



FINAL YEAR PROJECT 2 : Demo Research Result

CLASSIFICATION OF KNEE OSTEOARTHRITIS 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

Demo Result OUTLINE

1

DataAutoSelection.py

- Sharpness Measurement to select X-ray images

2

DataPreparation.py

- Handcrafted Preprocess
- CNN Preprocess
- Split Dataset
- Augment Training Set

3

FeatureExtraction.py

- VGG19 & ResNet101
- DWT , GLCM and LBP

4

Fusion and FFNN Train&Evaluate.py

- FFNN Model Architecture
- Feature-level Fusion
- Performance Evaluate of Feed-Forward Neural Network on 3 Models

5

Result Analysis & Comparative

- Confusion Matric on FFNN classify
- Comparison of FFNN Performance between 3 Models
- Discussion

Architecture Diagram

1 Data AutoSelection will be made before the "Preprocess" images

Step 1: Preprocess Data

2

Step 2: Split Dataset

Step 3: Augment Training Set Only

Step 4: Extract CNN Features on Preprocessed CNN Images

3

Step 5: Extract Handcrafted Features on Preprocessed Hand Images

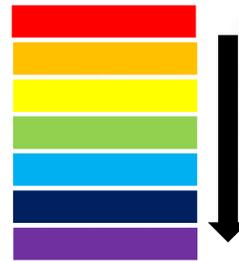
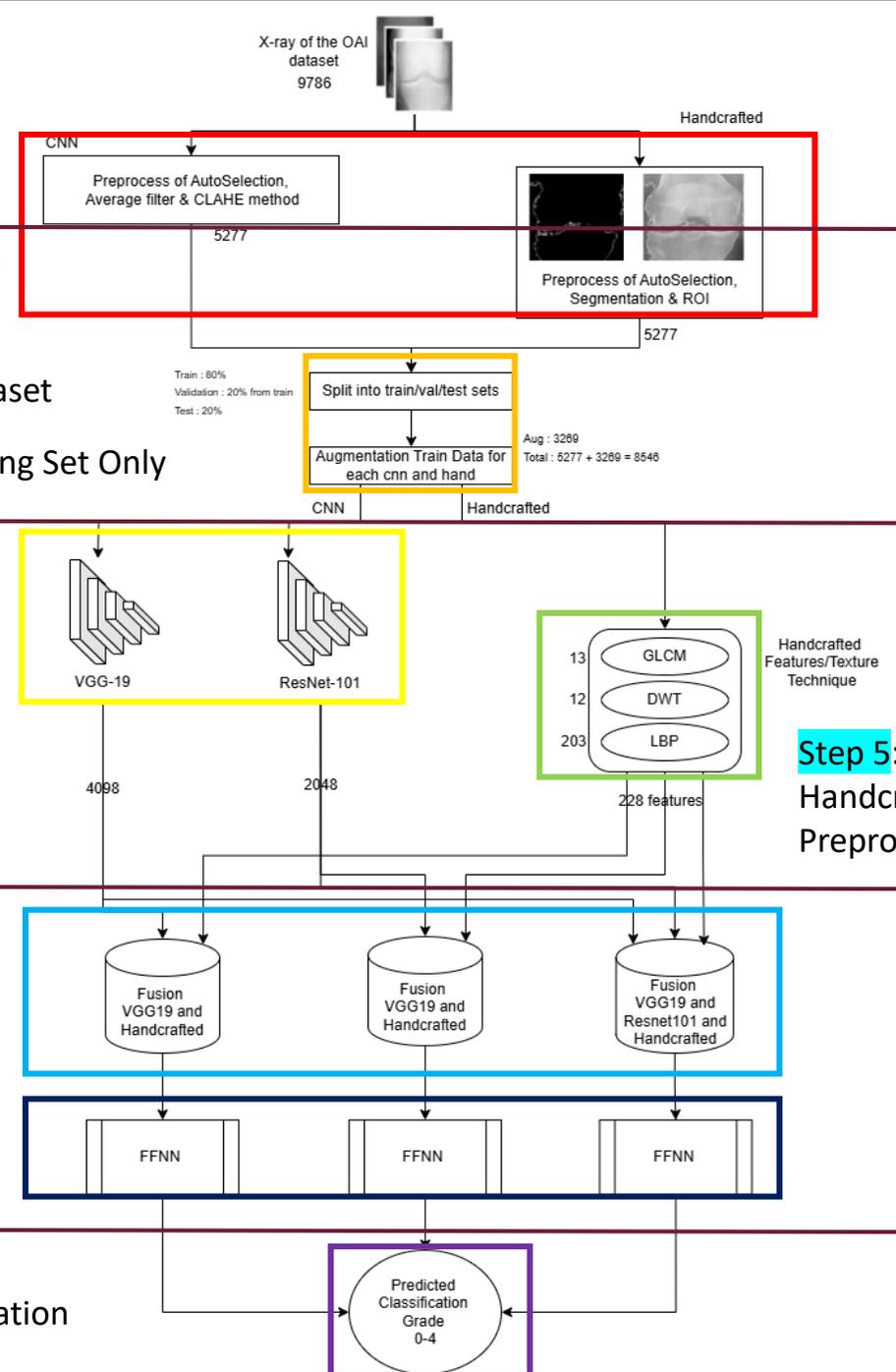
4

