Click me to view **Presentation** Video

Click me to view **Demo Result** Video

FINAL YEAR PROJECT 2

CLASSIFICATION OF KNEE OSTEOTHRITIS USING DEEP LEARNING MODELS FUSED WITH HANDCRAFTED FEATURES

PREPARED BY : LIM YONG WEI (B22EC0025)

SUPERVISOR: TS.DR.CHAN WENG HOWE

COORDINATOR: DR.ROZILAWATI BINTI DOLLAH @ MD.ZAIN

EXAMINER 1: DR. AZURAH BT. A.SAMAH

EXAMINER 2: DR. SHAMINI A/P RAJA KUMARAN

PRESENTATION OUTLINE

1 Problem formulation & Data collection Literature review

- Research Problem
- Research Goal and Objective
- Research Scope and Importance
- Literature Review
- Data Collection

2 Research Design and Implementation

- Architecture Diagram & Propose Model
- Pre-process Images
- Split & Aug Data
- Convolutional Neutral Network
- Handcrafted Method
- Feature-level Fusion

3 Develop FFNN Model

- Feed-Forward Neutral Network on 3 Models

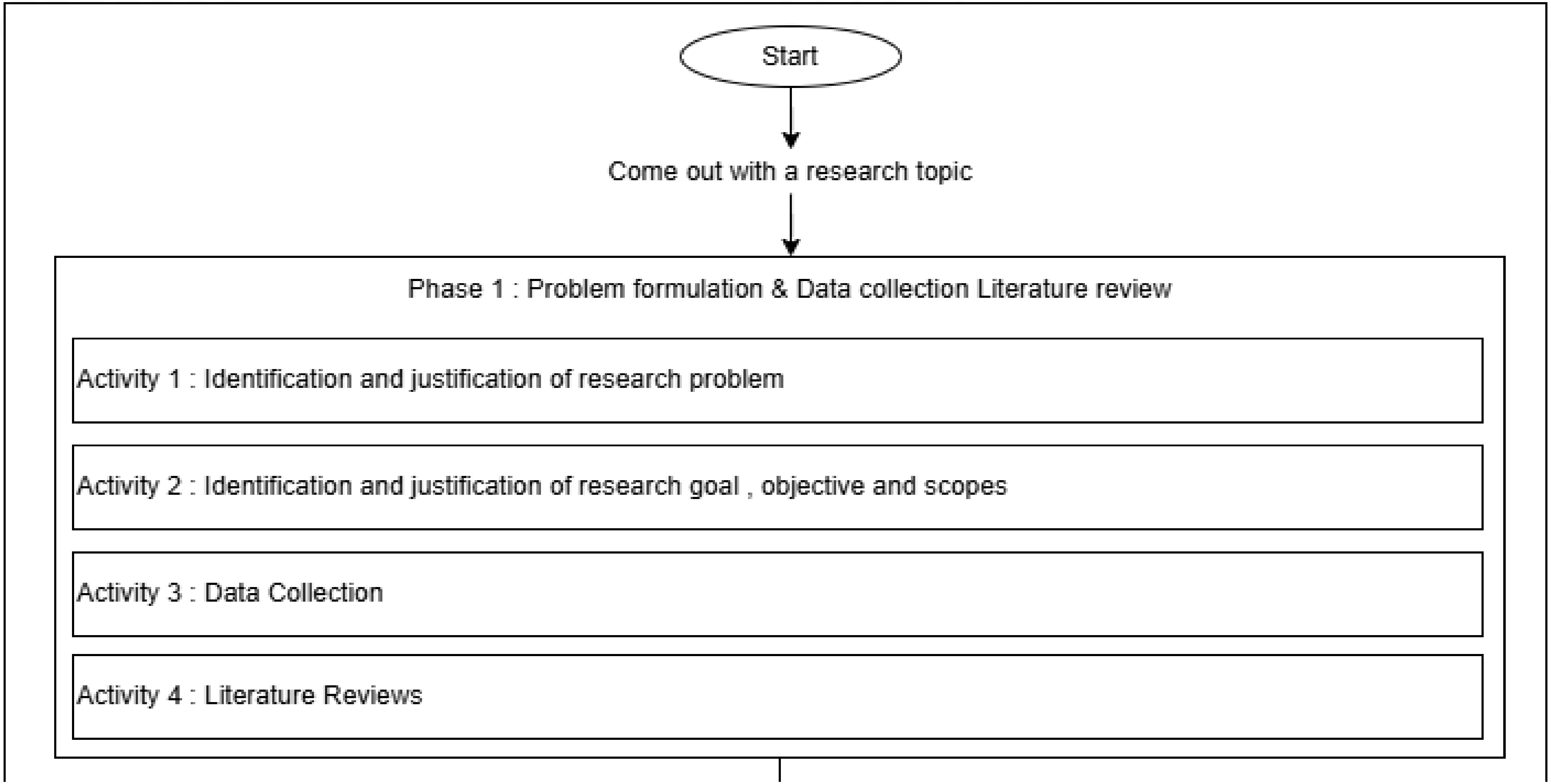
4 Testing and Evaluation

- Performance measurement on FFNN classify
- Comparison of FFNN performance
- Discussion

5 Conclusion

- Achievement on Objective
- Research Constraint
- Suggest Improvement and Future Work

Research Framework – Phase 1



Background of Study

What is KOA ?

knee osteoarthritis, is a common **musculoskeletal disease**

Why does KOA occur ?

due to the **degeneration of joint cartilage** in the knee, which leads to **joint pain** and disability

Who will interact with KOA?

primarily affects **elderly individuals**, but it can also impact those with risk factors such as **obesity, prior joint injuries, or a family history of osteoarthritis**

Problem Statement

1. Need for ***Improved Diagnostic*** Techniques

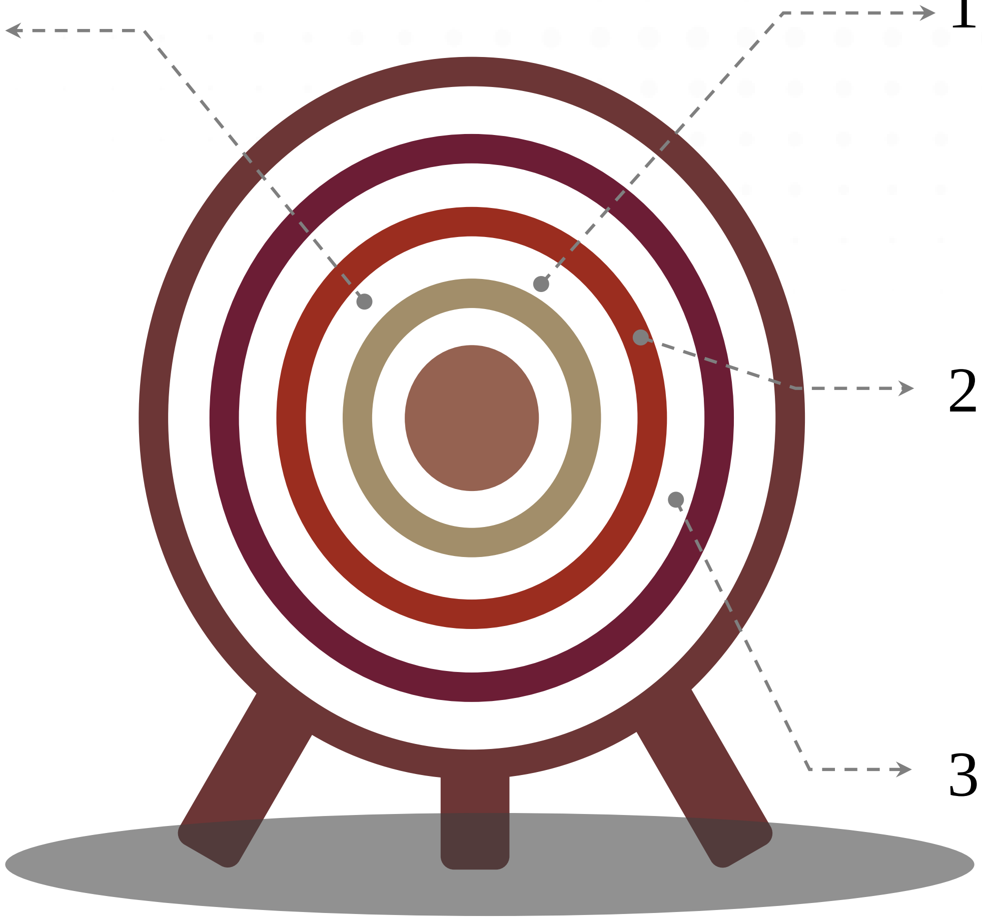
Predicting KOA is crucial because it's still caused by ***many factors***, ***affects quality of life, and varies widely in how it develops***.

Source : An Evolutionary Machine Learning Approach for KOA

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8000487/

Goal

1. To ***improve the accuracy and reliability of KOA grading***, thereby facilitating ***early detection and appropriate management strategies*** for knee osteoarthritis.



Objective

1. To ***extract features vector from deep learning models (CNN)*** for later fusion with handcrafted features.
2. To ***fuse handcrafted features with network features*** using features level fusion.
3. To ***implement Feed-Forward Neutral Network for classification*** of KOA using fused features.

Scope

1. Covers KOA analysis using **feature extraction, classification methods, evaluation metrics, and comparison of different approaches** for accurate diagnosis and severity classification.

Importance

1. Enhances KOA diagnosis by **combining deep learning and handrcfated features**, aiming for higher accuracy, reliability, and improved patient care.

Studies for KOA using Features

Types	Reference	Approach	Methodology	Features
Deep Learning	Chen et al.,(2020)	Fully automatic knee osteoarthritis severity grading using deep neural networks with a novel ordinal loss.	This study detect knee joints using a customized one-stage YOLOv2 network then fine-tune the most popular CNN models to classify the detected knee joint images with a novel adjustable ordinal loss	DenseNet , InceptionV 3 , Resnet , VGG
Fusion	Prashantha and Prakash (2022)	Feature level fusion framework for brain MR image classification using supervised deep learning and handcrafted features	This study propose an efficient fusion framework for brain magnetic resonance (MR) image classification using deep learning and handcrafted feature extraction method	Feature Level Fusion
Hand-crafted	Khalid et al.,(2023)	Automatic Analysis of MRI Images for Early Prediction of Alzheimer's Disease Stages Based on Hybrid Features of CNN and Handcrafted Features	This study extract MRI image that by deep learning technique and combine with DWT , GLCM and LBP.	DWT , GLCM , LBP , CNN

Table 1 Studies KOA in DL

Studies for KOA using Machine Learning

Reference	Approach	Methodology	Accuracy / F1 Score
Heisinger et al.,(2020)	Predicting total knee replacement from symptomology and radiographic structural change using artificial neural networks—data from the osteoarthritis initiative (OAI)	This study uses longitudinal assessments of radiographic changes, knee pain, function, and quality of life, then classifies by artificial neural networks **(ANN)**	81.2%
Kokkotis et al.,(2020)	Identification of risk factors and machine learning-based prediction models for knee osteoarthritis patients.	This study uses a robust feature selection approach combining filter, wrapper, and embedded techniques, followed by classification with **SVM**	74.07%
Su et al.,(2023)	Improved Prediction of Knee Osteoarthritis by the Machine Learning Model XGBoost.	This study uses a dataset combining clinical parameters, imaging data, and patient history, then classifies by Model **XGBoost**	**0.553** (F1 Score)

Table 2 Studies KOA in ML

Studies for KOA using Deep Learning

Reference	Approach	Methodology	Accuracy
Tiwari, Poduval and Bagaria (2021)	Using deep learning methods for orthopedic radiography, artificial intelligence models for osteoarthritis of the knee are evaluated.	This study uses **AI and DL** to develop a neural network-based assessment for classifying KOA severity from radiographic images annotated with KL grades	74%
Yunus et al.,(2022)	YOLOv2 is used to identify knee osteoarthritis (KOA), and convolutional neural networks are employed for classification.	This study uses 3D radiographic images with LBP features and deep features from Alex-Net and Dark-net-53, then classifies by **CNN**	90.6%
Mohammed et al.,(2023)	Knee Osteoarthritis Detection and Severity Classification Using Residual Neural Networks on Preprocessed X-ray Images	Propose the application of six pretrained **DNN** models, namely, VGG16, **VGG19**, **ResNet101**, MobileNetV2, InceptionResNetV2, and DenseNet121 for KOA diagnosis using images obtained from the Osteoarthritis Initiative (OAI) dataset.	64% (VGG19) 69% (Resnet101)

Table 3 Studies KOA in DL

Data Collection

Kellgren and Lawrence grades

Grade 0, **no reactive changes** + healthy knee.

Grade 1, indicating a **possible development of osteophytes**

Grade 2, **definite osteophytes**

Grade 3, exhibits **moderate osteophyte growth**,

Grade 4, exhibits a **large amount of osteophyte growth**.

Description of Datasets - OAI

- Osteoarthritis Initiative (University of California, San Francisco's Osteoarthritis Initiative (OAI))

Categories	Explanation of KL Grading	Ori Amount	Aft_Select Amount
Grade 0	X-ray that is Healthy	3857	1468
Grade 1	X-rays showing joint restriction with tipping osteophytes	1770	855
Grade 2	Evidence of minimal osteoarthritis with constrained joint spaces and osteophytes.	2578	1643
Grade 3	X-rays show numerous osteophytes, mild sclerosis, and moderate osteoarthritis with joint space stenosis.	1286	986
Grade 4	Extensive injuries visible on X-rays include massive osteophytes, extensive sclerosis, and obvious joint constriction.	295	295
Total		9786	5277

Table 4 Description of data OAI

Research Framework – Phase 2

Phase 2 : Data Pre-process then Generate Features and Fused Features

Activity 5 : Auto Selection X-ray images then pre-process

Activity 6 : Improve X-ray images

Activity 7 : Generate handcrafted features (GLCM , DWT , LBP) then combines 3 of them

Obj 1

Activity 8 : Generate network features (VGG-19 and ResNet-101) through convolutional layers , pooling layers and some auxiliary layers after applying data augmentation technique

Obj 2

Activity 9 : Generate 3 fusion features by fusing VGG-19 with handcrafted features , ResNet-101 with handcrafted features and VGG-19 with Resnet-101 with handcrafted features by using feature-level fusion

Architecture Diagram

Step 1: Preprocess Data

Step 2: Split Dataset

Step 3: Augment Training Set Only

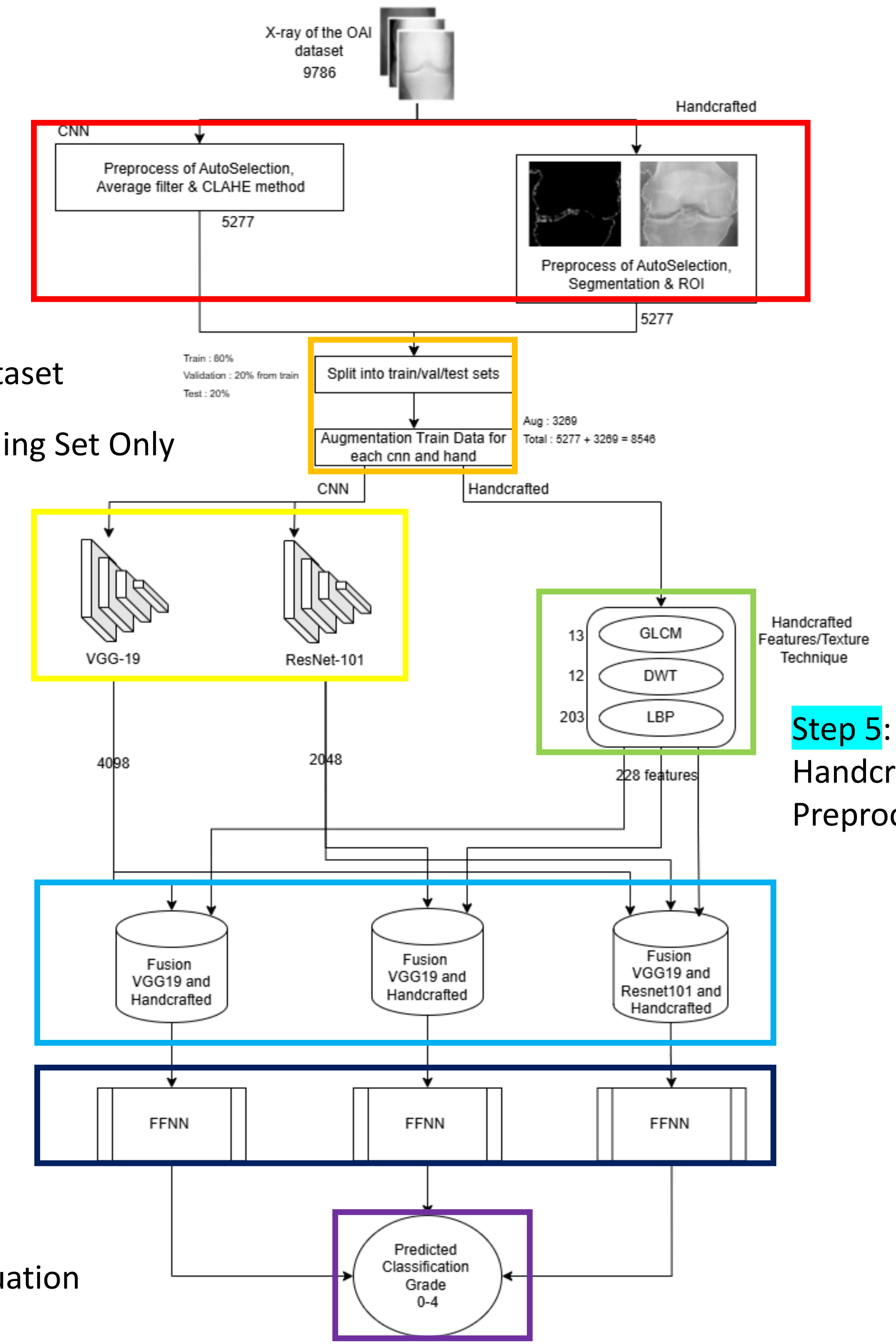
Step 4: Extract CNN Features on Preprocessed CNN Images

