Proposed Methodology

Problem definition

Anterior Cruciate Ligament (ACL) injuries are severe and common among athletes, often requiring extensive treatment and rehabilitation. Timely and accurate prediction of ACL injury risk from images can significantly aid in preventive measures and improve outcomes for athletes. This project aims to develop a deep learning-based model that can predict the risk of ACL injury from images. Utilizing a pre-trained VGG16 model, the project leverages transfer learning to enhance the accuracy and efficiency of injury detection. The model will classify images into two categories: 'injury' and 'no injury', providing a probability score indicating the risk level. The ultimate goal is to create a user-friendly interface using Streamlit, where users can upload images and receive an immediate assessment of ACL injury risk. This tool can be invaluable for athletes, coaches, and medical professionals in making informed decisions and implementing appropriate preventive strategies.

Data Collection and Preprocessing

Data Collection

1. **Image collection and Data preparation**

Sources: Images were sourced from publicly available sports datasets, repositories, and sports event footage. This ensured a diverse dataset, capturing various contexts and environments where ACL injuries might occur. Both professional and amateur sports environments were included to cover a wide range of scenarios.

Categories: The images were categorized into two primary classes: 'injury' and 'no injury'. The 'injury' category included images where athletes sustained ACL injuries, either during matches or practice sessions. The 'no injury' category consisted of images from similar contexts where no injuries occurred. This binary classification helped in simplifying the problem and focusing on distinguishing between injury and no injury cases.

Manual Labeling: Each image was meticulously labeled by a team of experts. These experts included analysts who could accurately identify signs of ACL injuries. The labeling process was guided by kabaddi sports expert opinions. All the labeling process and data collection was handled by sports experts and experienced players

Some examples of dataset(JPG formate)

Figure 1

Figure 2

Data Preprocessing

1. **Loading the Images**

Directory Structure: The images were organized into directories based on their labels ('injury' and 'no injury'). This hierarchical structure facilitated easy access and management of the dataset, ensuring that images were properly segregated according to their classes.

File Reading: Images were read from these directories using appropriate libraries, ensuring that the data was correctly loaded into the system for further processing.

2. **Splitting the Dataset**

Training and Validation Split: The dataset was split into training and validation sets. Typically, 80% of the data was used for training the model, while 20% was reserved for validation. This split was done randomly but with a fixed seed to ensure reproducibility. The training set was used to teach the model, and the validation set was used to evaluate its performance on unseen data, helping to monitor for overfitting.

3. **Image Augmentation**

Image augmentation was employed to artificially enlarge the training set and improve the model's generalization capabilities. This process involves applying random transformations to the training images, such as rotations, translations, and flips, to create new, slightly altered versions of the images. Common augmentation techniques included rescaling pixel values, applying shear transformations, zooming in and out, and flipping images horizontally. These transformations ensured that the model could handle variations in the input data and was not overly reliant on specific features.

4. **Normalization and Data Augmentation for Training**

Pixel Value Scaling: All image pixel values were normalized to a range of 0-1. This was done by dividing the pixel values by 255.0, the maximum value for a pixel in an 8-bit image. Normalization ensures that the model's input values are consistent, aiding in faster and more stable convergence during training. During model training, real-time augmentation was performed to continually provide the model with varied training images. This approach ensured that each training epoch received a slightly different set of images, improving the model's robustness and ability to generalize to new, unseen data. This continuous variation helped prevent overfitting and made the model more adaptable to different image contexts.

Dataset example after preprocessing (0 is injury and 1 is no_injury)

Figure 3. Dataset Example

Model Selection and Architecture

Model Selection

1. Transfer Learning

Rationale: Transfer learning was chosen for this project due to its ability to leverage pre-trained models that have already learned rich feature representations from large datasets like ImageNet. This approach helps in achieving better performance with limited computational resources and smaller datasets, which is often the case in specialized fields like sports injury prediction. VGG16, a widely used convolutional neural network (CNN), was selected for its proven effectiveness in various image classification tasks. VGG16 is known for its simplicity and deep architecture, which helps in learning intricate patterns in the images.

Model Architecture

1. Base Model - VGG16:

Layers: VGG16 consists of 16 weight layers, including 13 convolutional layers followed by 3 fully connected layers. The convolutional layers use small 3x3 filters, which are convolved with the input image to detect low-level features like edges and textures. These features are then passed through

deeper layers to detect higher-level patterns. Model was initialized with weights pre-trained on the ImageNet dataset, providing a robust starting point that captures a wide variety of visual features.

Freezing Layers: To retain the learned features, the initial layers of the base model were frozen. Specifically, the first 15 layers were set to non-trainable, meaning their weights would not be updated during training. This preserves the general visual features learned from the large-scale ImageNet dataset, allowing the model to focus on learning task-specific features in the subsequent layers.