Step 6: Concatenate Features

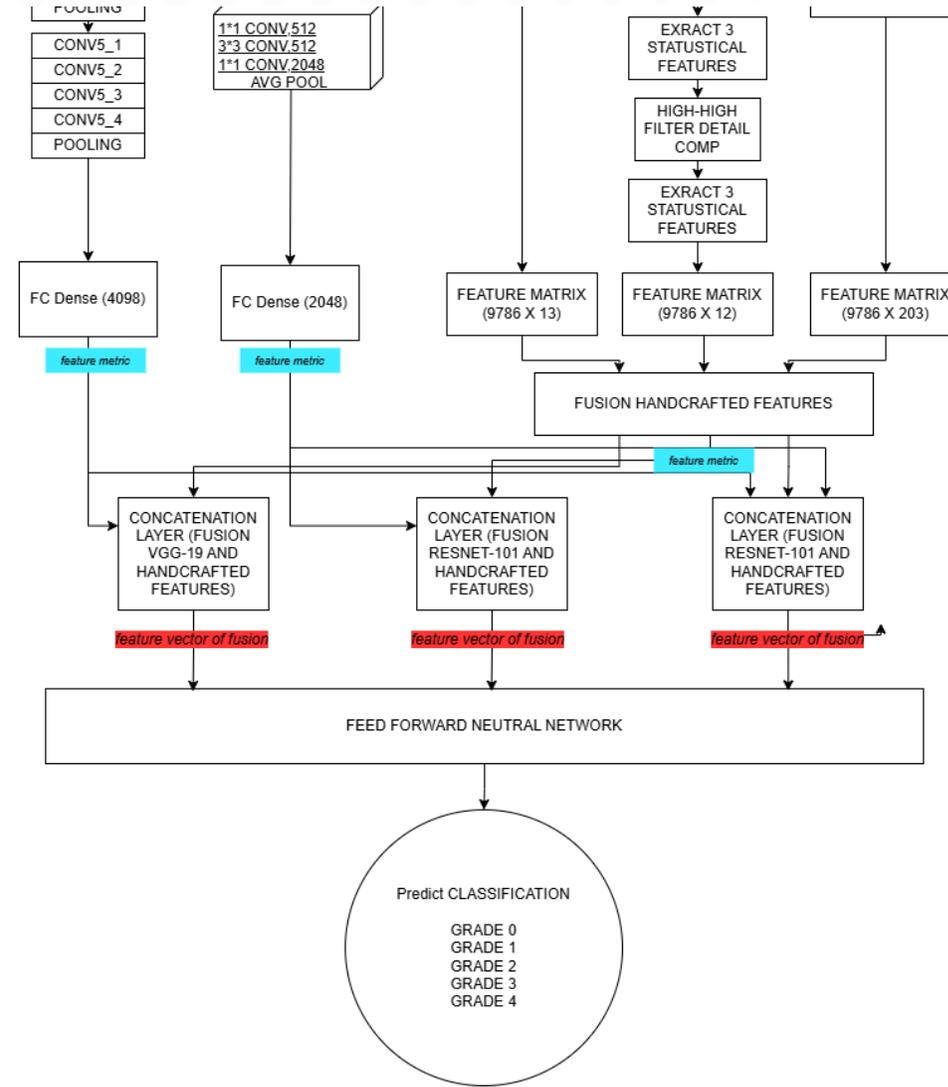