Step 5: Extract Handcrafted Features on Preprocessed Hand Images

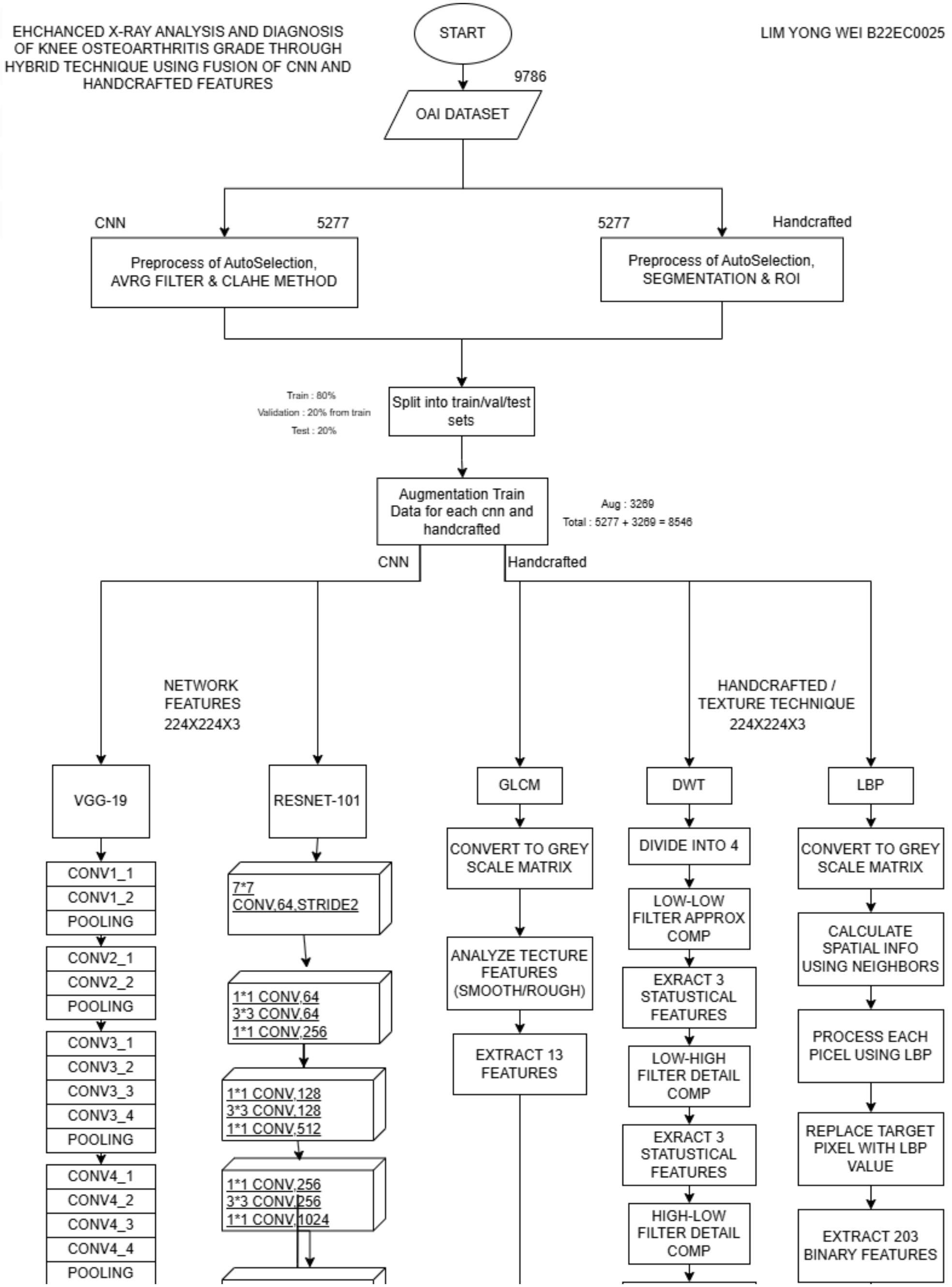
Step 6: Concatenate Features

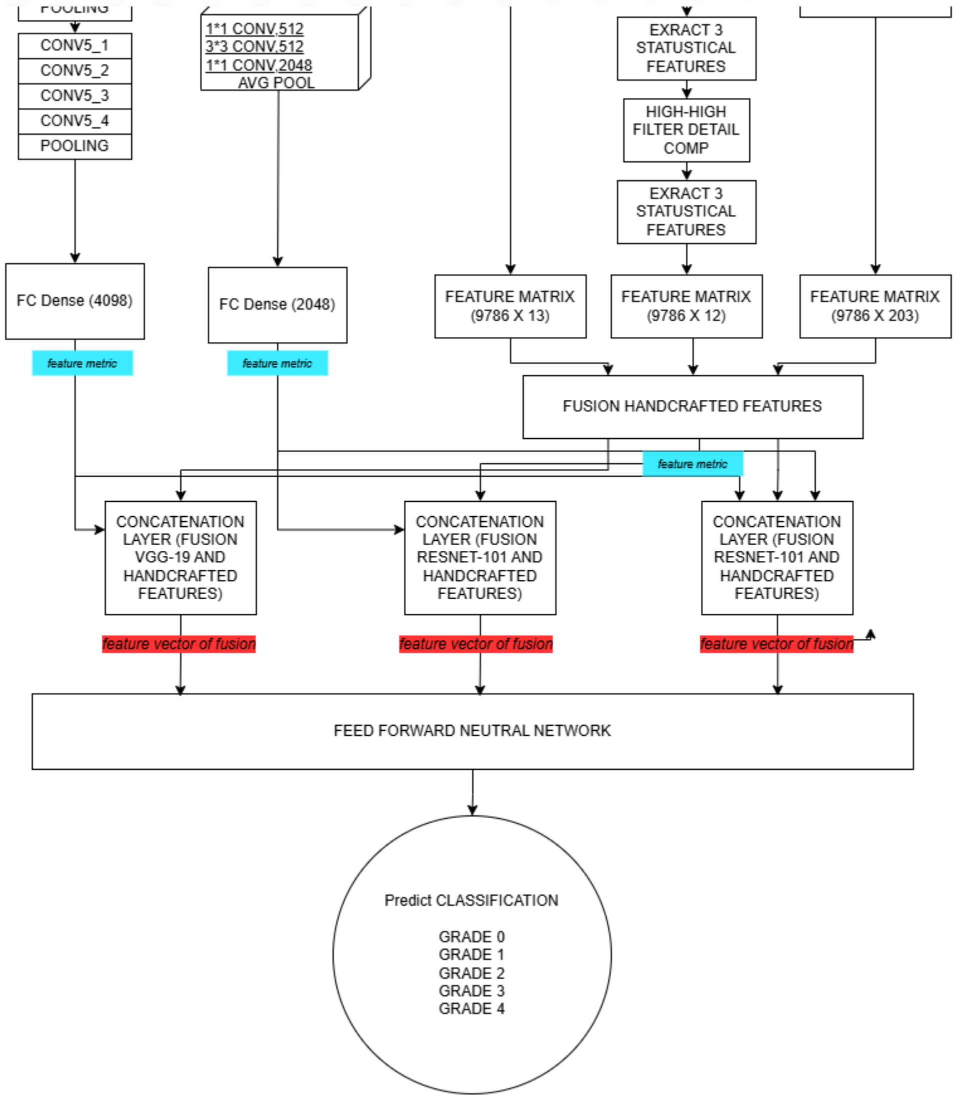
Step 7: FFNN Classification

Step 8: Final Comparative Evaluation

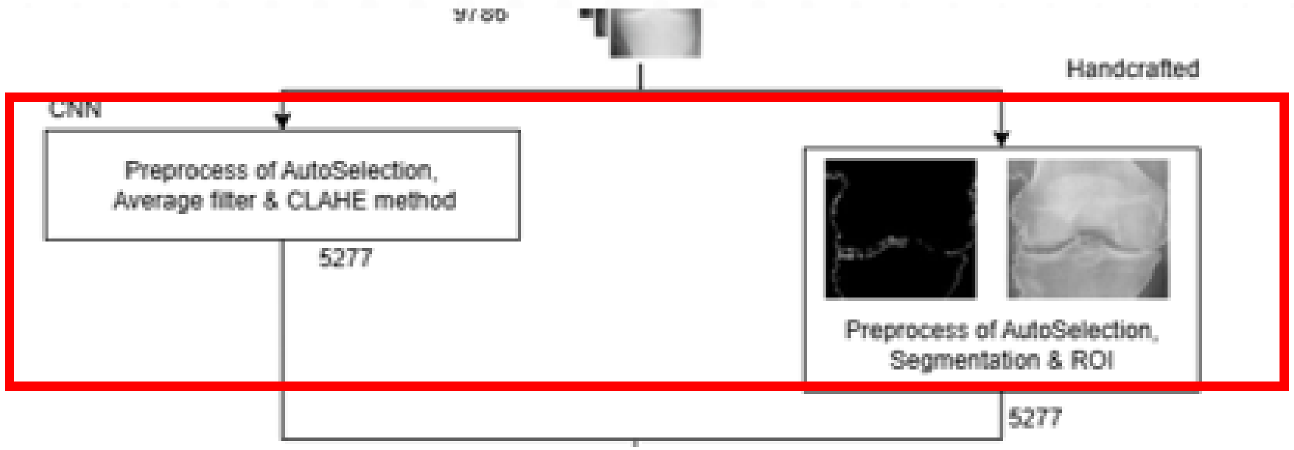


Proposed Model Diagram





Step 1: Preprocess Data



Datasets Pre-Processing – AutoSelection

•**Auto-selection Objective**:
Select **only the clearest images** to ensure high-quality input for deep learning.

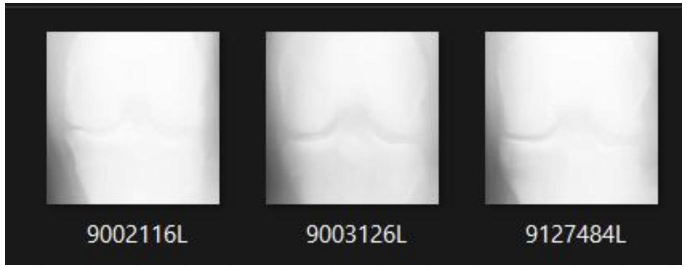
•**Sharpness Measurement**:
Used **variance of the Laplacian operator** to evaluate image clarity.
Higher variance = sharper image.

•**Selection Process**:
From 9,786 images, the **top sharpest 5,277 images** were retained.
Ensured balanced representation across KL grades.

Input and output size is 240 × 240

```
loating-point round-off errors from different computation orders. To turn them off, set the environmen
Loading and selecting data...
Total: 5277 images [0: 1468, 1: 885, 2: 1643, 3: 986, 4: 295]
After selection: 5277 images
```

Exp Non-clear ORI Images



Categories	Ori Amount	Aft_Select Amount
Grade 0	3857	1468
Grade 1	1770	855
Grade 2	2578	1643
Grade 3	1286	986
Grade 4	295	295
Total	**9786**	**5277**

Table 5 Dataset OAI bef and aft selection

Datasets Pre-Processing – Segmentation & ROI

Objective:
Enhance image clarity by isolating the knee joint to ensure accurate and **consistent handcrafted feature extraction**.

Input size is 240 × 240

Step 1 : Segmentation:
- Convert image to **grayscale**.
- Apply **binary thresholding** to isolate the knee joint.
- Detect **contours** and identify the **largest contour** as the knee region.

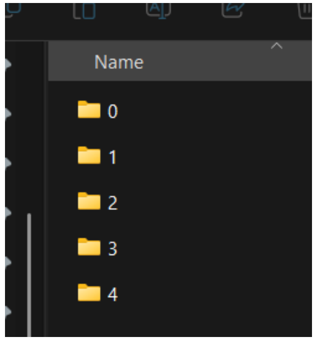
Step 2 : Region of Interest (ROI) Cropping:
- Crop a **rectangular region** around the detected knee joint.
- Add **10-pixel padding** around the bounding box to retain anatomical context.
- Resize the cropped image to a **fixed resolution of 224 × 224 pixels**.

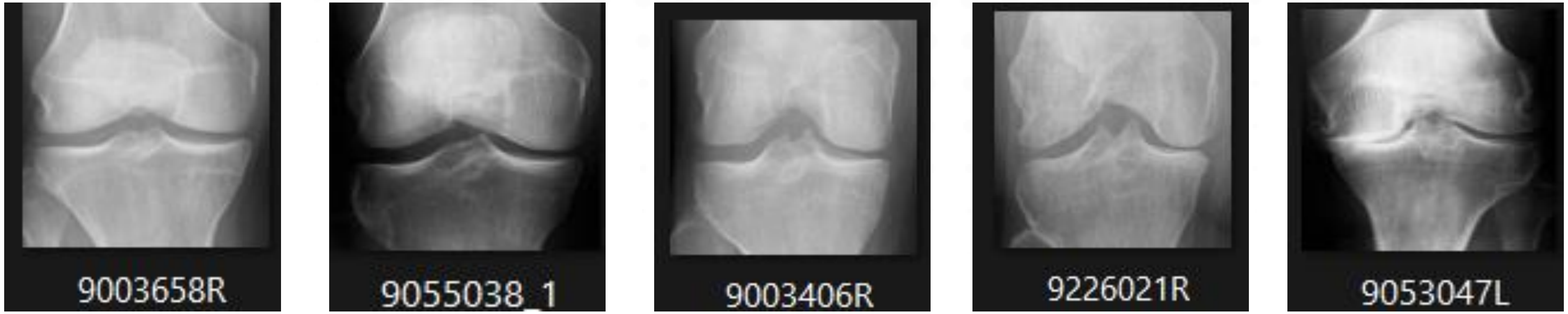
Combined Benefit:
Ensures that handcrafted features (like texture and shape) are extracted from the **most meaningful part** of the image.

Post-Processing Step for Handcrafted images:
All enhanced images are saved and organized **into separate folders based on their KOA grade** (e.g., 0, 1, 2, 3, 4).
Preprocessed images are now **ready for dataset splitting** into **training, validation, and testing sets**.

Visual Reference– Segmentation & ROI

\ori data\FYP data\hand\preproAuto





0 1 2 3 4

Datasets Pre-Processing – Average Filter & CLAHE

🔍 **Objective**:
Enhance visual quality of the grayscale image for better feature extraction.

Input size is 240 × 240

Step 1: CLAHE (Contrast Limited Adaptive Histogram Equalization)

- **Improves contrast** to make bones and joints clearer, especially helpful for medical images with uneven lighting.

- **Step 2: Average Filtering**
- Smooths the image by **reducing noise** while keeping important details.

🖼 **Final Processing**:

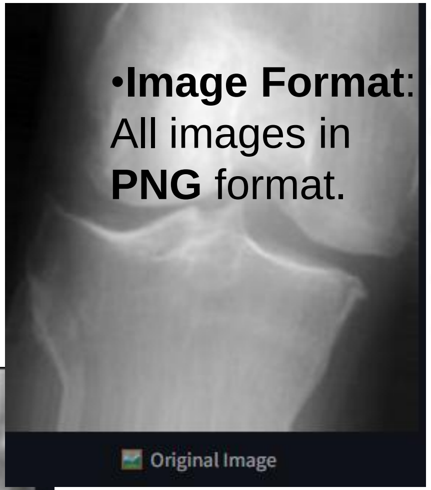
- Resize to **224 × 224 × 3**.
- Convert to **three-channel image** to match CNN input format.

☑ **Combined Benefit**:

- Balances **contrast enhancement** and **noise reduction**.
- Produces a **clearer and cleaner image** for downstream feature extraction.

🗁 **Post-Processing Step for CNN images:**

All enhanced images are saved and organized **into separate folders based on their KOA grade** (e.g., 0, 1, 2, 3, 4).
Preprocessed images are now **ready for dataset splitting** into **training, validation, and testing sets**.



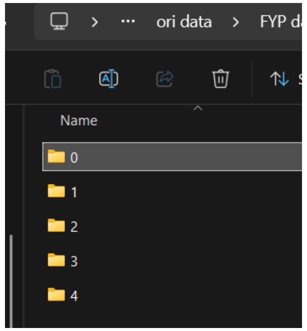
Original Image

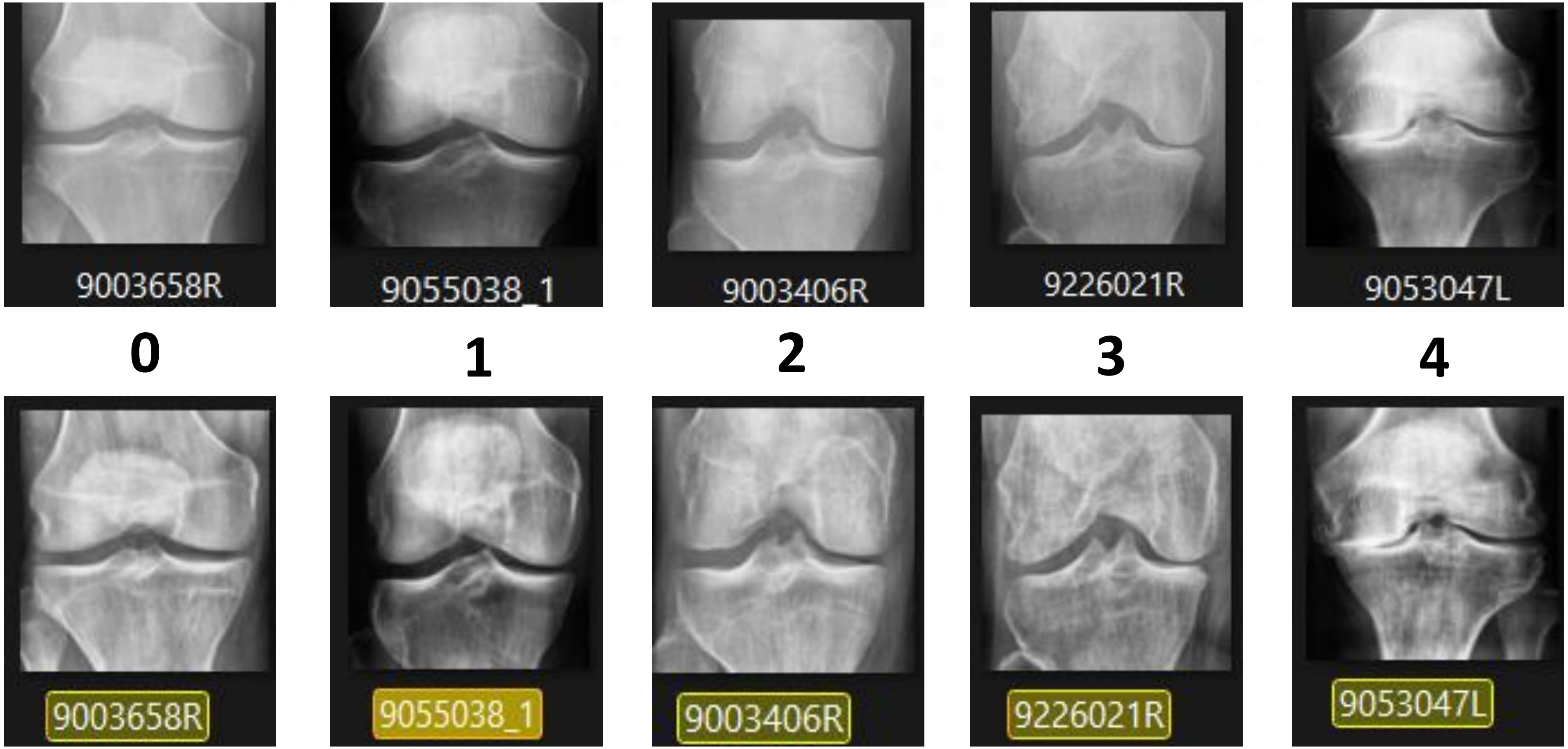

After CLAHE


After Average Filter

Visual Reference – Average Filter & CLAHE

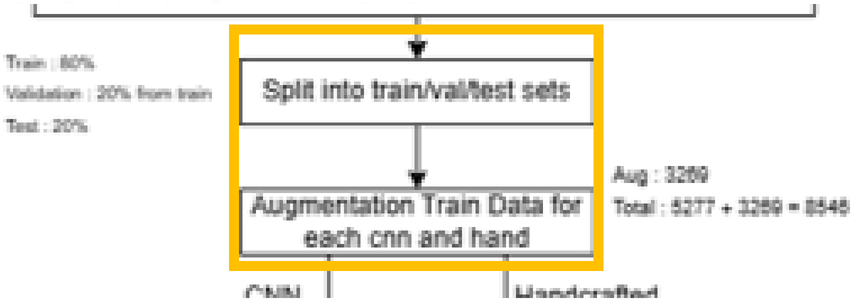
\ori data\FYP data\cnn\preproAuto





Step 2: Split Dataset

Step 3: Augment Training Set Only



Data Division and Arrangement

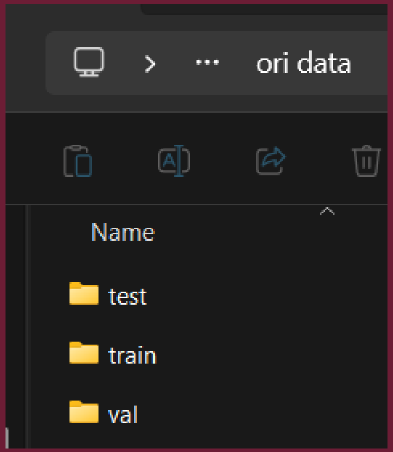
The OAI dataset will be partitioned into ***80:20 (train test) and validation sets will be obtained from 20% of training data*** segments because to its substantial size. Accordingly, training will occupy **80% of the data**, with the **remaining 20% going toward testing and validation** for each.

