



# **INTELLIGENT RADIOLOGIST ASSISTANT**

**An optimized web application designed specifically for  
classifying tears in knee MRI images.**

Presented by Arup Sankar Roy

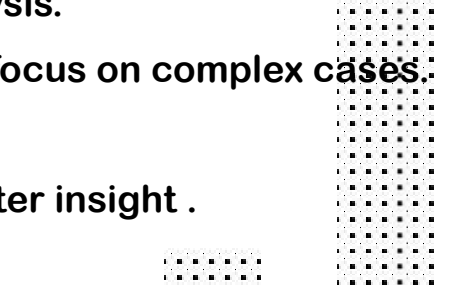
Date :21-05-2024



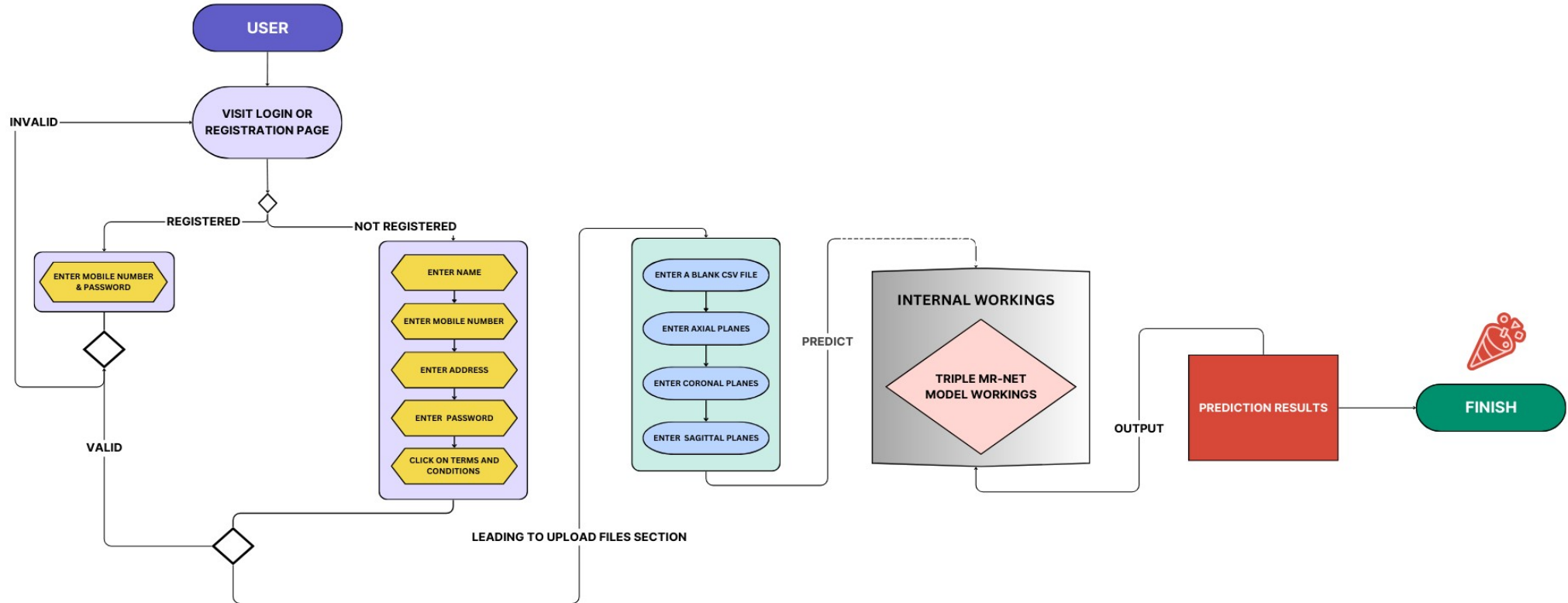
## Objective

To develop an automated web application that interprets knee MRI scans using deep learning, enhancing diagnostic accuracy and efficiency for radiologists.