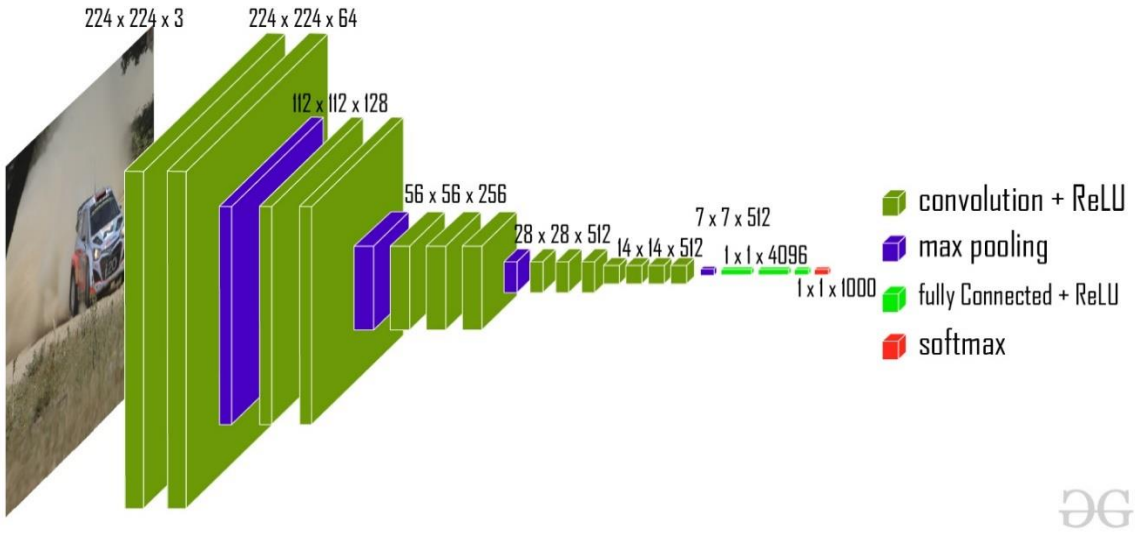

Figure 4. ***VGG16 Architecture***

2. Custom Layers

Global Average Pooling: Instead of using fully connected layers immediately after the convolutional layers, Global Average Pooling (GAP) was added. GAP reduces each feature map to a single value by taking the average, which significantly reduces the number of parameters and helps prevent overfitting. It effectively summarizes the spatial information of each feature map into a single number, making the model more robust to spatial translations of the input image.

Dense Layer: A dense layer with 1024 neurons and ReLU activation was added after the GAP layer. This layer introduces non-linearity and helps in learning complex patterns that are specific to the ACL injury prediction task.

Output Layer: The final layer is a dense layer with a single neuron and sigmoid activation, providing a probability score for the binary classification task (injury vs. no injury). The sigmoid activation ensures that the output is between 0 and 1, representing the probability of injury

3. Model Compilation, Model Training and Training Strategy

Binary cross-entropy was used as the loss function, appropriate for binary classification tasks. It measures the difference between the predicted probability and the actual label (0 or 1). The Adam optimizer was chosen for its adaptive learning rate capabilities and efficient handling of sparse gradients. Initially, a lower learning rate of 0.0001 was set to ensure fine-tuning of the custom layers

without significantly altering the pre-trained weights. Accuracy was used as the primary metric to evaluate the model's performance during training and validation.

The model was trained using the augmented training dataset, which helps in learning a wide variety of patterns and reduces overfitting. The validation dataset was used to monitor the model's performance and adjust hyperparameters accordingly.

An early stopping callback was implemented to monitor the validation loss. If the validation loss did not improve for 3 consecutive epochs, training would be stopped, and the best model weights would be restored. This prevents overfitting and ensures the model generalizes well to new data. The model was trained for up to 15 epochs, with early stopping likely terminating the training process earlier if no further improvement was observed in the validation loss.

Model Summery

Layer (type)	Output Shape	Param #
input_layer_4 (Input Layer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_average_pooling2d_4 (GlobalAveragePooling2D)	(None, 512)	0
dense_8 (Dense)	(None, 1024)	525,312
dense_9 (Dense)	(None, 1)	1,025

Total params: 30,452,549 (116.17 MB)
Trainable params: 7,605,761 (29.01 MB)
Non-trainable params: 7,635,264 (29.13 MB)
Optimizer params: 15,211,524 (58.03 MB)

