

**Deep Learning Model for Knee Joint Disease Analysis**

**Supervision:**

* **Dr/ Ahmed Ezz**
* **Eng. Ahmed Essa**

**Team members:**

1. **Hisham Said Fangary (T.L)**
2. **Mahmoud Hamed**
3. **Nour Eldin**
4. **Yasmin Ashraf**
5. **Evana Boles**
6. **Rodaina Moawad**
7. **Hager Said**

Agenda

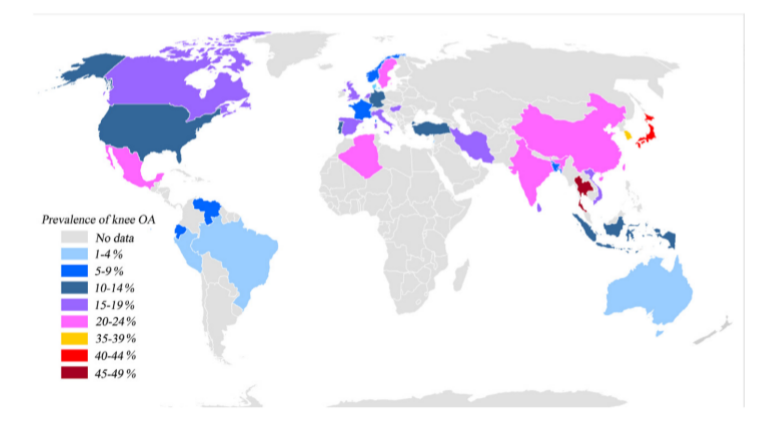
1. Introduction and Objectives
2. Brief about the disease and treatment
3. Implementation methods
4. Software – Hardware Requirements
5. Brief about used methodologies
6. First Methodology (transfer learning)
7. Dataset Overview
8. Dataset preprocess
9. Models' comparisons
10. Brief about obstacles
11. Results of Transfer Learning
12. Second Methodology
13. Dataset of second Methodology
14. New Model Architecture
15. Model Evaluation
16. Snippest of final Code
17. Conclusion
18. Applications
19. Limitations and Future Work
20. References

Introduction:

10,000 Person per year are affected by Osteoporosis with vary degrees The disease is caused due to weakness of bones’ edges that lead to increase of fragility, so the density of bone becomes low and easily affected by overwork or using stairs.

So, discovering early is especially important to help avoid the harm and less activity of patient.

As result of that, Our project revolves around the problem of osteoporosis that people face, as it requires absolute accuracy to measure the percentage of fragility in the bone, which may be difficult to the doctors, and thus the idea came, where we use machine learning to measure these percentages very accurately, and we seek to make the error rate less than now.



⦁ Causes and Risk Factors:

Knee osteoarthritis develops gradually due to the wear and tear on the knee joint over time. Several factors contribute to its onset, including age, genetics, joint injuries, repetitive stress from certain occupations or activities, obesity, and gender, with women being more susceptible after the age of 50. The degeneration of cartilage results in bones rubbing against each other, causing pain, swelling, and reduced mobility.

⦁ Symptoms:

The primary symptoms of knee osteoarthritis include:

⦁Pain during or after movement

⦁Stiffness, particularly after periods of inactivity

⦁Swelling caused by inflammation or excess joint fluid

⦁Loss of flexibility and difficulty in moving the knee

⦁A grating sensation or popping sounds during movement

⦁Bone spurs, or extra bits of bone, that form around the affected joint

⦁Diagnosis and Grading

Diagnosing knee osteoarthritis involves a combination of clinical evaluation, patient history, physical examination, and imaging techniques such as X-rays and MRI. X-rays are particularly useful in grading the severity of knee osteoarthritis, which is commonly assessed using the Kellgren-Lawrence (KL) grading system. This system classifies knee osteoarthritis into five grades:

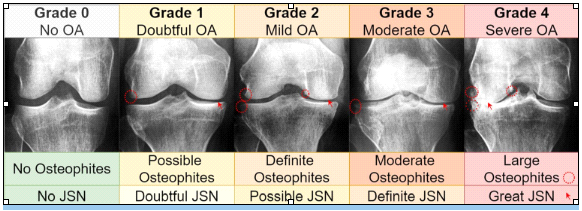
⦁ Grade 0 (Normal): No signs of osteoarthritis.

⦁ Grade 1 (Doubtful): Minute osteophytes (bone spurs) with doubtful clinical significance.

⦁ Grade 2 (Mild): Definite osteophytes and narrowing of the joint space.

⦁ Grade 3 (Moderate): Moderate multiple osteophytes, definite narrowing of joint space, some sclerosis, and deformity of bone ends.

⦁ Grade 4 (Severe): Large osteophytes, marked narrowing of joint space, severe sclerosis, and definite deformity of bone ends.



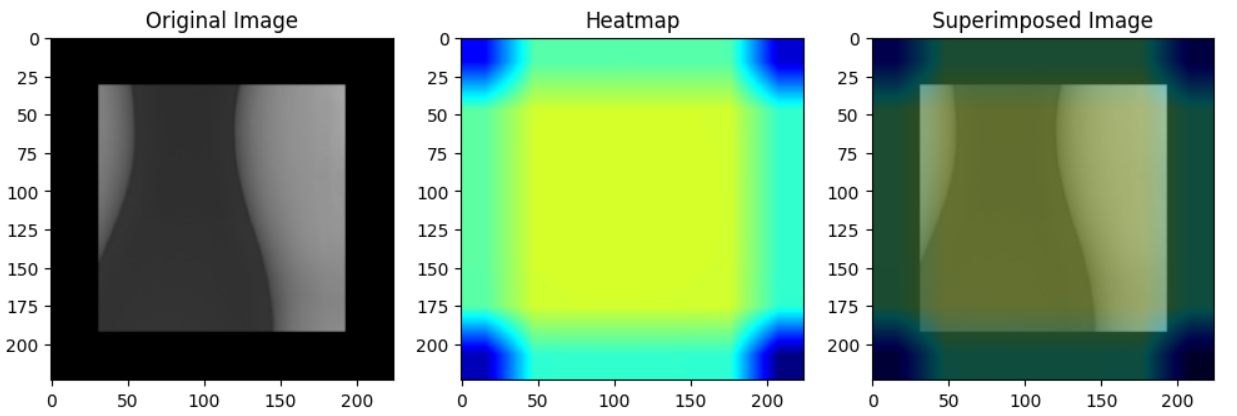
in this context, specially trained Models were used to differentiate between the types of disease and the rates for each case, and they were divided through a database containing each type with illustrative images on which the model was trained so that it could predict the rate of the disease.

Some medical procedures are necessary to identify this pathology, such as X-rays or magnetic resonance imaging, in which it is possible to assess the loss in joint spacing, thus indicating the severity of the disease.

In upcoming steps, we are showing our work timeline, Methodology and Excerpts from the work environment.

Purpose:

The purpose of this project is to correctly classify the severity of osteoarthritis based on X-ray.



Implementation:

The software used is Google Collab due to its reliability and availability.

The needed hardware is GPU 8vCore

We had two methodologies to work on

1. Transfer Learning
2. Using research papers

**The transfer learning revolved around using pretrained models like VGG 16, 19 and MobileNet**

Methodology:

The following methodology has been proposed to correctly classify the degree of osteoarthritis based on X-ray images:

1. Data preparation
2. Model training
3. Model evaluation

1. Data preparation

The dataset consists of about 1700 X-ray images of the knee obtained to classify the degree of severity.