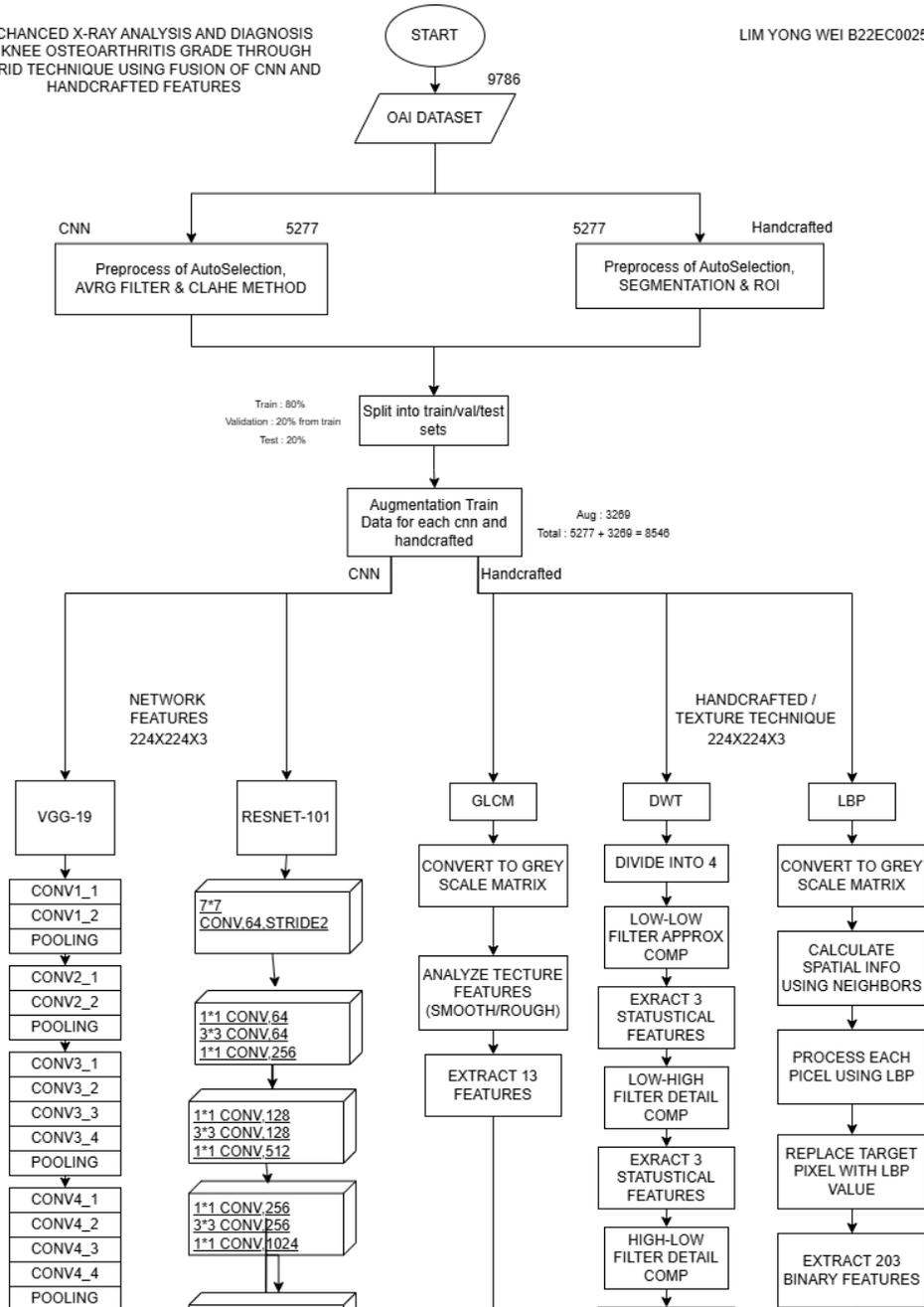
Step 7: FFNN Classification

