University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

ACL Injury Detection and Classification utilizing Magnetic Resonance Imaging Scans and Deep Learning Techniques

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Karlos Lorenzo S. Tuazon

July 2023

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "ACL Injury Detection and Classification utilizing Magnetic Resonance Imaging Scans and Deep Learning Techniques" prepared and submitted by Karlos Lorenzo S. Tuazon in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Vincent Peter C. Magboo, M.D., M.Sc.
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Avegail D. Carpio, M.Sc.		
2. Richard Bryann L. Chua, Ph.D. (*cand.*)		
3. Perlita E. Gasmen, M.Sc. (*cand.*)		
4. Ma. Sheila A. Magboo, Ph.D. (*cand.*)		
5. Marbert John C. Marasigan, M.Sc. (*cand.*)		
6. Geoffrey A. Solano, Ph.D.		

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Vio Jianu C. Mojica, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

Marie Josephine M. De Luna, Ph.D.
Chair
Department of Physical Sciences and Mathematics

Maria Constancia O. Carrillo, Ph.D.
Dean
College of Arts and Sciences

Abstract

One of the most common injuries incurred through physical activity is the anterior cruciate ligament tear or ACL tear. ACL injuries tend to have a low self-recovery rate and as such, they usually require surgery in order to reconstruct or repair the torn ligament. The study aims to improve upon the already established methods when trying to diagnose and classify potential ACL injuries through the use of convolutional neural networks (CNNs) and transfer learning. Different parameters are tested to find the optimum and best-performing model based on specific performance metrics. A web-based decision support tool for assessing and diagnosing ACL injury utilizing knee MRIs integrating the best performing CNN model is developed serving as a valuable decision support tool in different healthcare applications.

Keywords: anterior cruciate ligament, convolutional neural networks, transfer learning, decision support tool

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

Sports and athletics are recognized to be penchant activities among individuals. These activities have tremendous physical and mental health benefits such as improving cardiovascular endurance, weight management, emotion management, and improved sleeping patterns to name a few.

However, a certain risk accompanies engaging in physical activities and sports which are injuries. Injuries are not only a physical burden caused by pain and disability but also a mental and economic burden as well. The individual might exhibit adverse psychological responses such as fear of reinjury, lack of confidence, devastation, and restlessness [5]. In addition, monetary costs for tests, medication, and rehabilitation fees cannot be avoided since treatment is of the essence [6].

One of the most common injuries incurred through physical activity is the anterior cruciate ligament tear or ACL tear. The anterior cruciate ligament is responsible for stabilizing the knee, connecting the femur and tibia. It is part of the four primary ligaments of the knee which are: the anterior cruciate ligament, posterior cruciate ligament, lateral collateral ligament, and medial collateral ligament [7]. ACL injuries tend to have a low self-recovery rate and as such, they usually require surgery in order to reconstruct or repair the torn ligament in order to prevent any additional complications such as osteoarthritis (OA). This injury is common among athletes and active individuals and poses a great burden physically, mentally, and economically for rehabilitation and surgery is an inevitable requirement.

The detection of these injuries is usually done through two methods: clinical physical exams and the use of magnetic resonance imaging (MRI). Clinical physical exams are typically performed by an expert physician that is adept and well-versed in dealing with these potential injuries. The MRI on the other hand is usually used to confirm knee-joint injuries. Specific shapes and characteristics are

visually assessed by a musculoskeletal - trained radiologist in order to determine if there is damage in the ligament [8].

B. Statement of the Problem

The study aims to improve upon the already established methods when trying to diagnose and classify potential ACL injuries. The diagnosis is made through consulting with an expert physician or by getting a reading of an MRI scan from a trained radiologist, and although these detection methods are widely established, they require individuals with particular expertise in order to accurately assess the injury. Misdiagnosis may occur if inexperienced physicians or radiologists perform the tests and assess the readings which may lead to consequences such as prolonged and further injury. As such, through the proposed use of a convolutional neural network (CNN) and transfer learning, the detection and classification of ACL injuries could provide additional decision-making capabilities in assessing if an individual is suffering from ACL damage or injury. The research questions that guide the objectives are as follows:

1. How well would the proposed CNN diagnose and classify ACL injuries using MRI scans?

2. Which CNN models would have the best performance in ACL injury detection and classification?

3. What would be the effects of data augmentation techniques on the performance of the proposed CNN?

C. Objectives of the Study

The use of machine learning and deep learning in the medical field is already of great prevalence. Some specific applications are early disease prediction, rehabilitation, and disease diagnosis to name a few. In line with this, the study's main objective is to employ a CNN model and transfer learning to detect and classify

if an individual has an ACL injury through MRI scans. Four pre-trained CNN models are used based on related literature, and the method of transfer learning is employed. The CNN models are compared to each other, testing different parameters to find the optimum and the best-performing model based on specific performance metrics. Specifically, the objectives of the study are:

1. Compare the performance of the following pre-trained CNN models:
    (a) ResNet-18
    (b) VGG-16
    (c) InceptionV3
    (d) AlexNet
2. Determine the optimum parameters for the CNN model based on:
    (a) Input Size (75 x 75, 256 x 256)
    (b) Epochs (30, 50, 100)
    (c) Batch Size (8, 16, 32)
3. Determine the effects of random geometric data augmentation techniques on the performance of the proposed CNN model such as:
    (a) Horizontal Flipping
    (b) Shifting
    (c) Scaling
    (d) Rotation (0° - 360°)
    (e) Translation
4. Use specific metrics to quantitatively determine the performance of each model such as:
    (a) Confusion Matrix

(b) Precision

(c) Sensitivity

(d) F1-Score

(e) Specificity

(f) Accuracy

(g) AUC

(h) Negative Predictive Value (NPV)

Furthermore, a web-based decision-support tool for assessing and diagnosing ACL injury utilizing knee MRIs is developed using the best-performing CNN model. The functionalities of the system are:

1. Allow a user to:

   (a) Upload an MRI scan of the knee which includes an ACL.

   (b) View the results of the ACL diagnosis if it is: healthy, partially torn, or fully torn.

   (c) Save the results in a PDF file with the uploaded image produced by the system.

D. Significance of the Project

The findings of the study aim to improve upon the current methods used by physical therapists, physicians, and radiology technicians in assessing knee-joint injuries, specifically ACL injuries. Since the current methods have limitations in terms of expertise and ability of the assessor, the findings of the study would be especially beneficial for medical practitioners, specifically those who specialize in the practice of orthopedics. The proposed framework could address the problem of a lack of experts in this field causing inexperienced physicians to perform the assessments which could lead to misdiagnosis. In addition, health institutions and

hospitals that these medical practitioners and experts work in gain an advantage from the proposed study as well. Consequentially, the direct beneficiaries of the findings of the study would of course be the patients that are being diagnosed with potential ACL injuries. The application of the proposed model entails less risk of getting a misdiagnosis, prolonging the rate of healing and potentially making the injury more severe or causing new injuries. The delay caused by waiting for a diagnosis from expert physicians and radiologists is also lessened and mitigated, ensuring the immediate start of treatment prior to the injury assessment.

E. Scope and Limitations

The study focuses on the use and application of CNNs in order to effectively detect and classify ACL injuries based on MRI scans. The main dataset that would be used would mainly consist of different MRI scans from varying knee-joint injuries. These could include scans of healthy ligaments, sprained ligaments, partially torn ligaments, or fully torn ligaments. Since the acquisition of MRI scans requires ethical permissions and the use of an MRI machine is not accessible and costly, the dataset to be used would be from a pre-collected open-source resource, specifically the KneeMRI dataset from the Clinical Hospital Centre Rijeka, Croatia. The use of this dataset would, however, limit the specificity of the background of the subjects being studied and there would be no control over the geographical region the study would encompass. The pre-trained CNN models used for transfer learning are only ResNet-18, VGG-16, InceptionV3, and AlexNet. The optimum parameters tested are input size (75x75, 256x256), epoch size (30, 50, 100), and batch size (8, 16, 32). The random augmentation techniques tested are only horizontal flipping, shifting, scaling, rotation, and translation. In addition, time constraints pose a potential constraint in the overall findings of the study.

F. Assumptions

1. The web application's user has received the necessary training to properly understand the results of the model.

2. The system will only be utilized as a decision support tool for the detection and classification of anterior cruciate ligament (ACL) injuries.

3. The file uploaded would be an image file of an MRI scan of a knee that has a visible ACL.

II. Review of Related Literature

Anterior cruciate ligament (ACL) injuries are clinically common and are seen especially in individuals that participate in competitive or leisure sports. It has been estimated that 70% of ACL injuries occur due to non-contact circumstances. Numerous factors play a role in these incidences, some being extrinsic, intrinsic, and sometimes out of our control, and as such, there is always an inherent risk [9]. Traditional diagnostic techniques such as arthroscopy are utilized to detect ACL injury. These techniques, however, are highly sensitive and intrusive. As such, magnetic resonance imaging (MRI) which offers high spatial resolution, clearly displaying the overall structure of the knee joint, is mostly preferred by patients. It is also capable of offering high sensitivity and accuracy rates reaching 90% compared to arthroscopic detection in diagnosing complete and partial ACL tears [10].

Numerous studies have been conducted relating to the utilization of machine learning and deep learning techniques to detect and classify patterns using MRI scans in the medical field. Machine learning and deep learning techniques, though quite similar, have key differences. Both techniques have the ability to self-learn through repetitions and perform predictions without extensive explicit programming, however, deep learning, which is a subset of machine learning, differs in the amount of human intervention required to improve outcomes. Typical machine learning uses individual experts to select the image features that seem to best describe the visual input, while deep learning does not require feature selection. As they have been modeled after the human brain, deep learning models consist of artificial neural networks capable of increasingly complex tasks compared to machine learning models. Multiple computational units known as artificial neurons make up artificial neural networks, and they receive input signals multiplied by the synapses' strength termed weights. Artificial neural networks and convolutional neural networks (CNNs) are quite similar, with the explicit presumption that the inputs are images. This presumption enables us to incorporate specific characteristics into the CNN architecture and as such, CNNs are predominantly used in

different computer vision tasks such as classification, detection, and segmentation [11].

CNNs are considered to be excellent feature extractors and using them to classify medical images can avoid complicated and costly feature engineering. It is vastly used in the medical field and has numerous applications. A study by [12] utilized the use of a CNN model for the early diagnosis of Alzheimer's Disease (AD) using MRI images. The model they created achieved an accuracy of 99%, utilizing 25 epochs which they found to produce the highest result. This result is significantly higher compared to other techniques and models that they compared which also utilized MRI scans. A study by [13] used R-CNN or region-based convolutional neural networks on brain MRI images to detect and classify regions containing tumors. It produced an accuracy result of 91.66% which was found to be higher compared to the literature they compared it with which used the same dataset. It has also been found that some studies utilizing CNNs in medical image classification achieved similar or even better performances when compared to their human counterparts. CheXNet which is a CNN with 121 layers that were trained on a dataset of more than 100,000 frontal-view chest X-rays outperformed four radiologists on average [14]. A diagnostic tool for screening patients with common treatable binding retinal diseases based on deep learning approaches and transfer learning systems was also compared with the performance of human experts. The trained model's ROC curve was used to map the sensitivities and specificities of the experts, and it was found that the differences in diagnostic performance between the model and the experts were statistically similar within a 95% confidence interval [15].

