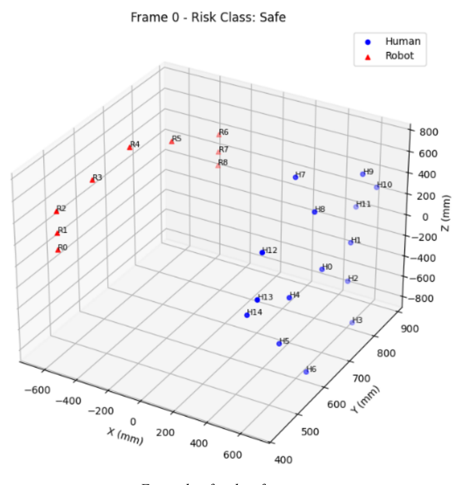
$$d_{min}(t) = min_{i,j}\, d_{(i,j)}(t)$$

This minimum distance is used as a proxy for collision risk.

Based on $d_{\min}(t)$, each frame is assigned to one of three discrete risk classes:

- Safe (0):

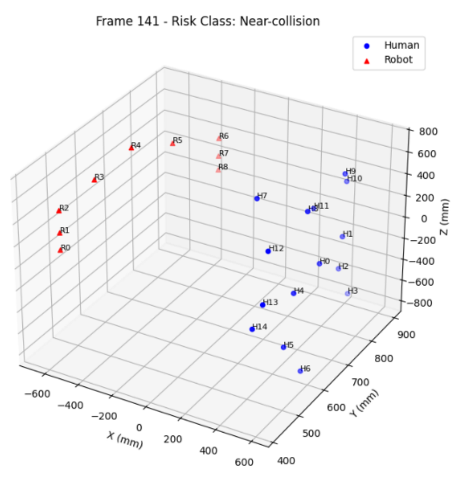
$$d_{\min}(t) > 630 \text{ mm}$$



Example of risk safe

- Near-Collision (1):

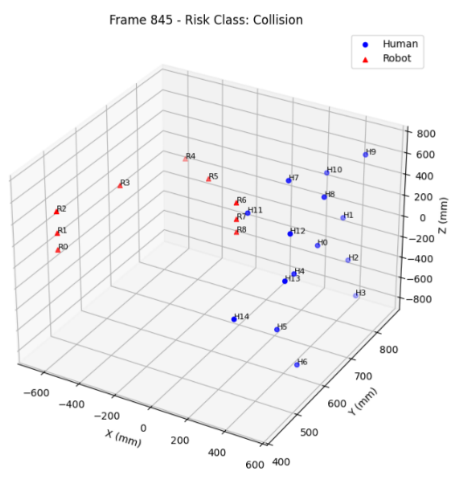
$$130 \text{ mm} < d_{\min}(t) \le 630 \text{ mm}$$



Example of risk near-collision

- Collision (2):

$$d_{\min}(t) \leq 130 \text{ mm}$$



Example of risk collision

The labeling process is applied frame by frame to all sequences in the dataset. For each original frame $[H_t,\ R_t]$, a new frame structure is created:

$$frame_t^{risk} = [H_t, R_t, y_t]$$

where $y_t \in \{0,1,2\}$ is the automatically assigned risk label.

The geometric thresholds used to derive frame-level risk labels are not arbitrary, but follow the CHICO annotation protocol and established practices in industrial HRC. In particular, a minimum human–robot joint distance of 130 mm is used to define a Collision, while distances up to 630 mm correspond to Near-Collision states. These values reflect typical safety margins adopted in collaborative robotics and are consistent with prior work on pose forecasting for HRC

2.3 Handling Data Imbalance

The collision risk prediction task considered in this work is inherently affected by severe class imbalance. Frames labeled as *Safe* and *Near-Collision* largely dominate the dataset, while *Collision* events are very rare. This imbalance poses a critical challenge in safety-oriented learning, as standard training procedures tend to bias the model toward the majority class, increasing the risk of missed collision detections, which are particularly costly in human–robot collaboration.

To address this issue, a safety-driven data balancing strategy has been adopted. The goal is not merely to equalize class frequencies, but to ensure that collision-related patterns are sufficiently represented during training, both in terms of quantity and variability.

2.3.1 Data Augmentation for Single-Frame Classification

The data augmentation strategy focuses on generating geometric variations that simulate sensor noise and subject variability without altering the semantic risk label. For the minority class (Collision), we apply four types of transformations to the normalized joint coordinates:

- Gaussian Noise (*σ=0.02*) per joint
- Random Scaling $\in [0.95, 1.05]$ to simulate different subjects heights
- Vertical Axis Rotation $(\theta \sim \mathcal{U}[-15°, 15°])$ to improve robustness against different approach angles
- Joint Dropout, which randomly masks specific joints to simulate occlusion artifacts typical of depth cameras

These transformations are applied independently to each frame, expanding the collision dataset by a factor of 50 to prevent overfitting on static poses.

2.3.2 Weighted Sampling Strategy for Single-Frame Classification

We implement a Weighted Random Sampler that rebalances the class distribution at the batch level. Each sample is assigned a weight inversely proportional to its class frequency:

$$w_c = 1/N_c$$

where N_c is the total count of samples in class *c*.
During training, batches are drawn with replacement based on these weights, ensuring that each mini-batch contains an approximately uniform distribution of Safe, Near-Collision, and Collision examples.

