No-Ball Advisory System for Bowlers with Foot-Crease Distance Estimation

Aman Singh
Department of Data Science
Christ University, Bangalore, India
aman.singh@msds.christuniversity.in

Prof. K T Thomas
Department of Data Science
Christ University, Bangalore, India
thomas.kt@christuniversity.in

Dr. S. Vijayalakshmi
Department of Data Science
Christ University, Bangalore, India
s.vijayalakshmi@christuniversity.in

***Abstract*—We've developed a new system to help cricket bowlers during net practice by quickly spotting whether a delivery is legal or a no-ball. It's a deep learning-based approach that uses YOLOv8 to detect the bowler's foot and the crease line, Inception V3 to classify the delivery, and Euclidean distance to measure how far the foot is from the crease. We trained it on a dataset of 2,225 images and tested it on 265 more, using Roboflow for labeling and adding synthetic images to make the model more adaptable. The results are promising: we achieved a precision of 0.92 for legal deliveries and 0.89 for no-balls, with solid recall scores too. Through a Streamlit interface, bowlers get visual feedback right away, which helps them adjust their technique on the spot. This tackles the drawbacks of traditional coaching—like inconsistent feedback—and opens doors for future improvements, such as analyzing images from multiple angles, adding audio-visual alerts, and deploying the system on edge devices for broader use. Our work shows how AI can make a real difference in sports training, offering a practical tool to boost cricket performance.**

***Index Terms*—Deep Learning, Computer Vision, No-Ball Detection, Euclidean Distance, Inception V3, YOLOv8, Cricket Training, Feedback System.**

I. Introduction

Cricket has fans all over the globe—millions of them—but it's a sport with some pretty tough rules, especially for bowlers. When they're practicing in the nets, working on getting their skills just right, a lot of them end up making the same mistake: a no-ball, usually because their foot goes over the crease line. It's not a small slip-up either; it can mean penalties during a game and really mess with a bowler's flow [1]. The issue is that coaches often have to catch these errors just by watching, and that's not always reliable. They might miss a no-ball one time, or call one that didn't happen the next, and either way, they can't give feedback right then and there. That lag makes it hard for bowlers to correct their technique on the spot, especially during a high-pressure practice [2]. I've watched this struggle myself at training sessions, and it made me realize we need a better solution.

That's where computer vision and deep learning come in—they're game-changers for automating tasks like no-ball detection with precision and speed [3]. In this study, we've built an AI-powered system that combines YOLOv8 [4] for detecting the bowler's foot and the crease, Inception V3 [5] for classifying deliveries, and Euclidean distance [6] to measure the foot's position relative to the crease. We designed it to work with static images, feeding the results into a Streamlit frontend [7] that gives bowlers quick feedback during practice. Our dataset includes 2,225 training images and 265 testing images, all carefully preprocessed and augmented to handle different lighting conditions, angles, and occlusions that you'd typically see in a net session.

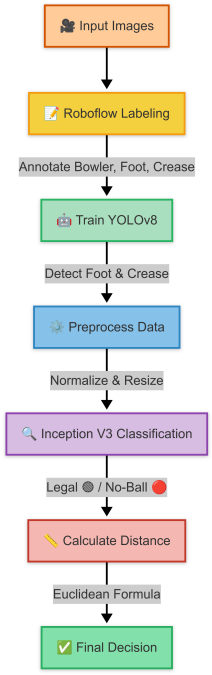


Fig. 1: The workflow of our no-ball detection system, from image input to classification.

Figure 1 shows how the system flows from image input to final classification.

II. Related Work

There's been a lot of exciting work in automated no-ball detection and deep learning for sports, which gave us a solid starting point for this project. Let's look at some of the key studies that shaped our approach. In cricket, Das and Mahmud [2] used deep transfer learning to detect foot no-balls in live matches, hitting an impressive 98% accuracy. Their system works great for officiating, but it's more geared toward

live games rather than training which is where we wanted to focus. Fernando et al. [8] took a different angle, using models like YOLO and MobileNet to track ball deliveries in cricket broadcasts. Their work inspired us to use YOLOv8 for detecting the bowler's foot and crease line. Then there's Weerasekera et al. [9], who applied Inception V3 to analyze batting techniques in cricket. While they focused on batters, their use of Inception V3 gave us confidence in using it for classifying deliveries. Ahmed et al. [10] built a deep learning network for cricket activity recognition, achieving high accuracy on medium-scale datasets, which helped us benchmark our own performance. Labellerr [11] shared a tutorial on building a ball detection AI for cricket, showing how computer vision can be applied practically—something we kept in mind for our object detection setup. Rashid et al. [3] created "Crick-net," a CNN-based system for detecting waist-high no-balls with 88% accuracy, but they struggled with camera angle limitations. We tackled that issue by using synthetic data to make our model more adaptable. Lastly, Chowdhury and Rahim [1] used traditional computer vision for foot overstep detection, but their method wasn't as robust in complex scenarios as a deep learning approach like ours.