As we have discussed, the use of deep learning techniques and CNNs is predominant in the field of computer vision tasks involving medical images given its inherent architectural characteristics fitted for image inputs. This trend is also observed in the related studies we will discuss specifically similar to our objective which is ACL injury detection and/or classification. A study I found however that

utilized traditional machine learning techniques, instead of CNNs was performed by [16]. Support Vector Machines (SVM) and Random Forests (RF) were used on sagittal plane MRI scans in semi-automatic detection of ACL injuries, detecting healthy, injured, or completely ruptured cases. Manually extracted regions of interest (ROI) were implemented in the methodology of the study which is why the researchers deemed the detection methods to be semi-automated. Since traditional machine learning techniques were used and the task involves images, feature extraction was a requirement. The image volumes cannot be directly handled by the models and as such histogram of oriented gradient (HOG) feature descriptors and scene spatial envelope descriptors were used to preprocess and extract smaller amounts of usable features from the data. The results suggest that it effectively differentiates complete ACL tears from other cases with an area under the receiver operating characteristic curve (AUC) of 0.943. In the cases of differentiating non-injured from other cases, however, a decrease was seen having an AUC of 0.894 which could be caused by the difficulty to separate many of the partially injured patients from the non-injured ones in the available data.

Moving on, the majority of studies addressing the same problem utilized deep learning techniques and CNNs. The study by [17] utilized three CNN architectures to differentiate the performances of varying input field-of-views (full-slice, cropped slice, dynamic patch-based sampling) and dimensionality (single slices, three slices, five slices) in detecting between normal ACLs and complete ACL tears. The coronal imaging plane of the knee was used to train and test the network. It was found that images cropped to the area of the cruciate ligament and the utilization of five-slice networks yielded the highest accuracy. Based on the results, two important findings were found. Firstly, cropping the image limits the input field of view and effectively reduces the image search space improving the algorithm's performance. Secondly, the use of 3D data as input is vital especially for ACL information because there are 3D oblique orientations of the ligament fibers that are critical in making a diagnosis. Another study that utilized the

coronal plane of the MRI images was also conducted by [18]. A smaller sample size was however used having 120 samples each for normal, partially torn, or completely torn cases. The accuracy produced by the CNN in their study was 94.7%. Another study on the detection of ACL injury was conducted by [8] and introduced the utilization of arthroscopic findings as a reference standard. They employed a CNN architecture based on the 3D DenseNet. Like the study by [17], 3D and cropped images were used for evaluation and analysis based on the same reasons. They did, however, use a different imaging view, utilizing the sagittal plane. It was found that the proposed model that took combined inputs consisting of cropped images and cropped images with an ROI of the outlined ligament obtained the highest accuracy at 0.957. This was compared with the performance of inexperienced and senior radiologists which had an accuracy of 0.814 and 0.899 respectively, proving again to be higher than their human counterparts. Lastly, the study by [7] which is the latest study in this field, introduced balancing and real-time data augmentation in the pre-processing process. A hybrid approach was employed, performing under-sampling, and over-sampling on the dataset to address the inherent imbalance. From 2,315 images of healthy tears, 580 images of partial tears, and 186 images of fully ruptured tears the dataset consisted of 1,487, 1027, and 1,238 images of healthy, partial, and fully ruptured ACL images respectively after balancing. In terms of dealing with the scarcity of the dataset, real-time data augmentation was implemented, generating different images after each epoch run. The researchers used a custom Residual Network of 14 layers (ResNet-14) as their CNN architecture and garnered an overall accuracy result of 0.90 and AUCs of 0.98, 0.97, and 0.99 for healthy, partial tears, and fully torn cases respectively. It could be inferred that hybrid class balancing and data augmentation immensely helped improve the results.

It is apparent that the use of CNNs is prevalent in the domain of our problem. A specific approach in the utilization of CNNs however would be explored which is the use of transfer learning. Large, labeled datasets are often needed for training

CNNs to achieve a specific level of classification accuracy. In order to eliminate the issue of needing large datasets, transfer learning is employed which reuses pre-trained CNN models which are models that have been trained on huge datasets like ImageNet, and transfers the knowledge the model gained from these enormous datasets to another similar problem. Additionally, this allows us to save on training time and at times performs better than CNN models created from scratch, especially in biomedical applications [19] [20]. A study by [21] utilized transfer learning on the Inception-V3 model to detect Terry Nail abnormalities on nail images. The image dataset of the nails consisted of only 115 Terry's nails and 100 healthy nails which in total is a small dataset hence the use of transfer learning. Different batch sizes and epochs were also tested in the hyperparameters of the model. The best-performing model variation had 90% as its validation data and 10% as its training data acquiring accuracy and precision results of 95.24% and 100% respectively. Another study by [22] also utilized the method of transfer learning but used more pre-trained models namely Alexnet, Resnet50, GoogLeNet, VGG-16, Resnet101, VGG-19, InceptionV3, and InceptionResNetV2, and compared the performance of each in classifying brain MRI scans. The last three layers of the models have been replaced so that they would be applicable to brain image classification. It was found that the AlexNet model performed the best garnering accuracy results of 100%, 94%, and 95.92% on three datasets.

The articles discussed gave us insights into the significant determinants that must be taken into account when performing this study. The use of CNN and transfer learning is the most obvious choice for the application of image classification, as it performs far better compared to traditional machine learning techniques and allows auxiliary alteration to make the model more fit depending on the architecture, pre-processing techniques, or dataset. To add, there is an abundance of articles tackling computer vision tasks with the use of CNNs, especially in the medical field, providing multiple points of basis and reference. Another recurring pattern observed is the use of 3D and cropped images as inputs to the model.

All articles exhibited an increase in performance based on the evaluation metrics when these inputs were used. The utilization of an ROI to the cropped MRI images must also be considered, though this requires the acquisition of a trained radiologist which could be a limiting factor in the study. The choice of which plane of view of the MRI must also be deliberated as differing planes offer different characteristics for the model to classify, affecting the diagnostic accuracy. Lastly, as [7] mentioned, the incorporation of balancing and data augmentation techniques helps alleviate the overfitting problem and improve the performance of the CNN model to be chosen.

III. Theoretical Framework

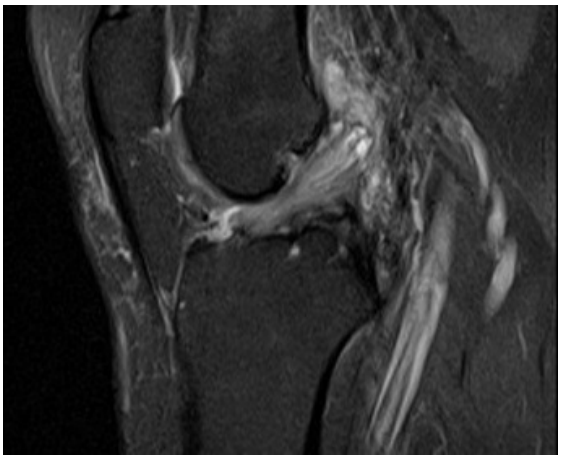
A. Anterior Cruciate Ligament (ACL) and ACL Injuries

The knee has four major ligaments: the lateral collateral ligament (LCL), medial collateral ligament (MCL), posterior cruciate ligament (PCL), and anterior cruciate ligament (ACL). The LCL and the MCL are on the outer area of the knee while the PCL and the ACL are in the inner part. The anterior cruciate ligament is a ligament that is found in the knee joint and has an essential role in ensuring that the knee is properly stabilized. This ligament is responsible for connecting the femur to the tibia, giving the knee stability during rotation movements, and preventing anterior and posterior tibial dislocation [7]. ACLs, however, are commonly injured especially for individuals participating in sports or physical activity for it is primarily caused by sudden changes in direction or movement. Since it is a ligament, it could be sprained, partially torn, or completely torn and it, unfortunately, has a low capacity for self-healing which promotes a greater risk of developing knee osteoarthritis (OA), and more meniscus and cartilage tears. This could make walking challenging and include knee discomfort, inflammation, and deformity. Surgical intervention, such as repair or construction, is the only guaranteed method to fix the ACL [8].

There are numerous methods for assessing if an individual is suffering from an ACL injury. There are the anterior drawer test, Lachman test, and pivot-shift test which are physical examinations done on the patient by a medical expert [23]. There is also arthroscopy which is a traditional diagnostic method that has a high sensitivity. The procedure, however, is highly invasive for it involves making a minute incision on the joint and inserting a narrow tube with a video camera to diagnose the injury. The most optimal method is the utilization of MRI scans. Sprains, partial tears, and complete tears could be distinguished through MRI readings done by an expert radiologist by inspecting different planes of views of the scan such as the coronal, axial, and sagittal planes.

Figure 1: MRI of normal ACL from kneeMRI dataset

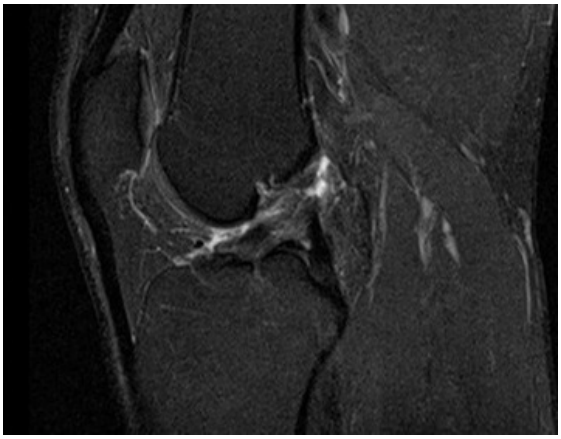
Figure 2: MRI of partially torn ACL from kneeMRI dataset



B. Convolutional Neural Networks (CNN)

Convolutional neural networks are a subclass in the hierarchical nomenclature used to describe artificial intelligence, machine learning, and deep learning. Specifically, it is a subcategory of artificial neural networks which are the primary building blocks of deep learning. Artificial neural networks are modeled after biological neural systems and are composed of numerous computational units called artificial neurons. These neurons receive input signals and based on the weighted total of their inputs, the activation function output controls the firing of neurons. Artificial neural networks and CNNs are quite similar, with the explicit presumption

Figure 3: MRI of fully torn ACL from kneeMRI dataset



that the inputs are images, allowing us to incorporate specific characteristics into the CNN architecture. CNNs are composed of several layers that enable them to learn how to encode an image's features hierarchically and transform the image into output class scores. The main layers that comprise a conventional CNN architecture are the convolutional layer, pooling layer, nonlinearity layer, and the fully connected layer.

Firstly, the convolutional layer is the foundational component of a CNN and is made up of a set of learnable filters that make up the convolutional layer's parameters. Each filter is convolved across the width and height of the input volume during the forward pass, computing dot products between the filter entries and the input at any position. These filters can be spatially minimal yet still cover the entire depth of the input volume. A series of filters are contained in one convolutional layer, and each filter will result in a different map. These maps are stacked along the depth dimension in the output volume. Thus, each item in the output volume may also be seen as the output of a neuron that scans a tiny area in the input and uses parameters that are shared by neurons in the same activation map.

In order to limit the number of parameters and computations in the network and to prevent overfitting, the pooling layer gradually shrinks the representation's spatial dimension. The pooling layer usually performs down-sampling of the spatial dimension and is usually inserted periodically between convolution layers. The MAX pooling function, which employs the highest value from each cluster of neu-

rons at the previous layer to produce a new neuron in the following layer, is the most widely used pooling function.