Classes	Grade 0	Grade 1	Grade 2	Grade 3	Grade 4	Total
	1468	885	1643	986	295	**5277**

Phase/Classes	Training 80% (Validation 20%)	Testing (20%)
Grade 0	940 (234)	294
Grade 1	567 (141)	177
Grade 2	1052 (263)	328
Grade 3	630 (159)	197
Grade 4	189 (47)	59
Total	**3378 (844)**	**1055**

Table 6 Data Division and Calculation

\cnn\augBefSplit
&
\hand\augBefSplit



Balancing with Augmentation Data on Train Split

🔍 **Purpose**:

Enhance **model robustness** and **generalizability**, Artificially expand the dataset by creating **new, diverse samples** from existing images.

☑ **Applied Only to**:

The **training set**

(✕ not on validation or test sets because to reflect **true model performance** in real-world scenarios).

•**Augmentation Techniques Used**: →

```
datagen = ImageDataGenerator
   (rotation_range=30,
    horizontal_flip=True,
    vertical_flip=True,
    width_shift_range=0.2,
    height_shift_range=0.2,
   shear_range=0.2,
    zoom_range=0.2 )
```

•**Why It's Important**:

•Helps in **medical imaging**, where labeled data is often limited.

•Prevents **overfitting** by increasing diversity in training data.

•Encourages learning of **general features** useful for unseen cases.

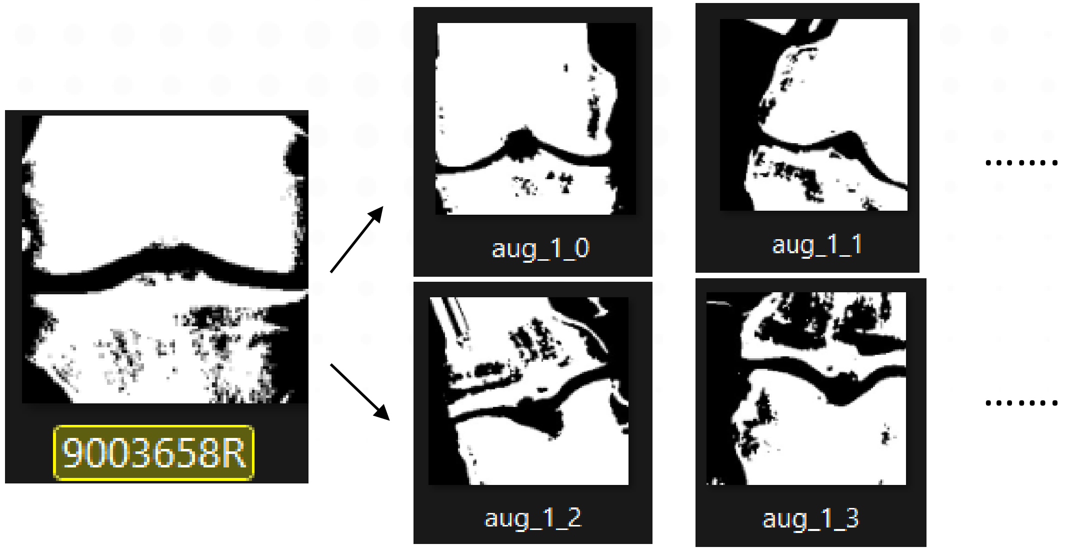
🗁 **Post-Augmentation Step for CNN images and Handcrafted images:**

All enhanced augmentation images are saved and organized **into separate folders based on their KOA grade** (e.g., 0, 1, 2, 3, 4).

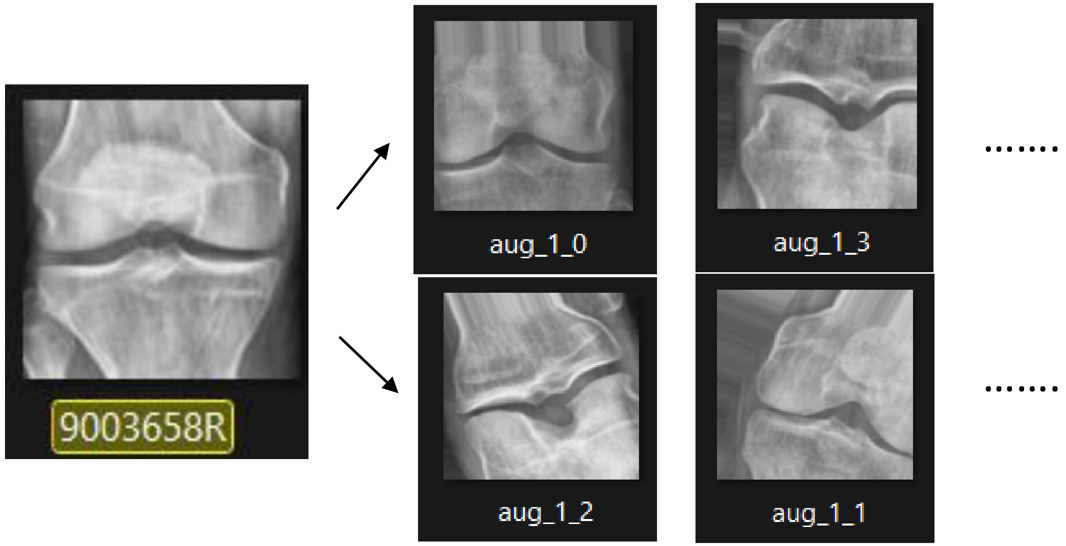
Then waiting for next step : **Overall Data Partitioning**

Visual Reference - Balancing with Augmentation Data on Train Split

Pre-Processing – Segmentation & ROI

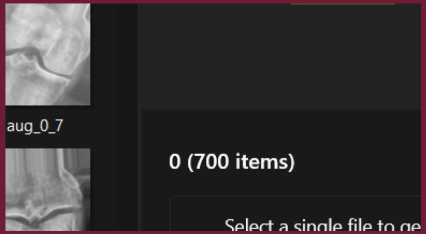


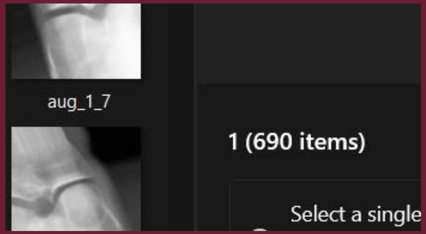
Pre-Processing – Average Filter & CLAHE

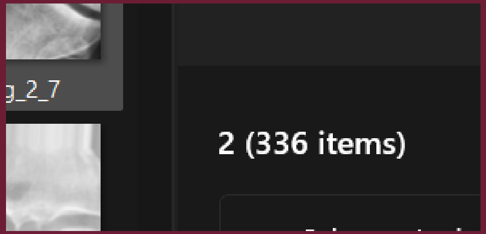


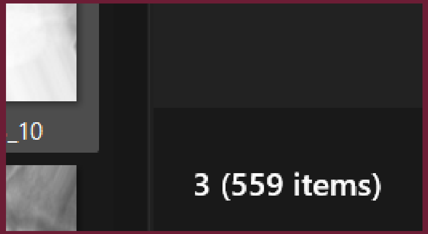
Visual Reference - Augmentation on Training Split

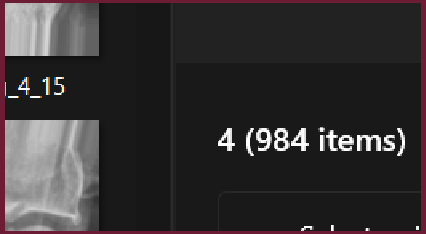
Most classes had roughly one image added per original, but augmentation was applied more selectively. For example, **Grade 4**, the least represented, had around **five** images added per original, while **Grade 2**, despite having more original images, received fewer augmentations (only +336).











\hand\Aug
&
\cnn\Aug

Training Phase						
Classes	Grade 0	Grade 1	Grade 2	Grade 3	Grade 4	**Total**
Bef-aug	940	567	1052	630	189	**3378**
aug	+700	+690	+336	+559	+984	**3269**
Aft-aug	1640	1257	1388	1189	1173	**6647**

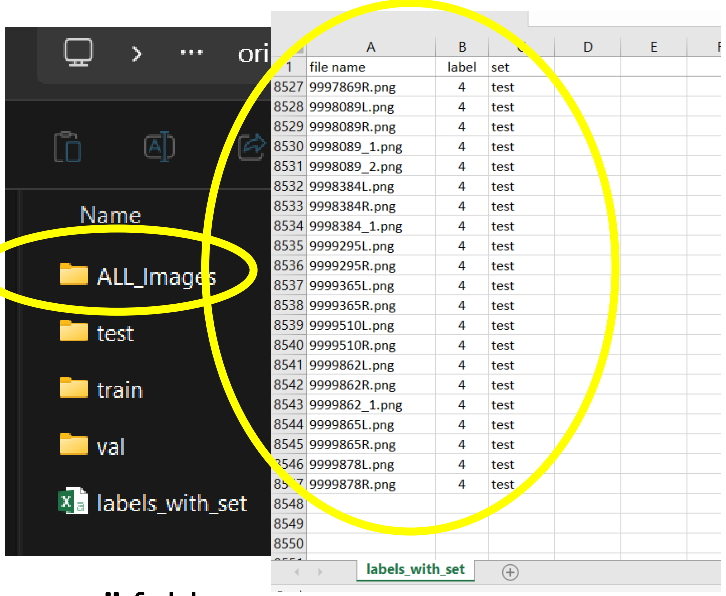
Table 7 Training Split Specification Summarization

Data Partitioning Overall After Augmentation

Phase/Classes	Training 80% + Aug	Validation (20%)	Testing 20%	Total
Grade 0	940 + 700 = 1640	234	394	**2168**
Grade 1	567 + 690 = 1257	141	177	**1575**
Grade 2	1052 + 336 = 1388	263	328	**1979**
Grade 3	630 + 559 = 1189	159	197	**1545**
Grade 4	189 + 984 = 1173	47	59	**1279**
Total	3378 + 3269 = **6647**	**844**	**1055**	**8546**

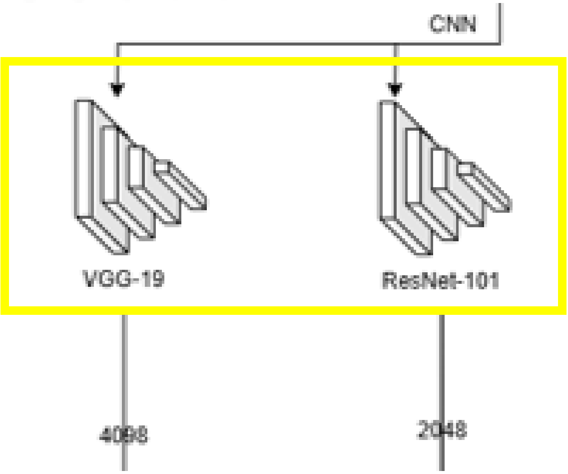
Table 8 Data Partitioning

\hand\augAftSplit
&
\cnn\augAftSplit



Save all each processed and aug images (CNN and Hand) in **"All Images"** folder.
Create **"labels_with_sets.csv"** with **filename, label, set** (train/val/test).

Step 4: Extract CNN Features on Preprocessed CNN Images



Network Features Extraction

Obj 1

Activity 8 : Generate network features (VGG-19 and ResNet-101) through convolutional layers , pooling layers and some auxiliary layers after applying data augmentation technique

VGG19 and **ResNet101** used as feature extractors.

Used : CNN-based **feature extraction pipeline** before classification
Purpose : Run inference to extract features (not training)

☞ **Input:** Use cnn preprocessed knee X-ray images.

```
from tensorflow.keras.applications import VGG19, ResNet101
# ……………………………………………………………..
# VGG19 Layer used ‘FC2’ which flattened 4096D vector
model = Model(inputs=base_model.input, outputs=base_model.get_layer('fc2').output)

# Resnet1019 Layer used ‘GAP’ which is 2048D vector
model = Model(inputs=base_model.input, outputs=base_model.get_layer(‘gap').output)
```

Model Summary

Model: "functional_1"

Layer (type)	Output Shape	Param #
image (InputLayer)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 112, 112, 64)	9,472
max_pooling2d (MaxPooling2D)	(None, 56, 56, 64)	0
batch_normalization (BatchNormalization)	(None, 56, 56, 64)	256
conv2d_1 (Conv2D)	(None, 56, 56, 64)	4,160
conv2d_2 (Conv2D)	(None, 56, 56, 64)	36,928
conv2d_3 (Conv2D)	(None, 56, 56, 256)	16,640
batch_normalization_1 (BatchNormalization)	(None, 56, 56, 256)	1,024
conv2d_4 (Conv2D)	(None, 28, 28, 128)	32,896
conv2d_5 (Conv2D)	(None, 28, 28, 128)	147,584
conv2d_6 (Conv2D)	(None, 28, 28, 512)	66,048
batch_normalization_2 (BatchNormalization)	(None, 28, 28, 512)	2,048
conv2d_7 (Conv2D)	(None, 14, 14, 256)	131,328
conv2d_8 (Conv2D)	(None, 14, 14, 256)	590,080
conv2d_9 (Conv2D)	(None, 14, 14, 1024)	263,168
batch_normalization_3 (BatchNormalization)	(None, 14, 14, 1024)	4,096
conv2d_10 (Conv2D)	(None, 7, 7, 512)	524,800
conv2d_11 (Conv2D)	(None, 7, 7, 512)	2,359,808
conv2d_12 (Conv2D)	(None, 7, 7, 2048)	1,050,624
batch_normalization_4 (BatchNormalization)	(None, 7, 7, 2048)	8,192
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0

Total params: 5,249,152 (20.02 MB)
Trainable params: 5,241,344 (19.99 MB)
Non-trainable params: 7,808 (30.50 KB)

Model Summary

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 224, 224, 64)	1,792
re_lu (ReLU)	(None, 224, 224, 64)	0
conv2d_1 (Conv2D)	(None, 224, 224, 64)	36,928
re_lu_1 (ReLU)	(None, 224, 224, 64)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_2 (Conv2D)	(None, 112, 112, 128)	73,856
re_lu_2 (ReLU)	(None, 112, 112, 128)	0
conv2d_3 (Conv2D)	(None, 112, 112, 128)	147,584
re_lu_3 (ReLU)	(None, 112, 112, 128)	0
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_4 (Conv2D)	(None, 56, 56, 256)	295,168
re_lu_4 (ReLU)	(None, 56, 56, 256)	0
conv2d_5 (Conv2D)	(None, 56, 56, 256)	590,080
re_lu_5 (ReLU)	(None, 56, 56, 256)	0
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590,080
re_lu_6 (ReLU)	(None, 56, 56, 256)	0
conv2d_7 (Conv2D)	(None, 56, 56, 256)	590,080
re_lu_7 (ReLU)	(None, 56, 56, 256)	0
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 256)	0
conv2d_8 (Conv2D)	(None, 28, 28, 512)	1,180,160
re_lu_8 (ReLU)	(None, 28, 28, 512)	0
conv2d_9 (Conv2D)	(None, 28, 28, 512)	2,359,808
re_lu_9 (ReLU)	(None, 28, 28, 512)	0
conv2d_10 (Conv2D)	(None, 28, 28, 512)	2,359,808
re_lu_10 (ReLU)	(None, 28, 28, 512)	0
conv2d_11 (Conv2D)	(None, 28, 28, 512)	2,359,808
re_lu_11 (ReLU)	(None, 28, 28, 512)	0
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 512)	0
conv2d_12 (Conv2D)	(None, 14, 14, 512)	2,359,808
re_lu_12 (ReLU)	(None, 14, 14, 512)	0
conv2d_13 (Conv2D)	(None, 14, 14, 512)	2,359,808
re_lu_13 (ReLU)	(None, 14, 14, 512)	0
conv2d_14 (Conv2D)	(None, 14, 14, 512)	2,359,808
re_lu_14 (ReLU)	(None, 14, 14, 512)	0
conv2d_15 (Conv2D)	(None, 14, 14, 512)	2,359,808
re_lu_15 (ReLU)	(None, 14, 14, 512)	0
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 4096)	102,764,544
re_lu_16 (ReLU)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16,781,312
re_lu_17 (ReLU)	(None, 4096)	0
feature_output (Dense)	(None, 2048)	8,390,656

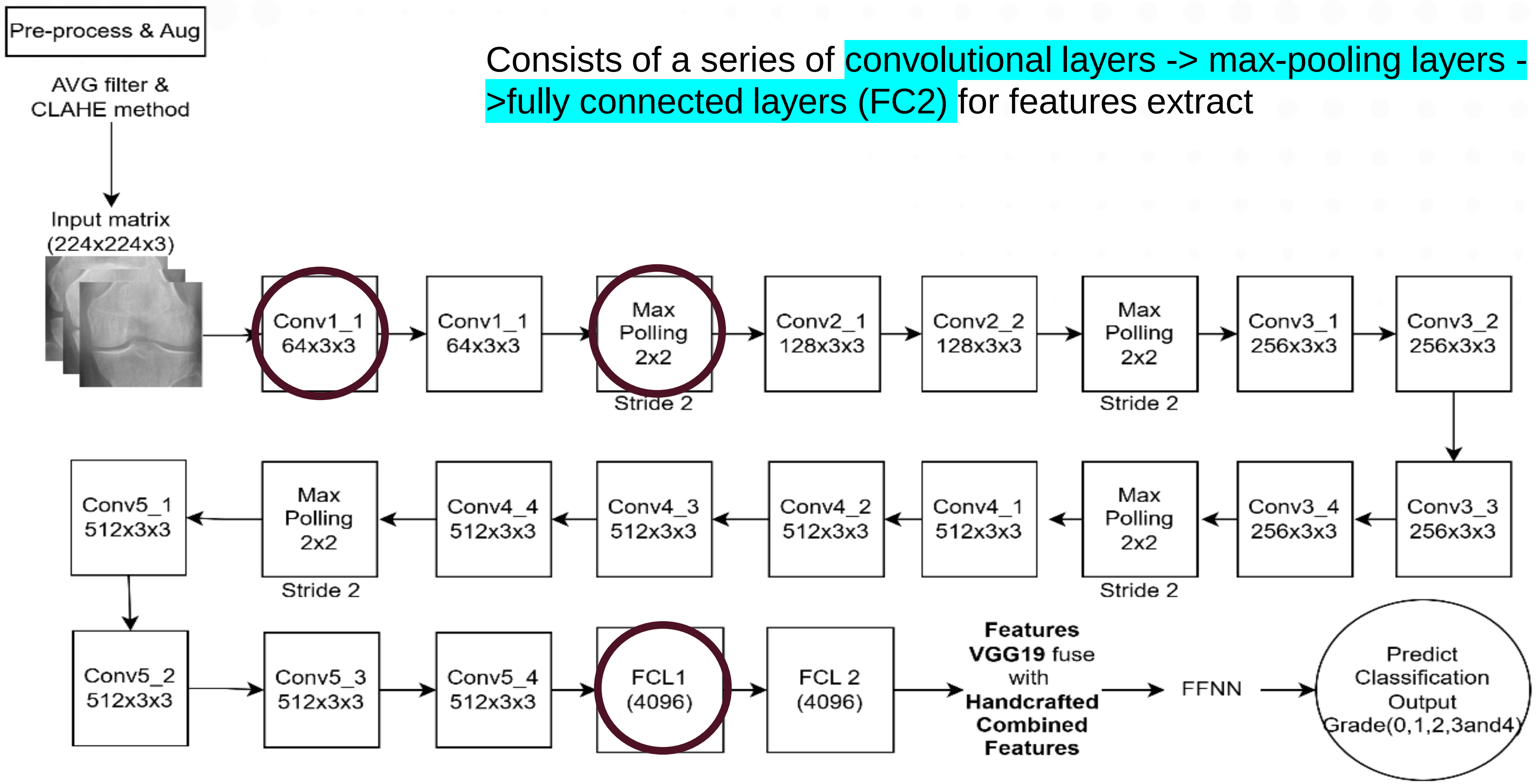
Total params: 147,960,896 (564.43 MB)
Trainable params: 147,960,896 (564.43 MB)
Non-trainable params: 0 (0.00 B)

CNN (VGG-19)

Visual Geometry Group 19-layer network

It's a deep convolutional neural network with 19 layers.