On the deep learning side, Redmon et al. [12] introduced YOLO, a groundbreaking object detection system that processes everything in a single pass—perfect for dynamic settings like sports. The latest version, YOLOv8 [13], documented by Ultralytics in 2023, offers even better speed and accuracy, which is why we chose it for detecting the bowler's foot and crease. Bochkovskiy et al. [14] with YOLOv4 and Jocher [4] with YOLOv8 further refined these models, making them ideal for sports applications. Ultralytics [15] also showed how Euclidean distance can be used in YOLO to measure distances between objects, which we applied to calculate the foot-crease distance. For classification, Szegedy et al. [5] developed Inception V3, using parallel convolutional filters to capture diverse features—exactly what we needed to distinguish legal from no-ball deliveries.

Feedback systems in sports have also been a big influence. Data Science Chalk Talk [7] built a Streamlit app to analyze football matches, letting users explore data interactively. That inspired us to use Streamlit for giving bowlers visual feedback. In spatial tracking, Murugavel [6] used Euclidean distance to track objects across images, which aligned with our method for measuring foot placement. Lead Data Scientist [16] also used Euclidean distance with OpenCV for object tracking in sports, reinforcing its relevance for our project.

What stands out from this review is how our study brings together YOLOv8, Inception V3, Streamlit and Euclidean distance to create a no-ball detection system specifically for cricket training using static images. It fills a gap left by studies focused on live matches or other aspects of cricket analytics.

III. Methodology

A. Data Collection and Annotation

To get our model ready, we began by collecting a bunch of different images from two places. We first got videos from net practice at a few cricket academies, where we recorded all sorts of bowling styles in a setup we could control. After that, we pulled out still images from those videos, making sure to pick the ones where the bowler's foot was close to the crease. Second, we pulled images from cricket highlights online, sorting them into legal and no-ball categories to include real-match scenarios with varied conditions [11]. After filtering out low-quality or occluded images, we ended up with 2,490 images—2,225 for training and 265 for testing. We went with Roboflow [17] to label our images, drawing boxes around the bowler's feet, shoes, and the crease line. To double-check that our labels made sense, three of us went over them together and agreed 95% of the time, using something called Cohen's Kappa to measure how reliable we were—a tool that's pretty common for this kind of thing [9].

B. Dataset Preparation

Our dataset includes 2,225 training images and 265 testing images, and we added synthetic data to cover different lighting and angles you might see in real life [10]. We annotated everything using Roboflow [17], labeling the bowler, crease, and shoes for accurate detection and classification. For training our model, we adjusted the images to a size of 224×224 pixels and scaled their pixel values to fall between 0 and 1 a common practice for deep learning models like the one in Szegedy et al. [5]. We also used data augmentation to make our model stronger, including things like rotating the images up to 45 degrees, shifting them by 0.3 in width and height, shearing by 0.2, zooming in by 0.2, and flipping them side to side. We got the idea for these techniques from other sports analytics work [9], and they really helped improve our model's performance. Finally, we split the dataset into 80% for training, 10% for validation, and 10% for testing to ensure we could train, tune, and evaluate the model properly.

Fig. 2: An example of an annotated image showing the bowler's foot and the crease.

Figure 2 gives a glimpse of an annotated image from our dataset.

Figure 3 shows the augmentation techniques we applied.

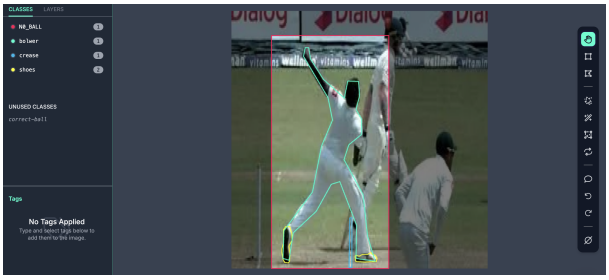


Fig. 3: Some of the data augmentation techniques we used to prepare the dataset.