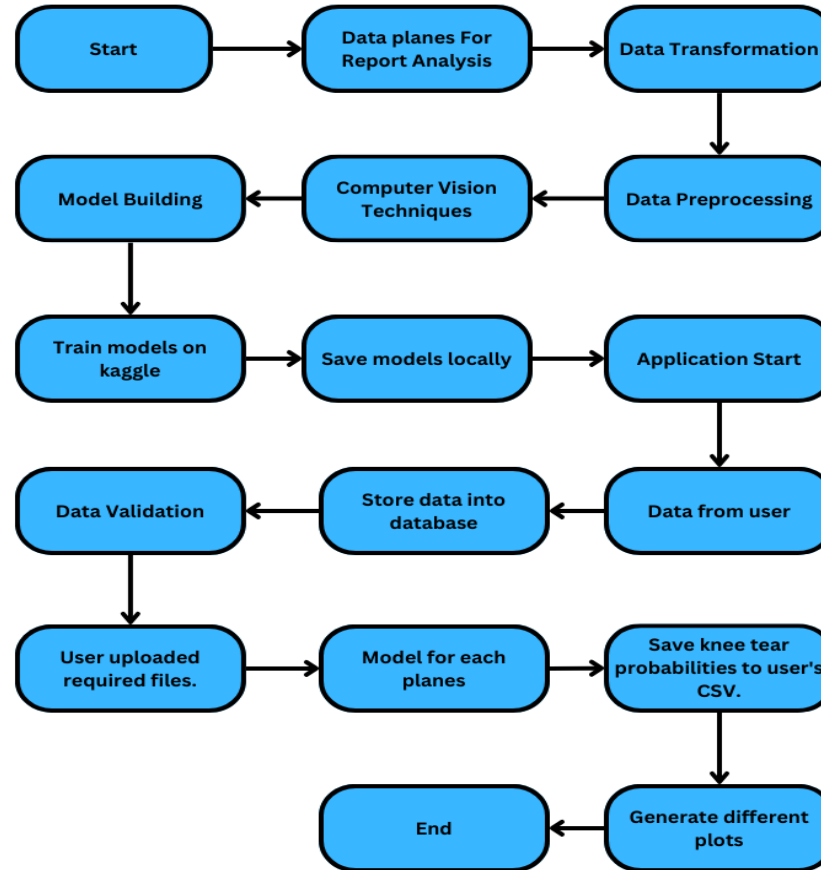
## Benefits

- **Increased Accuracy:** Reduces diagnostic errors by providing reliable MRI analysis.
  - **Time Efficiency:** Speeds up the interpretation process, allowing radiologists to focus on complex cases.
  - **User-Friendly:** Easy-to-use interface suitable for healthcare professionals.
  - **Data Visualization:** Offers detailed visual reports with charts and graphs for better insight .
  - **Real-Time Analysis:** Enables quick analysis and reporting of MRI scans.
- 

# WORKFLOW DIAGRAM



# Architecture



# User basic information Insertion in Database

- The web application uses MySQL to manage user data, facilitating secure registration and login processes. On the registration page, users input their name, mobile number, address, and password. The system validates the inputs, hashes the password, and stores the data in the database. For login, users provide their mobile number and password, which are validated against the stored credentials. Upon successful authentication, users can proceed to the next steps. Proper error handling and session management ensure a smooth and secure user experience.