5

Step 8: Final Comparative Evaluation



Proposed Model Diagram

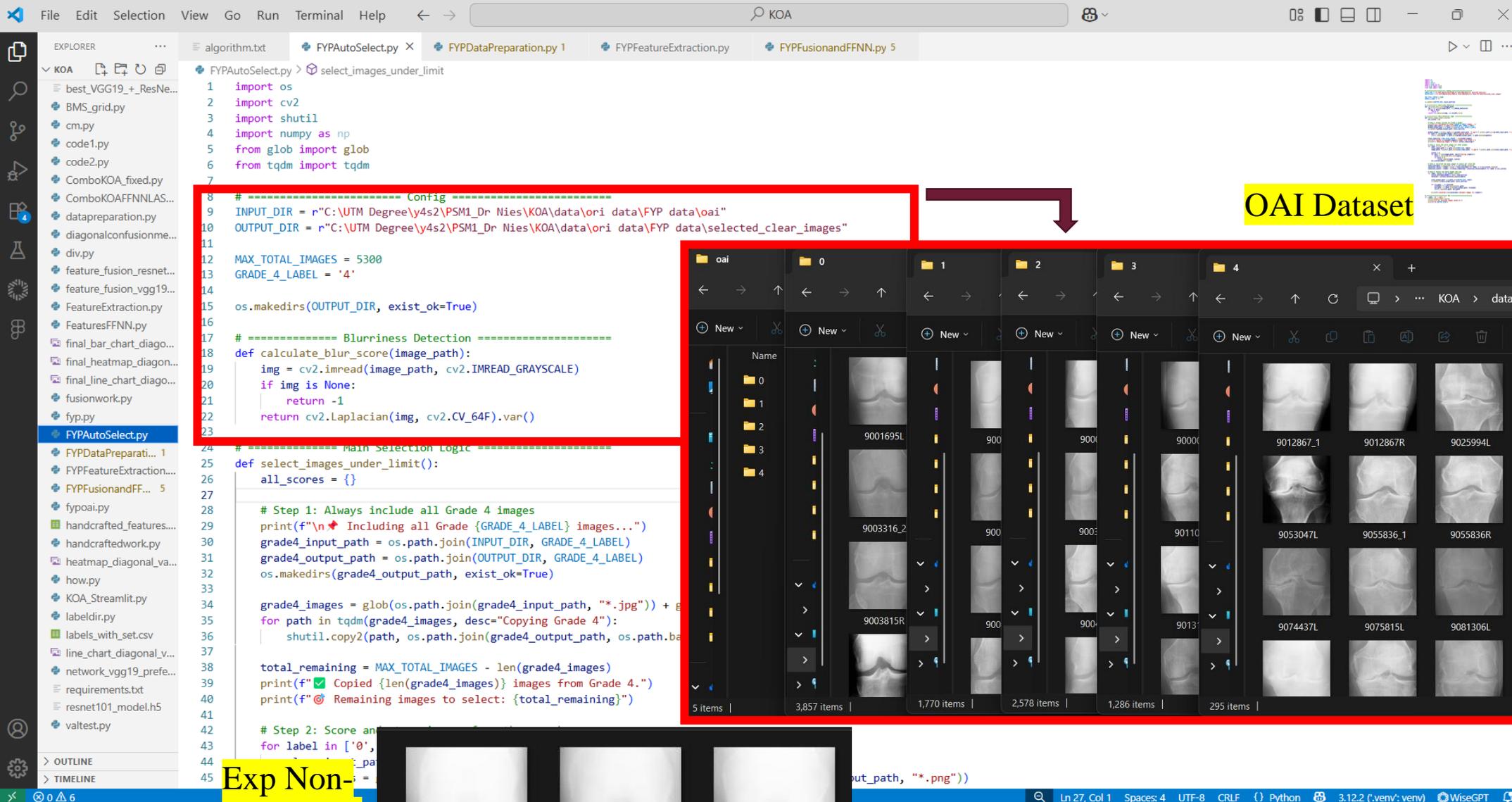


1

Data AutoSelection will be made before the “Preprocess” images



Datasets AutoSelection – Sharpness Measurement to select X-ray Images

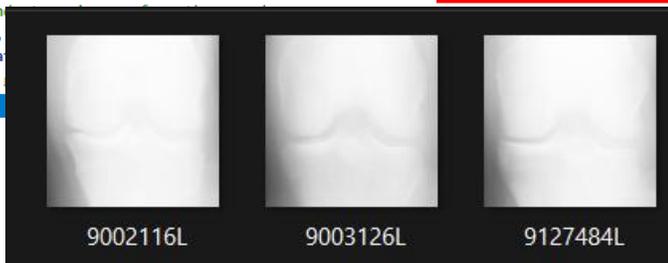


The image shows a Python IDE with a file explorer on the left and a code editor in the center. The code editor displays the following Python code:

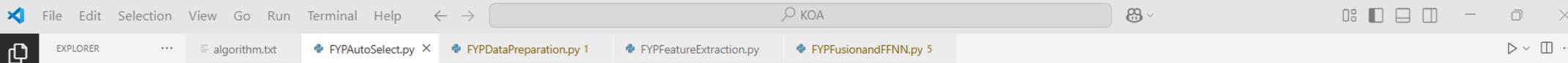
```
1 import os
2 import cv2
3 import shutil
4 import numpy as np
5 from glob import glob
6 from tqdm import tqdm
7
8 # ===== Config =====
9 INPUT_DIR = r"C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\oai"
10 OUTPUT_DIR = r"C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\selected_clear_images"
11
12 MAX_TOTAL_IMAGES = 5300
13 GRADE_4_LABEL = '4'
14
15 os.makedirs(OUTPUT_DIR, exist_ok=True)
16
17 # ===== Blurriness Detection =====
18 def calculate_blur_score(image_path):
19     img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
20     if img is None:
21         return -1
22     return cv2.Laplacian(img, cv2.CV_64F).var()
23
24 # ===== Main Selection Logic =====
25 def select_images_under_limit():
26     all_scores = {}
27