Consists of a series of convolutional layers -> max-pooling layers ->fully connected layers (FC2) for features extract



- **Input layer**
- **Conv Block 1**: 2 Conv layers (64 filters)
- **Conv Block 2**: 2 Conv layers (128 filters)
- **Conv Block 3**: 4 Conv layers (256 filters)
- **Conv Block 4**: 4 Conv layers (512 filters)
- **Conv Block 5**: 4 Conv layers (512 filters)
- **Fully Connected (FC) Layers**:
- FC1: 4096 units
- FC2: 4096 units
- FFNN
- **Total = 2 + 2 + 4 + 4 + 4 + 3 = 19 layers**

convolution filters throughout the network, which allows for deeper architectures

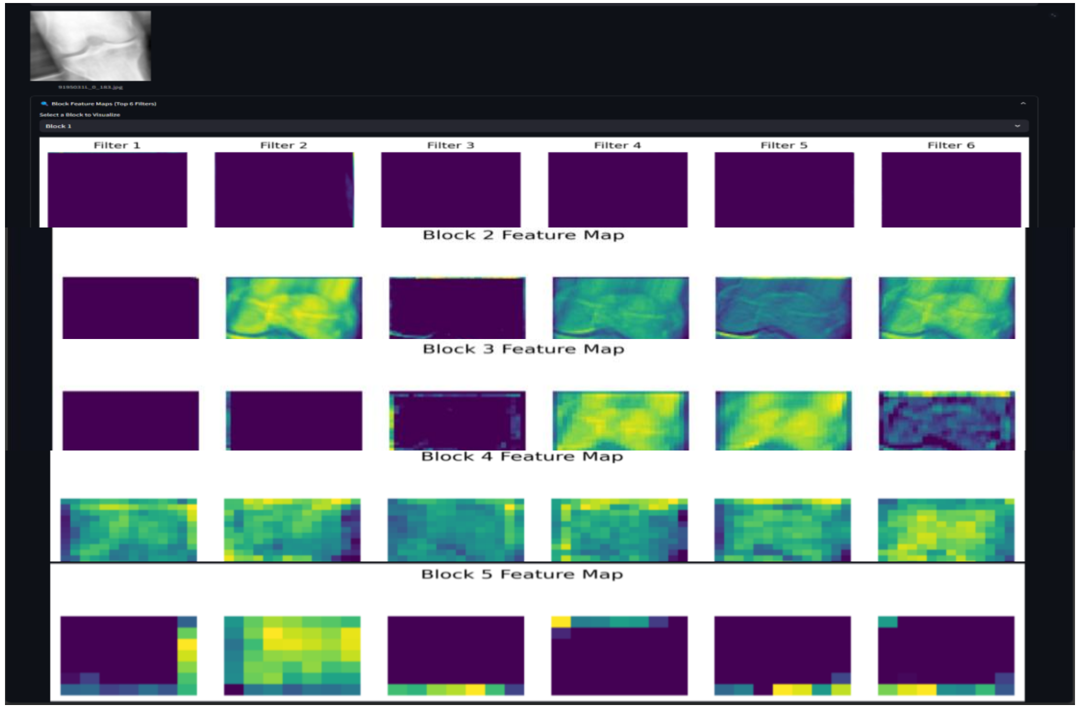
CNN (VGG-19) – Result Visualization show in Streamlit & Proposed Model

```python
st.title("Optimized KOA Feature Extraction Pipeline")
handcrafted_images_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\hand\augSplitAfter\ALL_Images'
handcrafted_labels_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\hand\augSplitAfter\labels_with_set.csv'
cnn_images_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\cnn\augSplitAfter\ALL_Images'
cnn_labels_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\cnn\augSplitAfter\labels_with_set.csv'
# Output directory
save_dir = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\AugPre\features'
os.makedirs(save_dir, exist_ok=True)
```

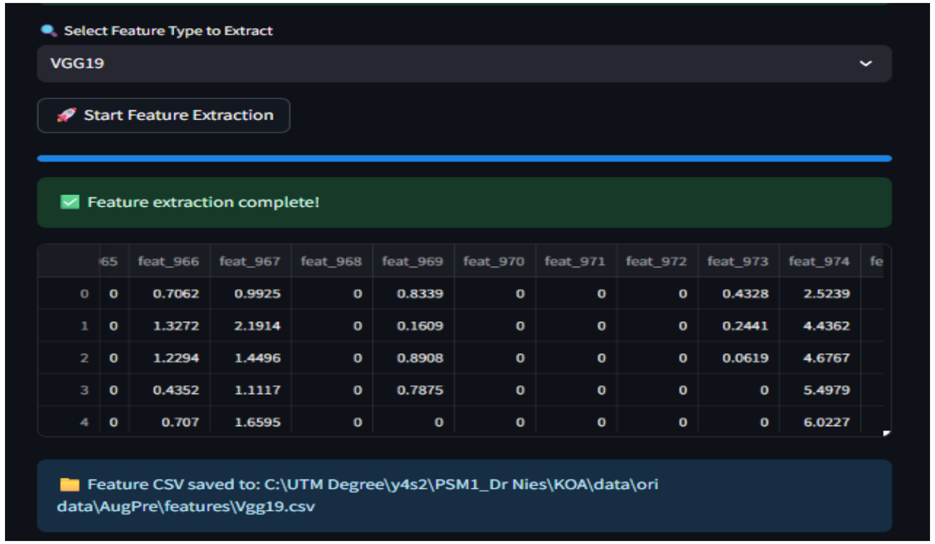
Input method to extract features

Method of a pre-trained **Keras model**

Stage	Samples Size	Batch	Batch per Epoch
Network Features Extraction	6277	4	6277/4 = **1662**




```
Extracting VGG19 features...
2025-06-23 13:58:01.029363: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available
rformance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
1662/1662 ━━━━━━━━━━━━━━━━━━━━ 2196s 1s/step
Extracting ResNet101 features...
```



	65	feat_966	feat_967	feat_968	feat_969	feat_970	feat_971	feat_972	feat_973	feat_974	fe
0	0	0.7062	0.9925	0	0.8339	0	0	0	0.4328	2.5239	
1	0	1.3272	2.1914	0	0.1609	0	0	0	0.2441	4.4362	
2	0	1.2294	1.4496	0	0.8908	0	0	0	0.0619	4.6767	
3	0	0.4352	1.1117	0	0.7875	0	0	0	0	5.4979	
4	0	0.707	1.6595	0	0	0	0	0	0	6.0227	

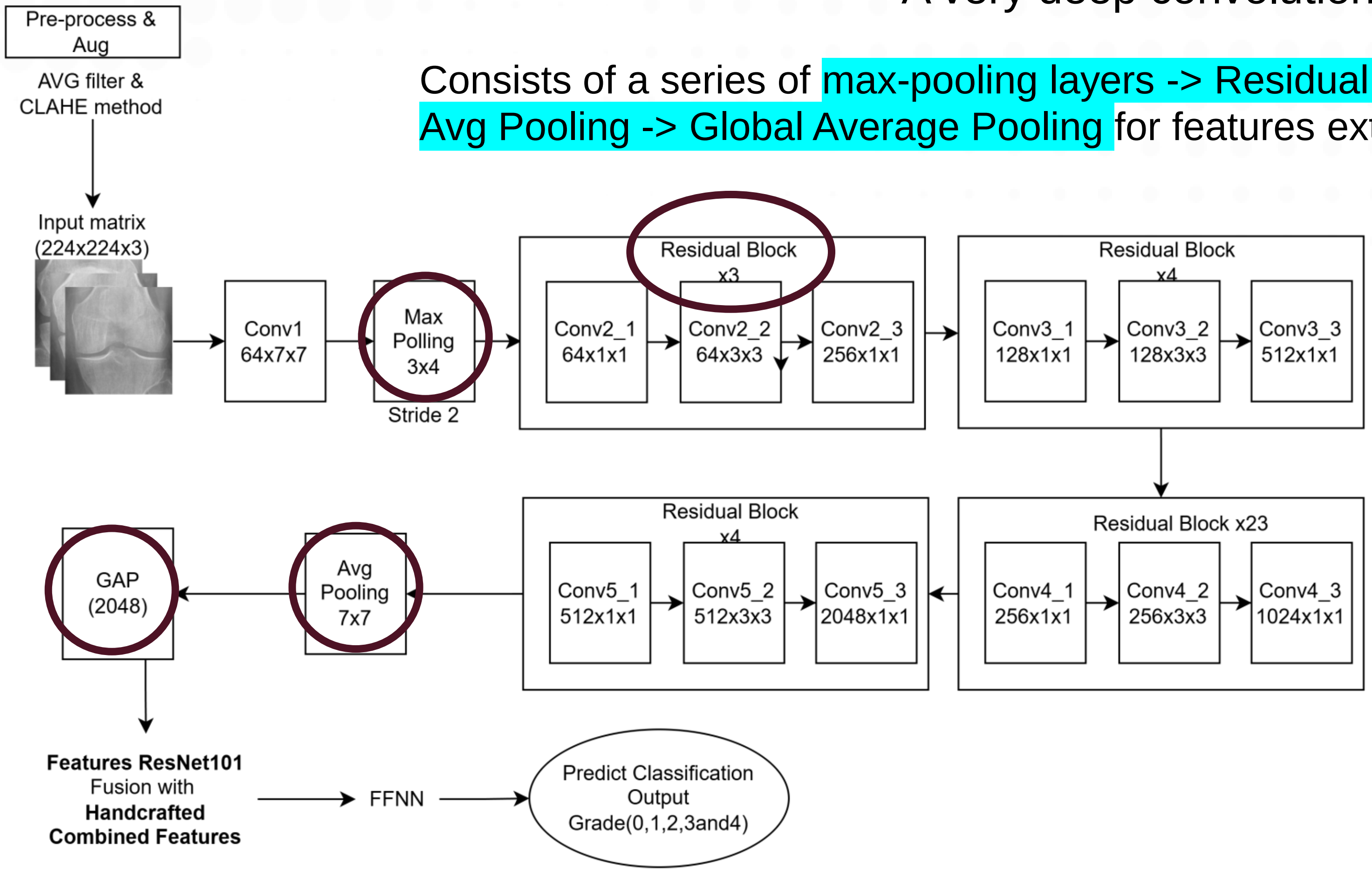
Show top 5 exp output in Streamlit

Auto save in path dir

CNN (ResNet-101) Residual Network with 101 layers.

A very deep convolutional neural network with 101 layers.

Consists of a series of max-pooling layers -> Residual Block-> Avg Pooling -> Global Average Pooling for features extract



- Input Layer
- 1 conv layer (Conv1)
- 3 blocks (conv2_x) × 3 layers = 9
- 4 blocks (conv3_x) × 3 layers = 12
- 23 blocks (conv4_x) × 3 layers = 69
- 3 blocks (conv5_x) × 3 layers = 9
- **Average Pooling Layer** (after Conv5_x)
- Global Average Pooling Layer
- fully connected layer (FFNN)

Total = 1 + 9 + 12 + 69 + 9 + 1 = 101 layers

A **residual block** contains a few convolutional layers **plus a shortcut (skip connection),** allows the network to learn **residuals** (what needs to change), rather than mapping everything from scratch to help avoid **vanishing gradients**

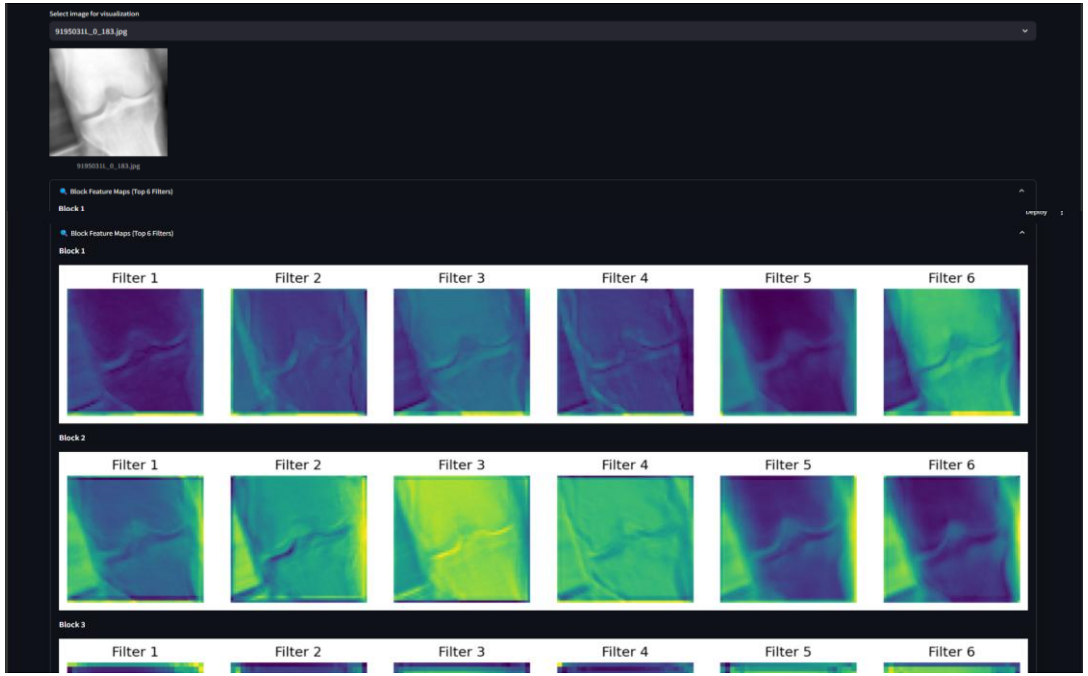
CNN (Resnet-101) – Result Visualization show in Streamlit & Proposed Model

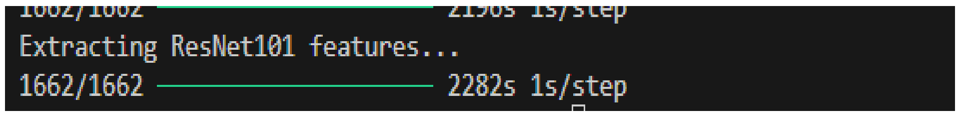
```python
st.title("Optimized KOA Feature Extraction Pipeline")
handcrafted_images_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\hand\augSplitAfter\ALL_Images'
handcrafted_labels_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\hand\augSplitAfter\labels_with_set.csv'
cnn_images_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\cnn\augSplitAfter\ALL_Images'
cnn_labels_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\cnn\augSplitAfter\labels_with_set.csv'
# Output directory
save_dir = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\AugPre\features'
os.makedirs(save_dir, exist_ok=True)
```

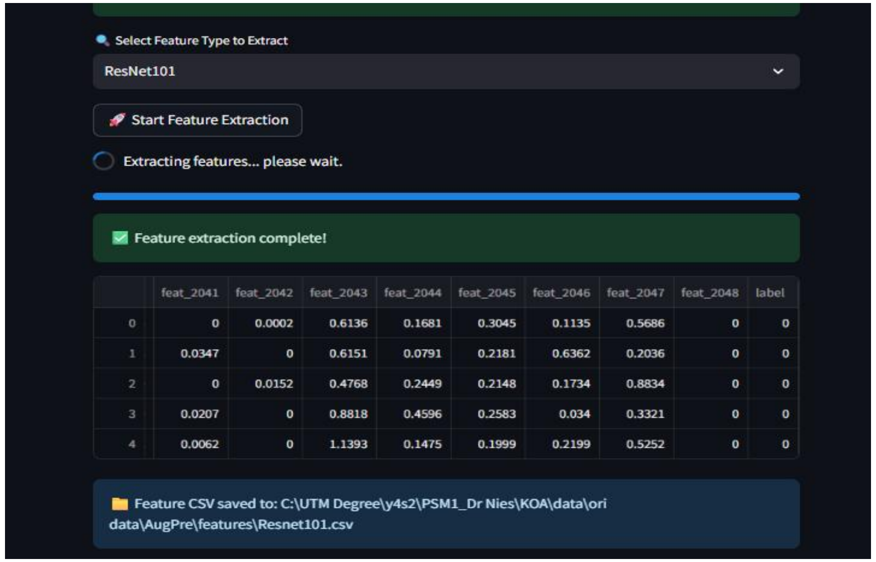
Input method to extract features

Method of a pre-trained **Keras model**

Stage	Samples Size	Batch	Batch per Epoch
Network Features Extraction	6277	4	6277/4 = **1662**



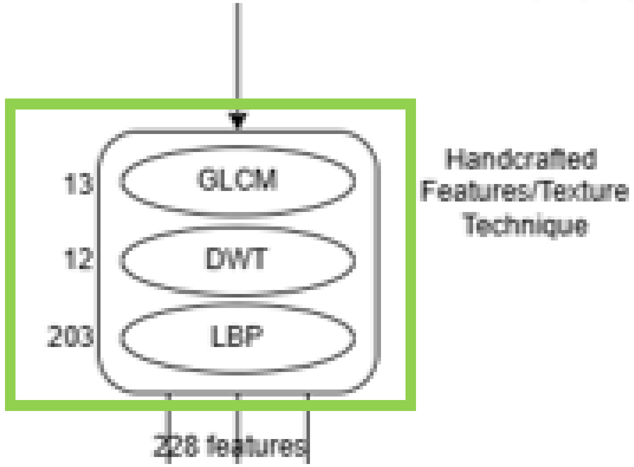




	feat_2041	feat_2042	feat_2043	feat_2044	feat_2045	feat_2046	feat_2047	feat_2048	label
0	0	0.0002	0.6136	0.1681	0.3045	0.1135	0.5686	0	0
1	0.0347	0	0.6151	0.0791	0.2181	0.6362	0.2036	0	0
2	0	0.0152	0.4768	0.2449	0.2148	0.1734	0.8834	0	0
3	0.0207	0	0.8818	0.4596	0.2583	0.034	0.3321	0	0
4	0.0062	0	1.1393	0.1475	0.1999	0.2199	0.5252	0	0