We chose a lot of sources to be a reference for our x-ray images, then we separated the work for models to obtain best results.

Three strategies were implemented to reduce the impact that the unbalanced base can have on the models:

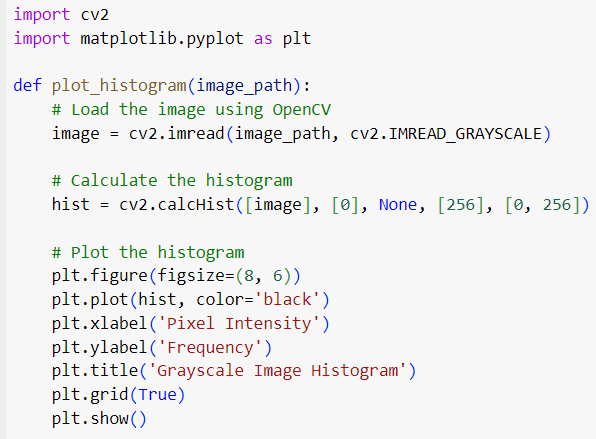
class weight

data augmentation (horizontal\_flip, brightness\_range, width\_shift\_range, zoom\_range)

preprocessing features of pre-trained networks

Then we move to next step in which the preprocess is done:

As shown in the figure [9] the provided method with the VGG 19 model



And in figure [10] for the ResNet model

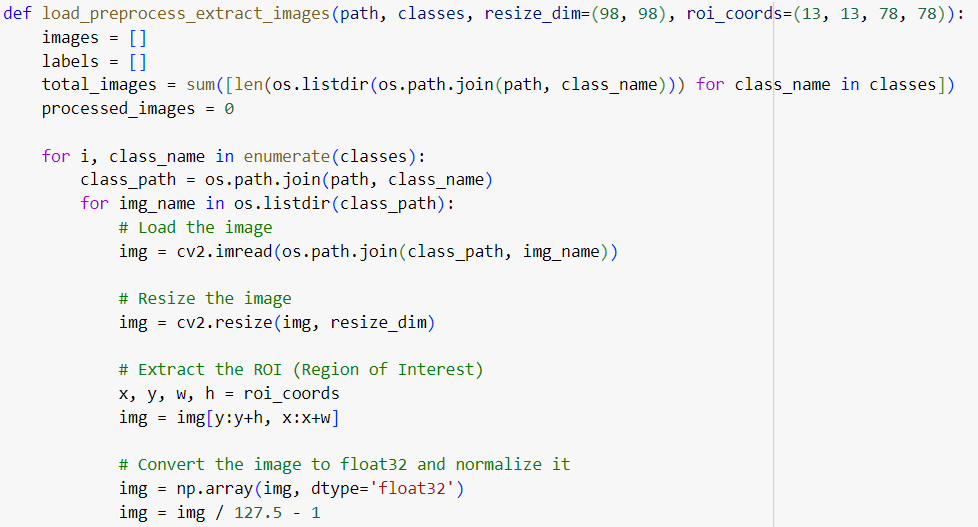
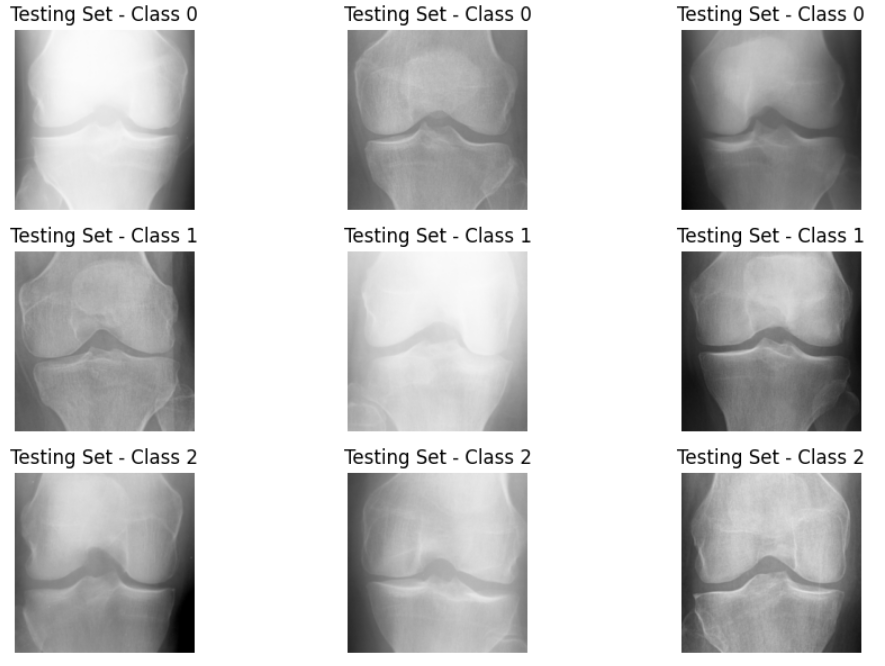


Figure [11] shows random result

The Last step is to get histogram about the preprocessed data to ensure that all is right, and we can train the models on it

As shown in figure [12] the provided code for the histogram function and the next figure for the result

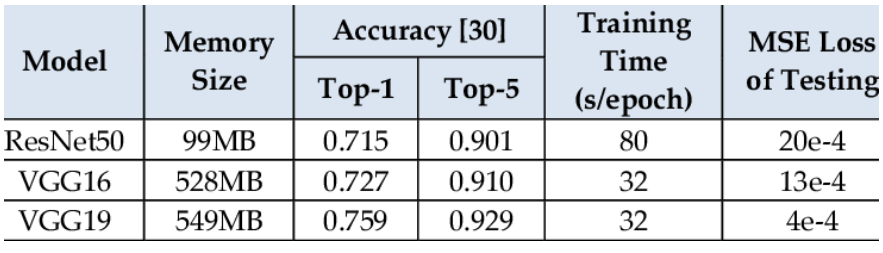


**2. Model training**

Pre-trained Networks

Two pre-trained networks were chosen: VGG 19 and ResNet-50 e Inception after the differentiation between models and below a figure showing the priority of each model

Figure [14]

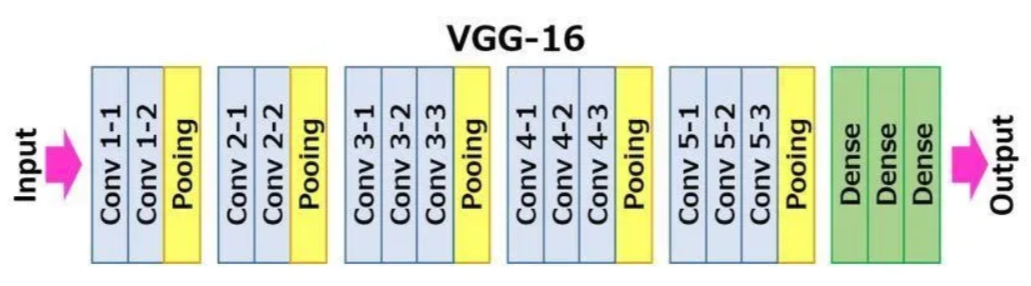


In the beginning of work, we built our own model and accuracy was 28%

This step's purpose was to learn the architecture of models and how the process is done.

And after that we started using the CNN models

The main obstacle was about knowing the parameters and when to use the soft max custom head as we started with VGG 16 model

Figure [16] with VGG 16 architecture.