28     # Step 1: Always include all Grade 4 images
29     print(f"\n* Including all Grade {GRADE_4_LABEL} images...")
30     grade4_input_path = os.path.join(INPUT_DIR, GRADE_4_LABEL)
31     grade4_output_path = os.path.join(OUTPUT_DIR, GRADE_4_LABEL)
32     os.makedirs(grade4_output_path, exist_ok=True)
33
34     grade4_images = glob(os.path.join(grade4_input_path, "*.jpg")) + g
35     for path in tqdm(grade4_images, desc="Copying Grade 4"):
36         shutil.copy2(path, os.path.join(grade4_output_path, os.path.ba
37
38     total_remaining = MAX_TOTAL_IMAGES - len(grade4_images)
39     print(f"✓ Copied {len(grade4_images)} images from Grade 4.")
40     print(f"🕒 Remaining images to select: {total_remaining}")
41
42     # Step 2: Score an
43     for label in ['0',
44
45
```

The file explorer on the right shows a directory structure with folders labeled 0, 1, 2, 3, and 4. Each folder contains X-ray images. A yellow box highlights the text "OAI Dataset" in the top right corner. A red box highlights the code in the editor. A red arrow points from the code to the file explorer. At the bottom, three X-ray images are shown with labels 9002116L, 9003126L, and 9127484L. A yellow box highlights the text "Exp Non-clear ORI Images" in the bottom left corner.