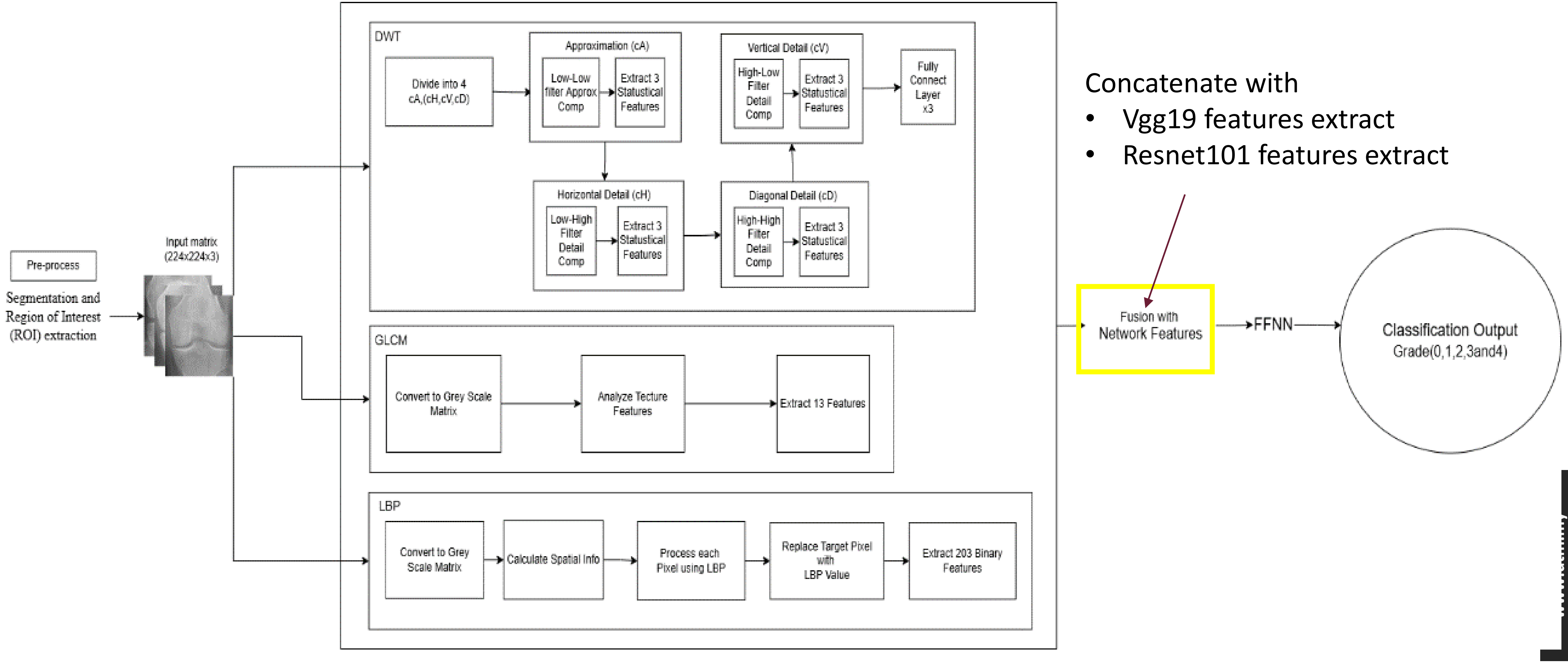
Show top 5 exp output in Streamlit

Auto save in path dir



Step 5: Extract Handcrafted Features on Preprocessed Hand Images

Proposed Model – Handcrafted (DWT , GLCM and LBP)



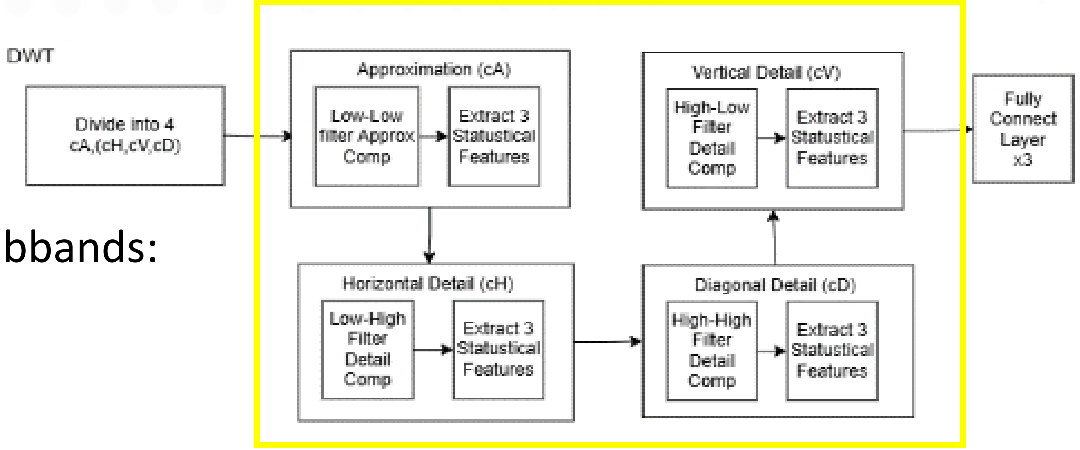
Concatenate with

- Vgg19 features extract
- Resnet101 features extract

Handcrafted Features (DWT)

🔍 **Purpose**: Captures both frequency and spatial details of the image.

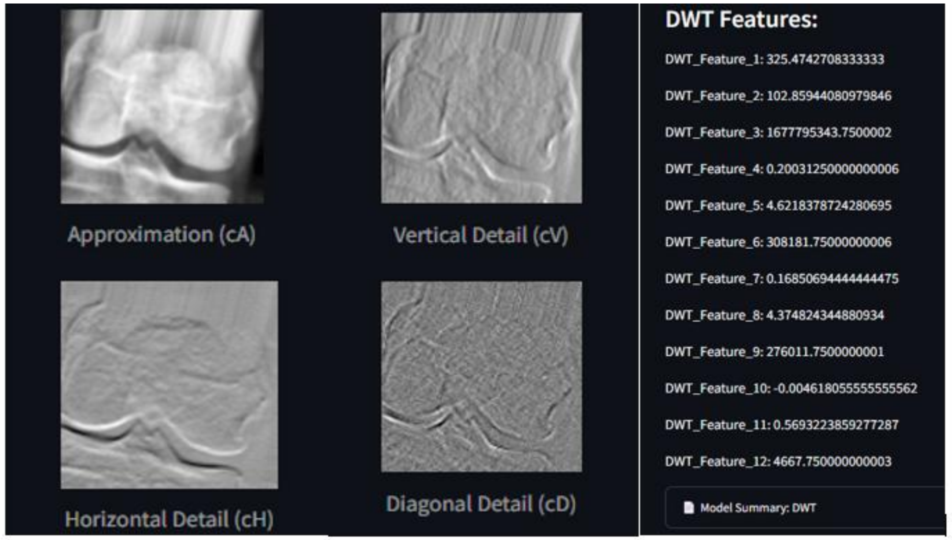
👉 **Input:** Use handcrafted preprocessed knee X-ray images.



DWT Decomposition:

Apply Discrete Wavelet Transform (**Haar**) to split the image into four subbands:

- Approximation (cA)
- Horizontal (cH)
- Vertical (cV)
- Diagonal (cD)



Feature Calculation (per subband):

Extract 3 statistical features from each subband:

- Mean
- Standard Deviation
- Energy

☑ **Output:**

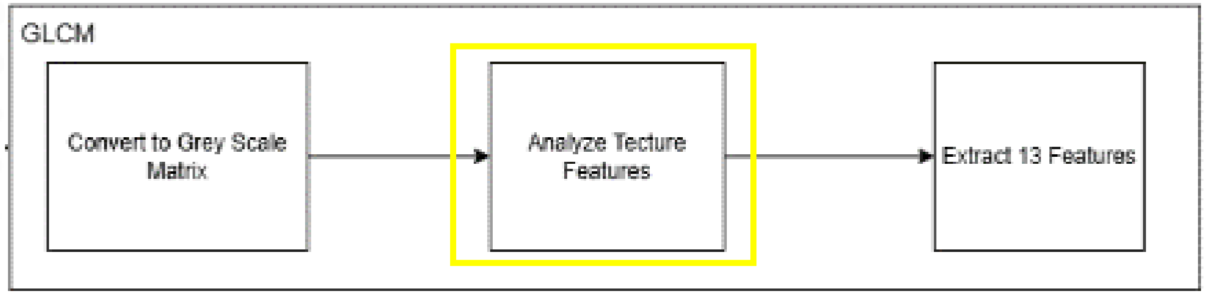
Combine all stats into a 12-dimensional feature vector (4 subbands × 3 stats).

This captures both spatial and frequency details for classification.

Handcrafted Features (GLCM)

🔍 **Purpose**: Measures texture by analyzing pixel intensity relationships.

👉 **Input:** Use handcrafted preprocessed knee X-ray images.



GLCM Generation :
Create a **Gray-Level Co-occurrence Matrix** using pixel pairs at:
- 1-pixel distance
- 4 angles: 0°, 45°, 90°, 135°

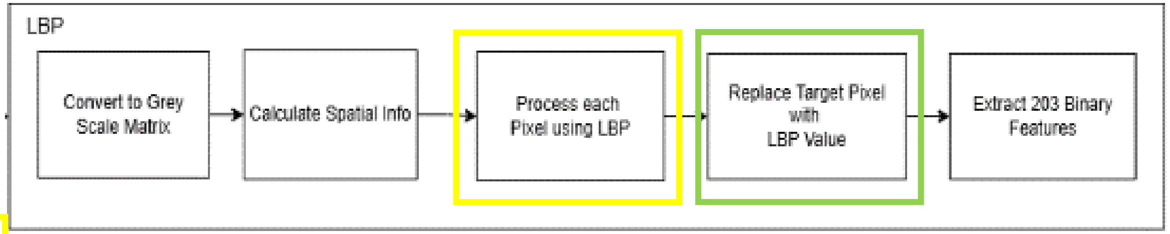
Feature Calculation :
From the GLCM, extract 5 texture features across angles:
- Contrast
- Dissimilarity
- Homogeneity
- Energy
- Correlation

☑ **Output:**
Combine into a 13-dimensional feature vector (padded if needed).
This captures spatial texture patterns important for diagnosis.

Handcrafted Features (LBP)

🔍 **Purpose**: Detects local patterns and textures around each pixel.

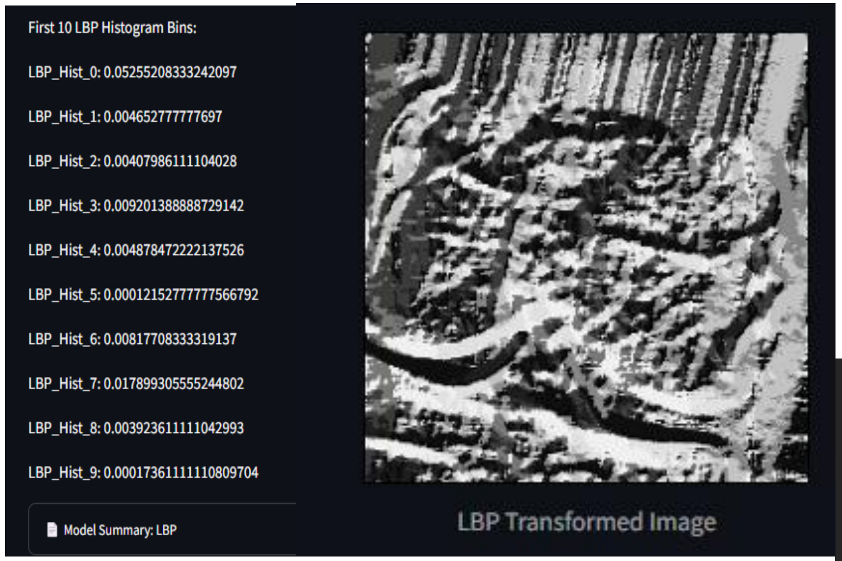
👉 **Input:** Use handcrafted preprocessed knee X-ray images.



LBP Calculation:

For each pixel, compare it with its 8 surrounding neighbors.

- If **neighbor > center pixel → assign 1**, else 0.
- Convert the 8-bit binary pattern into a decimal value (0–255).



Histogram Generation:

- Build a histogram of all LBP values across the image.
- Normalize the histogram to ensure uniformity.

☑ **Output:**

- Extract the **first 203 values** from the histogram as the LBP feature vector.
- Captures local texture patterns and edge structures important in medical imaging.

UNIVERSITI TEKNOLOGI MALAYSIA

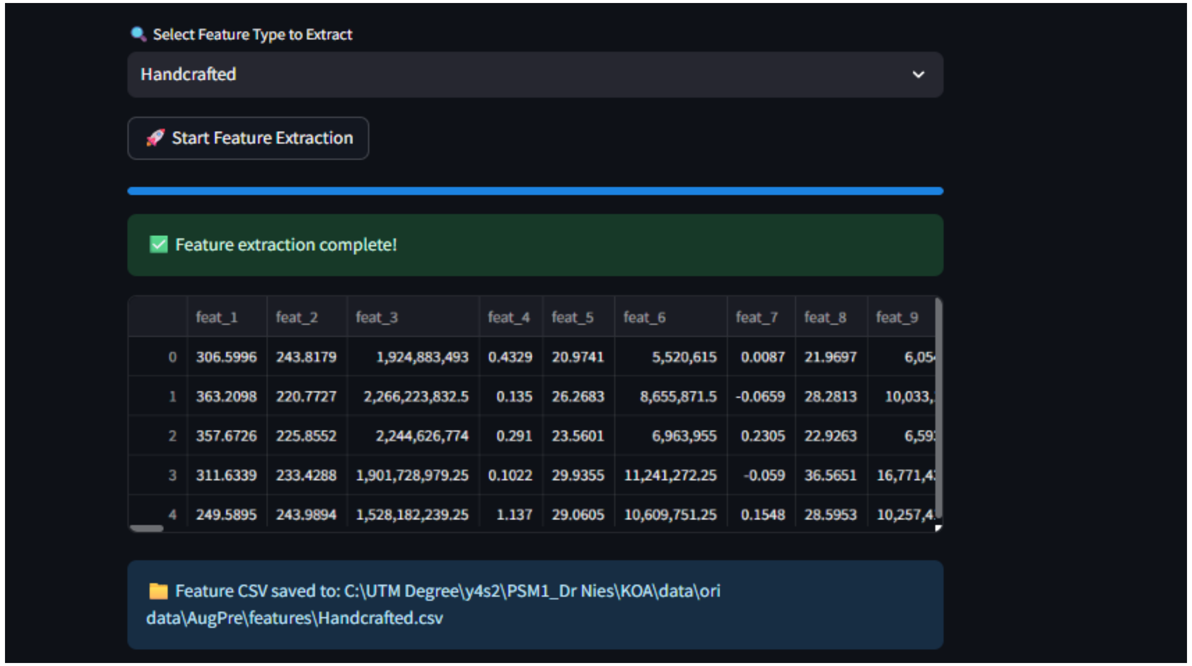
Handcrafted Features Extraction in Streamlit (DWT, GLCM and LBP)

```
st.title("Optimized KOA Feature Extraction Pipeline")
handcrafted_images_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\hand\augSplitAfter\ALL_Images'
handcrafted_labels_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\hand\augSplitAfter\labels_with_set.csv'
cnn_images_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\cnn\augSplitAfter\ALL_Images'
cnn_labels_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\cnn\augSplitAfter\labels_with_set.csv'
# Output directory
save_dir = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\AugPre\features'
os.makedirs(save_dir, exist_ok=True)
```

Input method to extract features

Show top 5 exp output in Streamlit

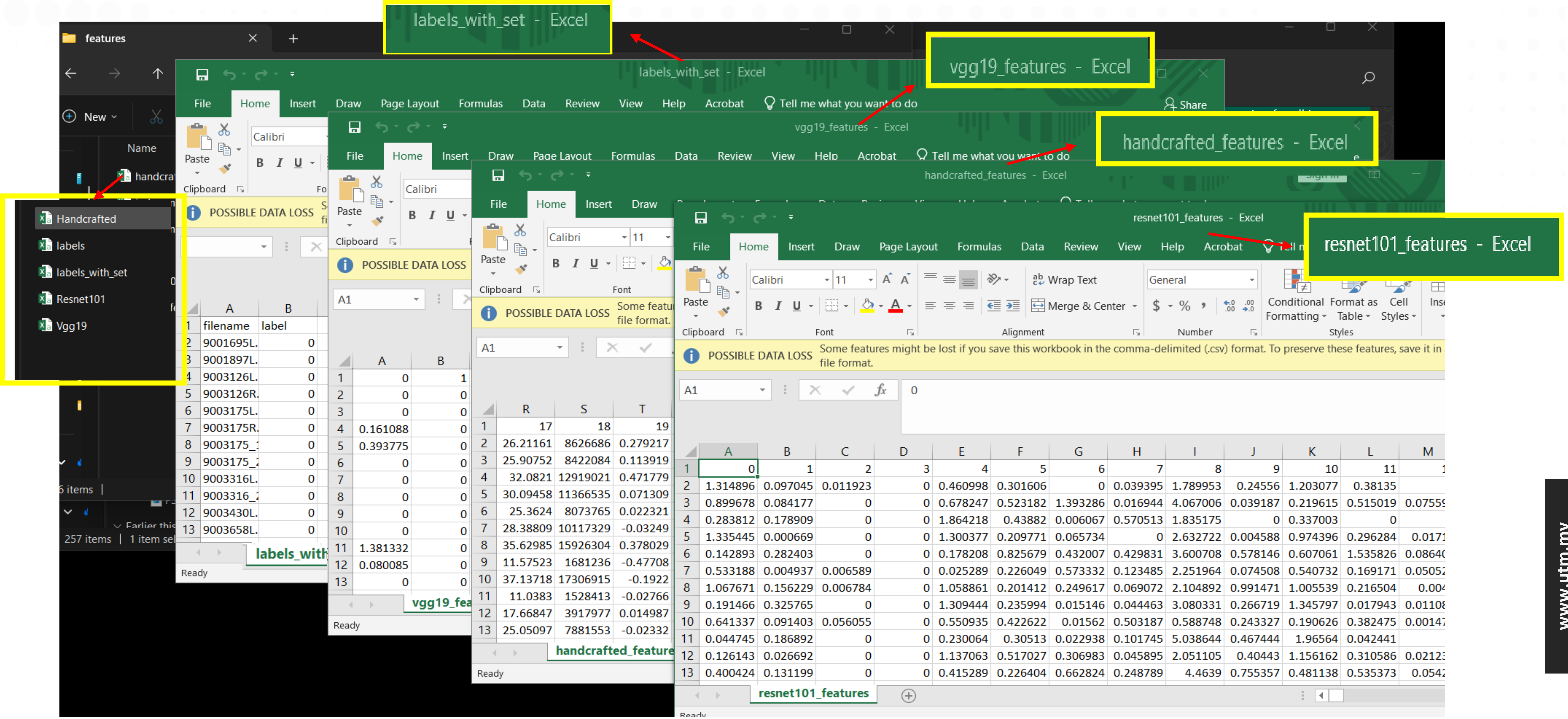
Auto save in path dir



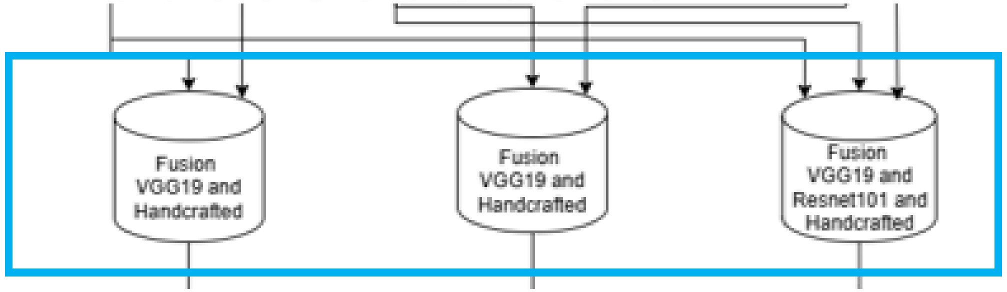
Exp :

feat_0	feat_1	feat_2	.....	feat_0	feat_1	feat_2	....	feat_0	feat_1	....	Label
...	...	....	....	....	....	....	....	....	....	....	0