The nonlinearity layer employs element-wise nonlinearity through the use of a particular activation function such as the ReLU function which is considered to be the most common

Lastly, as observed in traditional neural networks, the fully connected layer or completely connected layer is a layer of neurons with complete connections between the different layers. It is usually placed before the output layer, forming the last few layers of the CNN. The fully connected layer acts as the classifier, while the convolution and pooling layers retrieve features from the input image [11].

C. Pre-trained CNN Models

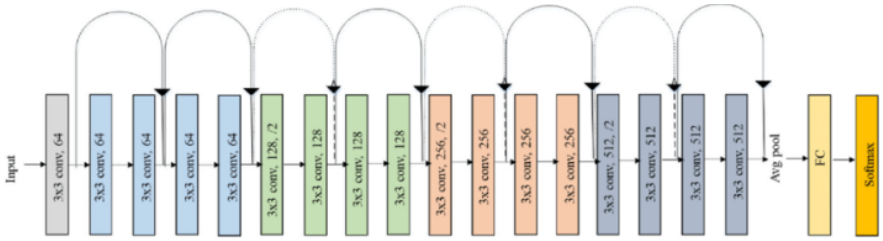
1. ResNet-18

   ResNet or Residual Network ResNet-18 is a pre-trained CNN architecture that consists of 18 layers. First introduced in the paper by He et. Al, this model facilitates the training of networks that are far deeper than those already employed previously. Adding multiple layers to a neural network makes it difficult to train, degrading and saturating the accuracy of the model. ResNet is made up of residual blocks which tackle this very issue of training very deep networks. The core of residual blocks is a connection known as a "skip connection" wherein it skips some layers of the model. By providing an alternative shortcut path for the gradient to flow through, ResNet's skip connections technique addresses the issue of the vanishing gradient in deep CNNs. Additionally, the skip connection is useful since it allows regularization to skip any layers that negatively impact architecture performance [24].

2. VGG-16

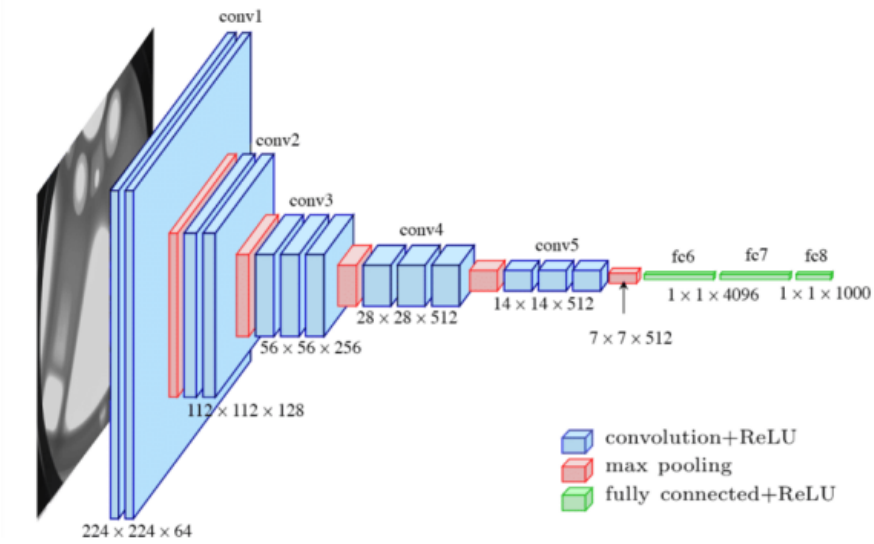
   VGG-16 is a CNN architecture that consists of 16 layers. It was showcased by Karen Simonyan & Andrew Zisserman from University of Oxford in the

Figure 4: ResNet-18 Architecture Diagram [1]



ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in which it won 1st and 2nd place in object detection and classification. The model suggested using tiny 3 × 3 receptive filters with a 1-pixel stride over the whole network which means a larger receptive area can be substituted by combining many 3 x 3 filters. As a result, the decision-making processes are more sensitive, and the network would gain the potential to converge more quickly. To add, it considerably lowers the number of weight parameters in the model. By doing this, the network's potential to over-fit during the training exercise would be diminished [25].

Figure 5: VGG-16 Architecture Diagram [2]



3. InceptionV3

Introduced by Szegedy et al. in the paper Rethinking the Inception Architecture for Computer Vision, it is a product of Google and is the 3rd version in the Inception family of Deep Learning Convolutional Architectures. It

has made numerous improvements from past versions of Inception CNNs. Specifically, it has introduced the techniques of label smoothing, factorized 7x7 convolutions, and the utilization of an auxiliary classifier to propagate label information lower in the network. These improvements allow the network to have high-quality training. Additionally, the computation costs required are lessened, especially on relatively large datasets [3].

Figure 6: Inception-V3 Architecture Diagram [3]

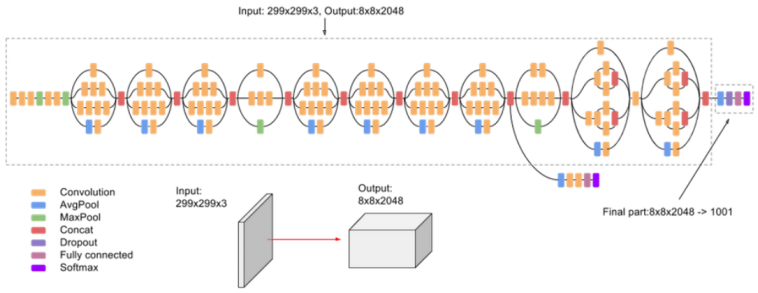
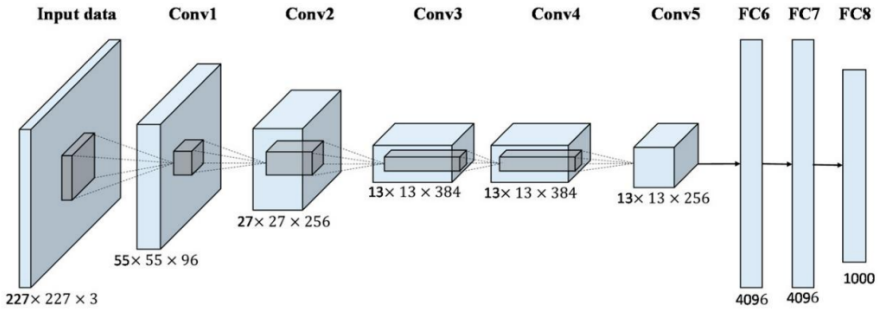


Figure 7: AlexNet Architecture Diagram [4]



4. AlexNet

AlexNet was designed mainly by Alex Krizhevsky but published with Ilya Sutskever Krizhevsky's doctoral advisor Geoffrey Hinton in the paper ImageNet Classification with Deep Convolutional Neural Networks. Having a significant impact in the field of deep learning, it was the first CNN to utilize a graphical processing unit (GPU) in boosting its performance. It is a CNN that has 60 million parameters and 650,000 neurons and consists of 5 convolutional layers, 3 max-pooling layers, 2 normalization layers, 2 fully connected layers, and 1 softmax layer. It was able to make training faster

through non-saturating neurons and efficient GPU implementation as well as reduce overfitting using the "dropout" regularization method. [26][27]

D. Transfer Learning

Training CNNs typically require extensive labeled datasets to have a certain level of classification accuracy. Resolving the dilemma of requiring large datasets, transfer learning is the use of pre-trained CNN models such as ResNet-18, VGG-16, etc. which have been trained on large datasets like ImageNet and transferring the knowledge the model learned through these extensive datasets to another similar problem. In the first layer of a neural network, edges are typically detected by the model, shapes and forms are then detected in the middle layer, and task-specific features are learned in the latter layers. In employing transfer learning, the final task-specific classification layer of the pre-trained models is retrained using the new dataset and appropriate labels needed for our task. This methodology allows us to save training time as well as resolve the problem of large data requirements. In some cases, it also outperforms CNN models trained from scratch, some in biomedical applications [22][19][20].

E. Performance Metrics

1. Confusion Matrix

   Summarizes a classifier's classification performance in relation to specific test data. Represented as a two-dimensional matrix with the true class of an object as its index in one dimension and the class the classifier assigns to it in the other. The four cells are designated as true positives (TP), false positives (FP), true negatives (TN) and, false negatives (FN) [28].

2. Precision

   The quality of a positive prediction made by the model, precision refers to the number of true positives divided by the total number of positive predic-

tions (i.e., the number of true positives plus the number of false positives) [29].

$$Precision = \frac{TP}{TP + FP}$$

3. Sensitivity

   The ability of a test to correctly classify an individual as 'diseased' [30].

$$Sensitivity = \frac{TP}{TP + FN}$$

4. F1 – Score

   It is the performance metric for classification and is calculated as the harmonic mean of precision and recall [29].

$$F1\text{–}Score = \frac{Precision \times Recall \times 2}{Precision + Recall}$$

5. Specificity

   The ability of a test to correctly classify an individual as disease-free [30].

$$Specificity = \frac{TN}{TN + FP}$$

6. Accuracy

   The percentage of pixels that are correctly categorized across all classes, lesions, and backgrounds [31].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

7. AUC

   AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1) [32].

8. Negative Predictive Value (NPV)

   The percentage of patients with a negative test who do not have the disease [30].

$$NPV = \frac{TN}{FN + TN} \times 100$$

IV. Design and Implementation

A. KneeMRI Dataset

The kneeMRI dataset (http://www.riteh.uniri.hr/ istajduh/projects/kneeMRI/) comprises of 917 sagittal view plane DICOM MRI of left or right knees in 12-bit grayscale volumes from the Clinical Hospital Center Rijeka, Croatia. The volumes were recorded from the year 2007 to 2014 and a 1.5T Siemens MR scanner was used, utilizing a proton density-weighted fat suppression method for its collection having 3.6 mm between slices (z-axis), 3 mm slice thickness, and 0.56 mm in-plane spacing (x and y axes). In a double-blind process, each volume record received a diagnostic about the state of the anterior cruciate ligament. Each volume record was given one of three labels, depending on the ligament's health: healthy with a label of 0, slightly ruptured with a label of 1, or fully ruptured with a label of 2. From the original volumes, a larger rectangular region of interest (ROI) was manually extracted and annotated. The dataset also included a metadata file which gives further information about the different volumes of the dataset. The attributes included per volume are examId, seriesNo, aclDiagnosis, kneeLR, roiX, roiY, roiZ, roiHeight, roiWidth, roiDepth, and volumeFilename [16].

B. Preprocessing Techniques

Using the findings and preprocessing techniques done by [7] for they utilized the same dataset, considering that the region of interest (ROI) of the MRI volumes varied in sizes such as their widths, heights, and, depths, this variation could negatively affect the performance of our proposed CNN architecture. As we have established in the study of [17], limiting the input field-of-view improves the performance of the model, limiting the required image search space for the model to work on. Having consistent input sizes would also help our model perform more optimally, as such we would extract and crop the ROI and rescale the volumes. Two input sizes would be tested: 75 x 75 and 256 x 256, comparing which input

size combination would result in higher model performance. Afterward, upon exploratory data analysis of the metadata CSV file, it was observed that there is an imbalance present in the dataset. There are 690 volumes diagnosed to be healthy, 172 volumes diagnosed to be partially ruptured, and 55 volumes diagnosed to be fully ruptured. The dataset is imbalanced towards the healthy class comprising of 75% of the total dataset. The proposed CNN model would have a bias in the training set to the class that has a larger amount of data. Multiple data augmentation techniques would also be explored as to which would perform best on the base CNN model. As such, a random combination of horizontal flipping, scaling, shifting, rotation, and translation would be implemented.