2.3.3 Data Augmentation for Sequence Forecasting

For the LSTM model, applying static transformations independently to each frame would destroy the temporal coherence of the motion. Therefore, we developed a sequence-consistent augmentation pipeline, considering an augmentation factor of 30 samples. Transformations are applied globally to the entire input window ($T = 10$ frames) to preserve the kinematic derivatives (velocity and acceleration).
The pipeline includes:

- Time Warping, which resamples the sequence length to simulate variations in execution speed (faster or slower movements)
- Global Scene Rotation, rotating the entire trajectory around the vertical axis to simulate different camera viewpoints while maintaining the relative human-robot distance
- Trajectory Shifts, applying small translation vectors to the whole sequence. This ensures that the augmented sequences represent physically plausible motions, crucial for learning valid forecasting dynamics.

2.3.4 Balanced Sequence Sampling for Sequence Forecasting

Balancing sequence data is more complex than static frames, as a single sequence may contain transitions between multiple risk states. We adopt a Safety-First Labeling strategy for sampling: a sequence is assigned the label of the highest risk occurring within its prediction horizon (e.g., a sequence ending in a collision is treated as a collision sample, even if it starts as safe).
Based on these high-level labels, we implement a Balanced Sequence Batch Sampler that forces each training batch to contain an equal number of sequences for each risk category ($\approx$ 33% each). Furthermore, within the collision category, preference is given to sequences containing a higher density of collision frames, prioritizing the learning of sustained impacts over transient or grazing contacts.

2.4 Single-Frame (MLP) vs Window-Sequence (LSTM)

Data samples were generated according to the temporal requirements of each architecture.

Although convolutional architectures (e.g., CNN-LSTM or TCN-based models) are widely used for spatio-temporal pattern extraction, we opted for a pure MLP + LSTM design. Our input is a compact vector of 3D joint coordinates, which does not exhibit a regular grid structure as in images or spectrograms, making convolutional operators less natural and less efficient. Moreover, collision risk depends on global geometric relations between human and robot rather than on local spatial patterns. Introducing convolutions would therefore increase computational complexity without a clear representational benefit.

For the static MLP (Single-Frame) model, the dataset consists of independent input–output pairs (x_t, y_t), where $x_t \in \mathbb{R}^{72}$ represents the flattened joint coordinates at frame t, and y_t is the corresponding instantaneous risk label.
For the LSTM (Window-Sequence) model, a sliding window approach was employed to construct input–output pairs (X, Y) representing motion history and future risk evolution. Let F_t denote the feature vector at time t. The input sequence X is defined as a tensor of shape $(T_{\text{in}} \times 72)$, covering the observation window $[F_{t-T_{\text{in}}+1}, \dots, F_t]$, with $T_{\text{in}} = 10$.
The target Y is a vector of shape (T_{out}), containing the risk labels $[y_{t+1}, \dots, y_{t+T_{\text{out}}}]$ over the prediction horizon, with $T_{\text{out}} = 25$. Sequences were extracted using a stride s (e.g. $s = 10$ for training) to balance dataset size and temporal overlap.

3 Static Risk Classification (MLP)

3.1 Architecture

3.1.1 Input representation

Each frame is characterized by two main components: the positions of the human joints and those of the robot joints. Specifically:

- Human joints are represented by 15 joints, each with 3D coordinates (x, y, z), resulting in 45 numerical values.
- Robot joints are represented by 9 joints, also with 3D coordinates, contributing an additional 27 values.

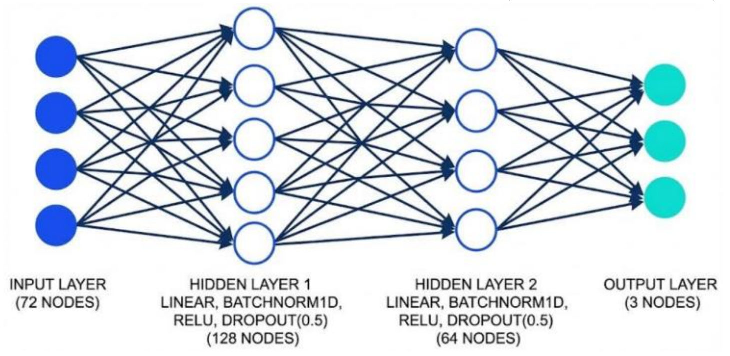
The coordinates are concatenated into a single 72-dimensional vector, which captures the relative spatial configuration between the human operator and the robot at a given point in time. This vector representation allows the model to learn risk patterns based on geometric proximity.

3.1.2 Model

The MLP model architecture is implemented as a Multi-Layer Perceptron (MLP). The model is designed to classify collision risk into three categories: Safe, Near-Collision, and Collision, based on inputs of size, corresponding to features extracted from 3D skeletal data.
The architecture is composed of three main linear layers:

- An input layer receiving vectors $x \in \mathbb{R}^{72}$.
- Two hidden layers with sizes 128 and 64 neurons, respectively.
- An output layer with 3 neurons, corresponding to the risk classes.

(18.179 Parameters)



MLP architecture

3.2 Methodology

The methodology is based on three pillars: a weighted cost function, a threshold-based inference logic, and a hierarchical model selection.