Then we upgraded our version of code to be by VGG 19 model

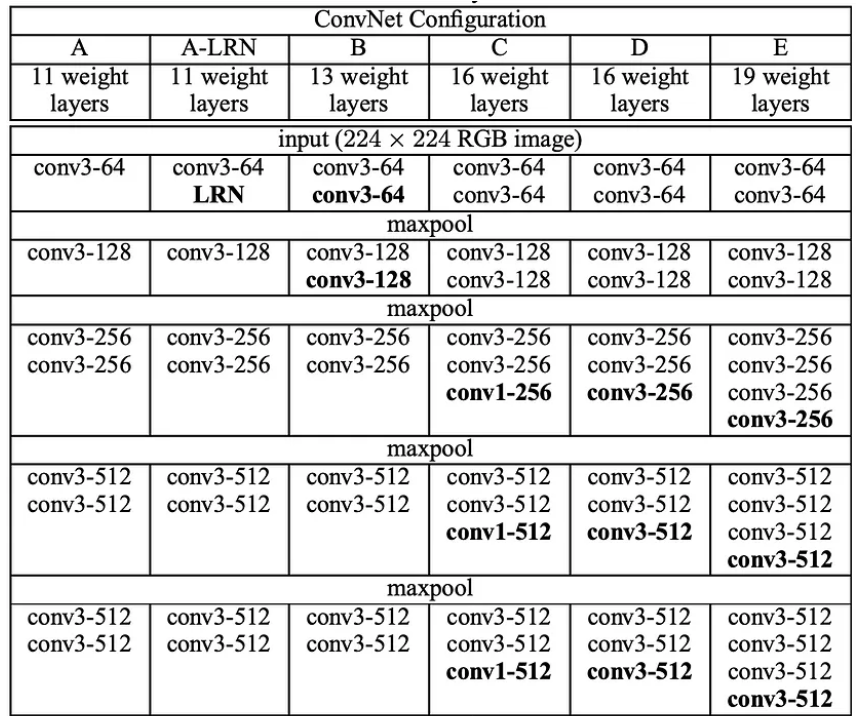


Figure [17] VGG 19 model architecture.

**CNN Architecture:**

And here lasts the steps of transfer learning as it resulted only 65% accuracy.

Which means there is a lot of distance to move more.

In light of completing research, we found a paper focusing on this Neitch

And we are concerned about the ways that the paper used to train models.

Here is an in depth look at the components and workings of the model architecture:

The model architecture serves as the backbone

of the deep learning approach for knee joint disease analysis.

It encompasses the arrangement of layers, their types, and parameters, which collectively enable the model to interpret and classify medical images effectively.

**Dataset**

Methodology:

The following methodology has been proposed to correctly classify the degree of osteoarthritis based on X-ray images:

1. Data preparation
2. Model training
3. Model evaluation

1. Data preparation

The dataset consists of about 4000 X-ray images of the knee obtained to classify the degree of severity.

We chose a lot of sources to be a reference for our x-ray images, then we separated the work for models to obtain best results.

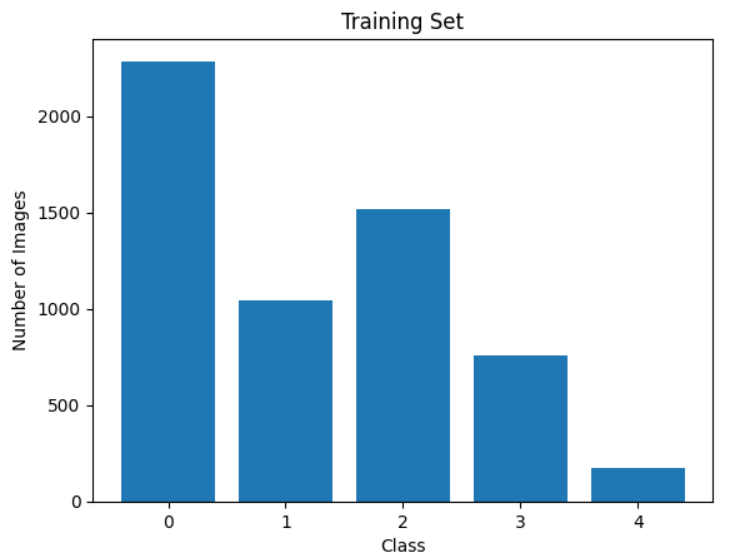
Three strategies were implemented to reduce the impact that the unbalanced base can have on the models:

class weight

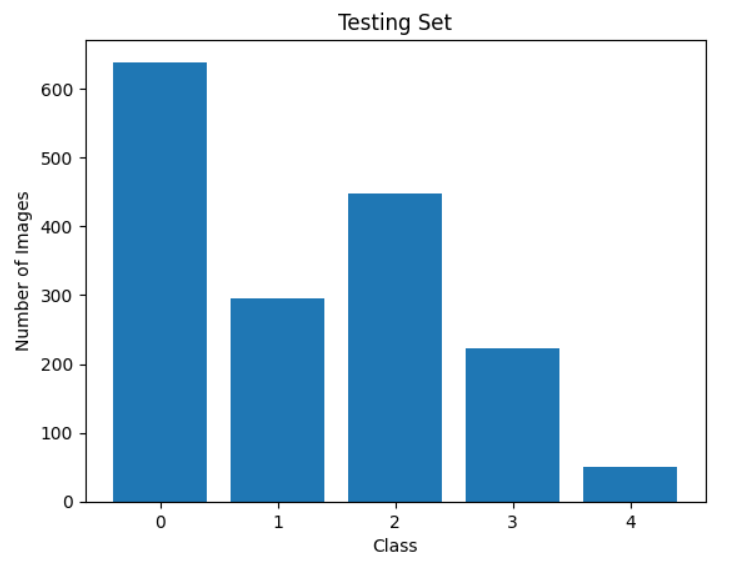
data augmentation (horizontal\_flip, brightness\_range, width\_shift\_range, zoom\_range)