C. Transfer Learning Approach

After applying the necessary pre-processing steps, we would now utilize the pre-trained CNN models that we chose based on the related literature. The models to be used would be: ResNet-18, VGG-16, InceptionV3, and, AlexNet. The methodology to be employed in the study would be similar to [22]. Firstly, the dataset would be partitioned into a testing, validation, and training set, 75% would be taken into the training set, 15% into the validation set, and 10% would be taken into the testing set. Afterwards, the last three layers of the pre-trained models would be replaced by a fully connected layer, softmax layer, and classification output layer. The transferred layer would then be connected to the new layers. Afterwards, a combination of parameters based on previous literature and studies would be explored and applied on the prepared pre-trained CNN models. The epoch size would be varied from 30, 50 and 100. Different batch sizes would also be tested from 8, 16, and 32. Each combination would be run on the data and compared to each other based on the different performance metrics namely: precision, sensitivity, F1-score, specificity, accuracy, area under the curve (AUC), and NPV.

D. Technical Architecture

The proposed model would be coded using Python as there are extensive libraries that offer deep learning libraries and tools which allows us to create our own CNN models as well as use and modify pre-trained CNN models depending on the task we are required to do. Keras, Tensorflow, and PyTorch would be the libraries utilized. Keras is a high-level deep learning Python API created by Google to implement neural networks. It is an open-source library and runs on top of TensorFlow [33]. TensorFlow is again another open-source library created by Google. It could be used primarily for deep learning applications but also has support for various machine learning applications [34]. Lastly, PyTorch is an open-source deep learning framework created by Facebook that allows us to build deep learning models [35]. The code would be run on Google Colab, which is a Jupyter notebook environment, allowing us to run our code in the cloud. Google Colab gives us free access to TPU and GPU resources to train, test, and validate our deep learning models for these models typically require significant hardware requirements.

The proposed web application diagnostic and classification system would utilize Python and the web framework Django. Django is an open-source Python based web framework that allows swift production and development of secure websites and is based on the model-templates-views architectural pattern [36].

V. Results

The study assessed the performance of 4 pre-trained Convolutional Neural Network (CNN) models under different parameter configurations. A total of 144 models were trained, and the obtained results are presented below: Tables 1 and 1.1 showcase the outcomes for models without oversampling and data augmentation, while Tables 2 and 2.1 displays the results for models incorporating oversampling and data augmentation techniques. Additionally, Tables 5 to 7 present an overview of the performance of various model configurations based on different input sizes, epochs, and batch sizes.

Table 1: Without Oversampling and Data Augmentation (VGG16, ResNet18)

Model	Input Size	Epochs	Batch Size	AUC	F1 Score	Recall	Accuracy
VGG16	75x75	30	8	75.16 %	76.13 %	76.92 %	76.92 %
			16	75.69 %	75.05 %	75.82 %	75.82 %
			32	78.79 %	75.51 %	74.73 %	74.73 %
		50	8	73.78 %	74.79 %	74.73 %	74.73 %
			16	72.07 %	75.52 %	78.02 %	78.02 %
			32	72.27 %	74.21 %	76.92 %	76.92 %
		100	8	74.44%	68.96 %	73.63 %	73.63 %
			16	71.74 %	72.62 %	74.73 %	74.73 %
			32	72.66 %	72.10 %	73.63 %	73.63 %
	256x256	30	8	47.04 %	5.88 %	18.68 %	18.68 %
			16	82.67 %	80.21 %	80.22 %	80.22 %
			32	80.83 %	73.72 %	71.43 %	71.43 %
		50	8	57.11 %	69.83 %	76.92 %	76.92 %
			16	62.45 %	69.80 %	75.82 %	75.82 %
			32	77.93 %	72.63 %	71.43 %	71.43 %
		100	8	50.00 %	00.57 %	05.49 %	05.49 %
			16	83.27 %	61.65 %	56.04 %	56.04 %
			32	85.57 %	71.32 %	69.23 %	69.23 %
ResNet18	75x75	30	8	65.58 %	66.18 %	68.13 %	68.13 %
			16	74.64 %	73.27 %	76.92 %	76.92 %
			32	82.35 %	79.30 %	82.42 %	82.42 %
		50	8	78.29 %	75.40 %	78.02 %	78.02 %
			16	69.63 %	69.87 %	75.82 %	75.82 %
			32	81.23 %	77.85 %	81.32 %	81.32 %
		100	8	73.98 %	68.99 %	69.23 %	69.23 %
			16	73.62 %	73.32 %	78.02 %	78.02 %
			32	79.12 %	77.35 %	81.32 %	81.32 %
	256x256	30	8	66.93 %	62.66 %	59.34 %	59.34 %
			16	69.83 %	66.41 %	67.03 %	67.03 %
			32	70.16 %	54.34 %	48.35 %	48.35 %
		50	8	73.78 %	72.68 %	73.63 %	73.63 %
			16	63.67 %	63.77 %	60.44 %	60.44 %
			32	72.30 %	71.94 %	73.63 %	73.63 %
		100	8	71.93 %	67.67 %	68.13 %	68.13 %
			16	70.16 %	69.38 %	68.13 %	68.13 %
			32	74.24 %	66.00 %	62.64 %	62.64 %

Table 1.1: Without Oversampling and Data Augmentation (InceptionV3, AlexNet)

Model	Input Size	Epochs	Batch Size	AUC	F1 Score	Recall	Accuracy
InceptionV3	75x75	30	8	59.62 %	66.06 %	69.23 %	69.23 %
			16	55.40 %	66.81 %	71.43 %	71.43 %
			32	59.03 %	66.62	69.23 %	69.23 %
		50	8	61.46 %	71.72 %	74.73 %	74.73 %
			16	52.08 %	64.47 %	68.13 %	68.13 %
			32	47.23 %	64.48 %	70.33 %	70.33 %
		100	8	52.40 %	68.52 %	70.33 %	70.33 %
			16	53.52 %	64.70 %	67.03 %	67.03 %
			32	59.19 %	72.15 %	74.73 %	74.73 %
	256x256	30	8	64.82 %	67.87 %	69.23 %	69.23 %
			16	61.92 %	70.27 %	70.33 %	70.33 %
			32	63.64 %	70.42 %	70.33 %	70.33 %
		50	8	65.55 %	65.01 %	63.74 %	63.74 %
			16	61.20 %	65.13 %	64.84 %	64.84 %
			32	66.07 %	62.42 %	62.64 %	62.64 %
		100	8	61.89 %	65.85 %	67.03 %	67.03 %
			16	58.86 %	63.51 %	64.84 %	64.84 %
			32	64.00 %	66.56 %	65.93 %	65.93 %
AlexNet	75x75	30	8	60.87 %	13.24 %	13.19 %	13.19 %
			16	80.11 %	71.07 %	76.92 %	76.92 %
			32	74.21 %	70.02 %	72.53 %	72.53 %
		50	8	65.58 %	66.18 %	68.13 %	68.13 %
			16	80.24 %	74.51 %	74.73 %	74.73 %
			32	81.03 %	74.15 %	73.63 %	73.63 %
		100	8	71.15 %	45.65 %	41.76 %	41.76 %
			16	78.66 %	73.68 %	74.73 %	74.73 %
			32	73.45 %	75.76 %	80.22 %	80.22 %
	256x256	30	8	48.91 %	64.60 %	68.13 %	68.13 %
			16	70.62 %	71.81 %	74.73 %	74.73 %
			32	61.26 %	45.08 %	41.76 %	41.76 %
		50	8	59.35 %	15.02 %	21.98 %	21.98 %
			16	60.01 %	61.76 %	57.14 %	57.14 %
			32	71.74 %	68.21 %	65.93 %	65.93 %
		100	8	65.15 %	47.62 %	42.86 %	42.86 %
			16	61.92 %	48.25 %	43.96 %	43.96 %
			32	63.50 %	67.86 %	70.33 %	70.33 %

Table 2: With Oversampling and Data Augmentation (VGG16, ResNet18)

Model	Input Size	Epochs	Batch Size	AUC	F1 Score	Recall	Accuracy
VGG16	75x75	30	8	72.13 %	67.54 %	70.33 %	70.33 %
			16	74.57 %	72.70 %	73.63 %	73.63 %
			32	73.25 %	72.60 %	76.92 %	76.92 %
		50	8	72.99 %	70.55 %	72.53 %	72.53 %
			16	75.69 %	70.99 %	74.73 %	74.73 %
			32	73.39 %	73.25 %	76.92 %	76.92 %
		100	8	69.96 %	70.51 %	74.73 %	74.73 %
			16	74.31 %	68.42 %	75.82 %	75.82 %
			32	72.99 %	69.98 %	73.63 %	73.63 %
	256x256	30	8	79.78 %	72.16 %	74.73 %	74.73 %
			16	80.50 %	77.74 %	76.92 %	76.92 %
			32	77.27 %	72.65 %	72.53 %	72.53 %
		50	8	80.57 %	67.19 %	64.84 %	64.84 %
			16	82.94 %	73.00 %	72.53 %	72.53 %
			32	78.66 %	76.04 %	78.02 %	78.02 %
		100	8	51.58 %	66.84 %	75.82 %	75.82 %
			16	78.46 %	74.17 %	73.63 %	73.63 %
			32	82.35 %	80.56 %	80.22 %	80.22 %
ResNet18	75x75	30	8	69.27 %	67.55 %	69.23 %	69.23 %
			16	70.36 %	64.15 %	63.74 %	63.74 %
			32	76.94 %	68.21%	68.13 %	68.13 %
		50	8	74.21 %	69.82 %	72.53 %	72.53 %
			16	67.39 %	71.69 %	74.73 %	74.73 %
			32	74.60 %	72.38 %	74.73 %	74.73 %
		100	8	74.77 %	72.72 %	75.82 %	75.82 %
			16	68.51 %	66.91 %	68.13 %	68.13 %
			32	71.08 %	76.40 %	79.12 %	79.12 %
	256x256	30	8	68.77 %	68.24 %	68.13 %	68.13 %
			16	78.52 %	65.68 %	61.54 %	61.54 %
			32	71.67 %	65.96 %	61.54 %	61.54 %
		50	8	74.67 %	68.96 %	68.13 %	68.13 %
			16	72.86 %	65.43%	61.54 %	61.54 %
			32	75.76 %	71.84 %	73.63 %	73.63 %
		100	8	78.33 %	73.33 %	72.53 %	72.53 %
			16	77.14 %	70.91 %	70.33 %	70.33 %
			32	71.64 %	68.30 %	68.13 %	68.13 %

Table 2.1: With Oversampling and Data Augmentation (InceptionV3, AlexNet)