3.2.1 Weighted Cross-Entropy

Given the strong imbalance between the classes and the severity of a missed collision detection, we adopted a Weighted Cross-Entropy Loss (WCE). This function differentially penalizes classification errors, assigning a significantly higher cost to errors made on the highest-risk classes.
Defining $\mathcal{C}$ as the set of classes:
{0: *Safe*, 1: *Near-Collision*, 2: *Collision*},
the loss function $\mathcal{L}$ for a single sample is formalized as:

$$\mathcal{L} = -\sum_{c\in C} w_c \cdot y_c \cdot log(\hat{y_c})$$

Where:

- $y_c \in \{0,1\}$ is the real one-hot label.
- $\hat{y_c}$ is the probability predicted by the network
- $w = [\hat{w}_0, \hat{w}_1, \hat{w}_2]$ is the weight vector

3.2.2 Inference via Safety Thresholding and Hyperparameters

The choice of hyperparameters falls back on learning rate $\alpha=0.001$ with *Adam* optimizer and $w = [1, 4, 10]$. To maximize system sensitivity during validation and testing, the standard *argmax* decision rule is replaced by a Safety Threshold $\tau=0.1$ logic. In this way, the system prioritizes collision probability and minimize False Negatives (FN) by shifting the error distribution toward False Positives (FP), ensuring a "fail-safe" profile for operational deployment.

3.3.3 Hierarchical Model Selection

To guarantee that the final model is as safe as possible is followed a precise hierarchy of three criteria:

Priority I: Maximization of Collision Recall. The primary objective is to correctly identify the highest number of real impacts:

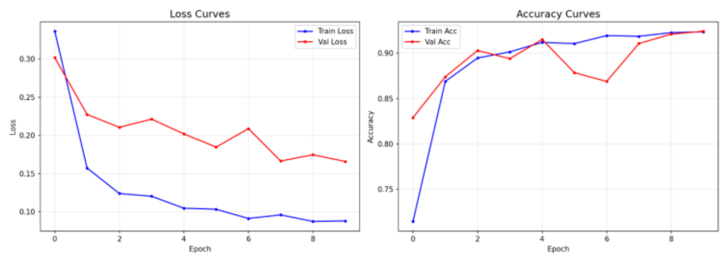
$$maximize\ R_{coll} = \frac{TP_{coll}}{TP_{coll} + FN_{coll}}$$

Priority II: Minimization the critical error of classifying a *Near-Collision* as *Safe*. This reduces the risk of missed warnings (unreported "Near-Misses").

$$\text{minimize} \sum_i 1(y_{true} = 1 \wedge \hat{C} = 0)$$

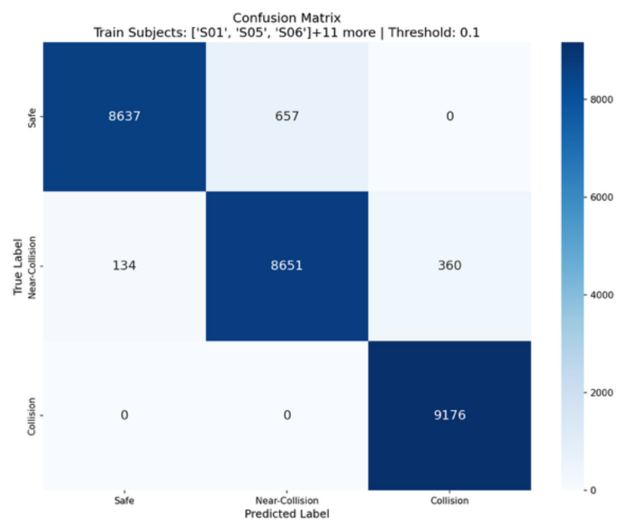
Priority III: Minimization of Validation Loss: Only as a tertiary criterion, given equal safety metrics, is the overall model generalization optimized.

3.3 Metrics and Worst Cases Analysis

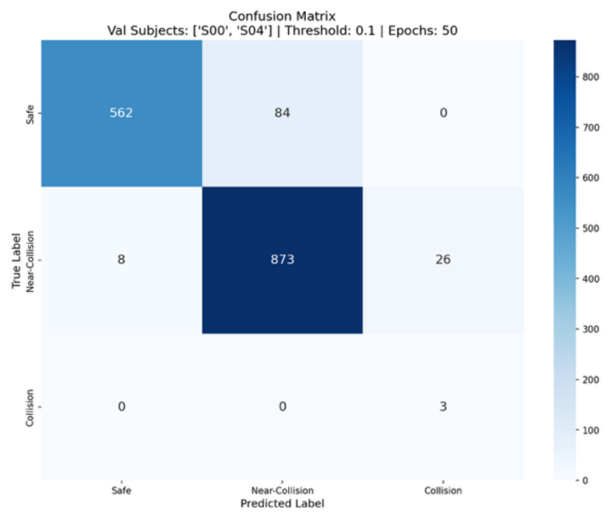


Training losses

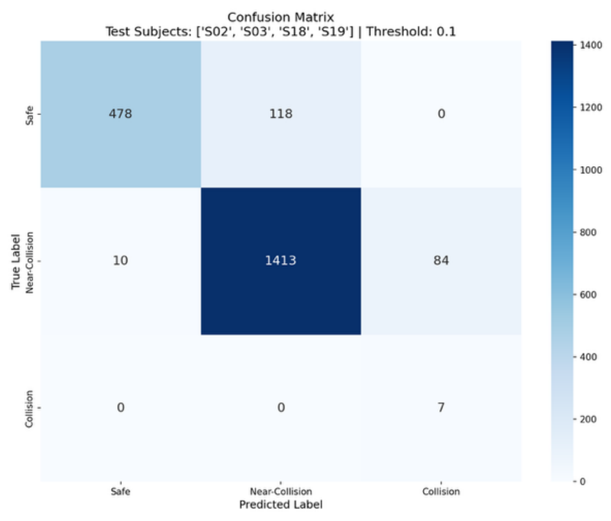
The training curves demonstrate rapid convergence, with accuracy exceeding 90% within the first few epochs. Although a slight divergence in validation loss appears after epoch 7, the applied regularization techniques effectively stabilized the model, preventing significant overfitting.