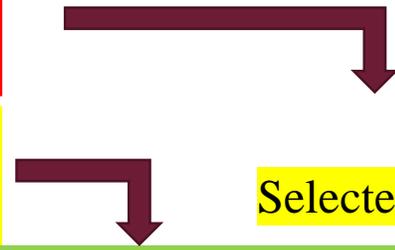
Exp Non-clear ORI Images



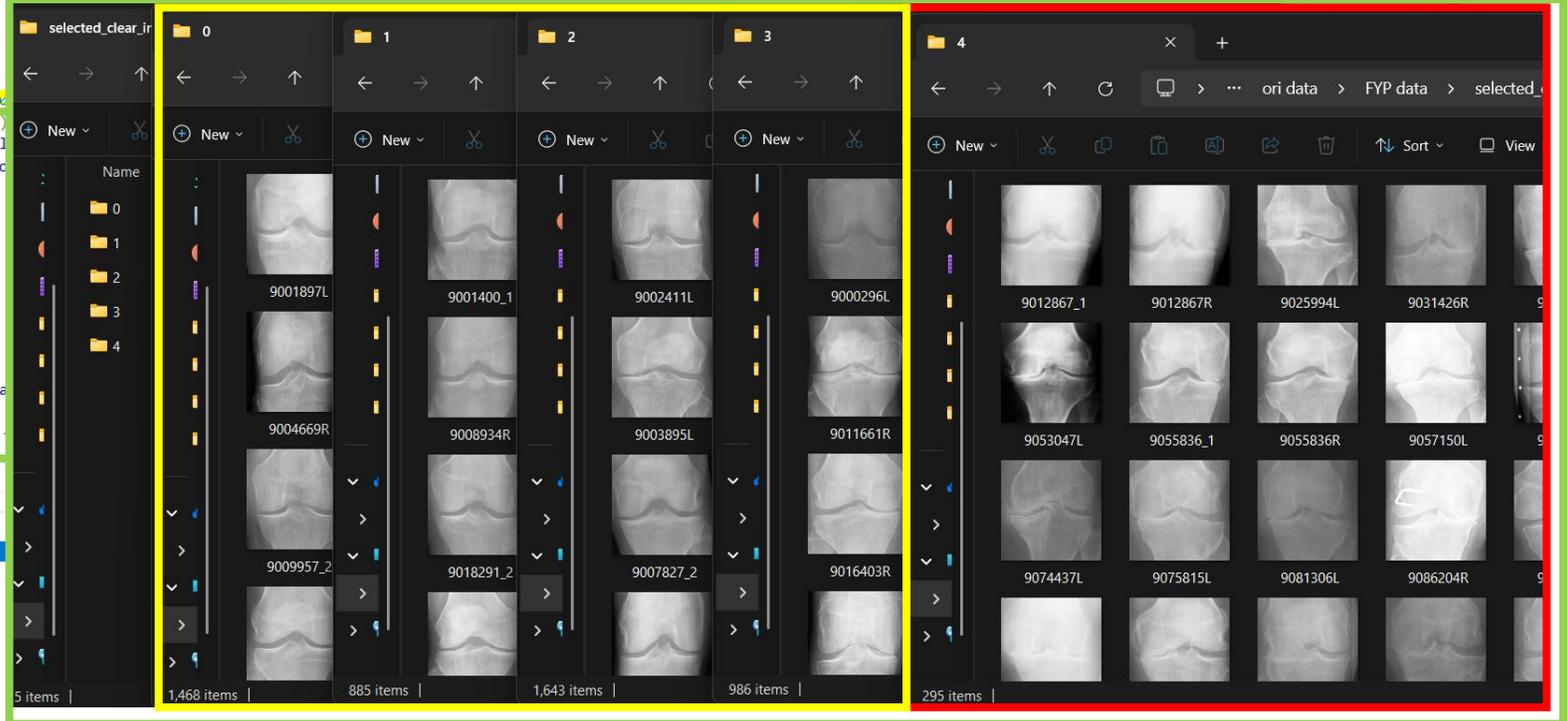
Datasets AutoSelection - Result



```
25 def select_images_under_limit():
26     all_scores = {}
27     # Step 1: Always include all Grade 4 images
28     print(f"\n* Including all Grade {GRADE_4_LABEL} images...")
29     grade4_input_path = os.path.join(INPUT_DIR, GRADE_4_LABEL)
30     grade4_output_path = os.path.join(OUTPUT_DIR, GRADE_4_LABEL)
31     os.makedirs(grade4_output_path, exist_ok=True)
32     grade4_images = glob(os.path.join(grade4_input_path, "*.jpg")) + glob(os.path.join(grade4_input_path, "*.png"))
33     for path in tqdm(grade4_images, desc="Copying Grade 4"):
34         shutil.copy2(path, os.path.join(grade4_output_path, os.path.basename(path)))
35
36     total_remaining = MAX_TOTAL_IMAGES - len(grade4_images)
37     print(f"✓ Copied {len(grade4_images)} images from Grade 4.")
38     print(f"⊗ Remaining images to select: {total_remaining}")
39     # Step 2: Score and store images for other grades
40     for label in ['0', '1', '2', '3']:
41         class_input_path = os.path.join(INPUT_DIR, label)
42         image_paths = glob(os.path.join(class_input_path, "*.jpg")) + glob(os.path.join(class_input_path, "*.png"))
43
44         scores = []
45         for path in tqdm(image_paths, desc=f"Scoring {label}"):
46             score = calculate_blur_score(path)
47             if score >= 0:
48                 scores.append((path, score))
49         all_scores[label] = scores
50     # Step 3: Calculate how many images to select per class
51     total_available = sum(len(v) for v in all_scores.values())
52     selection_ratios = {label: len(v) / total_available for label, v in all_scores.items()}
53     selection_counts = {label: int(total_remaining * selection_ratios[label]) for label in selection_ratios}
54
55     # Step 4: Select top sharp images and copy
56     for label, scores in all_scores.items():
57         scores.sort(key=lambda x: x[1], reverse=True)
58         selected = scores[:selection_counts[label]]
59         class_output_path = os.path.join(OUTPUT_DIR, label)
60         os.makedirs(class_output_path, exist_ok=True)
61         for src_path, _ in selected:
62             filename = os.path.basename(src_path)
63             dst_path = os.path.join(class_output_path, filename)
64             shutil.copy2(src_path, dst_path)
65             print(f"✓ Selected {len(selected)} sharpest images")
66
67 if __name__ == "__main__":
68     select_images_under_limit()
69     print(f"\n* All selected images saved to:")
```