Model	Input Size	Epochs	Batch Size	AUC	F1 Score	Recall	Accuracy
InceptionV3	75x75	30	8	54.28 %	56.41 %	52.75 %	52.75 %
			16	56.36 %	61.73 %	60.44 %	60.44 %
			32	54.94 %	59.83 %	57.14 %	57.14 %
		50	8	55.27 %	62.11 %	60.44 %	60.44 %
			16	53.95 %	64.11 %	62.64 %	62.64 %
			32	46.51 %	55.14 %	51.65 %	51.65 %
		100	8	54.48 %	51.78 %	46.15 %	46.15 %
			16	54.87 %	59.30 %	56.04 %	56.04 %
			32	53.59 %	60.57 %	58.24 %	58.24 %
	256x256	30	8	55.60 %	65.18 %	64.84 %	64.84 %
			16	58.83 %	66.82 %	68.13 %	68.13 %
			32	58.96 %	64.03 %	62.64 %	62.64 %
		50	8	60.34 %	63.99 %	61.54 %	61.54 %
			16	60.31 %	64.52 %	64.84 %	64.84 %
			32	59.52 %	60.34 %	59.34 %	59.34 %
		100	8	57.35 %	62.53 %	61.54 %	61.54 %
			16	60.11 %	67.18 %	65.93 %	65.93 %
			32	55.93 %	62.91 %	61.54 %	61.54 %
AlexNet	75x75	30	8	85.70 %	73.95 %	71.43 %	71.43 %
			16	77.67 %	65.16 %	61.54 %	61.54 %
			32	79.38 %	62.72 %	57.14 %	57.14 %
		50	8	85.14 %	75.91 %	73.63 %	73.63 %
			16	79.15 %	74.27 %	73.63 %	73.63 %
			32	80.76 %	57.73 %	53.85 %	53.85 %
		100	8	81.92 %	75.18 %	75.82 %	75.82 %
			16	79.51 %	71.98 %	69.23 %	69.23 %
			32	80.43 %	70.58 %	69.23 %	69.23 %
	256x256	30	8	75.23 %	53.68 %	49.45 %	49.45 %
			16	82.21 %	73.24 %	72.53 %	72.53 %
			32	76.75 %	72.20%	71.43 %	71.43 %
		50	8	78.99 %	72.50 %	74.73 %	74.73 %
			16	80.04 %	57.04 %	51.65 %	51.65 %
			32	79.38 %	69.23 %	69.23 %	69.23 %
		100	8	77.08 %	73.16 %	72.53 %	72.53 %
			16	78.06 %	75.40 %	75.82 %	75.82 %
			32	76.91 %	76.51 %	76.92 %	76.92 %

Table 3: Without Oversampling and Data Augmentation - Best Model per Architecture

Model	Best Model Configuration	AUC	F1 Score	Recall	Accuracy
VGG16	256x256, Epochs 100, BS 32	85.57 %	71.32 %	69.23 %	69.23 %
ResNet18	75x75, Epochs 30, BS 32	82.35 %	79.30 %	82.42 %	82.42 %
InceptionV3	256x256, Epochs 50, BS 32	66.07 %	62.42 %	62.64 %	62.64 %
AlexNet	75x75, Epochs 50, BS 32	81.03 %	74.15 %	73.63 %	73.63 %

Table 4: With Oversampling and Data Augmentation - Best Model per Architecture

Model	Best Model Configuration	AUC	F1 Score	Recall	Accuracy
VGG16	256x256, Epochs 100, BS 32	82.35 %	80.56 %	80.22 %	80.22 %
ResNet18	256x256, Epochs 100, BS 8	78.33 %	73.33 %	72.53 %	72.53 %
InceptionV3	256x256, Epochs 100, BS 16	60.11 %	67.18 %	65.93 %	65.93 %
AlexNet	75x75, Epochs 50, BS 8	85.14 %	75.91 %	73.63 %	73.63 %

Table 5: Best Model Configurations per Architecture based on Input Size

Model	Input Size	Model Configuration	AUC	F1 Score	Recall	Accuracy
VGG16	75x75	W/o Oversampling, Epochs 30, BS 32	78.79 %	75.51 %	74.73 %	74.73 %
	256x256	W/o Oversampling, Epochs 100, BS 32	85.57 %	71.32 %	69.23 %	69.23 %
ResNet18	75x75	W/o Oversampling, Epochs 30, BS 32	82.35 %	79.30 %	82.42 %	82.42 %
	256x256	With Oversampling, Epochs 100, BS 8	78.33 %	73.33 %	72.53 %	72.53 %
InceptionV3	75x75	W/o Oversampling, Epochs 50, BS 16	52.08 %	64.47 %	68.13 %	68.13 %
	256x256	W/o Oversampling, Epochs 30, BS 8	64.82 %	67.87 %	69.23 %	69.23 %
AlexNet	75x75	With Oversampling, Epochs 50, BS 8	85.14 %	75.91 %	73.63 %	73.63 %
	256x256	With Oversampling, Epochs 30, BS 16	82.21 %	73.24 %	72.53 %	72.53 %

Table 6: Best Model Configurations per Architecture based on Epoch Size

Model	Epochs	Model Configuration	AUC	F1 Score	Recall	Accuracy
VGG16	30	256x256, W/o Oversampling, BS 16	82.67 %	80.21 %	80.22 %	80.22 %
	50	256x256, W/o Oversampling, BS 32	77.93 %	72.63 %	71.43 %	71.43 %
	100	256x256, With Oversampling, BS 32	82.35 %	80.56 %	80.22 %	80.22 %
ResNet18	30	75x75, W/o Oversampling, BS 32	82.35 %	79.30 %	82.42 %	82.42 %
	50	256x256, With Oversampling, BS 32	75.76 %	71.84 %	73.63 %	73.63 %
	100	75x75, W/o Oversampling, BS 32	79.12 %	77.35 %	81.32 %	81.32 %
InceptionV3	30	256x256, W/o Oversampling, BS 32	63.64 %	70.42 %	70.33 %	70.33 %
	50	75x75, W/o Oversampling, BS 8	61.46 %	71.72 %	74.73 %	74.73 %
	100	75x75, W/o Oversampling, BS 32	59.19 %	72.15 %	74.73 %	74.73 %
AlexNet	30	75x75, With Oversampling, BS 8	85.70 %	73.95 %	71.43 %	71.43 %
	50	75x75, With Oversampling, BS 8	85.14 %	75.91 %	73.63 %	73.63 %
	100	75x75, With Oversampling, BS 8	81.92 %	75.18 %	75.82 %	75.82 %

Table 7: Best Model Configurations per Architecture based on Batch Size

Model	Batch Size	Model Configuration	AUC	F1 Score	Recall	Accuracy
VGG16	8	256x256, With Oversampling, Epochs 30	79.78 %	72.16 %	74.73 %	74.73 %
	16	256x256, With Oversampling, Epochs 30	80.50 %	77.74 %	76.92 %	76.92 %
	32	256x256, W/o Oversampling, Epochs 100	85.57 %	71.32 %	69.23 %	69.23 %
ResNet18	8	256x256, With Oversampling, Epochs 100	78.33 %	73.33 %	72.53 %	72.53 %
	16	256x256, With Oversampling, Epochs 100	77.14 %	70.91 %	70.33 %	70.33 %
	32	75x75, W/o Oversampling, Epochs 30	82.35 %	79.30 %	82.42 %	82.42 %
InceptionV3	8	256x256, W/o Oversampling, Epochs 50	65.55 %	65.01 %	63.74 %	63.74 %
	16	256x256, W/o Oversampling, Epochs 30	61.92 %	70.27 %	70.33 %	70.33 %
	32	256x256, W/o Oversampling, Epochs 30	63.64 %	70.42 %	70.33 %	70.33 %
AlexNet	8	75x75, With Oversampling, Epochs 50	85.14 %	75.91 %	73.63 %	73.63 %
	16	256x256, With Oversampling, Epochs 30	82.21 %	73.24 %	72.53 %	72.53 %
	32	75x75, W/o Oversampling, Epochs 50	81.03 %	74.15 %	73.63 %	73.63 %

The web-based system for the classification of Anterior Cruciate Ligament (ACL) injuries based on uploaded Magnetic Resonance Imaging (MRI) images of the knee are also presented. The system interface consists of a homepage, a results page, and a downloadable PDF report.

A. Home Page

The homepage serves as an introduction to the system, providing a concise overview of ACL injuries and a description of the system's functionality. Positioned on the right-hand side of the page is a file upload form, which prompts users to upload an MRI image of a knee featuring an ACL that requires classification. Upon selecting a file, the chosen filename is displayed for user confirmation. The upload button initiates the classification process.

Figure 8: Home Page

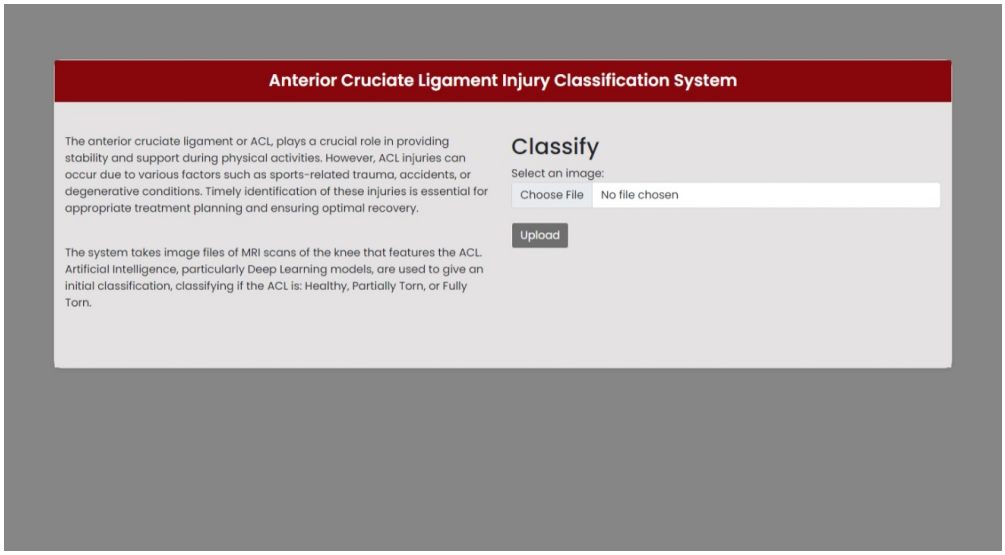
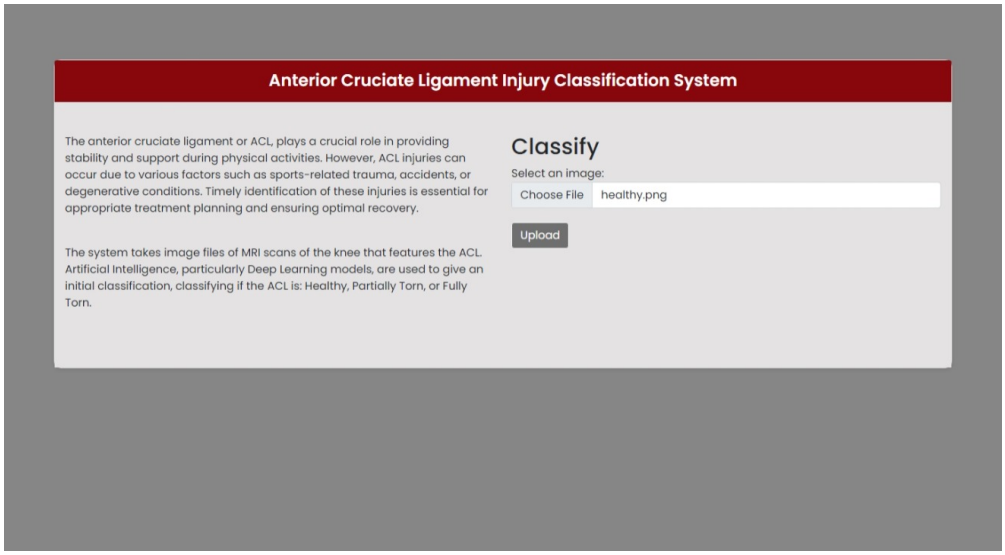