Train confusion matrix

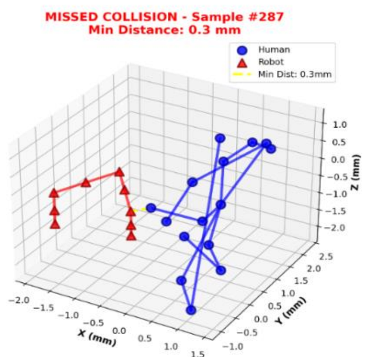


Validation confusion matrix



Test confusion matrix

Crucially for safety-critical applications, the analysis of the Test Set confusion matrix confirms the effectiveness of the weighted loss strategy. The model achieved 100% Recall on the Collision class (detecting 7 out of 7 events), ensuring that no hazardous impact was missed. Furthermore, the model exhibits conservative behavior: while it occasionally misclassifies 'Safe' states as 'Near-Collision' (false positives), it never classifies a 'Collision' as 'Safe'.



worst case analysis

Preliminary error analysis revealed that the model reduced sensitivity to collision scenarios involving non-frontal interaction angles, highlighting a lack of spatial generalization. To address this limitation and enhance robustness against variations in the operator's orientation relative to the robot, we implemented a specific rotational augmentation.

For a given skeletal pose $P \in \mathbb{R}^{Jx3}$, we apply a random rotation around the vertical axis (Y). The rotation angle θ is sampled from a uniform distribution $\theta \sim \mathcal{U}(-15°, 15°)$, simulating slight deviations in the approach angle.

The augmented pose P' is computed as:

$$P' = P \cdot R_y(\theta)^T$$

$$\mathrm{R_y}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

where $R_y(\theta)$ is the standard rotation matrix for the $Y - axis$.

This strategy effectively expands the training dataset, forcing the network to learn collision features that are invariant to small azimuthal rotations.

4 Phase 2: Dynamic Risk Forecasting (LSTM)

4.1 From Detection to Prediction:

While static detection provides immediate hazard recognition, proactive safety requires anticipating the future evolution of human-robot interactions. To address this, we introduce a temporal forecasting framework based on Recurrent Neural Networks (RNNs). Specifically, we adopt a Long Short-Term Memory (LSTM) architecture designed to perform multi-step risk forecasting.

Unlike single-frame classifiers, this model ingests a short temporal window of past motion observations to predict a sequence of future risk states. The system follows an encoder–decoder paradigm: temporal information from recent frames is compressed into a latent representation and subsequently decoded into a trajectory of future risk predictions. This design enables the model to capture both short-term motion dynamics and higher-level interaction patterns, which are essential for proactive safety assessment.

4.2 Architecture

4.2.1 Input representation

The input representation is constructed as a fixed-length temporal window of historical observations, designed to capture the dynamic evolution of the human–robot interaction. In this study, we define an observation horizon of $T = 10$ frames. Within this sequence, each time step is encoded as a feature vector summarizing the instantaneous spatial configuration of the system, encapsulating both skeletal posture and relative proximity cues.

Formally, the input is defined as:

$$X \in \mathbb{R}^{T \times 72}$$

where T represents the number of past frames included in the observation window. This formulation allows the model to jointly exploit spatial features and short-term temporal dependencies, which are fundamental for anticipating the escalation of collision risks.

4.2.2 Model

The forecasting architecture is implemented as an LSTM-based Encoder-Decoder network. The model is designed to predict the evolution of collision risk into three categories (Safe, Near-Collision, Collision) over a future horizon of 25 frames, based on an input sequence of 10 past frames containing 3D skeletal features.

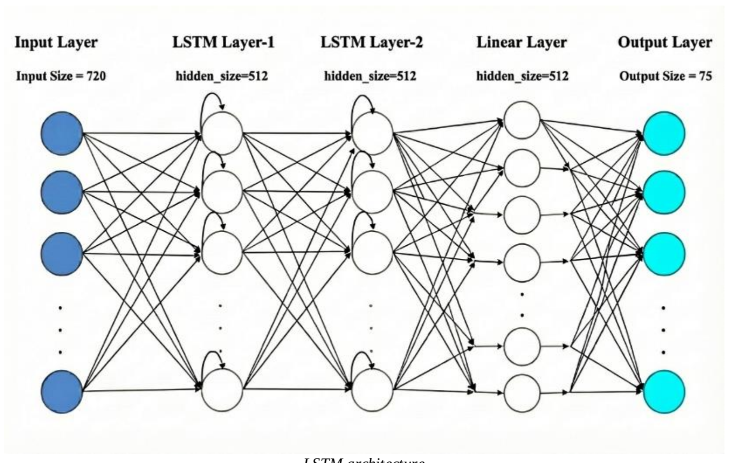
The architecture is composed of the following main components:

- An input layer receiving vectors $x \in \mathbb{R}^{720}$
- A LSTM Encoder with two hidden layers, each containing 512 neurons

- A MLP Decoder with a fully connected hidden layer (512 neurons), a Dropout Layer (0.2) and a final Output Layer that maps the latent representation to a flat vector of size 75, subsequently reshaped into a tensor

$$\hat{Y} \in \mathbb{R}^{25\times 3}$$

(Parameters 3.603.531)



LSTM architecture

4.2.3 Objective

The proposed LSTM architecture is designed to:

- effectively capture temporal dependencies in human–robot interactions,
- anticipate collision risk over an extended future horizon,
- support safety-oriented evaluation metrics such as collision recall and anticipation time,
- remain computationally efficient for near real-time deployment.

4.3 Temporal Methodology

To address the dual challenge of class imbalance and the inherent uncertainty of long-term forecasting, we designed a custom objective function: the Temporal Weighted Cross-Entropy Loss. Standard loss functions treat errors at every time step uniformly. However, in safety-critical HRC, an error in the immediate future (e.g.,$t + 1$ to $t + 15$) is operationally more severe than an error at the end of the prediction horizon (e.g., $t + 25$), where stochastic uncertainty dominates. The proposed loss function introduces a time-dependent modulation term λ_t to the standard weighted class-based cross-entropy. Formally, let $Y = \{y_1, \dots, y_P\}$ be the sequence of ground-truth labels and $\hat{Y} = \{\hat{y}_1, \dots, \hat{y}_P\}$ be the sequence of predicted probability distributions over the prediction horizon P. The total loss $\mathcal{L}$ is defined as:

$$\mathcal{L}(Y, \hat{Y}) = \frac{1}{P}\sum_{t=1}^{P} \lambda_t \cdot \mathcal{L}_{WCE}(y_t, \hat{y}_t)$$

Where $\mathcal{L}_{WCE}$ represents the class-weighted cross-entropy at time step t:

$$\mathcal{L}_{WCE}(y_t, \hat{y}_t) = -\sum_{c=1}^{C} w_c \cdot 1(y_t = c)\, log(\widehat{y_{t,c}})$$

Here, w_c denotes the static risk weight for class c (configured as $w = [1,1,5]$ or similar to mitigate imbalance), and $\mathbb{1}(\cdot)$ is the indicator function.

The key innovation is the temporal weight λ_t, which is modeled as a step-decay function to enforce a high-priority "urgency window":

$$\lambda_t = \begin{cases} 1.0 & if\, 1 \le t \le K_{urgent} \\ \gamma & if\, K_{urgent} < t \le P \end{cases}$$

In our experiments, we set the urgency horizon $K_{urgent} = 15$ frames (approx. 0.6 seconds) and the decay factor $\gamma = 0.3$.

This configuration ensures that the gradient descent optimization prioritizes accuracy in the short-term interval (where reliable pre-collision warnings are most critical) while preventing the high variance of long-term predictions ($t > 15$) from destabilizing the training process.

4.3.1 Inference via Safety Thresholding and Hyperparameters

Model optimization is performed using the Adam optimizer with a learning rate $\alpha = 10^{-3}$. To counteract the severity of missing hazardous events, the loss function incorporates class weights $w = [1,1,5]$ for Safe, Near-Collision, and Collision respectively.

Furthermore, to maximize system sensitivity during validation and testing, the standard argmax decision rule is superseded by a Safety Threshold logic with $\tau = 0.1$. Under this regime, any sequence where the predicted collision probability exceeds 0.1 is classified as a Collision, overriding the highest probability class.

This strategy explicitly prioritizes collision detection to minimize False Negatives (FN), consciously shifting the error distribution toward False Positives (FP) to ensure a "fail-safe" profile essential for operational deployment.

4.3.2 Hierarchical Model Selection

The selection process evaluates model checkpoints at each epoch based on a strict hierarchy of three criteria:

Priority I: Maximization of Minimum Class Recall (Maximin Principle). The primary objective is to improve the performance of the "weakest link" among the three classes. We calculate the recall for *Safe*, *Near-Collision*, and *Collision* independently, and identify the minimum value among them. This criterion forces the model to maintain high standards for *all* classes simultaneously, ensuring that high safety (Collision detection) is not achieved by sacrificing the recognition of Safe or Near-Collision state

$$\text{maximize}\left(\min_{c\in\{0,1,2\}} \text{Recall}_c^{(t\le 15)}\right)$$

Priority II: Maximization of Mean Class Recall. Given two models with the same minimum recall (e.g., both have a worst-class performance of 70%), the secondary criterion favors the model with the higher average performance across all classes (Macro-Average Recall).

$$\text{maximize}\left(\frac{1}{3}\sum_{c=0}^{2} \text{Recall}_c^{(t\le 15)}\right)$$

Priority III: Minimization of Validation Loss. As a final tie-breaker, if both recall metrics are equivalent, we select the model with the lowest overall validation loss to ensure better generalization and probability calibration.

4.4 Forecasting Metrics

To comprehensively evaluate the performance of the proposed forecasting model, we adopt a set of time-aware and safety-oriented metrics specifically designed for multi-step risk prediction. Standard classification accuracy alone is insufficient in this context, as it fails to capture temporal anticipation, early detection, and prediction stability.