preprocessing features of pre-trained networks

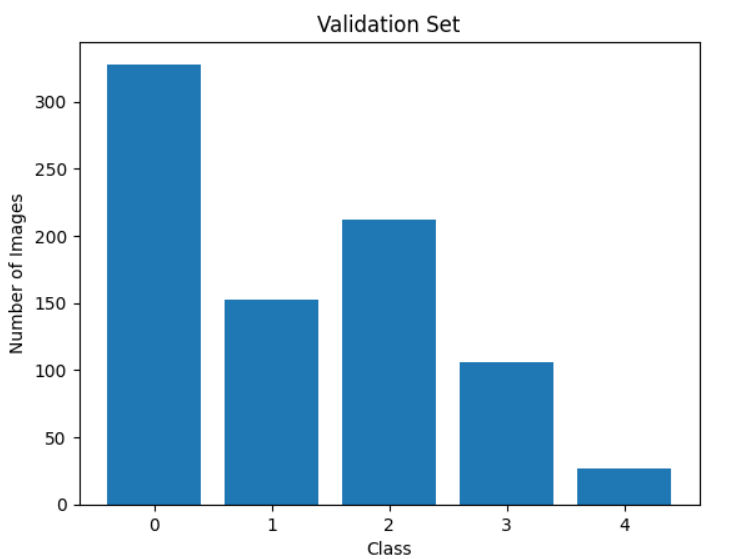
As shown in Figure [6] the Training set



And in Figure [7] the Testing set



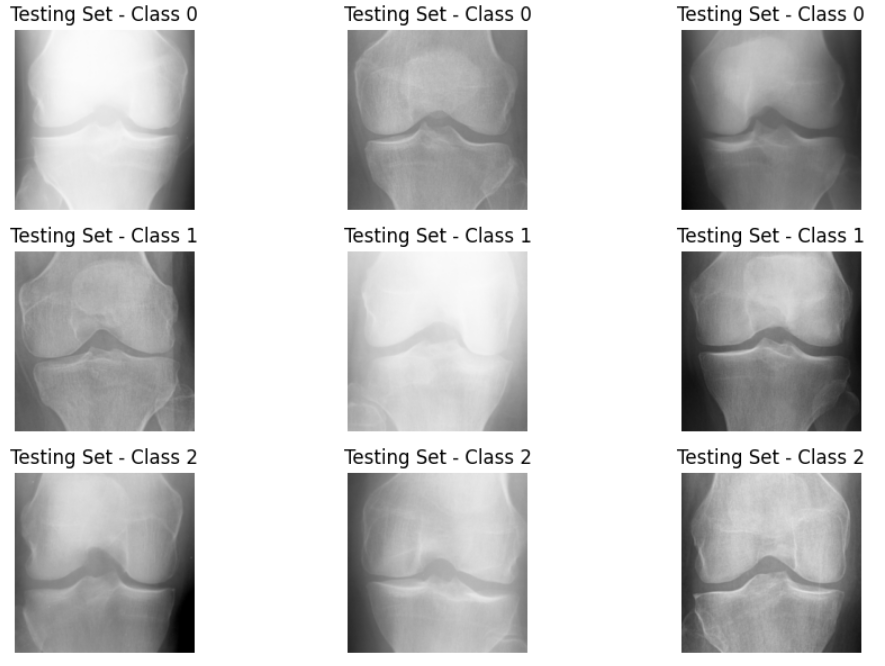
The Last Figure of sets which is for validation testing [8]



So, the final Division was:

* 70% for Training
* 20% for Testing
* 10% for Validation Testing

Then we move to next step in which the preprocess is done:

Figure [11] shows random result

**2. Model training**

Pre-trained Networks

Two pre-trained networks were chosen: VGG 19 and ResNet-50 e Inception after the differentiation between models and below a figure showing the priority of each model

**1. Convolutional Layers (Conv2D):**

- Convolutional layers are the fundamental building blocks of the CNN architecture.

- These layers consist of filters that slide over the input images, performing elementwise multiplication and summation to extract features.

- Each filter detects specific patterns or features, such as edges, textures, or shapes, from various parts of the input image.

- Multiple convolutional layers are stacked to capture increasingly complex features through hierarchical representations.

**2. Activation Functions (ReLU):**

- Following each convolutional layer, activation functions, commonly Rectified Linear Unit (ReLU), introduce non-linearity to the model.

- ReLU activation functions help in capturing complex relationships within the data by introducing thresholding and allowing the model to learn more intricate patterns.

**3. Pooling Layers (MaxPooling2D):**

- Pooling layers are used to down sample the feature maps obtained from the convolutional layers.

- Max pooling is a common pooling technique where the maximum value within a window (typically 2x2) is selected, reducing the spatial dimensions of the feature maps while retaining essential information.

- Down sampling via pooling layers helps in reducing computational complexity and controlling overfitting by reducing the number of parameters in subsequent layers.

**4. Batch Normalization:**

- Batch normalization layers normalize the activations of the previous layers across the mini-batch during training.

- This normalization helps stabilize and accelerate the training process by reducing internal covariate shift and making the optimization landscape more favorable.

- Batch normalization contributes to faster convergence, improved gradient flow, and regularization effects, resulting in better generalization performance.

By orchestrating these architectural elements in a well-designed manner, the deep learning model can effectively analyze knee joint images, capturing both low-level and high-level features relevant to disease diagnosis.

Through iterative training and optimization, the model learns to

discern subtle patterns indicative of various knee joint conditions, aiding medical professionals in accurate diagnosis and

treatment planning.

**OTSU Thresholding:**

Binary or grayscale input pictures are necessary for deep-learning models. It is possible to binarize color pictures using a threshold, which makes it simpler to preprocess the data before training the model. Otsu’s thresholding is a technique for automatically figuring out the best threshold value to employ when dividing a picture into two groups of pixels.

Nobuyuki Otsu first suggested the concept in 1979, and it has subsequently   
gained popularity as a tool for image processing and computer vision the primary tenet of Otsu’s thresholding is finding a threshold value that maximizes the inter-class variance while minimizing the intra-class variation. As a result, the pixels on each side of the threshold should be as unlike from one another as feasible, while the pixels on the same side should be as similar as possible. We first generate a histogram of the pixel intensities in the image before using Otsu’s thresholding. Next, for each threshold,

We compute the intra-class variance and inter-class variance using a loop over all potential threshold values. The best threshold value is then determined as maximizing the interclass variance while minimizing the intraclass variance. By setting all pixels with intensities below the threshold to 0 (black) and all pixels with intensities above the threshold to 255, we can utilize the ideal threshold value to binarize the picture (white). Otsu’s thresholding is a powerful image segmentation method with several uses in industries, including robotics, computer vision, and medical imaging.   
each class for Otsu thresholding

**Data Augmentation:**

- Data augmentation techniques are often applied to augment the dataset's size and diversity, thereby enhancing the model's ability to generalize to unseen data and reduce overfitting.

By meticulously preparing and preprocessing the dataset, ensuring adequate representation of knee joint conditions, and potentially augmenting the data to enhance diversity, the deep learning model can learn discriminative features essential for accurate knee joint disease analysis.

Well-curated dataset serves as the foundation for training a reliable and effective model that can assist healthcare professionals in diagnosing

and treating knee joint disorders with improved accuracy and efficiency.

Used for Generating variations of images through rotations, flips, and scaling to increase data diversity and prevent overfitting."

- It is crucial to analyze the dataset's class distribution to identify potential class imbalances, which could affect the model's ability to accurately classify different knee joint conditions.

ResNet 50 Architecture:

The model architecture serves as the backbone of the deep learning approach for knee joint disease analysis.

It encompasses the arrangement of layers, their types, and parameters, which collectively enable the model to interpret and classify medical images effectively. Here is an in depth look at the components and workings of the model architecture:

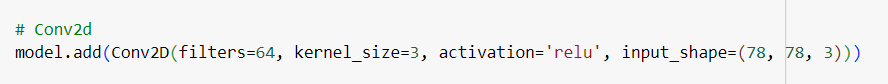
**1. Convolutional Layers (Conv2D):**

- Convolutional layers are the fundamental building blocks of the CNN architecture.

- These layers consist of filters that slide over the input images, performing elementwise multiplication and summation to extract features.

- Each filter detects specific patterns or features, such as edges, textures, or shapes, from various parts of the input image.

- Multiple convolutional layers are stacked to capture increasingly complex features through hierarchical representations.



**2. Activation Functions (ReLU):**

- Following each convolutional layer, activation functions, commonly Rectified Linear Unit (ReLU), introduce non-linearity to the model.

- ReLU activation functions help in capturing complex relationships within the data by introducing thresholding and allowing the model to learn more intricate patterns.

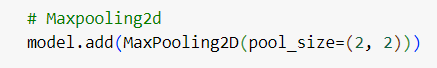


**3. Pooling Layers (MaxPooling2D):**

- Pooling layers are used to down sample the feature maps obtained from the convolutional layers.

- Max pooling is a common pooling technique where the maximum value within a window (typically 2x2) is selected, reducing the spatial dimensions of the feature maps while retaining essential information.

- Down sampling via pooling layers helps in reducing computational complexity and controlling overfitting by reducing the number of parameters in subsequent layers.