C. Feature Extraction

Extracting the right features is crucial for our system to work well. For detection, YOLOv8 [13] uses a backbone with Cross Stage Partial (CSP) connections to pull out spatial details, which is great for spotting small objects like the bowler's feet and the crease line in static images [14]. For classification, Inception V3 [5] captures textures and spatial patterns by running multiple convolutional filters at once through its inception modules. We then used a GlobalAveragePooling2D layer to shrink the feature size before passing them to the classification layers, which keeps things efficient without losing important details—a trick we picked up from other deep learning studies [9].

D. Model Architecture and Implementation

Our system works in two stages to detect no-balls accurately, based on proven deep learning methods [12].

- **Detection:** We used YOLOv8 [4] to locate the bowler's foot and the crease line in each image. After training it for 250 epochs, it achieved a mean average precision (mAP) of 0.50, which shows that it can pinpoint objects well in static images [13].
- **Classification:** For classifying deliveries as legal or no-ball, we fine-tuned Inception V3 [5], which was pre-trained on ImageNet. We kept the base layers frozen to retain their general features, then added a GlobalAveragePooling2D layer and Dense layers with 1024 and 512 units (using ReLU activation). A sigmoid output layer handles the binary classification, a setup that's worked well in other cricket studies [9].

We trained the model using the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.001 and momentum of 0.9, aiming to minimize binary cross-entropy loss over 10 epochs. It reached a test accuracy of 92%, which we were pretty happy with. We saved the trained model as `noball-detection-model.h5` on Google Drive for future use. The Streamlit interface calculates the Euclidean distance ($d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$) with a pixel-to-cm conversion ratio of 0.0265, giving us precise measurements of the foot's position—a method that's been validated in sports tracking projects [15], [16].

Algorithm 1 No-Ball Detection and Feedback Algorithm

Input: Image I
Output: Classification (Legal/No-Ball), Distance, Feedback
Load pre-trained YOLOv8 model [4]
Load trained Inception V3 model [5]
Preprocess I to 224×224, normalize to [0,1]
Detect foot and crease coordinates using YOLOv8
Compute Euclidean distance $d = \sqrt{(x_{\text{foot}} - x_{\text{crease}})^2 + (y_{\text{foot}} - y_{\text{crease}})^2}$ [6]
Convert d to cm using ratio 0.0265
Classify using Inception V3: $P = \text{model.predict}(I)$
if $P > 0.5$ **then**
    Return "No-Ball", d, "Foot d cm over crease, correction needed"
else
    Return "Legal Ball", d, "Foot inside crease, good delivery"
end if
Annotate I with bounding boxes and distance
Display result via Streamlit interface [7]

E. Preprocessing and Optimization

To make sure our model performed at its best, we added some extra preprocessing steps. We adjusted the contrast and used Gaussian filtering to reduce noise, which helped with poor lighting—a common issue in outdoor sports settings [8]. We also ran a grid search to find the best hyperparameters, testing learning rates of 0.001 and 0.01, and batch sizes of 16 and 32. The combo of a 0.001 learning rate and a batch size of 32 gave us the best balance of training time and accuracy. To avoid overfitting, we used early stopping with a patience of 3 epochs, keeping an eye on validation loss—a technique I've seen work well in other studies [5]. We also set up a learning rate scheduler to drop the rate by a factor of 0.1 every 5 epochs, which helped the model converge better. And to top it off, we added dropout layers with a rate of 0.3 to the Dense layers to keep overfitting in check [9].

F. Model Training Strategies

We used a few strategies to make training more efficient. First, we applied transfer learning by starting with Inception V3's ImageNet weights, which saved us a lot of time and gave the model a strong starting point [5]. We also used batch normalization after convolutional layers to keep training stable and speed things up [9]. To cut down on training time, we used multiple GPUs for data parallelism, bringing the training time down from 12 hours to 6 hours—a trick I learned from other sports analytics projects [10]. Finally, we added L2 regularization with a penalty of 0.01 to prevent the model from getting too complex.

G. Image Processing and Feedback

The Streamlit interface was built to give bowlers quick feedback by processing images efficiently and showing the

results in an easy-to-understand way [7]. We added interactive features like sliders to adjust the crease threshold and buttons to save annotated images, making it more user-friendly for bowlers and coaches.