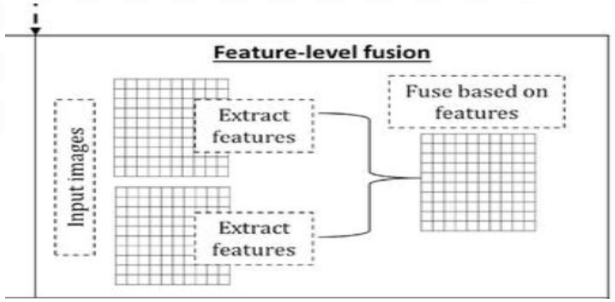
- All features extracted will store in csv file (Vgg19 , Resnet101 and Handcrafteds)



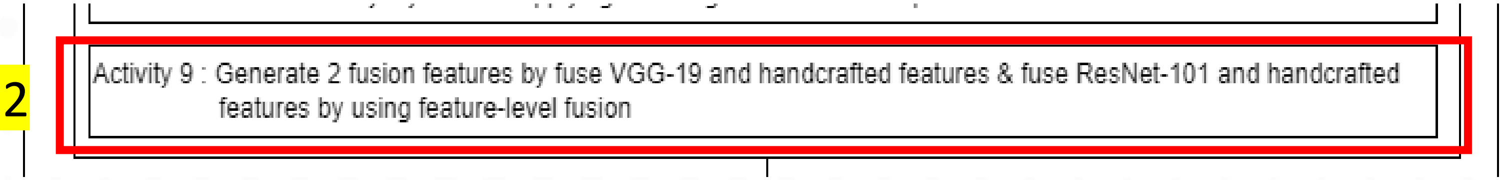
Step 6: Concatenate Features



Feature Level Fusion

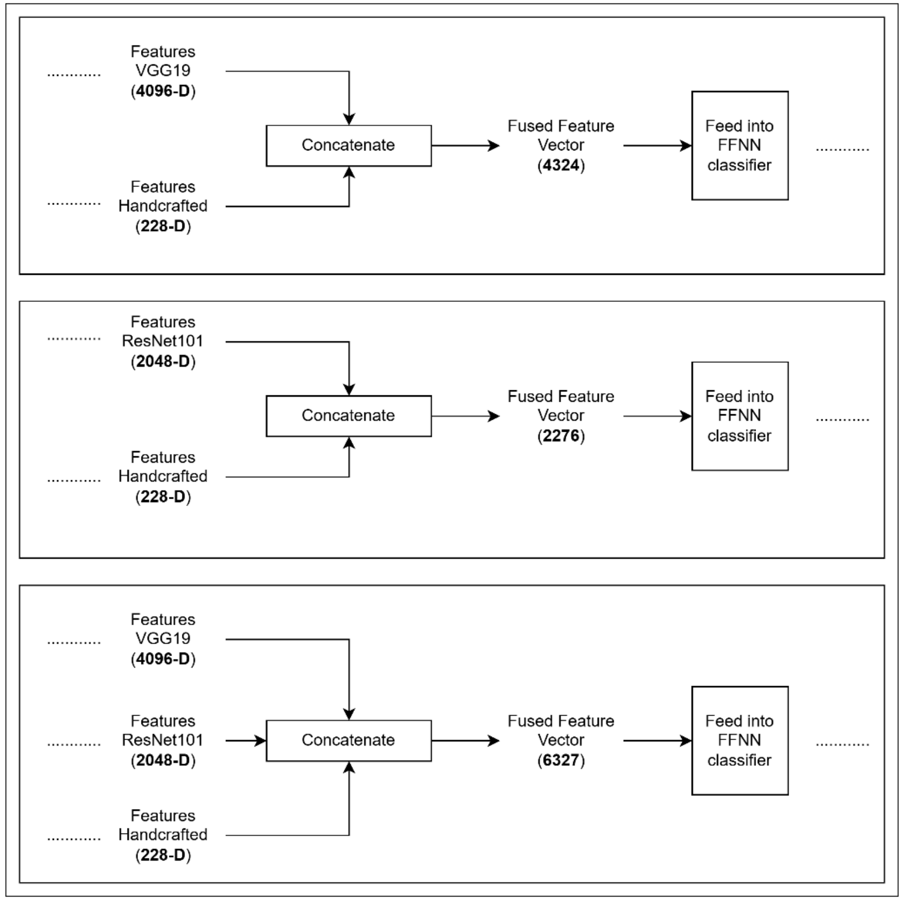


Obj 2


Activity 9 : Generate 2 fusion features by fuse VGG-19 and handcrafted features & fuse ResNet-101 and handcrafted features by using feature-level fusion

Fusion Features = Feature Level Fusion

is a technique where features extracted from multiple sources or models are combined into a single, extracted by from different models to create richer, more informative representations for the final classification task.



Input method to fusion features

```
st.header("Upload Feature CSVs and Labels CSV")
vgg_csv = st.file_uploader("Upload VGG19 features CSV", type=['csv'])
resnet_csv = st.file_uploader("Upload ResNet101 features CSV", type=['csv'])
handcrafted_csv = st.file_uploader("Upload Handcrafted features CSV", type=['csv'])
labels_csv = st.file_uploader("Upload Labels CSV", type=['csv'])
```

Feature Level Fusion – After ‘receive’ the features vector

Step 1: Load Features & Labels

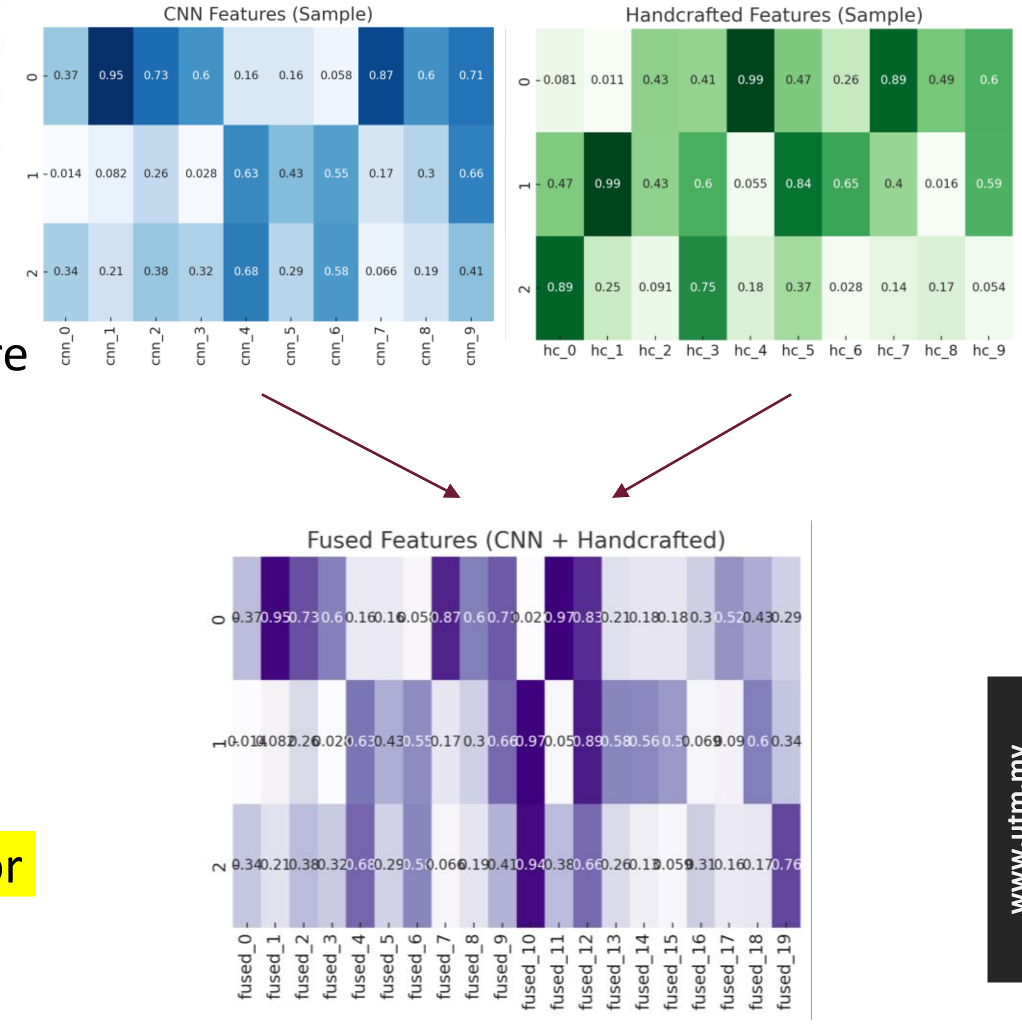
- Load VGG19, ResNet101, and handcrafted feature sets.
- Load the corresponding KOA grade labels.
- Convert the labels into a format suitable for classification.

Step 2: Combine Features

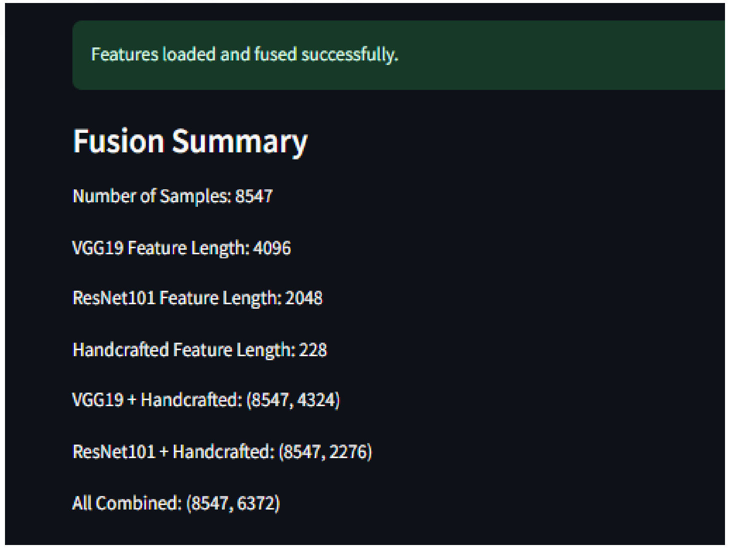
- **VGG19 + Handcrafted**: Merge both into one combined feature set.
- **ResNet101 + Handcrafted**: Merge these into another combined set.
- **All Features Together**: Combine VGG19, ResNet101, and handcrafted into a single rich feature set.

Step 3: Normalize Features

- Standardize each feature set so they are on the same scale for better learning performance.



Feature Level Fusion Visualization in Streamlit



Features loaded and fused successfully.

Fusion Summary

Number of Samples: 8547

VGG19 Feature Length: 4096

ResNet101 Feature Length: 2048

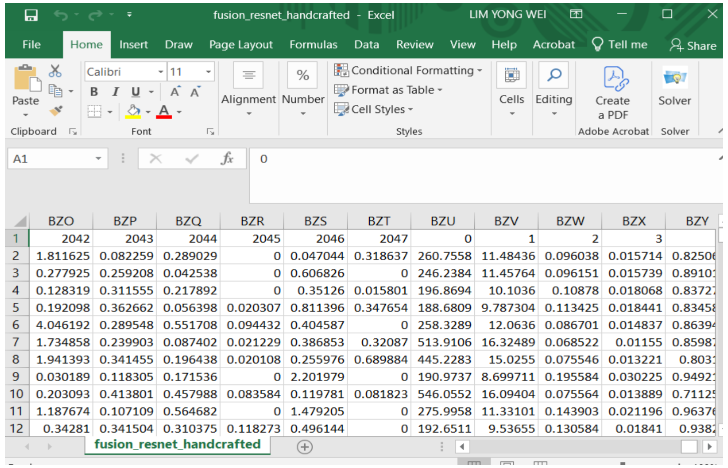
Handcrafted Feature Length: 228

VGG19 + Handcrafted: (8547, 4324)

ResNet101 + Handcrafted: (8547, 2276)

All Combined: (8547, 6372)

Exp Features Resnet101 + Handcrafted

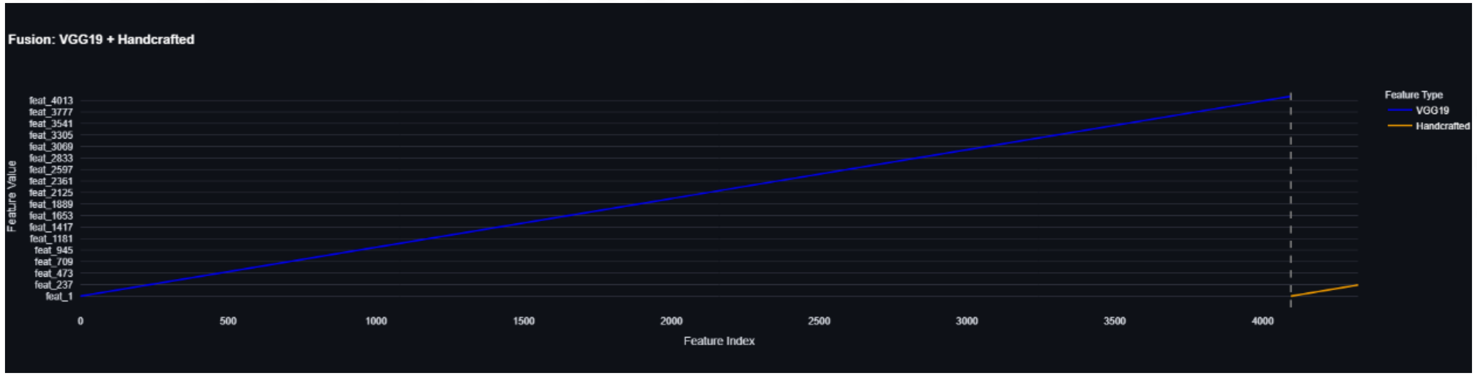


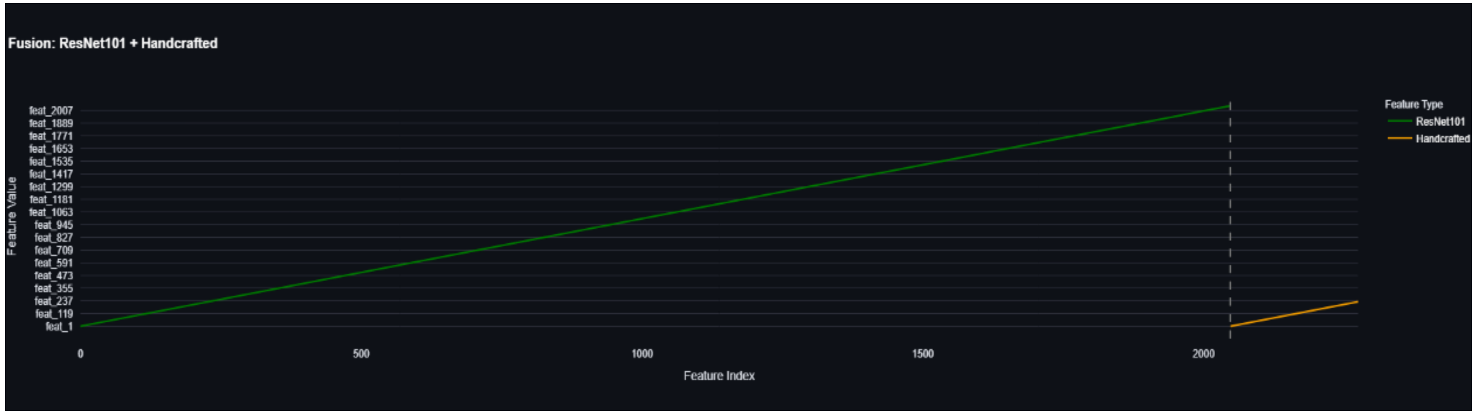
fusion_resnet_handcrafted - Excel

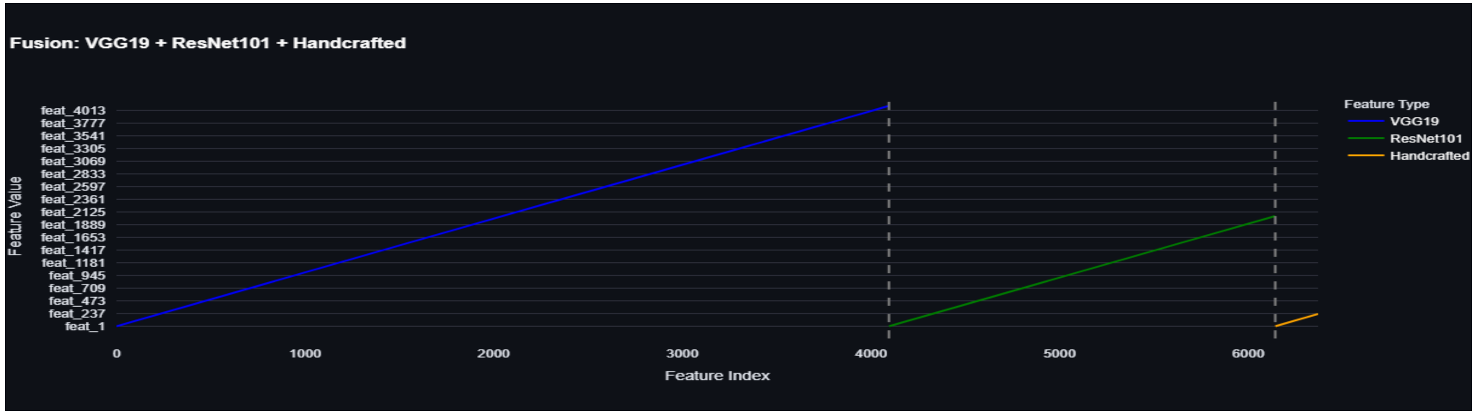
	BZO	BZP	BZQ	BZR	BZS	BZT	BZU	BZV	BZW	BZX	BZY
1	2042	2043	2044	2045	2046	2047	0	1	2	3	
2	1.811625	0.082259	0.289029	0	0.047044	0.318637	260.7558	11.48436	0.096038	0.015714	0.82506
3	0.277925	0.259208	0.042538	0	0.606826	0	246.2384	11.45764	0.096151	0.015739	0.8910
4	0.128319	0.311555	0.217892	0	0.35126	0.015801	196.8694	10.1036	0.10878	0.018068	0.8372
5	0.192098	0.362662	0.056398	0.020307	0.811396	0.347654	188.6809	9.787304	0.113425	0.018441	0.8345
6	4.046192	0.289548	0.551708	0.094432	0.404587	0	258.3289	12.0636	0.086701	0.014837	0.8639
7	1.734858	0.239903	0.087402	0.021229	0.386853	0.32087	513.9106	16.32489	0.068522	0.01155	0.8598
8	1.941393	0.341455	0.196438	0.020108	0.255976	0.689884	445.2283	15.0255	0.075546	0.013221	0.803
9	0.030189	0.118305	0.171536	0	2.201979	0	190.9737	8.699711	0.195584	0.030225	0.9492
10	0.203093	0.413801	0.457988	0.083584	0.119781	0.081823	546.0552	16.09404	0.075564	0.013889	0.7112
11	1.187674	0.107109	0.564682	0	1.479205	0	275.9958	11.33101	0.143903	0.021196	0.9637
12	0.34281	0.341504	0.310375	0.118273	0.496144	0	192.6511	9.53655	0.130584	0.01841	0.938

fusion_resnet_handcrafted

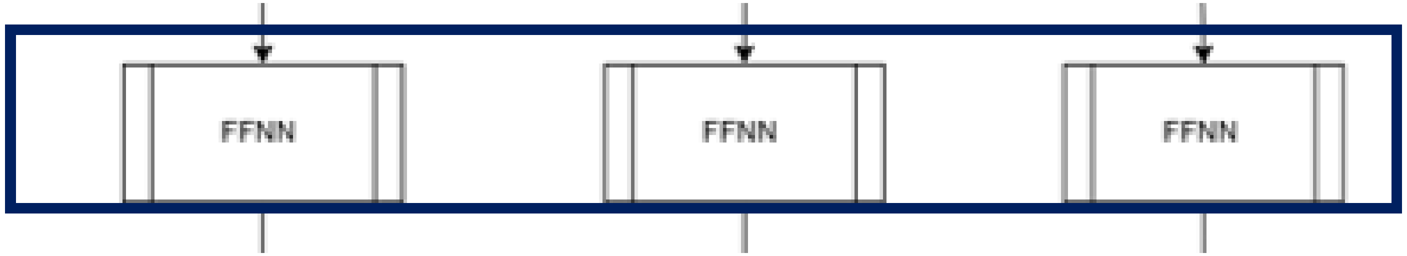
Feature Index Map (Single Sample)