Training and Validation Accuracy Graph

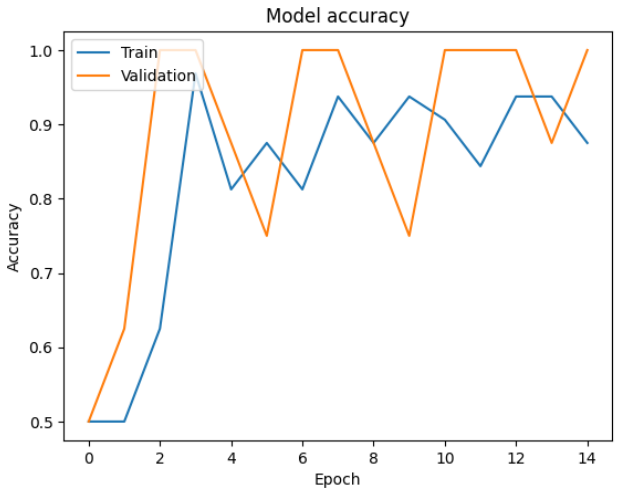

Figure 5. Model Accuracy

Training and Validation Loss Graph

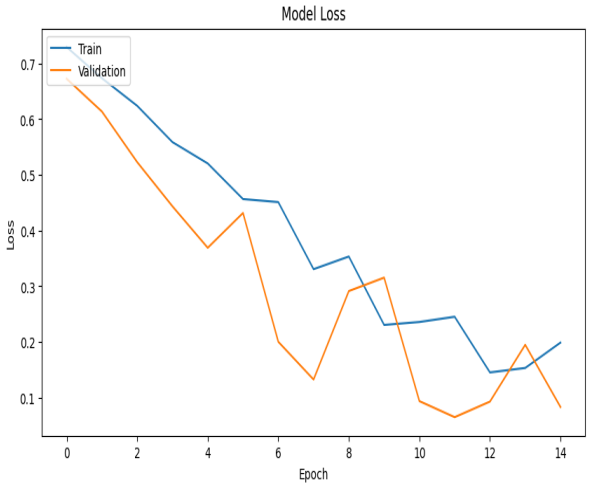

Figure 6. Model Loss

Confusion Matrix of the Model

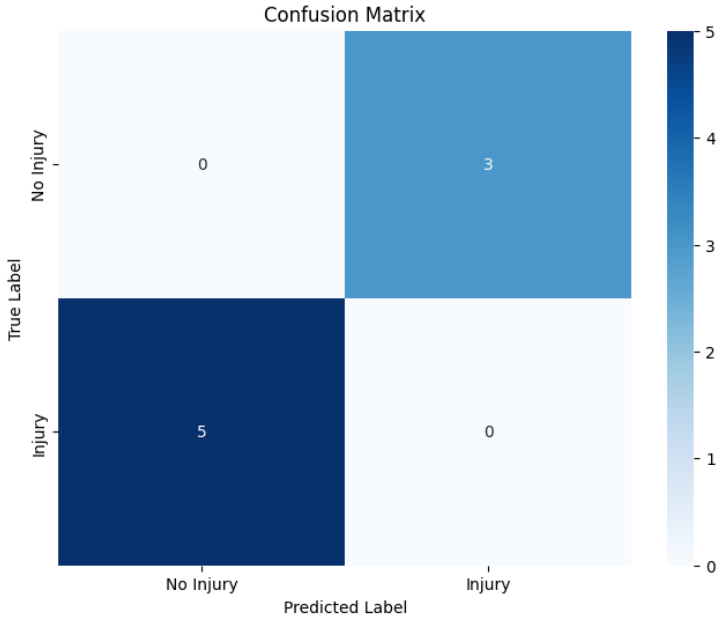

Figure 7. Confusion Matrix

Model Prediction and User Interface

Model Prediction

The prediction phase of our model involves several systematic steps that ensure accuracy and reliability. Initially, each input image is preprocessed to match the expected dimensions of the model, which is crucial for maintaining consistency. The images are resized to 224x224 pixels and normalized to scale the pixel values between 0 and 1. This preprocessing step is fundamental as it prepares the data for the neural network, ensuring optimal performance. The preprocessed images are then fed into the trained VGG16-based model. The model, equipped with a dense network architecture tailored for binary classification, outputs a probability score indicating the likelihood of injury. A sigmoid activation function at the final layer of the model ensures the output is a probability between 0 and 1. For classification, a threshold of 0.5 is applied: a score above this threshold indicates a prediction of 'injury,' while a score below it indicates 'no injury.' This approach balances sensitivity and specificity, making it adaptable for varying clinical needs. Visualization techniques, such as displaying images with their predicted labels and probabilities, alongside performance metrics like the confusion matrix and ROC curves, provide a comprehensive evaluation of the model's effectiveness.