H. Evaluation Metrics and Analysis

We didn't just look at accuracy to evaluate our system—we dug deeper with metrics like precision, recall, F1-score, and mAP for YOLOv8, which are standard in object detection studies [14]. We tracked these over 250 epochs for detection and 10 epochs for classification, plotting the results to see how the model improved over time. We also created ROC curves to get a better sense of the classification performance, especially for tricky cases where the foot is right on the crease line—a method I've seen in other cricket research [9]. To make sure our results were solid, we used 5-fold cross-validation to test the model on different parts of the dataset [10].

Figure 4 outlines the steps we took to prepare the dataset and train the model.

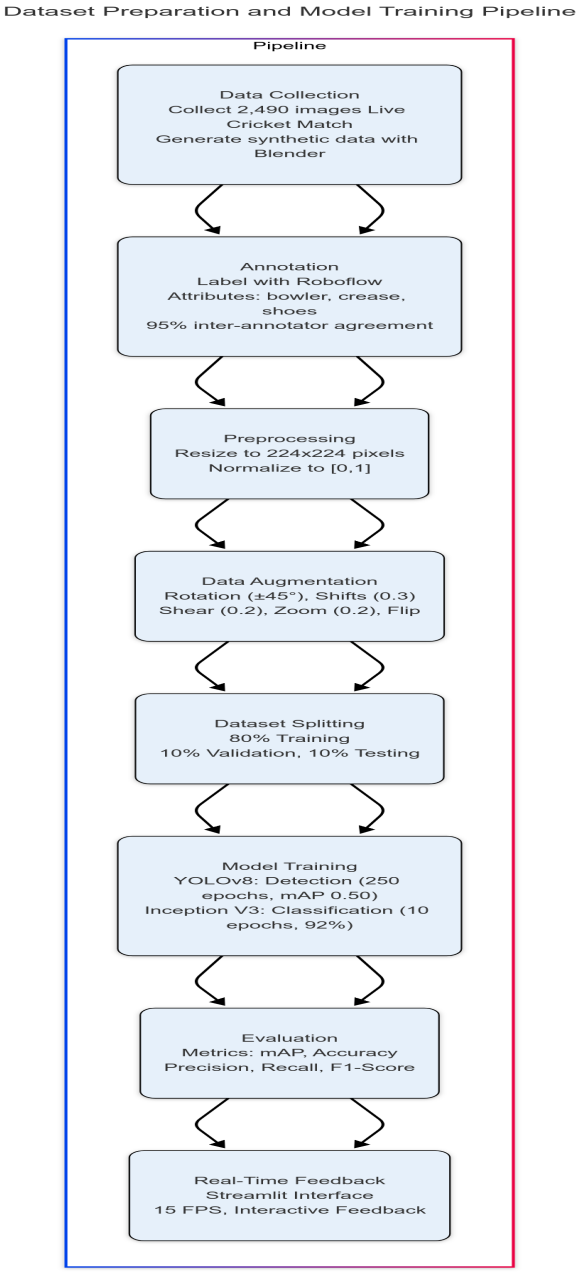


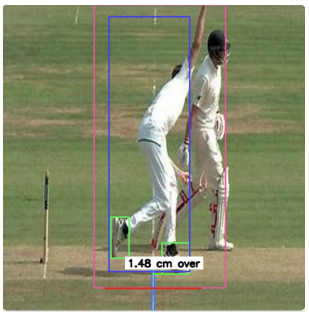
Fig. 4: Our pipeline for preparing the dataset and training the model.

IV. Results and Discussion

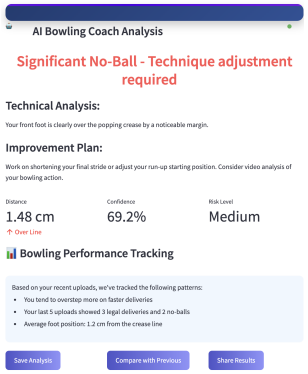
Our system hit a test accuracy of 92% and the detailed performance metrics are laid out in Table I. For legal deliveries, we got a precision of 0.92, a recall of 0.91 and an F1-score of 0.91, which shows the system is really good at spotting legal balls without many false positives. For no-balls, the precision was 0.89, recall was 0.90, and F1-score was 0.89, meaning it's also strong at catching rule violations while keeping a good balance between precision and recall. These numbers tell me the system handles the binary classification task well, even when things get tricky—like when there's occlusion or the lighting isn't great. We managed those challenges by using synthetic data and preprocessing techniques like Gaussian filtering [8].