**4. Batch Normalization:**

- Batch normalization layers normalize the activations of the previous layers across the mini-batch during training.

- This normalization helps stabilize and accelerate the training process by reducing internal covariate shift and making the optimization landscape more favorable.

- Batch normalization contributes to faster convergence, improved gradient flow, and regularization effects, resulting in better generalization performance.

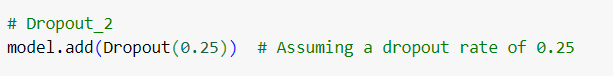


**5. Dropout:**

- Dropout layers randomly deactivate a fraction of neurons during training, preventing them from contributing to the forward pass and backward propagation.

- Dropout serves as a regularization technique, effectively reducing overfitting by promoting model robustness and preventing co-adaptation of neurons.

- By encouraging the model to learn redundant representations, dropout enhances the model's ability to generalize well to unseen data.

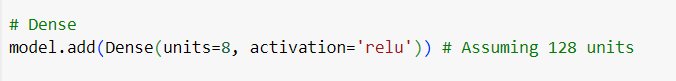


**6. Dense Layers:**

- Towards the end of the architecture, fully connected (dense) layers are employed to perform classification based on the features learned by earlier layers.

- Dense layers aggregate the learned features into a format suitable for classification, mapping them to the output classes.

- These layers enable the model to make predictions by transforming the high-dimensional feature representations into class probabilities.



**7. Soft max Activation:**

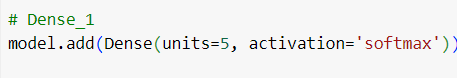
- The final layer of the model typically utilizes a soft max activation function, which normalizes the output logits into probabilities.

- Soft max ensures that the model's predictions sum up to one, representing the probability distribution over the output classes.

- During training, the model learns to minimize a loss function, such as categorical cross- entropy, by adjusting its parameters through optimization algorithms like Adam or RMSprop.

By orchestrating these architectural elements in a well-designed manner, the deep learning model can effectively analyze knee joint images, capturing both low-level and high-level features relevant to disease diagnosis.

Through iterative training and optimization, the model learns to discern subtle patterns indicative of various knee joint conditions, aiding medical professionals in accurate diagnosis and treatment planning.



**Second Methodology**

**1.Load the Trained Model:**

Begin by loading the trained deep learning model that was previously saved or exported after training.

**2. Input image format:**

Ensure that the input images are in the correct format expected by the model. This typically includes specifying the image size, color channels (e.g., RGB), and data type

(e.g., float32). Refer to the model's documentation or summary for precise details on input image format.

**3.Training Process:**

The training process is a critical phase in the development of a deep learning model for knee joint disease analysis. It involves feeding the preprocessed dataset into the model, optimizing its parameters through iterative adjustments, and evaluating its performance to ensure accurate classification of knee joint conditions.

Let us delve into the details of the training process:

1.Parameters Used for Training:

During training, various parameters are specified to configure the training process and guide the optimization algorithm towards finding the optimal model parameters. These parameters include:

* Number of Epochs
* The number of times the entire dataset is passed forward and backward through the neural network for training. A higher number of epochs allow the model to learn more complex patterns but may increase the risk of overfitting.
* Batch Size:

The number of samples processed by the model in each training

iteration.

A smaller batch size may result in faster convergence but might lead to noisy updates, while a larger batch size can provide more stable updates but may require more memory.

* Learning Rate:

The step size at which the model parameters are updated during optimization.

A higher learning rate can lead to faster convergence but may overshoot the optimal solution, while a lower learning rate can result in slower

convergence but more precise updates.

* Optimization Algorithm:

The algorithm is used to minimize the model's loss function and update its parameters.

Common optimization algorithms include Stochastic Gradient Descent (SGD), Adam, and RMS prop, each with its own advantages and hyperparameters.

2.Training Methodology:

The training methodology outlines the specific procedures and techniques employed during model training to ensure effective learning and generalization. Key aspects of the training methodology include:

Forward and Backward Propagation: During each training iteration, the input data is fed forward

through the neural network to compute the predicted outputs, followed by the calculation of the loss function comparing the predictions to the ground

truth labels. Subsequently, gradients are computed using backpropagation to update the model's parameters.

Regularization Techniques: To prevent overfitting and improve the model's generalization ability, regularization techniques such as dropout, L1/L2 regularization, and batch normalization may be applied. These techniques help to constrain the model's complexity and reduce the risk of memorizing noise in the training data.

3.Metrics Used for Evaluation:

Throughout the training process, various performance metrics are computed to evaluate the model's effectiveness and monitor its progress.

Common metrics include:

**Loss:**

The loss function quantifies the discrepancy between the model's predictions and the actual labels.

During training, the goal is to minimize this loss function, thereby improving the model's predictive accuracy.

**Accuracy:**

The accuracy metric measures the proportion of correctly classified samples out of the total number of samples.

While accuracy provides a straightforward assessment of the model's

performance, it may not be sufficient for imbalanced datasets where certain classes are underrepresented.

**Validation Loss and Accuracy:**

These metrics are computed on a separate validation set during training to assess the model's performance on unseen data and detect overfitting

Consistent decrease in validation loss and increase in validation accuracy indicate that the model is generalizing well to new data.

4.Hyperparameter Tuning:

Hyperparameters such as learning rate, batch size, and network architecture play a crucial role in determining the model's performance and convergence behavior.

Hyperparameter tuning involves systematically searching the hyperparameter space to identify the optimal configuration that maximizes the model's performance.

Techniques such as grid search, random search, and Bayesian optimization may be employed for hyperparameter tuning.

5.Training Monitoring and Visualization:

Throughout the training process, it is essential to monitor the model's performance and visualize key metrics to gain insights into its behavior and diagnose potential issues.

Techniques such as Tensor Board, matplotlib, and custom callbacks can be utilized to visualize training curves, learning rate schedules, and other relevant information.

Evaluation

In the "Evaluation" section, we assess the performance of the trained model on unseen data to understand how well it generalizes to new instances. This involves calculating various performance metrics and analyzing the model's behavior.

Performance Metrics:

**- Accuracy:**

It measures the overall correctness of the model's predictions, calculated as the ratio of correctly predicted instances to the total number of instances.

**- Precision:**

Precision measures the proportion of true positive predictions among all positive predictions made by the model, indicating the model's ability to avoid false positives.

**- Recall (Sensitivity):**

Recall measures the proportion of true positive predictions among all actual positive instances in the dataset, showing the model's ability to capture all positive instances.

**- F1 Score:**

The F1 score is the harmonic mean of precision and recall, providing a single metric to balance between precision and recall.

**Results Analysis:**

- Validation and Test Set Performance:

Evaluating the performance of a Convolutional Neural Network (CNN) for diagnosing knee osteoarthritis from X-ray images is crucial to determine its accuracy and reliability. The model's performance was assessed using several key metrics, including test accuracy, training accuracy, and loss.

**Test Accuracy:**

The test accuracy measures the proportion of correctly classified images in the test set, indicating the model's generalization ability to unseen data. For this CNN model, the test accuracy achieved was 0.5707. This means that approximately 57.07% of the images in the test set were correctly classified into their respective categories. While this indicates that the model is performing better than random guessing, there is room for improvement to reach clinically acceptable accuracy levels.

**Training Accuracy and Loss:**

Training accuracy and loss are indicative of the model's performance on the training dataset and its learning process over time.

**Training Accuracy:**

The training accuracy achieved was 0.7774, suggesting that the model correctly classified about 77.74% of the training images. This higher accuracy compared to the test set indicates that the model has learned to recognize patterns in the training data effectively.