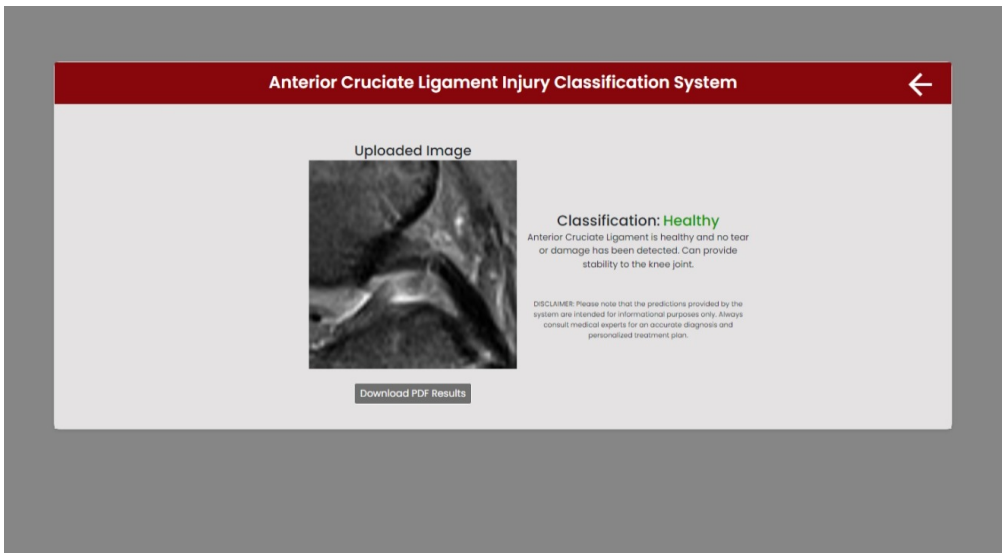
Figure 9: Home Page with Selected File



B. Results Page

Upon completion of the classification, users are redirected to the results page. This page allows users to inspect the uploaded image and view the ACL injury classification determined by the system. The classification can indicate whether the ACL is healthy, partially torn, or fully torn. Furthermore, a brief supplementary description explaining the classification is provided at the bottom of the classification, enhancing the user's understanding of the results.

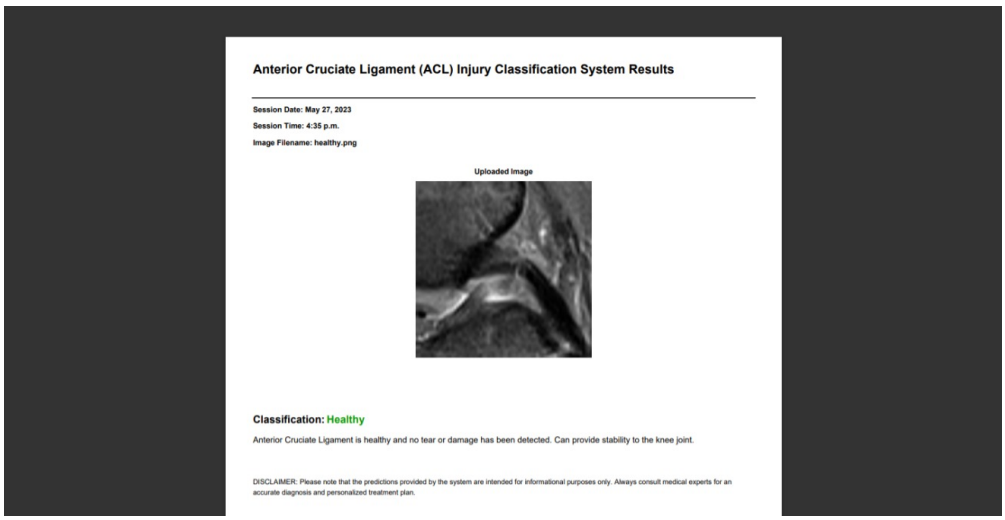
Figure 10: Results Page



C. PDF of Results

In order to facilitate further analysis and documentation, a "Download PDF Results" button is available on the results page. Users can utilize this feature to obtain a PDF file containing the information displayed on the results page. They can view as well as download the PDF file for future reference. The PDF report includes the session date and time when the user accessed the system, as well as the filename of the uploaded image used for classification.

Figure 11: PDF of Results



VI. Discussion

The Anterior Cruciate Ligament Injury Classification System is designed as a web application with the primary objective of serving as a decision support tool for the detection and classification of anterior cruciate ligament (ACL) injuries. This system utilizes an MRI image of the knee, specifically focusing on the ACL region, to accurately classify the extent of the injury.

The KneeMRI dataset was utilized, consisting of 917 sagittal view DICOM MRI images of either left or right knees obtained from the Clinical Hospital Center Rijeka, Croatia. These images are categorized into three labels: healthy, slightly ruptured or partially torn, and fully ruptured or fully torn, comprising 690, 172, and 55 samples, respectively. Preprocessing techniques described in [7] were applied. The samples were each determined a region of interest (ROI) encompassing the ACL. Each sample was subsequently cropped to its respective ROI, reducing the search space, and resized to standardized dimensions of 75x75 pixels and 256x256 pixels to ensure consistent input sizes. The first slice from each MRI volume was obtained and converted into a .PNG file and the images were also normalized.

In terms of the training, validation, and testing sets, the dataset is split following a 75%-15%-10% partitioning scheme, as outlined in [22]. Notably, an inherent class imbalance was observed, particularly in the healthy class consisting of 75% of the whole dataset. To mitigate this issue, class weights are incorporated during model training. Additionally, as a point of comparison, a hybrid oversampling and data augmentation technique is implemented to address the scarcity of samples in the partially torn and fully torn classes. This involved oversampling the minority classes by generating random augmented duplicates from the same respective classes. The augmentation process included various transformations, including horizontal flipping, scaling, shifting, rotation, and translation.

Given the imbalanced nature of the dataset, where the healthy class predominates, accuracy is insufficient to provide a comprehensive evaluation of the models.

Therefore, the primary metric employed in this analysis was the area under the curve (AUC), accompanied by the F1 score and recall as supplementary metrics. The AUC is particularly relevant as it captures the performance of the models across all thresholds, accounting for the trade-off between sensitivity and specificity.

Upon reviewing the results, first examining the without oversampling and data augmentation results in Tables 1 and 1.1, the VGG16 model with an input size of 256x256, 100 epochs, and a batch size of 32 emerged as the best-performing model overall, highlighted in gray in Table 1. In the with oversampling and data augmentation table, the same VGG16 model configuration also yielded the best results, highlighted in gray in Table 2. A comparison of the metrics between the two models reveals that although the model without oversampling exhibited a slightly higher AUC, the F1 score and recall of the model with oversampling demonstrated significant improvements, with an increase of approximately 9% to 10%. As such, between the two models, the VGG16 model incorporating oversampling and data augmentation techniques displayed superior overall performance, considering the improvements in the supplementary metrics. Additionally, Tables 3 and 4 contain the best model per architecture from the models ran without oversampling and data augmentation and from the models ran with oversampling and data augmentation.

Exploring the impact of each parameter on model performance, no clear trend emerged regarding the influence of higher or lower input sizes and batch sizes. The impact of input size on model performance was examined, revealing contrasting trends among the tested architectures. Notably, VGG16 and InceptionV3 exhibited improved AUC values when a larger input size was employed. Conversely, ResNet18 and AlexNet demonstrated enhanced performance with a smaller input size. In terms of batch sizes, a definitive pattern again could not be observed. VGG16 and ResNet18 performed the best with a batch size of 32 while InceptionV3 and AlexNet had the best AUC with a batch size of 8. It is evident

however, looking at the results of the models, there are only minimal variances in the AUC in the standard range of 8, 16 and 32 batch sizes.

Similarly, the relationship between the number of epochs and performance was investigated. Analyzing model performance at Table 8, the performance of the models did not necessarily improve as the epochs are increased or decreased and although some models had a lower AUC with more epochs, the differences are only marginal. That being said, generally it can be inferred that 30 epochs is sufficient for the models to learn the relevant features of the data. This finding also holds practical importance when training with CNNs and image data. Training CNN models can be computationally demanding and hardware-intensive. Therefore, selecting an optimal number of epochs that strikes a balance between model performance and resource utilization is important.

Moving on to the effects of oversampling with data augmentation, noteworthy enhancements were observed in the performance of the best model configuration. Despite a marginal decrease in the AUC, notable improvements were observed in the F1 score and recall for the model incorporating oversampling and data augmentation. This improvement signifies a more balanced performance, hence it is decided as the best model configuration. As such, in our case, we can say that including oversampling with data augmentation is able to enhance the outcome of the results.

The superior performance of the VGG16 model which included oversampling and data augmentation, characterized by an input size of 256x256, 100 epochs, and a batch size of 32, rendered it the most suitable choice for integration into the Anterior Cruciate Ligament Injury Classification System web application. To facilitate the development of the system, the Django web framework was employed as the backend, while HTML and CSS were utilized for the front-end interface. A file upload of an MRI image of the knee which includes an ACL allows the user to classify if the ligament is healthy, partially torn, or fully torn. Additionally, the user also has the ability to save and download a PDF file of the results.

The system is especially useful as a decision support tool in healthcare and orthopedic clinics. It can provide assistance and guidance to healthcare professionals when assessing ACL injuries, providing objective and consistent assessments. The system can also improve the efficiency of diagnosis as using the system is relatively simple and results are produced quickly. This could save valuable time for healthcare professionals, allowing them to allocate their resources effectively and focus on delivering timely and targeted interventions to address the actual injury.

VII. Conclusion

The presented study investigates the application of deep learning, specifically convolutional neural networks (CNNs), for the purpose of providing decision support in the detection and classification of anterior cruciate ligament (ACL) injuries. The primary objective is to assess the performance of four pre-trained CNN models: VGG16, ResNet18, InceptionV3, and AlexNet. Furthermore, the study identifies optimal parameters for each model by varying the number of epochs, input sizes, and batch sizes. In addition, the impact of employing oversampling and data augmentation techniques on model performance is explored and compared. A comprehensive analysis encompassing 144 different model configurations is conducted, utilizing a range of evaluation metrics to gauge the performance of each model configuration.

The study made use of the KneeMRI dataset, sourced from the Clinical Hospital Center Rijeka, Croatia, which comprised a total of 917 knee MRI images. Among these images, 690 were classified as healthy, 172 as partially torn, and 55 as fully torn. In order to prepare the dataset, several preprocessing techniques are employed. These techniques involved cropping the region of interest (ROI) within each image, resizing the images to dimensions of 75x75 pixels and 256x256 pixels, converting the first slice of each MRI volume into a .PNG file format, and applying normalization. Subsequently, the dataset is divided into training, validation, and testing subsets, following a partitioning scheme of 75%, 15%, and 10%, respectively.