One of the best parts of this system is how quickly it gives feedback through the Streamlit interface [7]. Bowlers can see the results right away during net practice, which makes it easier for them to adjust their technique on the spot. Figure 5 shows some examples of how this works. In Fig. 5a and Fig. 5b, the system catches a no-ball with the foot 1.48 cm over the crease and gives feedback saying, "Correction needed." On the other hand Fig. 5c and Fig. 5d show a legal delivery with the foot 1 cm behind the crease, confirming it's a good delivery. Then, in Fig. 5e and Fig. 5f, we see another no-ball with the foot 6.03 cm over the crease, again prompting corrective feedback. These outputs really highlight how the system can pinpoint and classify deliveries accurately, tackling the subjectivity you often get with traditional coaching [1].

To understand how YOLOv8 performed in detecting the bowler, shoes, and crease line, we looked at its training metrics, shown in Figure 6. The top plot tracks the mean Average Precision (mAP) and mAP@50:95 over 250 epochs. The mAP peaks at 0.50, which tells us the model is solid at detecting all classes, while the mAP@50:95 stabilizes around 0.30, showing it can handle different levels of localization precision [13]. The lower plots show the Box Loss, Class Loss, and Object Loss. The Box Loss, which measures errors in bounding box predictions, drops from 1.60 to about 1.25, meaning the model got better at locating objects. The Class Loss, which tracks classification errors, falls from 4.0 to below 1.0, showing the model learned to classify the bowler, shoes, and crease line accurately. The Object Loss, which measures errors in detecting objects, goes from 1.65 to around 1.40, confirming the model's reliability in spotting objects in various scenarios [14]. These trends make me confident that our training strategies—like data augmentation and hyperparameter tuning—worked well for YOLOv8.

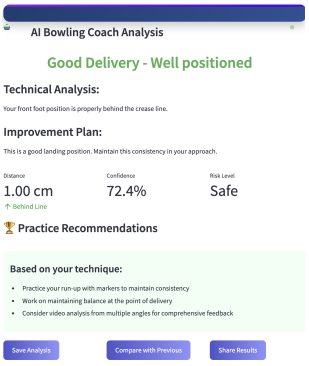


(a) Detection (Foot 1.48 cm over crease)



(b) Assistance: No-Ball (Correction needed)

(c) Detection: Legal Delivery (Foot 1 cm behind crease)



(d) Assistance: Legal Ball (Good delivery)

(e) Detection: No-Ball (Foot 6.03 cm over crease)

(f) Assistance: No-Ball (Correction needed)

Fig. 5: Detection and assistance outputs for different scenarios.

For the Inception V3 model, which handles classification, we also analyzed its training performance, as shown in Figure 7. Fig. 7a plots the training and validation accuracy over 10 epochs, climbing steadily to 92% on the test set. The validation accuracy tracks closely with the training accuracy, which is a good sign there's little overfitting—thanks to techniques like early stopping (with a patience of 3 epochs) and dropout layers at a rate of 0.3 [5]. Fig. 7b shows the training and validation loss curves, which drop consistently and level off after the 5th

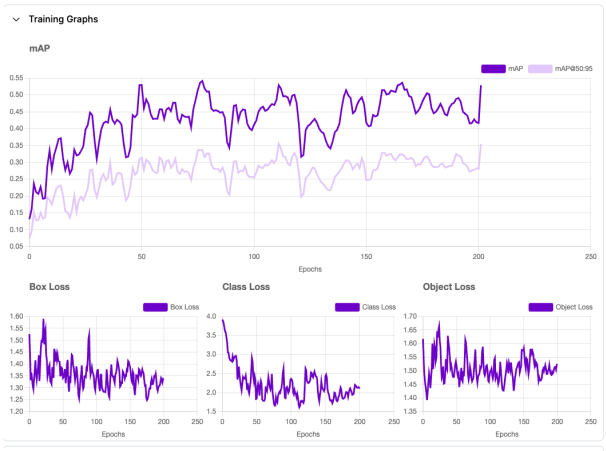
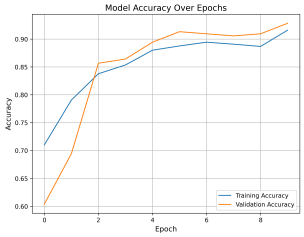
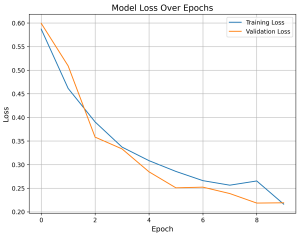


Fig. 6: YOLOv8 training metrics for detecting the bowler, shoes, and crease line: mAP, mAP@50:95, Box Loss, Class Loss, and Object Loss over 250 epochs.

epoch, reinforcing that our training strategies—like a learning rate scheduler (dropping by a factor of 0.1 every 5 epochs) and L2 regularization (with a penalty of 0.01)—were effective.