4.4.1 Per-Timestep Recall

For each future timestep $\boldsymbol{t} \in \{\mathbf{1}, \dots, \boldsymbol{P}\}$, recall is computed independently for each risk class. For class $\boldsymbol{c}$, recall at timestep $t$is defined as:

$$\text{Recall}_c(t) = \frac{TP_c(t)}{TP_c(t) + FN_c(t)}$$

where $TP_c(t)$ and $FN_c(t)$ denote the number of true positives and false negatives for class $c$at timestep t, respectively. This metric measures the model's ability to correctly identify a given risk class at a specific future offset.

4.4.2 Temporal Recall Degradation

By analyzing recall as a function of the prediction horizon, it is possible to assess how prediction quality degrades over time. This temporal analysis provides insight into:

- the effective anticipation range of the model,
- the reliability of early versus late predictions.

Emphasis is placed on the recall values within the first 15 future frames, where early detection is most valuable.

4.4.3 Mean Anticipation Time (MAT)

To explicitly quantify how early collisions are detected, we introduce the Mean Anticipation Time (MAT). For each sequence containing a collision, MAT is defined as:

$$\mathrm{MAT} = \frac{1}{\mathrm{N_a}} \sum_{\mathrm{i=1}}^{\mathrm{N_a}} \left(\mathrm{t}_{\mathrm{collision}}^{(\mathrm{i})} - \mathrm{t}_{\mathrm{prediction}}^{(\mathrm{i})}\right)$$

where $t_{\text{collision}}^{(i)}$is the first ground-truth collision frame of sequence i, and $t_{\text{prediction}}^{(i)}$ is the earliest predicted collision frame preceding it. Only valid anticipations are included in the average.

MAT directly measures the average temporal margin available for intervention, making it a key indicator for early-warning effectiveness.

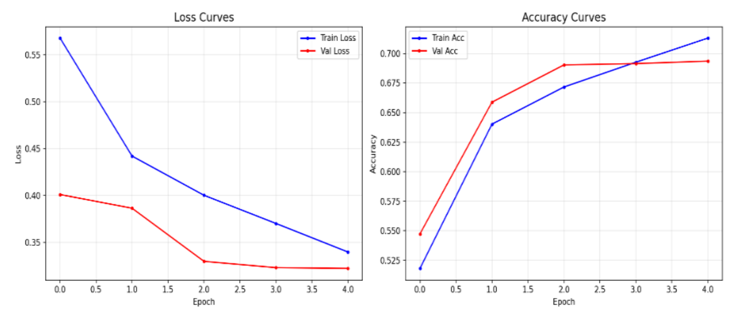
4.4.4 Prediction Stability (Flicker Rate)

To assess temporal consistency, we measure the Flicker Rate, defined as the number of class changes in the predicted sequence:

$$\mathrm{Flicker} = \sum_{\mathrm{t=2}}^{\mathrm{P}} \mathbb{1}[\hat{y}_{\mathrm{t}} \neq \hat{y}_{\mathrm{t-1}}]$$

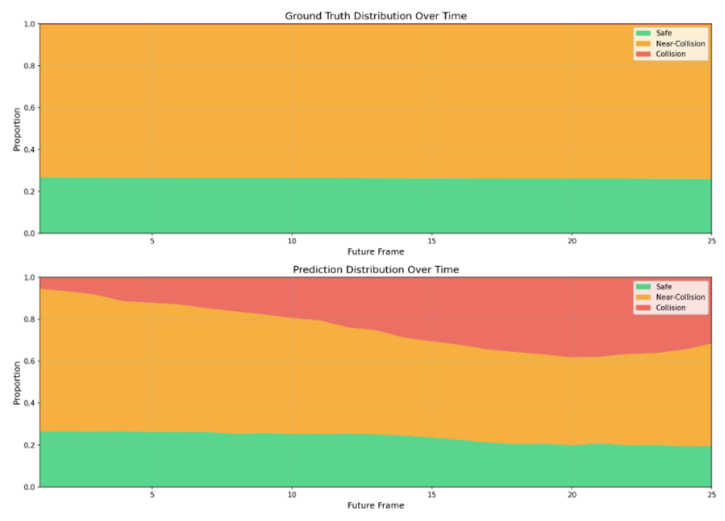
High flicker rates indicate unstable predictions, which are undesirable in real-time safety systems. Stable forecasts are essential to avoid oscillatory risk signals that could compromise decision-making.

5 Results and Analysis



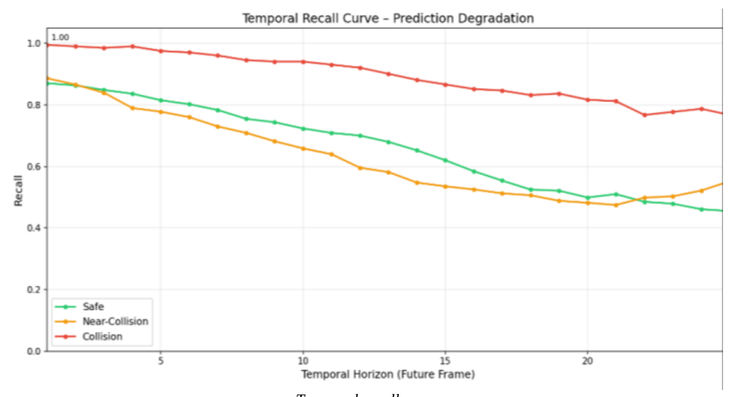
Training losses of LSTM architecture

The dynamics show a monotonic decrease in both training and validation losses, confirming that the model effectively captures temporal dependencies within the first few epochs while maintaining a robust alignment between training and validation performance.