Step 7: FFNN Classification



Feed-Forward Neutral Network

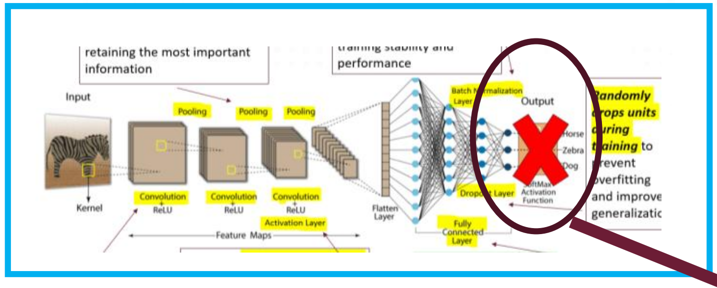
Phase 3 : Develop FFNN model

Obj 3

Activity 10 : Feed Forward Neutral Network (FFNN) with Model 1, VGG-19 with handcrafted features

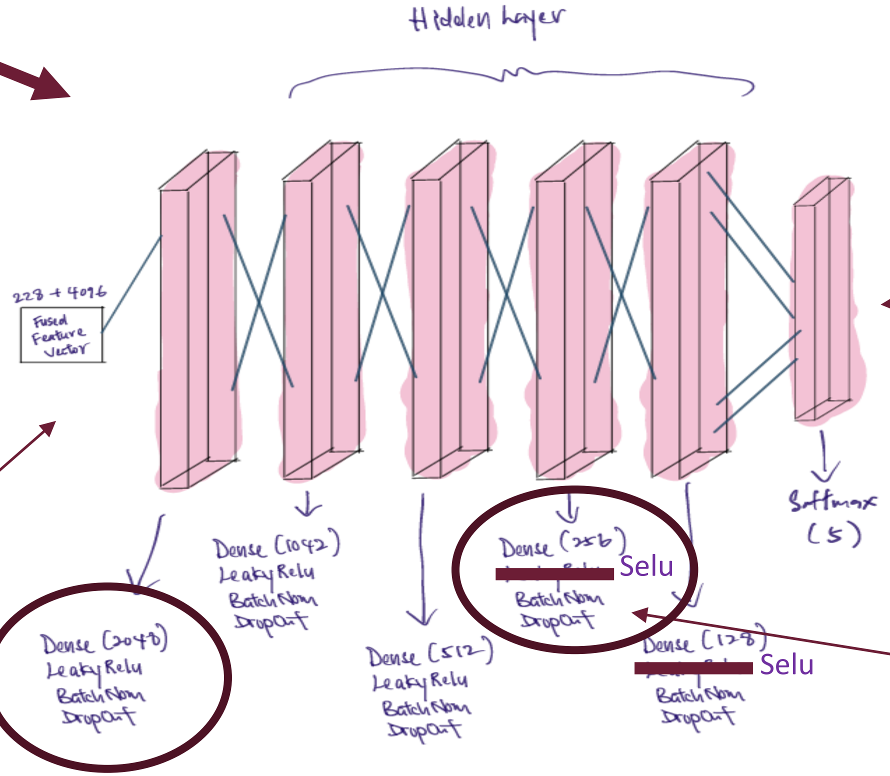
Activity 11 : Feed Forward Neutral Network (FFNN) with Model 2, ResNet-101 with handcrafted features

Activity 12 : Feed Forward Neutral Network (FFNN) with Model 3, VGG-19 with Resnet-101 with handcrafted features



FFNN
(Feed- Forward Neutral Network)

type of ***artificial neural network*** where each neuron in one layer is connected to every neuron in the next layer.



set of features extracted from an image

- Produces the final predictions
- For classification tasks, it might use a softmax activation function to output probabilities

- Consist of 5 hidden layers of neurons
- Learn to capture complex patterns and relationships in the data

After fusing the features, the dataset is still divided into:
- **Training set:** Used to train the model.
- **Evaluation set (Validation + Test):** Used to measure model performance.

ensure you get an honest estimate of model performance.

- The training features and labels are separated from the evaluation ones based on the "set" column (train, val, test).

FFNN Train

recognize patterns in your training data by adjusting its internal weights.

👉 FFNN Model Architecture

→ Define the network structure: dense layers(**2048** → **1024** → **512** → **256** → **128**), activation functions(**Leaky ReLU** and **SELU**), batch normalization, dropout and output layer (**Softmax**)

👉 Optimizer, Early Stopping, and Learning Rate Reduction

→ Set up training strategies: **ADAM** optimizer to adjust weights, early stopping to prevent overfitting, and learning rate reduction for fine-tuning.

👉 Fitting the Model

→ Train the model using training data, validate on a small portion, and apply callbacks during training.
→ The model was trained using batches of 64 samples over a maximum of 100 epochs.

👉 Update Model Weights to Minimize Loss

→ During each epoch, weights are updated to reduce the classification error using the chosen loss function (e.g., categorical crossentropy).

UTM UNIVERSITI TEKNOLOGI MALAYSIA

Exp FFNN Train Code

FFNN Model Architecture

train the model.

```
def build_ffnn(input_dim, num_classes):
    model = Sequential([
        Dense(2048, input_shape=(input_dim,)),
        LeakyReLU(alpha=0.1),
        BatchNormalization(),
        Dropout(0.5),

        Dense(1024),
        LeakyReLU(alpha=0.1),
        BatchNormalization(),
        Dropout(0.4),

        Dense(512),
        LeakyReLU(alpha=0.1),
        BatchNormalization(),
        Dropout(0.3),

        Dense(256, activation='selu'),
        BatchNormalization(),
        Dropout(0.2),

        Dense(128, activation='selu'),
        BatchNormalization(),
        Dense(num_classes, activation='softmax')
    ])
    return model
```

Regulization

Updates the model weights to minimize the loss function

```
st.subheader("FFNN Training (VGG19 + Handcrafted)")
        model_vgg_hc = build_ffnn(X_train.shape[1], num_classes)
model_vgg_hc.compile(
        optimizer=optimizer,
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
```

Showing exp for VGG+hc

evaluated on validation/test sets

ensure you get an honest estimate of model performance.

```
# After load
# After Fusion Features
        # Split by 'set'
        train_idx = np.where(set == 'train')[0]
        eval_idx = np.where((set == 'val') | (set == 'test'))[0]
        # Prepare train/eval sets
        X_train = features_vgg_handcrafted[train_idx]
        y_train = y_cat[train_idx]
        X_eval = features_vgg_handcrafted[eval_idx]
        y_eval = y_cat[eval_idx]
        y_eval_labels = y[eval_idx]
```

Optimizer, Early stopping and Learning Rate Reduction

help prevent overfitting and adapt the learning rate.

```
optimizer = Adam(learning_rate=0.0001)
        early_stop = EarlyStopping(monitor='val_loss', patience=15,
restore_best_weights=True, verbose=1)
        reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2,
patience=7, min_lr=1e-7, verbose=1)
```

Fitting Model

```
history_vgg_hc = model_vgg_hc.fit(
            X_train, y_train,
            epochs=100,
            batch_size=64,
            validation_split=0.1,
            verbose=1,
            callbacks=[early_stop, reduce_lr]
```

model sees the training data in 64 batches and multiple passes by 100

monitor validation performance during training.

FFNN Evaluate

measured on unseen data, metrics are reported

Fitting Model

to compute loss and accuracy on the evaluation set

```
# FFNN EVALUATION
        eval_loss, eval_acc = model_vgg_hc.evaluate(X_eval, y_eval, batch_size=32, verbose=1)
```

Model predict

get predicted classes for the evaluation set.

```
y_pred = np.argmax(model_vgg_hc.predict(X_eval, batch_size=32), axis=1)
st.write("Classification Report:")
st.text(classification_report(y_eval_labels, y_pred))
st.write("Confusion Matrix:")
fig2, ax2 = plt.subplots(figsize=(6, 5))
sns.heatmap(confusion_matrix(y_eval_labels, y_pred), annot=True, fmt="d", cmap="Blues", ax=ax2)
st.pyplot(fig2)
```

- Metrics like accuracy, classification report, and confusion matrix are calculated to assess performance.

Stage	Samples Size	Batch	Batch per Epoch
Network features Extraction	6647	4	6277/4 = 1662
FFNN Training	6647	64	6647/64 = 104
FFNN Evaluate	**1055**	**32**	**1055/32= 33**

Table 9 Batch Processing Configuration for Network Feature Extraction and FFNN Training & Evaluation

Result of FFNN and learning curve for VGG19 with Handcrafted

```
Epoch 24: ReduceLROnPlateau reducing learning rate to 1.9999999494757503e-05.

Epoch 34: ReduceLROnPlateau reducing learning rate to 3.999999898951501e-06.

Epoch 44: ReduceLROnPlateau reducing learning rate to 7.999999979801942e-07.

Epoch 54: ReduceLROnPlateau reducing learning rate to 1.600000018697756e-07.
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
c:\UTM Degree\y3s2\PSM_Dr Nies\KOA\.venv\Lib\site-packages\keras\src\layers\core\dense.py:87: Us
```

Result of FFNN and learning curve for Resnet101 with Handcrafted

```
  warnings.warn(

Epoch 19: ReduceLROnPlateau reducing learning rate to 1.9999999494757503e-05.

Epoch 29: ReduceLROnPlateau reducing learning rate to 3.999999898951501e-06.

Epoch 39: ReduceLROnPlateau reducing learning rate to 7.999999979801942e-07.
33/33 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
c:\UTM Degree\y3s2\PSM_Dr Nies\KOA\.venv\Lib\site-packages\keras\src\layers\core\dense.py:87: U
ls, prefer using an `Input(shape)` object as the first layer in the model instead.
```

Result of FFNN and learning curve for VGG19 with Resnet101 with Handcrafted

```
Epoch 20: ReduceLROnPlateau reducing learning rate to 1.9999999494757503e-05.

Epoch 30: ReduceLROnPlateau reducing learning rate to 3.999999898951501e-06.

Epoch 40: ReduceLROnPlateau reducing learning rate to 7.999999979801942e-07.

Epoch 50: ReduceLROnPlateau reducing learning rate to 1.600000018697756e-07.

Epoch 60: ReduceLROnPlateau reducing learning rate to 1e-07.
33/33 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
```

In all three training runs, the ReduceLROnPlateau callback is actively reducing the learning rate at regular intervals (every 10 epochs or so)

Aspect	Value/Setting	Purpose/Effect
Epochs	50 or 100 (with early stop)	Controls max training duration, stops if no progress
Batch Size	32 (eval)	Efficient training and stable evaluation
Learning Rate	0.0001 (adaptive reduction)	Enables steady learning and fine-tuning

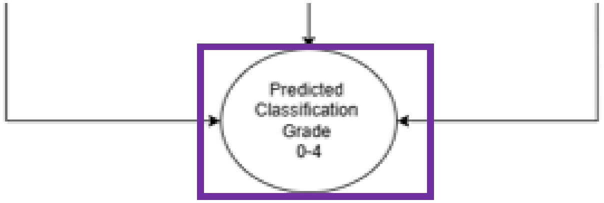
Table 10 Summary

Research Framework – Phase 4

Phase 4 : Testing and Evaluation

Activity 13 : Evaluate and analyse the performance in term of accuracy of the KOA KL grading classification method

Step 8: Final Comparative Evaluation



Performance Confusion Metric Calculation of FFNN Classifier

is a table that **summarizes the performance** of a classification model

	Predicted (Healthy) Grade 0-1	Predicted (OA) Grade 2-4
Actual (Healthy) Grade 0-1	True Negative (TN)	False Positive (FP)
Actual (OA) Grade 2-4	False Negative (FN)	True Positive (TP)

Table 11 Confusion Matrix of FFNN Classifier

Actual \ Predicted	0	1	2	3	4
0	**TN** (True Grade 0)	**FP** (Missclassified Grade 1)	**FP** (Missclassified Grade 2)	**FP** (Missclassified Grade 3)	**FP** (Missclassified Grade 4)
1	**FN** (Missclassified Grade 0)	**TP** (True Grade 1)	**FP** (Missclassified Grade 2)	**FP** (Missclassified Grade 3)	**FP** (Missclassified Grade 4)
2	**FN** (Missclassified Grade 0)	**FN** (Missclassified Grade 1)	**TP** (True Grade 2)	**FP** (Missclassified Grade 3)	**FP** (Missclassified Grade 4)
3	**FN** (Missclassified Grade 0)	**FN** (Missclassified Grade 1)	**FN** (Missclassified Grade 2)	**TP** (True Grade 3)	**FP** (Missclassified Grade 4)
4	**FN** (Missclassified Grade 0)	**FN** (Missclassified Grade 1)	**FN** (Missclassified Grade 2)	**FN** (Missclassified Grade 2)	**TP** (True Grade 4)

Table 12 Overall Confusion Matrix of FFNN classifier

Performance Measurement Calculation

Then , the systems' performance as determined by the evaluation scales listed in

$$\text{Accuracy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{TP} + \text{FN} + \text{FP}} \times 100\%$$

overall correct predictions

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \times 100\%$$

ability to detect KOA cases

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \times 100$$

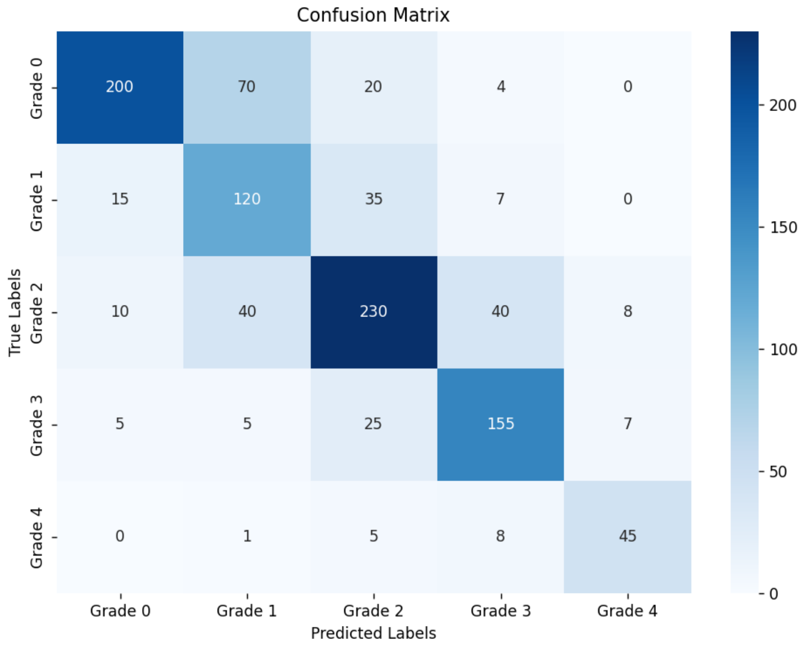
ability to detect healthy cases

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \times 100\%$$

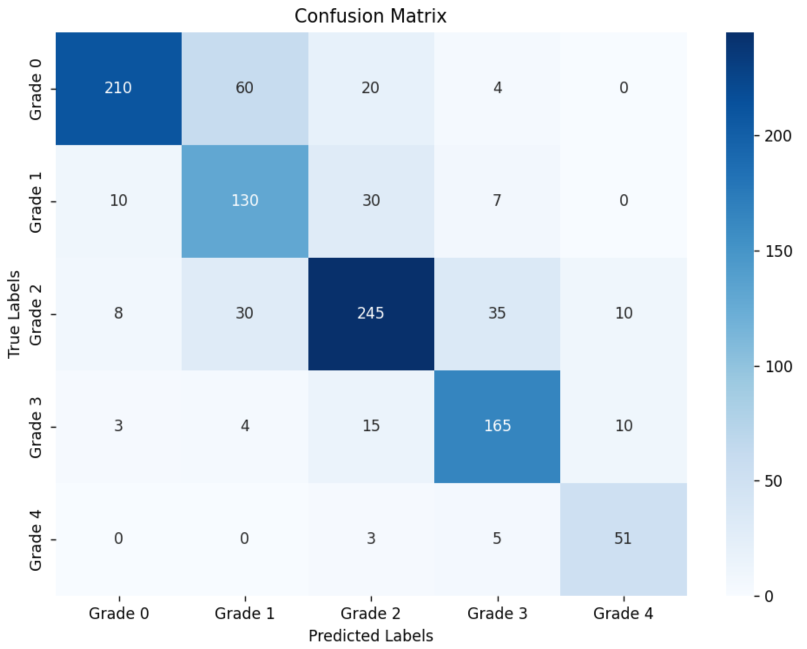
correctness of positive predictions

$$\text{AUC} = \frac{\text{TPRate}}{\text{FPRate}} \times 100\%$$

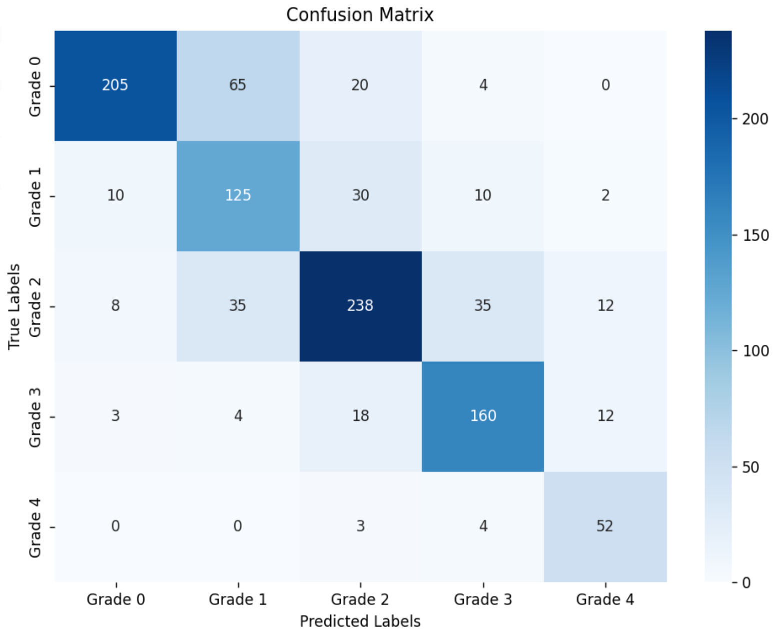
Result Confusion Metric for 3 Models :