User Interface

The user interface (UI) is designed to be intuitive and user-friendly, facilitating easy interaction for users with varying levels of technical expertise. Developed using Streamlit, a powerful framework for building interactive web applications, the UI allows users to upload images in JPEG, PNG, or JPG formats. Once an image is uploaded, it is displayed on the interface, providing immediate visual feedback to the user. The model then processes the image and generates a prediction, which is presented on the same interface. The predicted probability of injury is clearly displayed, expressed as a percentage to enhance interpretability. This percentage indicates the risk level, with higher values signifying greater injury risk. The UI ensures that users can quickly understand the results without delving into complex technical details. Additionally, the interface can display both the original and predicted labels for each image, aiding in visual verification and enhancing the transparency of the prediction process. By integrating these features, the UI not only simplifies the prediction process but also enhances the overall user experience, making it accessible for both clinical practitioners and researchers. This streamlined interaction is crucial for real-world applications, where quick and accurate assessments are essential for timely interventions.

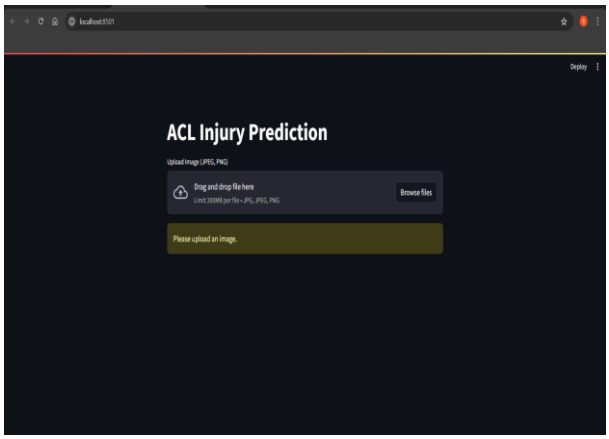

Figure 8. UI Output

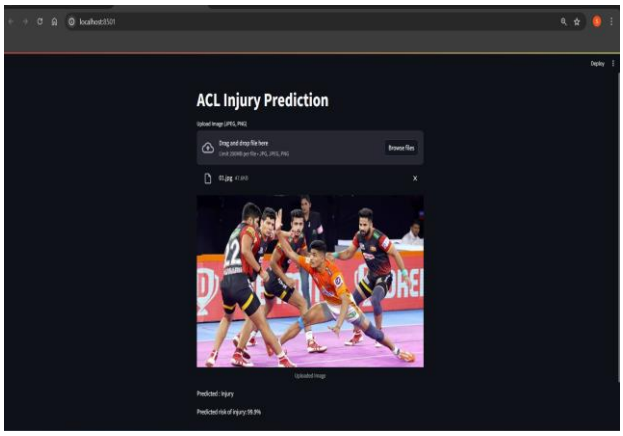

Figure 9. UI Output

Prediction of Testing Dataset with Labels

Figure 20. Prediction

Figure 11. Prediction

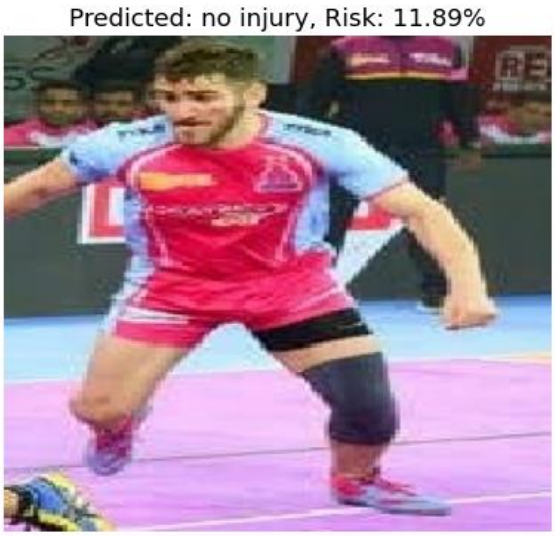

Figure 12. Prediction

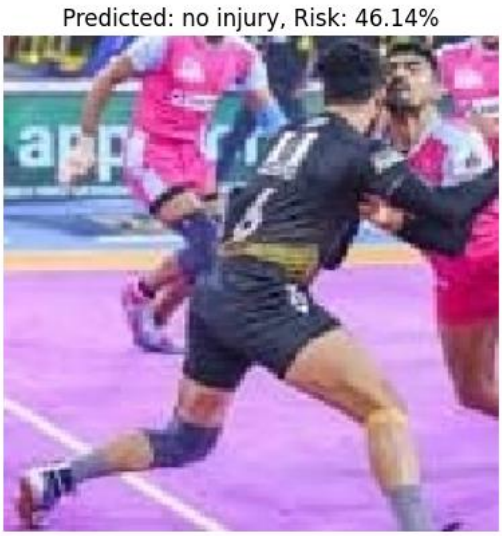

Figure 13. Prediction

For model code [click here]