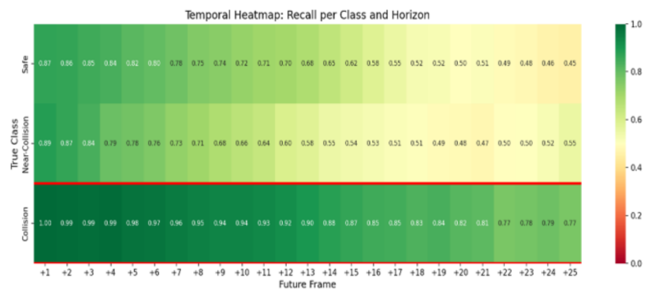
Temporal distribution of prediction

The analysis highlights the model's handling of temporal uncertainty. As the prediction horizon grows, the "Safe" proportion (green) remains relatively stable, whereas the "Near-Collision" proportion (orange) is progressively absorbed by the "Collision" class (red). This suggests that the model effectively distinguishes safe states even in the long term, but tends to upgrade "Near-Collision" scenarios to "Collision" when predictive confidence degrades. This behavior is a desirable property for safety-critical HRC, ensuring that ambiguous proximity is treated with maximum caution.



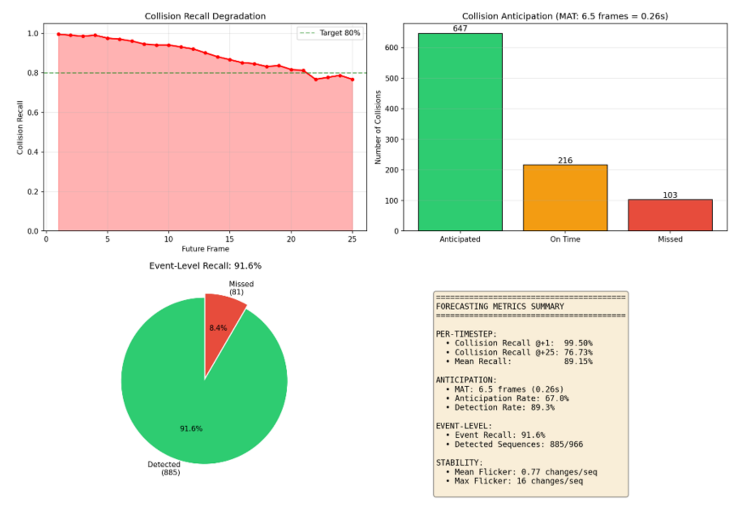
Temporal recall curve

The curve exhibits a "plateau" behavior in the initial phase (frames 1–10), where recall remains nearly constant at 1.0 for *Collision* classes. This aligns with the *"urgency window"* defined in the loss function parameters. The drop observed after frame 15 corresponds to the reduced penalty weight ($\gamma = 0.3$), confirming that the model successfully learned to focus its predictive capacity on the immediate, actionable future while managing long-term uncertainty through a conservative bias.



Temporal heatmap

The heatmap visualizes the system's operational safety profile. While prediction accuracy naturally decays over time due to the stochastic nature of human motion, the decay is asymmetric. The Collision recall (bottom row) decays significantly slower than the Safe recall (top row). This asymmetry is a direct result of the safety-weighted training strategy: the model effectively learns that in scenarios of high temporal uncertainty (frames +15 to +25), calculating a "false alarm" is preferable to missing a potential hazard. This ensures that the robot is more likely to slow down unnecessarily than collide unexpectedly.



```
========================================
FORECASTING METRICS SUMMARY
========================================

PER-TIMESTEP:
  • Collision Recall @+1:  99.50%
  • Collision Recall @+25: 76.73%
  • Mean Recall:           89.15%

ANTICIPATION:
  • MAT: 6.5 frames (0.26s)
  • Anticipation Rate: 67.0%
  • Detection Rate: 89.3%

EVENT-LEVEL:
  • Event Recall: 91.6%
  • Detected Sequences: 885/966

STABILITY:
  • Mean Flicker: 0.77 changes/seq
  • Max Flicker: 16 changes/seq
```

Summary

This composite figure summarizes the system's performance across four key dimensions:

1. *Temporal Robustness* (Top-Left): The collision recall curve starts at near-perfection (99.5% at $t = 1$) and degrades gracefully, staying above the 80% target threshold up to frame +21. This indicates a reliable prediction window of approximately 0.84 seconds.
2. *Proactivity* (Top-Right): The system demonstrates strong anticipatory capabilities. Out of 966 collision events, 67% were anticipated (detected before the actual impact), yielding a Mean Anticipation Time (MAT) of 6.5 frames (0.26s). This provides a critical time margin for preemptive robot braking.
3. *Reliability* (Bottom-Left): At the episode level, the model successfully flags 91.6% of all hazardous sequences, proving that sustained threats are rarely missed.
4. *Stability* (Bottom-Right): The low flicker rate (0.77 changes/sequence) confirms that the predictions are consistent and do not suffer from noise-induced oscillations, a crucial requirement for smooth robot control.