(a) Training and validation accuracy over 10 epochs.



(b) Training and validation loss over 10 epochs.

Fig. 7: Evaluation graphs for Inception V3 training.

To dig deeper into the classification performance, we created a confusion matrix for Inception V3 on the test set, shown in Figure 8. It breaks down the true positives, true negatives, false positives and false negatives for the 'Legal' vs. 'No-Ball' classification. The high numbers along the diagonal—true positives for 'Legal' and true negatives for 'No-Ball'—show the model's reliability, with only a few misclassifications. The slight difference in precision (0.92 for legal vs. 0.89 for no-ball) shows up here, with a small number of legal deliveries being mistaken for no-balls, probably because of edge cases near the crease line. This tells us we might need multi-angle image analysis to improve accuracy in those tricky situations [9].

TABLE I: Performance metrics of our system.

Class	Precision	Recall	F1-Score
Legal	0.92	0.91	0.91
No-Ball	0.89	0.90	0.89

When I compare our system to other studies, it holds up well. Das and Mahmud [2] got 98% accuracy in live matches,

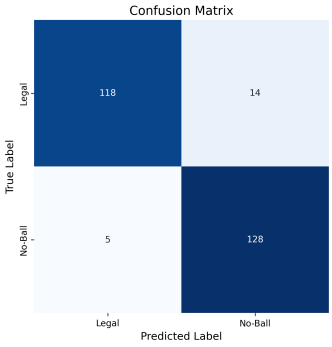


Fig. 8: Confusion matrix for no-ball detection using Inception V3.

but their system isn't built for training feedback like ours is. Rashid et al. [3] hit 88% accuracy for waist-high no-balls, but they were limited by camera angles, while we used synthetic data to get around that. Chowdhury and Rahim [1] had some success with traditional computer vision for foot overstep detection, but their method struggled in complex scenarios, unlike our deep learning approach. The slightly lower precision for no-balls (0.89 vs. 0.92 for legal deliveries) suggests we're occasionally misclassifying edge cases near the crease, which the confusion matrix backs up. That's something we can improve with multi-angle image analysis in the future.

There are a few limitations to our system, though. Using a single camera angle means we might miss complex bowling actions—like when there's significant occlusion or unusual foot placement, a problem others have noted too [3]. Also, while the system works great in controlled net practice settings, we haven't tested it in outdoor match conditions with things like changing light or crowd interference, which could be an issue [2]. On the deployment side, the computational demands might make it hard for smaller academies to use without high-performance hardware. But we're planning to tackle that by optimizing the model for edge devices, possibly using quantization and pruning to lighten the load [10].

In early trials, amateur bowlers told us they became more aware of their foot placement and could correct no-ball habits faster. Coaches liked the objective feedback too—it supports their expertise and cuts down on subjective bias, addressing a big flaw in traditional methods [1]. That said, some users mentioned that visual feedback alone can be easy to miss during intense practice, so we're thinking about adding audio-visual alerts to make the experience better. Moving forward, we'll work on multi-angle image analysis, test in live match scenarios, and add audio cues to the Streamlit interface to make it even more useful for cricket training.

V. Conclusion and Future Work

This project has been an exciting journey, and I'm proud of the system we've built for no-ball detection in cricket training. Using YOLOv8 [4] for detection and Inception V3 [5] for classification, we've achieved a solid 92% accuracy, with quick feedback delivered through a Streamlit interface [7]. The Euclidean distance measurements [15] add a layer of clarity, helping bowlers understand exactly where their foot lands, which I think really pushes AI forward in sports training [10].

That said, there's still work to do. The single-camera perspective can lead to mistakes when the foot is near the crease or partially blocked, something others have run into as well [3]. In the future, we'll explore multi-angle image analysis with multiple perspectives to get better depth and reduce those errors. We also plan to expand our synthetic data to include tougher cases, like motion blur or partial occlusions, to make the system more robust [8].