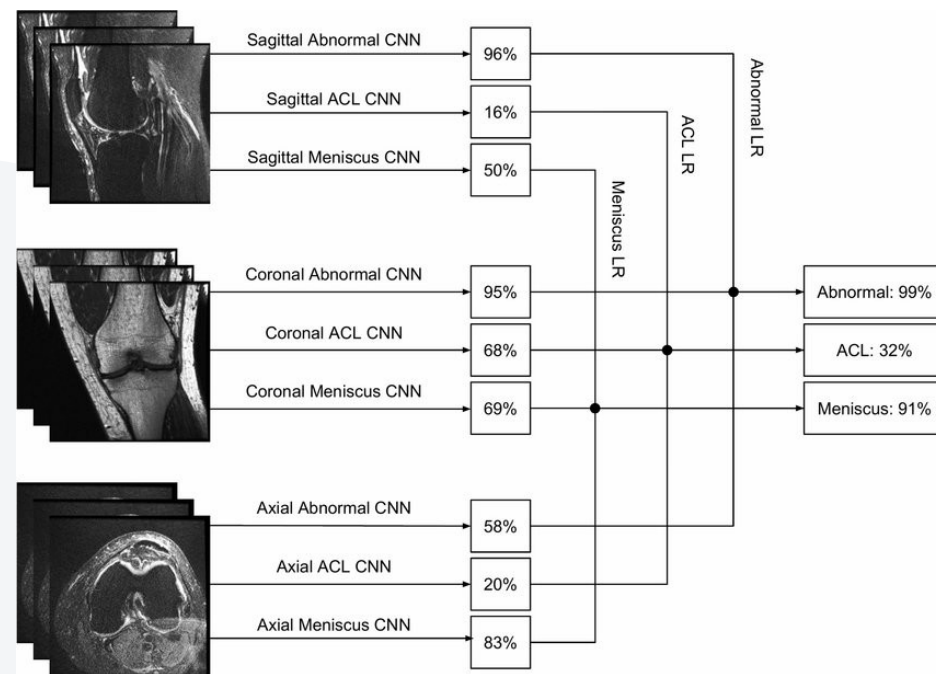
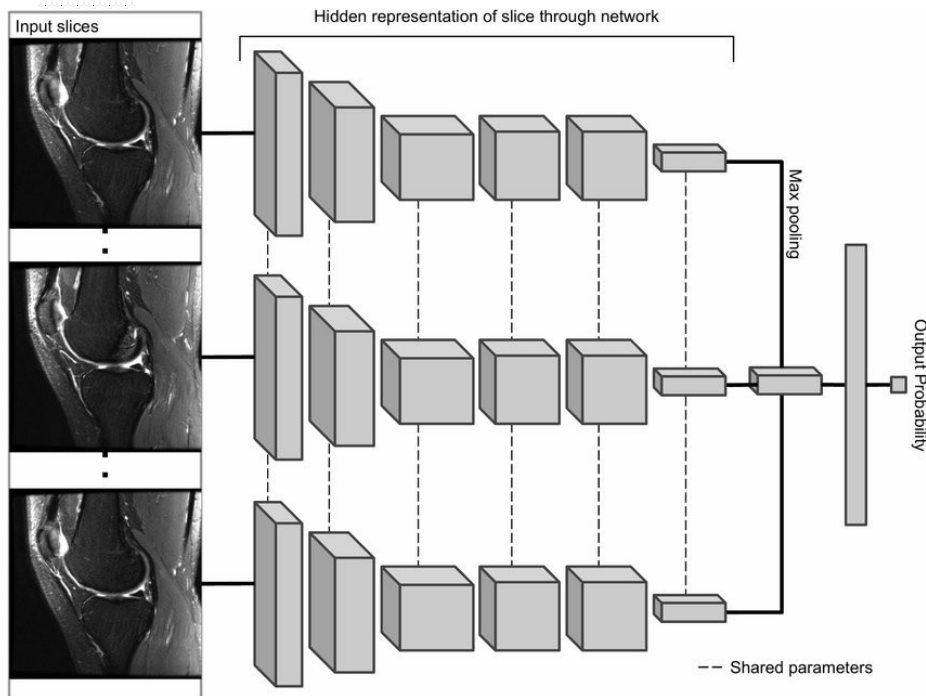
# MRI DATA PREPROCESSING FOR BUILDING MODEL

- The MRI dataset is sourced from Kaggle, ensuring access to diverse and high-quality data for analysis.
- Creating a “Dataset” Class to process the data.

## Data Preprocessing Steps :

- **Load MRI Planes:**  
Load axial, sagittal, and coronal MRI planes from `.npy` files.
- **Crop Volumes:**  
Compute the padding needed to center crop the volumes to the specified input dimensions.  
Apply the crop to each MRI plane.
- **Normalization:**  
Normalize each MRI plane by subtracting the minimum pixel value and dividing by the range (max - min) to scale pixel values between 0 and the maximum pixel value.
- **Standardization:**  
Standardize the normalized MRI planes by subtracting the mean and dividing by the standard deviation.
- **Channel Replication:**  
Stack each single-channel MRI plane three times along the channel dimension to simulate RGB channels (required by most pre-trained models).
- **Conversion to Tensors:**  
Convert the preprocessed MRI planes to PyTorch tensors for model input.
- **Label Conversion:**  
Convert the labels to PyTorch tensors for training.