Performance analysis is performed and it is determined that the VGG16 model that utilized oversampling with data augmentation with an input size of 256x256, 100 epochs, and a batch size of 32 is the best performing model configuration. No clear trend is observed between the input size, batch size, and model performance, however VGG16 and InceptionV3 exhibited improved AUC values when a larger input size was employed while ResNet18 and AlexNet demonstrated enhanced performance with a smaller input size. Additionally, VGG16 and ResNet18 per-

formed the best with a batch size of 32 while InceptionV3 and AlexNet had the best AUC with a batch size of 8. The optimal range for the number of epochs are found to be 30 epochs which is the balance for optimum performance and resource utilization.

The best model was chosen for integration into the Anterior Cruciate Ligament Injury Classification System web application. The system architecture was designed with the Django framework serving as the backend, while HTML and CSS were employed for the frontend development. Within the application, users are provided with the capability to upload an MRI image of a knee, enabling the system to classify the health status of the ACL as healthy, partially torn, or fully torn. Furthermore, the system offers the functionality to save the classification results in a PDF format for future reference.

The system serves as a valuable decision support tool in healthcare and orthopedic clinics, aiding healthcare professionals in assessing ACL injuries with objective and consistent assessments. Its user-friendly interface and prompt results enhance diagnostic efficiency, saving valuable time and enabling focused interventions for effective patient care. Although further enhancements can be implemented for both the web application and the employed CNN model, this study establishes a foundation for the development of a decision support system aimed at the accurate classification and detection of ACL injuries.

VIII. Recommendation

The model performance of the Anterior Cruciate Ligament Injury Classification System can be further improved by aiming for higher values of key evaluation metrics such as the area under the curve (AUC), F1 score, and recall. To achieve this, several recommendations can be considered:

1. Finding more images: Increasing the amount of images and samples to be used for training, especially for the minority classes can help alleviate the class imbalance problem and therefore improve the overall performance of the models.

2. Exploring different datasets using the same machine: Alternative datasets from different institutions can be explored, keeping in mind the importance of using MRI images captured with the same machine. Different MRI machines have their own unique capture and pre-processing protocols, resulting in differences in the produced MRI volumes. As such, it is important to ensure that the same machine was used to capture the MRI samples when looking at datasets from other institutions.

3. Explore other pre-trained architectures: As the study only focused on four pre-trained CNN architecures, it is also noteworthy to investigate alternative pre-trained models that may be more suitable for the task of ACL injury classification. This exploration can help identify models that can further enhance the system's performance and potentially uncover new insights in ACL injury assessment.

IX. Bibliography

[1] F. M. Ramzan, A. Rehmat, and Z. Mehmood, "A deep learning approach for automated diagnosis and multi-class classification of alzheimer's disease stages using resting-state fmri and residual neural networks," *ResearchGate*, 2019. Accessed: Dec. 06, 2022.

[2] M. Ferguson, R. ak, Y.-T. T. Lee, and K. H. Law, "Automatic localization of casting defects with convolutional neural networks," *ResearchGate*, 2017. Accessed: Dec. 06, 2022.

[3] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, "Rethinking the inception architecture for computer vision," 2015.

[4] X. Han, Y. Zhong, L. Cao, and L. Zhang, "Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification," *Remote Sensing*, vol. 9, p. 848, Aug. 2017.

[5] K. Haraldsdottir and A. M. Watson, "Psychosocial impacts of sports-related injuries in adolescent athletes," *Current Sports Medicine Reports*, vol. 20, pp. 104–108, Feb. 2021.

[6] G. P. Murphy and R. B. Sheehan, "A qualitative investigation into the individual injury burden of amateur rugby players," *Physical Therapy in Sport*, vol. 50, pp. 74–81, Jul. 2021.

[7] M. J. Awan, M. M. Rahim, N. Salim, M. Mohammed, B. Garcia-Zapirain, and K. Abdulkareem, "Efficient detection of knee anterior cruciate ligament from magnetic resonance imaging using deep learning approach," *Diagnostics*, vol. 11, p. 105, Jan. 2021.

[8] L. Zhang, M. Li, Y. Zhou, G. Lu, and Q. Zhou, "Deep learning approach for anterior cruciate ligament lesion detection: Evaluation of diagnostic per-

formance using arthroscopy as the reference standard," *Journal of Magnetic Resonance Imaging*, vol. 52, pp. 1745–1752, Jul. 2020.

[9] C. E. Pfeifer, P. F. Beattie, R. S. Sacko, and A. Hand, "Risk factors associated with non-contact anterior cruciate ligament injury: A systematic review," *International journal of sports physical therapy*, vol. 13, no. 4, pp. 575–587, 2018. Accessed: Nov. 30, 2022.

[10] M. Zhao and et al., "The accuracy of mri in the diagnosis of anterior cruciate ligament injury," *Annals of Translational Medicine*, vol. 8, pp. 1657–1657, Dec. 2020.

[11] S. Soffer, A. Ben-Cohen, O. Shimon, M. M. Amitai, H. Greenspan, and E. Klang, "Convolutional neural networks for radiologic images: A radiologist's guide," *Radiology*, vol. 290, pp. 590–606, Mar. 2019.

[12] A. W. Salehi, P. Baglat, B. B. Sharma, G. Gupta, and A. Upadhya, "A cnn model: Earlier diagnosis and classification of alzheimer disease using mri," in *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, Sep. 2020.

[13] E. Avşar and K. Salçin, "Detection and classification of brain tumours from mri images using faster r-cnn," *Tehnički glasnik*, vol. 13, pp. 337–342, Dec. 2019.

[14] S. S. Yadav and S. M. Jadhav, "Deep convolutional neural network based medical image classification for disease diagnosis," *Journal of Big Data*, vol. 6, Dec. 2019.

[15] D. S. Kermany and et al., "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, pp. 1122–1131.e9, Feb. 2018.

[16] I. Štajduhar, M. Mamula, D. Miletić, and G. Ünal, "Semi-automated detection of anterior cruciate ligament injury from mri," *Computer Methods and Programs in Biomedicine*, vol. 140, pp. 151–164, Mar. 2017.

[17] P. D. Chang, T. T. Wong, and M. J. Rasiej, "Deep learning for detection of complete anterior cruciate ligament tear," *Journal of Digital Imaging*, vol. 32, pp. 980–986, Mar. 2019.

[18] M. H. Razali, S. S. Mazlan, M. Mahmood, and M. Ayob, "Anterior cruciate ligament (acl) coronal view injury diagnosis system using convolutional neural network," *ResearchGate*, 2019. Accessed: Nov. 30, 2022.

[19] "What is transfer learning? exploring the popular deep learning approach." https://builtin.com/data-science/transfer-learning, 2022. Accessed: Dec. 03, 2022.

[20] P. Sharma, "Transfer learning — understanding transfer learning for deep learning." https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/, Oct. 2021. Accessed: Dec. 03, 2022.

[21] M. Yani, S. B. I. Si., and S. T. C. Setiningsih M.T., "Application of transfer learning using convolutional neural network method for early detection of terry's nail," *Journal of Physics: Conference Series*, vol. 1201, p. 012052, May 2019.

[22] T. Kaur and T. K. Gandhi, "Deep convolutional neural networks with transfer learning for automated brain image classification," *Machine Vision and Applications*, vol. 31, Mar. 2020.

[23] V. Musahl and J. Karlsson, "Anterior cruciate ligament tear," *New England Journal of Medicine*, vol. 380, pp. 2341–2348, Jun. 2019.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Thecvf.com*, pp. 770–778, 2016. Accessed: Nov. 30, 2022.

[25] K. Simonyan and A. Zisserman, "Published as a conference paper at iclr 2015 very deep convolutional networks for large-scale image recognition," 2015.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, May 2012.

[27] G. L. Team, "Alexnet: The first cnn to win image net." https://www.mygreatlearning.com/blog/alexnet-the-first-cnn-to-win-image-net/, Jun. 2020. Accessed: Dec. 05, 2022.

[28] T. R. Shultz and et al., "Confusion matrix," pp. 209–209, 2011.

[29] "Precision - c3 ai." https://c3.ai/glossary/machine-learning/precision/, Mar. 2022. Accessed: Nov. 30, 2022.

[30] R. Parikh, A. Mathai, S. Parikh, G. C. Sekhar, and R. Thomas, "Understanding and using sensitivity, specificity and predictive values," *Indian Journal of Ophthalmology*, vol. 56, no. 1, p. 45, 2008.

[31] M. G. F. Costa, J. P. M. Campos, G. d. Aquino e Aquino, W. C. d. A. Pereira, and C. F. F. Costa Filho, "Evaluating the performance of convolutional neural networks with direct acyclic graph architectures in automatic segmentation of breast lesion in us images," *BMC Medical Imaging*, vol. 19, no. 1, 2019.

[32] "Classification: Roc curve and auc — machine learning — google developers." https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc, 2022. Accessed: Nov. 30, 2022.

[33] "Keras documentation: About keras." https://keras.io/about/, 2022. Accessed: Nov. 30, 2022.

[34] "Why tensorflow." https://www.tensorflow.org/about, 2022. Accessed: Nov. 30, 2022.

[35] "What is pytorch?." https://www.nvidia.com/en-us/glossary/data-science/pytorch/, 2017. Accessed: Nov. 30, 2022.

[36] "Django overview." https://www.djangoproject.com/start/overview/, 2022. Accessed: Dec. 06, 2022.

X. Appendix

A. Source Code

views.py

```
from django.http import HttpResponse
from django.shortcuts import render, redirect
from django.core.files.storage import FileSystemStorage
from django.template.loader import get_template
import datetime

from xhtml2pdf import pisa

from keras.models import load_model
import tensorflow as tf
from tensorflow import Graph
import json
import numpy as np

img_height, img_width = 256, 256
with open('./models/labels.json','r') as f:
    labelInfo=f.read()

labelInfo=json.loads(labelInfo)

model_graph = Graph()
with model_graph.as_default():
    tf_session = tf.compat.v1.Session()
    with tf_session.as_default():
        model=load_model('./models/VGG16_256x256_yes_oversamp_epo_100_bs_32.h5')


def index(request):
    if request.method =='POST':
        # saving the image
        fileObj = request.FILES['filePath']
        fs = FileSystemStorage()
        filePathName = fs.save(fileObj.name, fileObj)
        filePathNamePDF = fs.path(filePathName)
        # contains the path of the image for viewing in index.html
        filePathName = fs.url(filePathName)
        request.session['filePathName'] = filePathName
        # get path for PDF
        request.session['filePathNamePDF'] = filePathNamePDF
        return redirect(predictImage)

    return render(request, 'index.html')


def predictImage(request):
    # prediction
    filePathName = request.session.get('filePathName')
    testimage='.'+filePathName
    img = tf.keras.utils.load_img(testimage, target_size=(img_height, img_width))
    x = tf.keras.utils.img_to_array(img)
    x=x/255
    x=x.reshape(1,img_height, img_width,3)
    with model_graph.as_default():
        with tf_session.as_default():
            predi=model.predict(x)


    predictedLabel=labelInfo[str(np.argmax(predi[0]))]

    # store predicted label in session for PDF
    request.session['predictedLabel'] = predictedLabel

    context={'filePathName':filePathName, 'predictedLabel':predictedLabel}
    return render(request,'prediction.html', context)


def renderPDF(request, *args, **kwargs):
    template_path = 'pdf.html'

    filePathNamePDF = request.session.get('filePathNamePDF')
    predictedLabel = request.session.get('predictedLabel')
    dateToday = datetime.date.today()
    timeToday = datetime.datetime.now().time()

    fileName = filePathNamePDF.split("media\\")

    context = {'title': 'Anterior Cruciate Ligament (ACL) Injury Classification System Results
        ',
               'filePathNamePDF' : filePathNamePDF,
               'predictedLabel' : predictedLabel,
               'dateToday' : dateToday,
               'timeToday' : timeToday,
               'fileName' : fileName[1]}

    response = HttpResponse(content_type='application/pdf')

    # if display
```

```
    response['Content-Disposition'] = 'filename="ACL Injury Prediction.pdf"'
    template = get_template(template_path)
    html = template.render(context)

    # create pdf
    pisa_status = pisa.CreatePDF(html, dest=response)
    # if error, show some view
    if pisa_status.err:
        return HttpResponse('Error <pre>' + html + '</pre>')
    return response
```

index.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="
      stylesheet" integrity="sha384-EVSTQN3/
      azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