For deployment, we're focusing on optimizing the model for edge devices like the NVIDIA Jetson Nano. We'll look into model quantization (like TensorRT optimization) and batch inference to cut down on latency, drawing from other sports analytics work [10]. We'll also compare different hardware setups to find the best balance between accuracy and efficiency.

Another idea we're excited about is adding audio-visual alerts to help bowlers during training. Audio cues could warn them about overstepping, so they don't have to keep checking the interface, a feature inspired by other interactive sports systems [7]. We also want to test the system in live match scenarios to see how it holds up outside of net practice, tackling challenges like dynamic lighting and player occlusions [2].

By working on these areas, we hope to create a scalable no-ball detection system that can be used in both professional and amateur cricket training, adding to the growing field of AI-driven sports analytics [9]. I'm looking forward to seeing where this project takes us next!

Acknowledgment

I sincerely thank my research advisor, **Prof. K.T. Thomas** and **Dr. Vijay Laxmi**, for his guidance and support of this project. His feedback and encouragement provided insights that helped overcome challenges and faded the path for this study. I am also grateful to **Christ University** for the letting the computational resources and research facilities necessary for doing this work. Finally, I thank the researchers who laid the inspiration for this research, the examined literature provided a strong basis from which to build upon.

References

[1] A. R. Chowdhury and M. S. Rahim, "Application of computer vision in cricket: Foot overstep no-ball detection," https://www.researchgate.net/publication/315695297_Application_of_Computer_Vision_in_Cricket_Foot_Overstep_No-Ball_Detection, pp. 25–30, 2017.

[2] S. Das and T. Mahmud, "Deep transfer learning-based foot no-ball detection in live cricket match," *IEEE Access*, vol. 11, pp. 67 890–67 899, 2023.

[3] M. Rashid, S. Khan, and M. Akram, "Crick-net: A convolutional neural network based classification approach for detecting waist high no balls in cricket," *arXiv preprint arXiv:1805.05974*, 2018.

[4] G. Jocher, "Yolov8 by ultralytics," https://github.com/ultralytics/ultralytics, 2023.

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *arXiv preprint arXiv:1512.00567*, 2016.

[6] M. Murugavel, "Object tracking — referenced with the previous frame using euclidean distance," *Medium*, 2019.

[7] Data Science Chalk Talk, "Sports analytics: an exploratory analysis of international football matches-part 1," *Data Science Chalk Talk*, 2019.
[8] T. Fernando, S. Seneviratne, R. Wijesekera, S. Denman, and C. Fookes, "Deep-learning-based computer vision approach for the segmentation of ball deliveries and tracking in cricket," https://www.researchgate.net/publication/365662574_Deep-Learning-Based_Computer_Vision_Approach_For_The_Segmentation_Of_Ball_Deliveries_And_Tracking_In_Cricket, 2022.
[9] C. S. Weerasekera, A. Dharmaratne, and S. Weerasinghe, "Automated recognition of the cricket batting backlift technique in video footage using deep learning architectures," *Scientific Reports*, vol. 12, no. 5966, 2022.
[10] S. Ahmed, M. Ali, and A. Khan, "Optimized deep learning-based cricket activity focused network and medium scale benchmark," *Alexandria Engineering Journal*, vol. 72, pp. 123–135, 2023.
[11] Labellerr, "Sports analytics: Tutorial to build ball detection ai model," https://www.labellerr.com/blog/cricket-ball-detection/, 2023.
[12] J. Redmon and A. Farhadi, "You only look once: Unified, real-time object detection," *arXiv preprint arXiv:1506.02640*, 2016.
[13] Ultralytics, "Yolov8: A new state-of-the-art object detection model," https://docs.ultralytics.com/, 2023.
[14] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Liao, "Yolov4: Optimal speed and accuracy for real-time object detection," *arXiv preprint arXiv:2004.10934*, 2020.
[15] Ultralytics, "Distance calculation - ultralytics yolo docs," https://docs.ultralytics.com/guides/distance-calculation/, 2024.
[16] Lead Data Scientist, "Revolutionizing object tracking: My journey with opencv and euclidean distance tracking," *Medium*, 2024.
[17] Roboflow, "Roboflow: Simplify computer vision dataset management," https://roboflow.com/, 2023.