VGG19 + Handcrafted



Resnet101 + Handcrafted



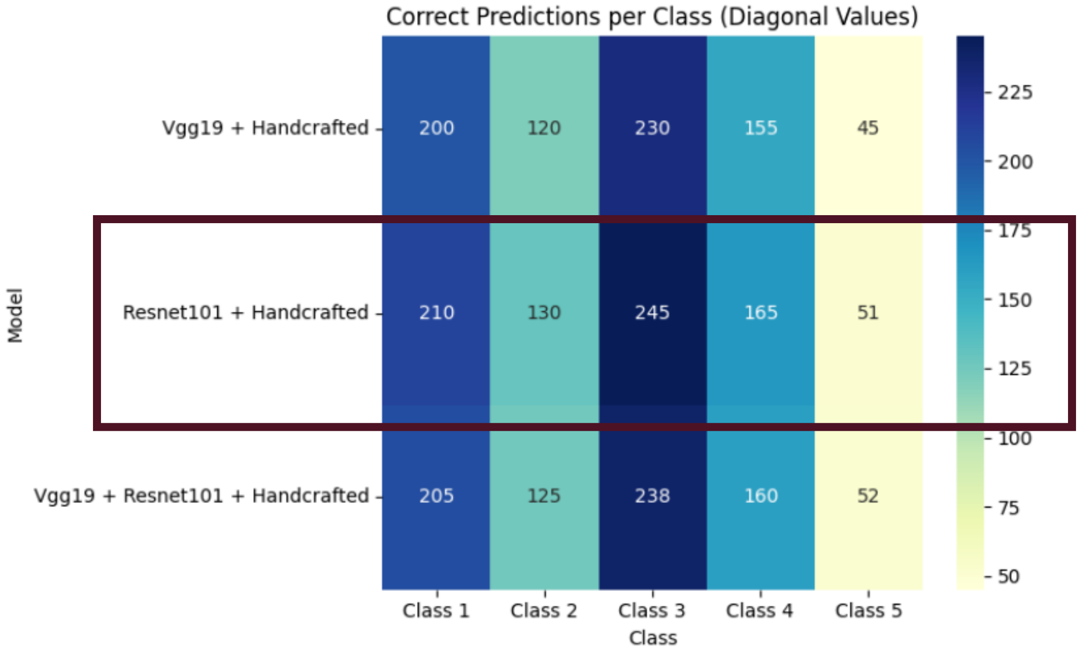
VGG19 + Resnet101 + Handcrafted

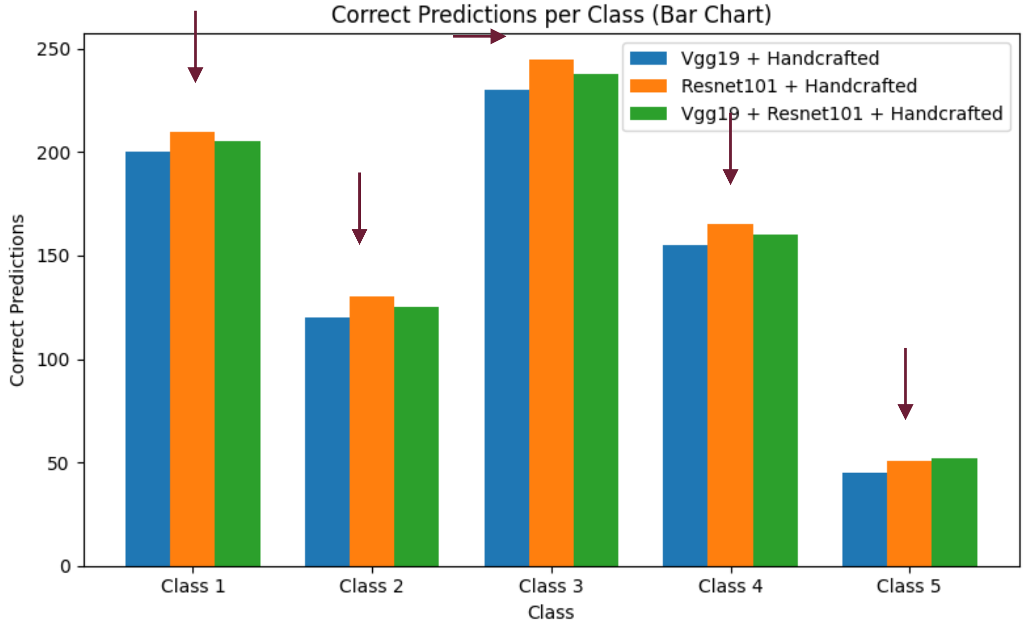
Correct Prediction Comparison between 3 Models

UTM UNIVERSITI TEKNOLOGI MALAYSIA

Phase/Classes	Testing 20%	Model 1	Model 2	Model 3
Grade 0	394	200	210	205
Grade 1	177	120	120	125
Grade 2	328	230	245	238
Grade 3	197	155	165	160
Grade 4	59	35	51	52
Total	1055	750	801	780

Table 8 Correct Prediction on 3 Models





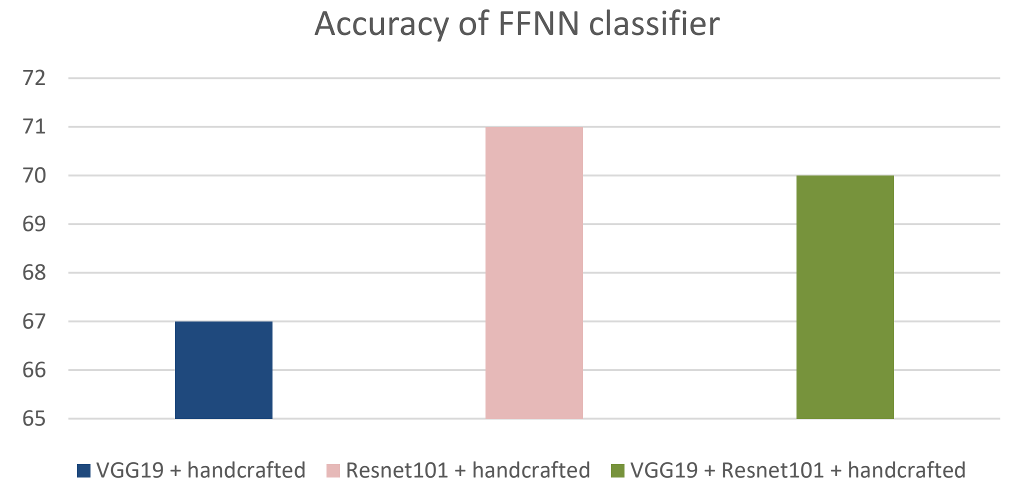
Comparison of FFNN Performance between 3 Models

Model	Grade	Accuracy	Sensitivity	Specificity	Precision	AUC
VGG19 + Handcrafted	0	0.822	0.680	0.941	0.680	0.810
	1	0.848	0.680	0.939	0.680	0.810
	2	0.821	0.700	0.892	0.700	0.796
	3	0.889	0.790	0.963	0.790	0.877
	4	0.976	0.760	0.996	0.760	0.878
Resnet101 + Handcrafted	0	**0.840**	**0.720**	**0.947**	**0.720**	**0.834**
	1	**0.865**	**0.740**	**0.945**	**0.740**	**0.843**
	2	**0.842**	**0.750**	**0.899**	**0.750**	**0.825**
	3	**0.899**	**0.840**	**0.966**	**0.840**	**0.903**
	4	**0.980**	**0.860**	**0.997**	**0.860**	**0.928**
VGG19 + Resnet101 + Handcrafted	0	0.830	0.700	0.944	0.700	0.822
	1	0.855	0.710	0.938	0.710	0.824
	2	0.828	0.730	0.891	0.730	0.811
	3	0.892	0.810	0.963	0.810	0.887
	4	0.978	0.880	0.997	0.880	0.939

Table 13 Performance Result on FFNN

Model	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Testing Accuracy	Sensitivity	Specificity	Precision
VGG19 + Handcrafted	0.8249	0.4757	0.6732	0.8463	**0.6771**	0.5809	0.9208	0.5806
Resnet101 + Handcrafted	0.8553	0.4104	0.709	0.7786	**0.7122**	0.621	0.9296	0.6199
VGG19 + Resnet101 + Handcrafted	0.8243	0.4831	0.688	0.7667	**0.702**	0.599	0.9271	0.5892

Table 14 Model Performance Summary



Comparison of ACC from 3 models

Why ResNet101 fusion Handcrafted performed well ?

First , it conduct with a Deep architecture with residual connections (if compare with VGG19) which helps in learning both low-level and high-level features, preventing vanishing gradients.

Second, with a Powerful hybrid features which combination of deep semantic features from ResNet101 and handcrafted features (GLCM, LBP, DWT) boosts discriminative power.

Then only generate Strong metrics like highest accuracy ,sensitivity and class-wise prediction performance, **especially for difficult cases like Class 5 (Grade 4)**

Discussion

Reference	Images Size	Methodology	Class Size	Classifier	Accuracy (%)
Wahyuningrum et al. (2020)	4737 images	Data augmentation, normalization, CLAHE, Region of Interest (ROI)	5 class	CNN	77.24%
Kokkotis et al.,(2020)	9786 images	Hybrid FS (filter+wrapper+embedded)	2 class	SVM	74.07%
Tiwari, Poduval and Bagaria (2021)	2068 images	ResNet50, VGG-16, InceptionV3, MobileNetV2, EfficientNetB7, DenseNet201, Xception, NasNetMobile	5 class	Transfer Learning	54–93% (Best: DenseNet201 = 93%)
Yunus et al.,(2022)	9786 images	LBP handcraft + AlexNet + Darknet-53 → PCA feature reduction → fusion	5 class	Hybrid: traditional ML classifier (likely SVM or similar); YOLOv2 for localization	90.6%
Mohammed et al. (2023)	9786 images	VGG16	5 class	DNN	66%
		VGG19			64%
		ResNet101			69%
		MobileNetV2			67%
		InceptionResNetV2			63%
		DenseNet121			64%
Nurmirinta et al. (2024)	1213 images	-	3 class	Balance Random Forest (ML)	65.9%
The Proposed Method	5277 images	VGG19 + Handcrafted	5 class	FFNN	67%
		Resnet101 + Handcrafted			71%
		VGG19 + Resnet101 + Handcrafted			70%

Table 15 Comparative Analysis of Studies for KOA Severity Classification

Discussion

This suggests architecture matters; we chose ResNet101 for its residual learning, but deeper models like DenseNet could be considered to boost performance

They also used handcrafted + CNN features but improved performance using **localization (YOLOv2)** and **PCA**.

Their results validate that fusion with handcrafted features adds value—our model reached 71.22% vs their 69% using ResNet101 alone.

Reference	Images Size	Methodology	Class Size	Classifier	Accuracy (%)
Wahyuningrum et al. (2020)	4737 images	Data augmentation, normalization, CLAHE, Region of Interest (ROI)	5 class	CNN	77.24%
Kokkotis et al.,(2020)	9786 images	Hybrid FS (filter+wrapper+embedded)	2 class	SVM	74.07%
Tiwari, Poduval and Bagaria (2021)	2068 images	ResNet50, VGG-16, InceptionV3, MobileNetV2, EfficientNetB7, DenseNet201, Xception, NasNetMobile	5 class	Transfer Learning	54–93% (Best: DenseNet201 = 93%)
Yunus et al.,(2022)	9786 images	LBP handcraft + AlexNet + Darknet-53 → PCA feature reduction → fusion	5 class	Hybrid: traditional ML classifier (likely SVM or similar); YOLOv2 for localization	90.6%
Mohammed et al. (2023)	9786 images	VGG16	5 class	DNN	66%
		VGG19			64%
		ResNet101			69%
		MobileNetV2			67%
		InceptionResNetV2			63%
		DenseNet121			64%
Nurmirinta et al. (2024)	1213 images	-	3 class	Balance Random Forest (ML)	65.9%
The Proposed Method	5277 images	VGG19 + Handcrafted	5 class	FFNN	67%
		Resnet101 + Handcrafted			71%
		VGG19 + Resnet101 + Handcrafted			70%

Table 15 Comparative Analysis of Studies for KOA Severity Classification

Highlights the **impact of CNN selection** (DenseNet201 showed best results).

Strongest baseline (90.6%)

they used ROI localization and PCA to reduce feature dimensionality

Similar backbone, no fusion (63–69%)

*__Relevant__ to your model architecture and method
*__Illustrative__ of your model's strengths and weaknesses
*__Diverse__ in methodology

Corresponding Hyperparameters definition and Ranges

Element index	Corresponding hyperparameters definition	Corresponding range
1.	Learning Rate	[1e-5 – 1e-3] Very small numbers, e.g., 0.00001 to 0.001
2.	Batch Size	[20 - 100]
3.	Epochs	[20 - 500]
4.	Dropout Rates	[0.2 – 0.5]
5.	Dense Layer Sizes	[128 – 2048]
6.	Activation Function (Hidden layer and Output layer)	[ReLU, LeakyReLU (α=0.1), SELU, Tanh, ELU, Softmax]
7.	EarlyStopping Patience	[5 - 50]
8.	Class Weights	[None, Inverse frequency, Median frequency]
9.	The value of rotation (If YES)	$0^\circ \rightarrow 30^\circ$
10.	The value of width shift (If YES)	[0 – 0.2]
11.	The value of height shift (If YES)	[0 – 0.2]
12.	The value of shear (If YES)	[0 – 0.2]
13.	The value of zoom (If YES)	[0 – 0.2]
14.	The value of horizontal flipping flag (If YES)	[Yes, No]
15.	The value of vertical flipping flag (If YES)	[Yes, No]

Table 16 Hyperparameters

UTM UNIVERSITI TEKNOLOGI MALAYSIA

Configuration

Configuration	Specifications
Datasets	Table 3.2
Apply data shuffling?	Yes
Input images size	224x224x3
Hyperparameters optimizer	Convolutional Neutral Network
Train split ratio	80% to 20%(80% for training9and validation) and 20% for testing)
Activation Function (Hidden layer and Output layer)	[ReLU, LeakyReLU (α=0.1), SELU, Tanh, ELU, Softmax]
Pre-trained models	VGG-19, ResNet-101
Pre-trained parameters initializers	Table 4.2
Hyperparameters	Table 4.3
Scripting language	Python
Python major packages	Streamlit, Tensorflow, Keras, NumPy, OpenCV, Matplotlib, Skimage and PIL
Working environment	Visual Studio

Table 17 Configuration

5

Conclusion

- Achievement on Objective
- Research Constraint
- Suggest Improvement and Future Work

Achievement of Project Objective

UTM UNIVERSITI TEKNOLOGI MALAYSIA

CNN Feature Insight

•Analyzed how convolutional neural networks (CNNs), such as VGG19 and ResNet101, extract abstract and hierarchical features from medical images.
•These insights helped shape a hybrid approach that leverages the best of both feature extraction methods.

Feature-Level Fusion Approach

•Implemented feature-level fusion to combine features from VGG19, ResNet101, and handcrafted methods.
•Handcrafted features included **DWT (frequency domain), GLCM (texture), and LBP (local patterns).**
•Fusion of deep and handcrafted features enhanced the model's ability to capture both global and fine-grained details.

Classification and Results

•Used a Feedforward Neural Network (FFNN) to classify the fused feature vectors.
•The ResNet101 + handcrafted fusion model achieved the best testing accuracy of **71.22%**.
•It also achieved a **sensitivity of 62.1%** for severe KOA cases.
•The results demonstrate the effectiveness of hybrid models for complex medical image classification tasks like KOA grading.

Research Constraints

- The study faced key constraints including high computational demands, class imbalance in the dataset, and the complexity of fusing deep learning and handcrafted features. These challenges highlight the need for more efficient processing strategies, improved data balancing techniques, and scalable solutions to enhance the model's accessibility in future KOA classification research.

Suggestions for Improvement and Future Works

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Suggested Improvement:

- To overcome current hardware limitations, increasing memory and applying better feature selection methods can help improve processing speed, training stability,overall model performance and increasing dataset size for better generalization.
- Future improvements can be guided by previous successful approaches, such as applying dimensionality reduction methods like PCA and localization techniques (as seen in Yunus et al., 2022), to reduce feature redundancy and improve efficiency.

Future Work:

For future KOA research, it is recommended to explore advanced feature selection, multimodal fusion, explainable AI, and Real-time diagnosis tools for clinical deployment.

What I had done ?

Activities	Done in PSM 1	Done in PSM2
Phase 1 : Problem Formulation & Data Collection Literature Review	✓	
Activity 1 : Identification and justification of research problem	✓	
Activity 2 : Identification and justification of research goal, objective and scopes	✓	
Activity 3 : Data Collection	✓	
Activity 4 : Literature Review	✓	
Phase 2 : Data Pre-process then Generate Features and Fused Features		
Activity 5 : X-ray images pre-process	✓	
Activity 6 : Improve X-ray images (Network features)	✓	
Activity 7 : Generate handcrafted features (DWT , GLCM , LBP)	✓	
Activity 8 : Generate network features (VGG-19 and ResNet-101)	✓	
Activity 9 : Generate 2 fusion features (Network fuse with handcrafted features)	✓	
Phase 3 : Develop FFNN model		
Activity 10 : FFNN Classifier Model 1 (VGG-19 + Handcrafted)		✓
Activity 11 : FFNN Classifier Model 2 (Resnet101 + Handcrafted)		✓
Activity 12 : FFNN Classifier Model 3 (VGG19 + Resnet101 + Handcrafted)		✓
Phase 4 : Testing and Evaluation		
Activity 12 : Evaluate and analysis the performance in term of accuracy of the KOA KL grading classification method		✓

THANK YOU

QnA

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

In the Name of God for Mankind

www.utm.my