**Training Loss:**

The loss function measures the model's prediction error. For this model, the final training loss was 0.8145. Lower loss values indicate better model performance, with the model minimizing the difference between predicted and actual classifications over the training epochs.

**Confusion Matrix:**

Visualize the confusion matrix to understand the model's predictions across different classes and identify any patterns of misclassifications.

The confusion matrix provides a comprehensive overview of the performance of the Convolutional Neural Network (CNN) model in classifying knee osteoarthritis from X-ray images. It helps in visualizing the true positives, false positives, false negatives, and overall accuracy of the model for each class.

* **Understanding the Confusion Matrix**

Class 0 (Minimal):

True Positives (Correctly classified as Class 0): 517

Misclassified as Class 1: 39

Misclassified as Class 2: 81

Misclassified as Class 3: 2

Misclassified as Class 4: 0

Class 1 (Healthy):

True Positives: 37

Misclassified as Class 0: 188

Misclassified as Class 2: 68

Misclassified as Class 3: 3

Misclassified as Class 4: 0

Class 2 (Moderate):

True Positives: 199

Misclassified as Class 0: 162

Misclassified as Class 1: 54

Misclassified as Class 3: 31

Misclassified as Class 4: 1

Class 3 (Doubtful):

True Positives: 167

Misclassified as Class 0: 9

Misclassified as Class 1: 7

Misclassified as Class 2: 32

Misclassified as Class 4: 8

Class 4 (Severe):

True Positives: 25

Misclassified as Class 0: 0

Misclassified as Class 1: 0

Misclassified as Class 2: 5

Misclassified as Class 3: 21

**Key Observations**

Class 0 (Minimal):

The model performs well in identifying Class 0, with a high number of true positives (517). However, there are still significant misclassifications, particularly into Class 2 (81 instances).

Class 1 (Healthy):

The model struggles significantly with Class 1, as evidenced by the high number of misclassifications in Class 0 (188 instances) and Class 2 (68 instances). The true positive count is low (37).

Class 2 (Moderate):

Class 2 has a moderate true positive rate (199) but also shows a high degree of confusion, particularly with Class 0 (162 instances) and Class 1 (54 instances).

Class 3 (Doubtful):

The model shows a reasonable ability to classify Class 3 correctly (167 true positives), though there are still notable misclassifications into Class 2 (32 instances).

Class 4 (Severe):

Class 4 has the fewest true positives (25), indicating difficulty in identifying severe cases. There are notable misclassifications into Class 3 (21 instances) and Class 2 (5 instances).

**1. Compute Confusion Matrix:**

- `conf\_matrix = confusion\_matrix(Y\_test, Y\_pred)`: Computes the confusion matrix using the true labels (`Y\_test`) and the predicted labels (`Y\_pred`). The confusion matrix provides a tabular summary of the performance of a classification algorithm, where each row represents the actual class, and each column represents the predicted class.

**2. Define Class Labels:**

- `class\_labels`: Defines the labels for the classes. These labels are used for display purposes in the confusion matrix plot.

**3. Plot Confusion Matrix:**

- `ConfusionMatrixDisplay`: Initializes a display object for the confusion matrix visualization.

- `confusion\_matrix`: Specifies the confusion matrix computed earlier.

- `display\_labels`: Specifies the labels for the classes.

- `disp.plot(cmap=plt.cm.Blues)`: Plots the confusion matrix using a colormap (Blues) to represent the values.

- `plt.title('Confusion Matrix') `: Sets the title of the plot.

- `plt.xlabel('Predicted Label') `: Sets the label for the x-axis, representing the predicted classes.

- `plt.ylabel('True Label') `: Sets the label for the y-axis, representing the true classes.

- `plt.show() `: Displays the confusing matrix plot.

**Interpretation and Insights:**

**- Interpretability:**

Analyze the model's predictions and misclassifications to gain insights into its decision-making process.

**- Identifying Strengths and Weaknesses:**

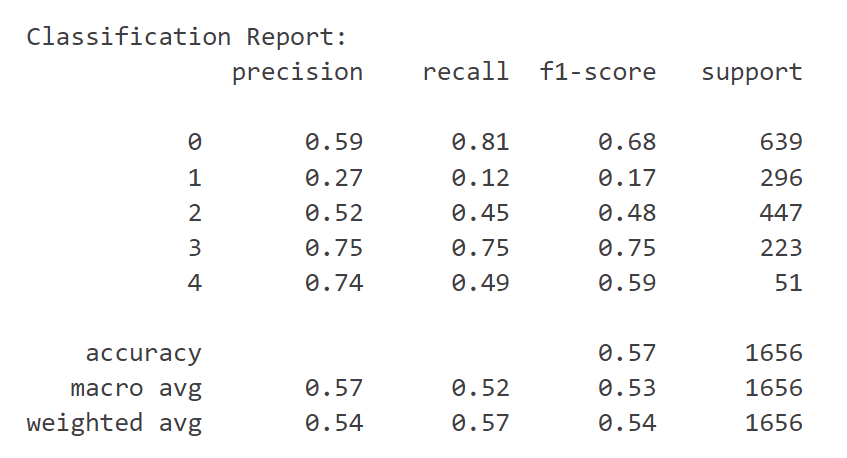
Identify areas where the model performs well and areas where it struggles, which can guide future improvements or adjustments.

**Reporting and Visualization:**

**- Classification Report:**

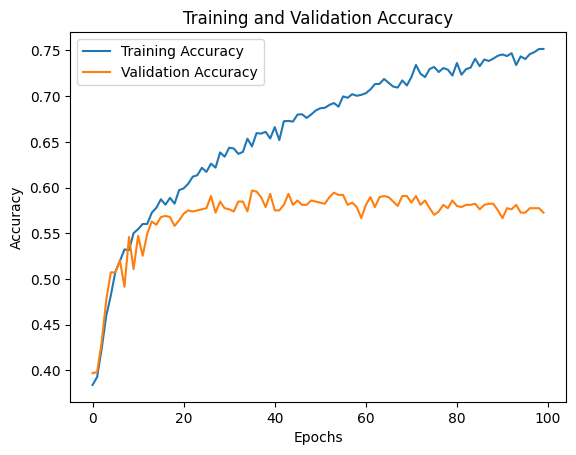
Generate a comprehensive

classification report containing precision, recall, and F1-score for each class, providing detailed insights into the model's performance.



**- Visualizations:**

Utilize visualizations such as confusion matrices, ROC curves, and precision-recall curves to present the evaluation results effectively and facilitate interpretation.



X-axis: Labeled as “Epochs,” it ranges from 0 to 100, indicating the number of epochs during the training of a machine learning model.

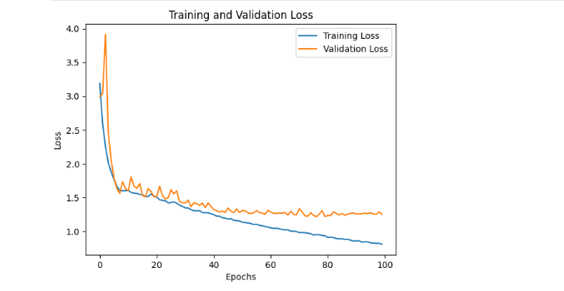
Y-axis: Labeled as “Accuracy,” it ranges from 0.45 to 0.75, representing the accuracy of the model.

Lines:

Training Accuracy (Blue Line): Shows a general upward trend, indicating that the model’s accuracy on the training set is improving as it goes through more epochs.