# Model Architecture





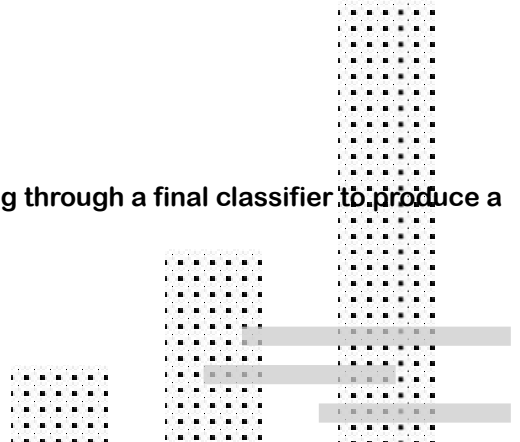
# Model Description

The primary model for classifying knee tears, known as TripleMRNet, is a fusion of three distinct MRNet models.

## MRNet:

- A single MRI plane classifier based on a modified AlexNet architecture.
- Includes a global average pooling layer and a custom linear classifier to output the final diagnosis.

## TripleMRNet:

- Extends the MRNet to handle three MRI planes (axial, sagittal, coronal) simultaneously.
  - Uses separate branches for each plane, sharing a common backbone (AlexNet or ResNet18).
  - Combines the features from each plane using adaptive average pooling and concatenation before passing through a final classifier to produce a diagnosis.
- 




# Model Training

- **Setup:**
  - Initialize directories for model checkpoints.
  - Set the device for training (CPU or GPU).
- **Data Loading:**
  - Load training and validation data.
- **Model Initialization:**
  - Instantiate TripleMRNet model with specified backbone architecture.
  - Move the model to the selected device.
- **Pre-trained Model Loading (Optional):**
  - Load the latest model checkpoint to resume training, if available.
- **Optimizer and Scheduler Initialization:**
  - Set up Adam optimizer with specified learning rate and weight decay.
  - Initialize learning rate scheduler to adjust learning rate dynamically.
- **Training Loop:**
  - Iterate through specified number of epochs.
  - Compute training loss and AUC score.
  - Evaluate model on validation dataset, calculate validation loss and AUC score.
  - Update learning rate based on validation loss.
  - Save model checkpoint if validation loss improves.
- **Execution:**
  - Call the train function with optional arguments (e.g., GPU usage) to begin training.




# Upload Scans

- **Functionality:** Allows users to upload knee MRI scans for analysis and diagnosis.
  - **Interface:** Provides a user-friendly form for scan upload with validation checks.
  - **Processing:** Scans are securely transmitted to the server and analyzed using the TripleMRNet model.
  - **Feedback:** Users receive analysis results, including diagnosis or probability scores for knee conditions.
- 



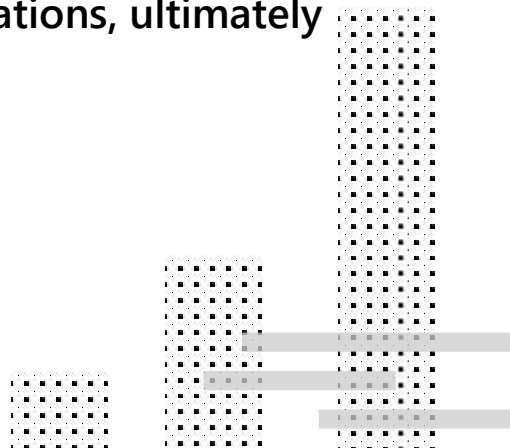
# Prediction Report

- **Functionality:** Displays analysis results and diagnostic reports based on uploaded MRI scans.
  - **Content:** Includes predicted knee condition, probability scores, and visualizations (e.g., pie plot, bar plot, stacked bar plot, line plot..etc).
  - **Interactivity:** Users can interact with the report, zoom in on visualizations, and download the report for reference.
  - **Integration:** Linked with the upload scans feature, allowing seamless transition from scan upload to report viewing.
- 



# Conclusion

In conclusion, the web application "Intelligent Radiologist Assistant" offers a comprehensive solution for knee MRI analysis and diagnosis. By leveraging advanced deep learning models, users can accurately assess knee conditions, prioritize high-risk cases, and streamline diagnostic workflows. With user-friendly features, seamless integration, and robust security measures, the application enhances the efficiency and accuracy of radiological interpretations, ultimately improving patient care in clinical settings.



**Thank you**