  <style>
    @import url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0
        ,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,9
        display=swap');

    body {
      background-color: #868686;
      font-family: Tahoma, sans-serif;
    }

    .container {
      margin-top: 0px;
    }

    .card {
      margin-top: 80px;
      border-radius: 10px;
      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    }

    .card-header {
      background-color: #88070c;
      color: #fff;
      border-radius: 8px 8px 0 0;
      padding: 15px;
    }

    .card-title {
      margin-bottom: 0;
      font-size: 24px;
      font-weight: 600;
      font-family: 'Poppins', sans-serif;
    }

    .card-body {
      padding: 15px;
      background-color: #e4e2e2;
    }

    .row{
      margin-top: 30px;
    }

    .form-group {
      margin-bottom: 20px;
    }

    .btn-primary {
      background-color: #6f6f6f;
      border-color: #feffff;
    }

    .btn-primary:hover {
      background-color: #4f4f4f;
      border-color: #4f4f4f;
    }

    h4{
      display: inline;
    }

    img{
      display: block;
    }

    #imageBox{
      padding-top: 10px;
    }

    #desc{
      margin-bottom: 50px;
      font-family: 'Poppins', sans-serif;
    }

    #uploadImage{
```

```
      font-family: 'Poppins', sans-serif;
    }
  </style>
  <title>ACL Injury Classification System</title>
</head>

<body>
  <div class="container">
    <div class="card">
      <div class="card-header">
        <h1 class="card-title text-center">Anterior Cruciate Ligament Injury Classification
            System</h1>
      </div>
      <div class="card-body">
        <div class="row">
          <div class="col-md-6" id="desc">
            <p>The anterior cruciate ligament or ACL, plays a crucial role in providing
                stability and support during physical activities.
              However, ACL injuries can occur due to various factors such as sports-related
                trauma, accidents, or degenerative conditions.
              Timely identification of these injuries is essential for appropriate treatment
                planning and ensuring optimal recovery.
            </p>
            <br>
            <p>The system takes image files of MRI scans of the knee that features the ACL.
              Artificial Intelligence, particularly Deep Learning models, are used to give an
                initial classification, classifying if the ACL is: Healthy, Partially Torn,
                or
            Fully Torn. </p>
          </div>

          <div class="col-md-6" id="uploadImage">
            <h2>Classify</h2>
            <form method="post" enctype="multipart/form-data" action=''>
            {% csrf_token %}
              <div class="form-group">
                <label for="image">Select an image:</label>
                <input class="form-control input-sm" type="file" id="file" name="filePath"/>
              </div>
              <button type="submit" class="btn btn-primary btn-md">Upload</button>
            </form>
          </div>

        </div>
      </div>
    </div>
  </div>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/js/bootstrap.bundle.min.js"></
      script>
</body>

</html>
```

pdf.html

```
{% load static %}
<html>
    <head>
        <meta charset="UTF-8">
        <style>
            h1{
                font-size: 18px;
            }

            h4{
                display: inline;
                font-size: 15px;

            }

            h3{
                margin-bottom: 0px;
            }

            #image{
                margin: auto;
                width: 100%;
                text-align: center;
            }

            #image_txt{
                margin: auto;
                width: 100%;
                text-align: left;
                margin-top: 80px;

            }

            #desc{
                font-size: 12px;
            }

            #disclaimer{
                margin-top: 40px;
            }

            #info{
                line-height: 7px;
```

```
            }
        </style>
        <title>PDF Results</title>
    </head>

    <body>
        <h1>{{title}}</h1>
        <hr>
        <div id="info">
            <h3>Session Date: {{dateToday}}</h3>
            <h3>Session Time: {{timeToday}}</h3>
            <h3>Image Filename: {{fileName}}</h3>
        </div>
        <br>
        <div id="image">
            <div id="image__img">
                <h3 id="img_title">Uploaded Image</h3>
                <img src="{{filePathNamePDF}}" alt="Predicted Image" width='250'>
            </div>
            <div id="image__txt">
                <h4 id="classification">Classification: </h4>
                {% if predictedLabel == "Healthy" %}
                <h4 style="color:rgb(13, 153, 3);" id="label">{{predictedLabel}}</h4>
                <br>
                <p id="desc">Anterior Cruciate Ligament is healthy and no tear or damage has
                    been detected.
                Can provide stability to the knee joint.</p>
                {% elif predictedLabel == "Partially Torn" %}
                <h4 style="color:rgb(206, 107, 0);" id="label">{{predictedLabel}}</h4>
                <p id="desc">Anterior Cruciate Ligament is partially torn. Ligament is
                    stretched and became loose,
                unlikely to provide full stability to the knee joint.
                </p>
                {% elif predictedLabel == "Fully Torn" %}
                <h4 style="color:rgb(168, 0, 0);" id="label">{{predictedLabel}}</h4>
                <p id="desc">Anterior Cruciate Ligament is fully torn. Ligament has split into
                    two pieces and provides
                    no stability to the knee joint.</p>
                {% endif %}
                <p id="disclaimer"> DISCLAIMER: Please note that the predictions provided by
                    the system are intended for
                    informational purposes only. Always consult medical experts for an
                    accurate diagnosis and personalized treatment plan.</p>
            </div>
        </div>
    </body>
</html>
```

prediction.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/css/bootstrap.min.
      css">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="
      stylesheet" integrity="sha384-EVSTQN3/
      azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
  <style>
    @import url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0
        ,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,9
        display=swap');
    body {
      background-color: #868686;
      font-family: Tahoma, sans-serif;
    }

    .container {
      margin-top: 0px;
    }

    .card {
      margin-top: 80px;
      border-radius: 8px;
      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    }

    .card-header {
      background-color: #88070c;
      color: #fff;
      border-radius: 8px 8px 0 0;
      padding: 15px;
    }

    .card-title {
      margin-bottom: 0;
      font-size: 24px;
      font-weight: 600;
      font-family: 'Poppins', sans-serif;
    }

    .card-body {
      padding: 15px;
      background-color: #e4e2e2;
```

```
      font-family: 'Poppins', sans-serif;
    }

    .form-group {
      margin-bottom: 20px;
    }

    .btn-primary {
      background-color: #6f6f6f;
      border-color: #feffff;
    }

    .btn-primary:hover {
      background-color: #4f4f4f;
      border-color: #4f4f4f;
    }

    h4{
      display: inline;
      font-size: 21px;
    }

    img{
      display: block;
    }

    .material-icons{
      position: absolute;
      display: inline;
      right: 20px;
      top: 7px;
      font-size: 50px;
      margin-bottom: 0;
      color: white;
    }

    .material-icons:hover{
      color: #4f4f4f;
    }

    .col-md-6{
      border: 3px blue;
      padding: 20px 40px 40px;
      max-width: 640px;
      display: flex;
      margin: auto;
      width: 100%;
      text-align: center;

    }

    .col-md-6__img{
      margin: 20px 30px 0 0;
      width: 200px;
      object-fit: contain;
      align-self: flex-start;
      border: 3px blue;
    }

    .col-md-6__txt {
      flex: 1 1 auto;
      left: 80px;
      position: relative;
      margin-top: 120px;
    }

    #desc{
      font-size: 13px;
    }

    #disclaimer{
      font-size: 10px;
    }

    #img_title{
      right: -50px;
      position: relative;
      font-size: 20px;
    }

    #pdf{
      right: -50px;
      top: 20px;
      position: relative;
      font-family: 'Poppins', sans-serif;
    }
  </style>
  <title>ACL Injury Classification System - Prediction</title>
</head>

<body>
  <div class="container">
    <div class="card">
      <div class="card-header">
        <h1 class="card-title text-center">Anterior Cruciate Ligament Injury Classification
            System</h1>
        <a href="javascript:history.go(-1)"><i class="material-icons">arrow_back</i></a>
      </div>
      <div class="card-body">
```

```
        <div class="row">

          <div class="col-md-6">
            <div class="col-md-6__img">
              <h4 id="img_title">Uploaded Image</h4>
              <img src="{{filePathName}}" alt="Predicted Image" width='300'>
              <a href="renderPDF" target="_blank" id="pdf">
                <button type="submit" class="btn btn-primary btn-sm">Download PDF Results</
                    button>
              </a>
            </div>

            <div class="col-md-6__txt">
              <h4>Classification:</h4>

              {% if predictedLabel == "Healthy" %}
              <h4 style="color:rgb(13, 153, 3);">{{predictedLabel}}</h4>
              <br>
              <p id="desc">Anterior Cruciate Ligament is healthy and no tear or damage has
                  been detected.
                Can provide stability to the knee joint.</p>
              {% elif predictedLabel == "Partially Torn" %}
              <h4 style="color:rgb(206, 107, 0);">{{predictedLabel}}</h4>
              <br>
              <p id="desc">Anterior Cruciate Ligament is partially torn. Ligament is stretched
                   and became loose,
                unlikely to provide full stability to the knee joint.
              </p>
              {% elif predictedLabel == "Fully Torn" %}
              <h4 style="color:rgb(168, 0, 0);">{{predictedLabel}}</h4>
              <br>
              <p id="desc">Anterior Cruciate Ligament is fully torn. Ligament has split into
                   two pieces and provides
                no stability to the knee joint.</p>
              {% endif %}
              <br>
              <p id="disclaimer"> DISCLAIMER: Please note that the predictions provided by the
                   system are intended for
                informational purposes only. Always consult medical experts for an
                accurate diagnosis and personalized treatment plan.</p>
            </div>
          </div>

        </div>
      </div>
    </div>
  </div>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/js/bootstrap.bundle.min.js"></
      script>
</body>

</html>
```

XI. Acknowledgment

I would like to take this moment to thank the people that helped me in my stay in UP and in my journey of completing this SP.

I would first like to thank my parents and my sister. Thank you for providing guidance and support, helping me with anything I need and being understanding throughout my college life.

I would also like to thank my SP adviser, Dr. Magboo. Thank you for the guidance and the lessons you have taught me. I have learned so much and would like to thank you very much, Sir!

I would like to thank my friends for always being there for me. For all the memories that we have created and all the fun we had while we were all going through college. Thank you for making the journey more memorable.

To the person closest to me, I am unbelievably lucky to have someone like you by my side. Thank you for accompanying me in everything and making the long days and late nights easier. I am grateful and thankful for everything.

My sincerest gratitude and appreciation to everyone!