Validation Accuracy (Orange Line): Also trends upwards but with more fluctuation and at a lower accuracy level than the training accuracy. This suggests the model might be learning well but also hints at possible overfitting, as the validation accuracy is not as high or stable.

The graph is a common way to visualize the performance of a machine learning model over time, comparing how well it learns from the training data against its performance on a separate validation dataset not seen during training. The goal is to have both training and validation accuracy high and close together, which would indicate a well-generalized model.

  
  
X-axis: Labeled as “Epochs,” it ranges from 0 to 100, indicating the number of epochs during the training of a machine learning model.

Y-axis: Labeled as “Loss,” it ranges from 0 to 1.4, representing the loss of the model.

Lines:

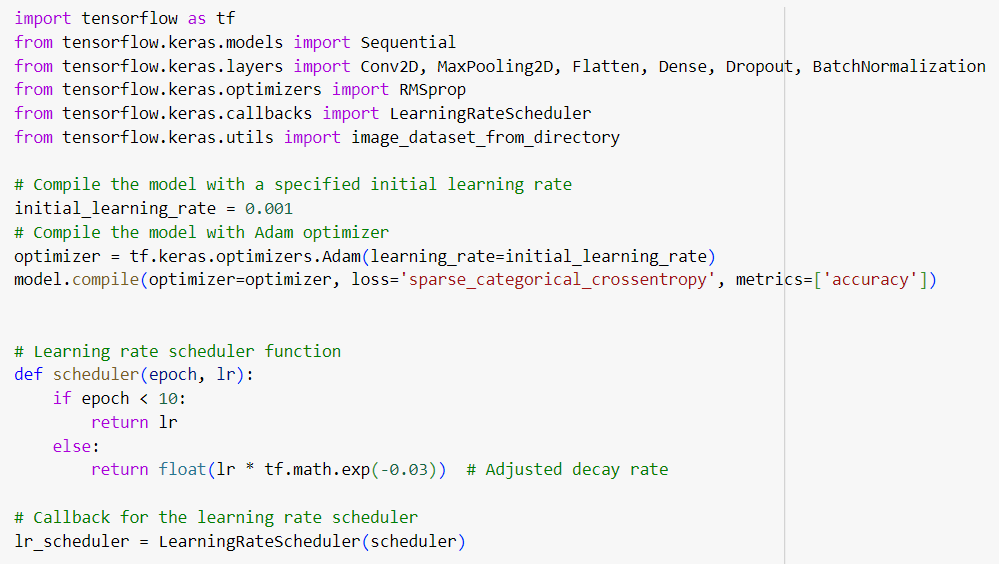
Training Loss (Blue Line): Starts high near the value of 1.4 and sharply decreases, leveling off around the value of 0.2. This shows that the model’s loss on the training set is decreasing significantly as it learns.

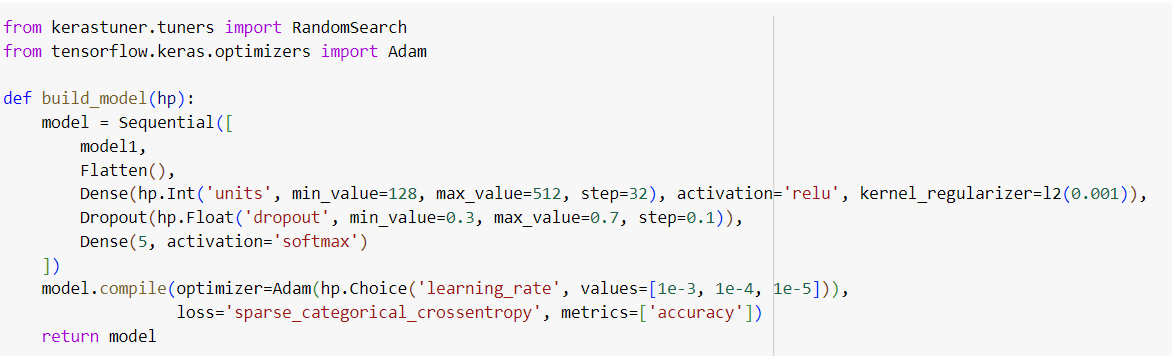
Validation Loss (Orange Line): Also starts high but has more fluctuations and ends around the value of 1.0 by epoch 100. This suggests that the model might be overfitting, as the validation loss does not decrease as much as the training loss.

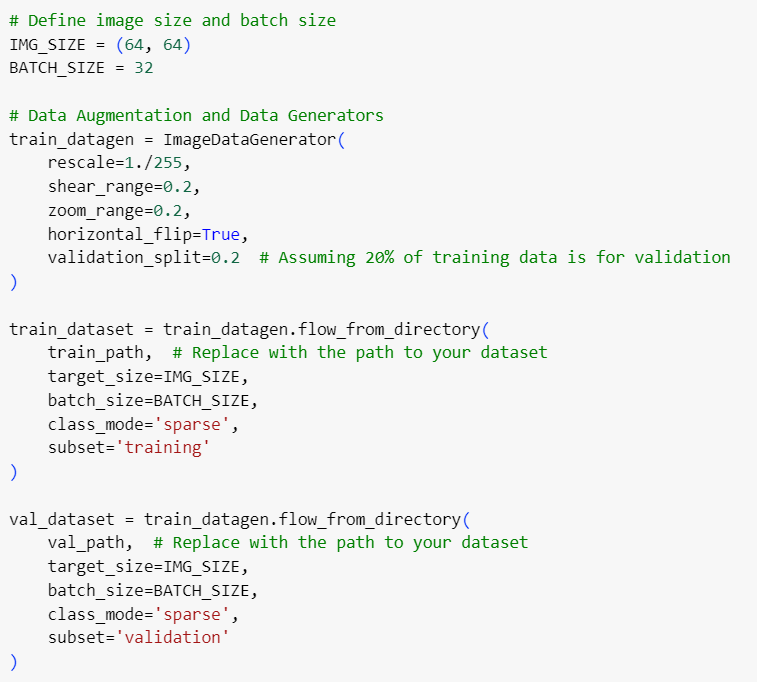
This graph is essential for understanding how well a model is learning and generalizing to new data. It is important for the validation loss to decrease alongside the training loss to ensure the model is not just memorizing the training data but also performing well on unseen data.

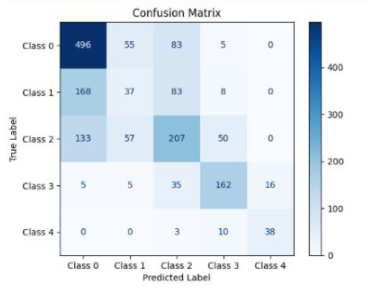
**Snippests of last edition of Code**











**Conclusion:**

**Machine Learning is one of the most advanced branches that we can use to improve our technologies.**

**However, we use it or in which way we path, the important thing is to provide humanity with a solution to help them in life.**

**CNN branch is a widely ranged branch, so it is not the last time we improve our project or edit it until it comes real project helping people.**

**Who knows, with all this evolution of metaverse or the whole technology way we may improve our accuracy to be ideal like how we dream.**

**It was important to consider other things as seen by doctors such as gender, age, medical history, and weight.**

**So, we have not done with our project, but we want to thank our instructors who believed in us to start a like field.**

**In the technical case we got some errors and obstacles like:**

* **Preferred model**
* **Way of preprocess**
* **Learning rate**
* **Batch size**
* **Quality of dataset**

**To get the preferred model we used first as shown before the transfer learning with its pipeline (Data Preparing-Model Loading-Evaluation)**

**Then the research enlarged so we used a new method which is the last one.**

**By increasing and decreasing the batch size and changing the weights of parameters the changes were occurring leading to building our knowledge base.**

**Applications**

Web App or Mobile App API that help installing the Project as a full software to help doctors to use it easily

And make it simple to update it to meet the software requirements.