6 Cross-Subject Evaluation Analysis

6.1 Introduction

To assess the generalization capabilities of the proposed models, relying on a single static train-test split is insufficient, particularly in datasets characterized by high inter-subject variability like CHICO. Randomly splitting frames across training and testing sets often leads to data leakage, where the model learns the specific kinematic idiosyncrasies of a subject rather than the general dynamics of the interaction.

To address this, we adopted a Subject-Independent K-Fold Cross-Validation protocol ($K = 5$). In this setup, the dataset is partitioned by subject ID rather than by sample index: in each fold, the model is tested on a group of users never seen during training. This approach strictly adheres to the Cross-Subject Evaluation criteria established by Shahroudy et al. [Ref] for 3D skeletal data analysis, ensuring that the performance metrics reflect the system's ability to operate safely with new, *unknown operators* in a real-world industrial environment.

6.2 Performance Analysis of the Static Risk Classifier

Test set	Mean	Dev. Std (±)	Min	Max
Collision Recall	96.00%	± 8.94%	80.00%	100.00%
Accuracy	84.02%	± 5.76%	75.46%	90.06%
Missed Warning	12.00	± 7.8%	5	25

The results of MLP model demonstrate exceptional robustness in detecting imminent hazards. Aggregated metrics across the five folds show a Mean Collision Recall of 96.00% (± 8.96%). The variability (standard deviation of 8.96%) indicates that while the model achieves 100% recall on certain subject groups, specific kinematic behaviors in others remain challenging.

6.3 Performance Analysis of the Forecasting Model

Test set @ 15 frames	Mean	Dev. Std (±)	Min	Max
Collision Recall	87.13%	± 4.98%	81.52%	94.22%
Min Class Recall	70.98%	± 3.94%	64.30%	75.80%
Mean Class Recall	79.94%	± 1.21%	78.26%	81.40%
Accuracy	63.50%	± 4.45%	56.86%	69.89%

Validating the LSTM model presents a greater challenge due to the stochastic nature of predicting future motion. The Cross-Validation yielded a Mean Collision Recall of 87.13% (± 4.98%). Considering the complexity of the forecasting task compared to static detection, this result is significant. The trade-off for this high sensitivity is a reduced overall Accuracy (63.5%), driven by a conservative "fail-safe" bias where ambiguous Near-Collision states are often classified as Collisions. Crucially, the low standard deviation ($\approx$ 5%) suggests that the LSTM generalizes better across different subjects than the MLP, likely due to learning temporal dynamics rather than static poses.

7 Inference Latency and Real-Time Feasibility

The prediction system is designed to operate in real time: at each time step t, a sliding window of the previous 10 frames is processed to update the risk forecast. To ensure compatibility with real-time control loops, we measured inference latency on a standard CPU, simulating an industrial controller environment, using a batch size of 1:

- LSTM : average latency of 3.6978 ms

This shows that the system’s inference requires only a small fraction of the control cycle, leaving ample time for data acquisition and safety-critical actuation. Even when using the more complex dynamic model, latency remains well within the 25 fps time budget (40 ms), confirming the feasibility of deploying the system in real-time industrial applications.

8 Conclusions

While both the MLP and LSTM models demonstrate strong performance within their respective operational scopes, their applicability in real-world safety-critical systems differs substantially.

The MLP model achieves highly reliable instantaneous collision detection, exhibiting conservative behavior that effectively minimizes false negatives on the test set. This makes the MLP suitable for reactive safety mechanisms, where the objective is to detect hazardous states as soon as they occur. Its simplicity and low computational overhead further support deployment in constrained environments or as a complementary safety layer.

However, the MLP’s frame-wise formulation inherently limits its ability to anticipate future risk. Predictions are driven solely by the current observation, resulting in reactive behavior that offers no temporal margin for preventive action. In real industrial settings, where safety interventions must be triggered before physical contact, this limitation reduces the operational usefulness of purely static classifiers.

The LSTM model addresses this limitation by enabling explicit temporal risk forecasting. Experimental results show that the LSTM consistently achieves high collision recall within the early portion of the prediction horizon, positive mean anticipation times, and temporally stable predictions. These properties demonstrate the model’s ability to identify hazardous interaction patterns before collisions occur, providing actionable lead time for intervention strategies such as speed reduction, trajectory adjustment, or emergency stopping.

The comparison between the two models confirms that temporal modeling is a key requirement for real-world predictive safety. While the MLP delivers robust instantaneous detection, the LSTM enables proactive safety assessment, aligning more closely with the operational needs of collaborative robotic systems. The observed improvements in anticipation capability and prediction stability make the LSTM-based approach better suited for deployment in real industrial environments, where early warning and reliability are critical.

Overall, the results suggest that frame-wise models may serve as effective reactive safeguards, whereas temporal models such as LSTMs are essential for achieving predictive, human-aware safety in dynamic human–robot collaboration scenarios.

9 References

- Alessio Sampieri, Guido D'Amely, Andrea Avogaro, Federico Cunico, Geri Skenderi, Francesco Setti, Marco Cristani, Fabio Galasso (2022). *Pose Forecasting in Industrial Human-Robot Collaboration*. https://arxiv.org/abs/2208.07308
- Amir Shahroudy, Jun Liu, Tian-Tsong Ng, Gang Wang (2016). *NTU RGB+D: A large scale dataset for 3D human activity analysis*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1010–1019. https://arxiv.org/abs/1604.02808