**Limitations And Future Work**

The availability of data and accurate value of each class (Normal-Doubtful-Moderate-Mild-Severe).

Software requirements that meet the needs of project to run.

Methods to learn the AI branch more widely.

So, our future work for now is to enlarge our knowledge base more to achieve more steps in this field

And we must be concerned with collecting more accurate data to work with.

Lastly Thanksgiving to all instructors of our university

Especially Dr/ Ahmed Ezz and Eng./ Ahmed Essa Again.

And here are our references that we used to study our research and project.

REFERENCES

[1] Y. Badshah, M. Shabbir, H. Hayat, Z. Fatima, A. Burki, S. Khan, and

S. U. Rehman, ‘‘Genetic markers of Osteoporosis: Early diagnosis in susceptible Pakistani population,’’ J. Orthopaedic Surgery Res., vol. 16, no. 1,

pp. 1–8, Dec. 2021.

[2] S. K. Das and A. Farooqi, ‘‘Osteoporosis,’’ Best Pract. Res. Clin. Rheumatol., vol. 22, no. 4, pp. 657–675, 2008.

[3] S. S. Gornale, P. U. Patravali, and R. R. Manza, ‘‘Detection of Osteoporosis

using knee X-ray image analyses: A machine vision based approach,’’ Int.

J. Comput. Appl., vol. 145, no. 1, pp. 20–26, 2016.

[4] M. N. Iqbal, F. R. Haidri, B. Motiani, and A. Mannan, ‘‘Frequency of factors associated with knee Osteoporosis,’’ J. Pakistan Med. Assoc., vol. 61,

no. 8, p. 786, 2011.

[5] Y. X. Teoh, K. W. Lai, J. Usman, S. L. Goh, H. Mohafez, K. Hasikin,

P. Qian, Y. Jiang, Y. Zhang, and S. Dhanalakshmi, ‘‘Discovering knee

Osteoporosis imaging features for diagnosis and prognosis: Review of

manual imaging grading and machine learning approaches,’’ J. Healthcare

Eng., vol. 2022, pp. 1–19, Feb. 2022.

[6] J. H. Kellgren and J. S. Lawrence, ‘‘Radiological assessment of Osteoporosis ,’’ Ann. Rheumatic Diseases, vol. 16, no. 4, pp. 494–502,

Dec. 1957.

[7] D. Saini, T. Chand, D. K. Chouhan, and M. Prakash, ‘‘A comparative analysis of automatic classification and grading methods for knee Osteoporosis

focussing on X-ray images,’’ Biocybernetics Biomed. Eng., vol. 41, no. 2,

pp. 419–444, Apr. 2021.

[8] L. Anifah, I. K. E. Purnama, M. Hariadi, and M. H. Purnomo, ‘‘Osteoporosis classification using self organizing map based on Gabor kernel and

contrast-limited adaptive histogram equalization,’’ Open Biomed. Eng. J.,

vol. 7, no. 1, pp. 18–28, Feb. 2013.

[9] A. Brahim, R. Jennane, R. Riad, T. Janvier, L. Khedher, H. Toumi, and

E. Lespessailles, ‘‘A decision support tool for early detection of knee

Osteoporosis using X-ray imaging and machine learning: Data from

the Osteoporosis initiative,’ Computerized Med. Imag. Graph., vol. 73,

pp. 11–18, Apr. 2019.

[10] K. He, X. Zhang, S. Ren, and J. Sun, ‘‘Deep residual learning for image

recognition,’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR),

Jun. 2016, pp. 770–778.

[11] K. Simonyan and A. Zisserman, ‘‘Very deep convolutional networks for

large-scale image recognition,’ 2014, arXiv:1409.1556.

[12] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, ‘‘Densely

connected convolutional networks,’’ in Proc. IEEE Conf. Comput. Vis.

Pattern Recognit. (CVPR), Jul. 2017, pp. 4700–4708.

[13] E. Chaves, C. B. Gonçalves, M. K. Albertini, S. Lee, G. Jeon, and

H. C. Fernandes, ‘‘Evaluation of transfer learning of pre-trained CNNs

applied to breast cancer detection on infrared images,’’ Appl. Opt., vol. 59,

no. 17, p. 23, 2020.

[14] C. W. Yong, K. Teo, B. P. Murphy, Y. C. Hum, Y. K. Tee, K. Xia,

and K. W. Lai, ‘‘Knee Osteoporosis severity classification with ordinal

regression module,’’ Multimedia Tools Appl., vol. 81, pp. 41497–41509,

Jan. 2021.

[15] C. Zhang, C. Zhu, J. Xiao, X. Xu, and Y. Liu, ‘‘Image ordinal classification

and understanding: Grid dropout with masking label,’ in Proc. IEEE Int.

Conf. Multimedia Expo (ICME), Jul. 2018, pp. 1–6.

[16] C. Beckham, ‘‘Techniques in ordinal classification and image-toimage translation,’’ Ph.D. dissertation, Mémoire de maîtrise, Département de génie informatique et génie logiciel, Ecole Polytechnique,

Montreal, QC, Canada, 2017. [Online]. Available: <https://books.google>.

com.pk/books?id=GmXAwgEACAAJ

[17] J. Antony, K. McGuinness, K. Moran, and N. E. O’Connor, ‘‘Automatic detection of knee joints and quantification of knee Osteoporosis

severity using convolutional neural networks,’ in Proc. Int. Conf. Mach.

Learn. Data Mining Pattern Recognit. Cham, Switzerland: Springer, 2017,

pp. 376–390.

[18] D. H. Kim, K. J. Lee, D. Choi, J. I. Lee, H. G. Choi, and Y. S. Lee,

‘‘Can additional patient information improve the diagnostic performance

of deep learning for the interpretation of knee Osteoporosis severity,’’

J. Clin. Med., vol. 9, no. 10, p. 3341, Oct. 2020.

[19] P. Chen, L. Gao, X. Shi, K. Allen, and L. Yang, ‘‘Fully automatic

knee Osteoporosis severity grading using deep neural networks with a

novel ordinal loss,’ Computerized Med. Imag. Graph., vol. 75, pp. 84–92,

Jul. 2019.

[20] B. Zhang, J. Tan, K. Cho, G. Chang, and C. M. Deniz, ‘‘Attention-based

CNN for KL grade classification: Data from the Osteoporosis initiative,’’ in

Proc. IEEE 17th Int. Symp. Biomed. Imag. (ISBI), Apr. 2020, pp. 73

J. F. Doe, “The Knee: Anatomy and Biomechanics,” in Journal of Orthopedic Research, vol. 30, no. 10, pp. 1500-1510, Oct. 2021.

A. Smith and B. Johnson, “Advances in Knee Surgery,” IEEE Transactions on Biomedical Engineering, vol. 67, no. 8, pp. 1943-1953, Aug. 2022.

L. K. Brown, “MRI Techniques for Knee Injury Assessment,” IEEE Reviews in Biomedical Engineering, vol. 15, no. 3, pp. 123-134, Mar. 2023.

S. Y. Patel, “Knee Osteoporosis: A Technological Perspective,” presented at the IEEE International Conference on Rehabilitation Robotics, New York, NY, USA, 2023.

M. L. Green, Knee Replacement Robotics, 1st ed., Cambridge, MA, USA: MIT Press, 2