Selected_Clear_images



2

Step 1: Preprocess Data

Step 2: Split Dataset

Step 3: Augment Training Set Only

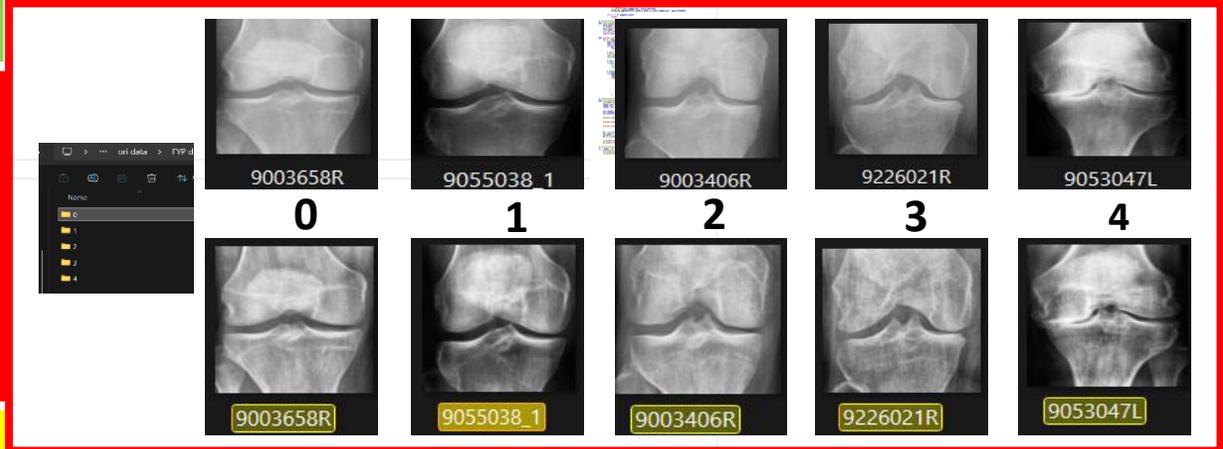


Data Preparation – Handcrafted and CNN Preprocess

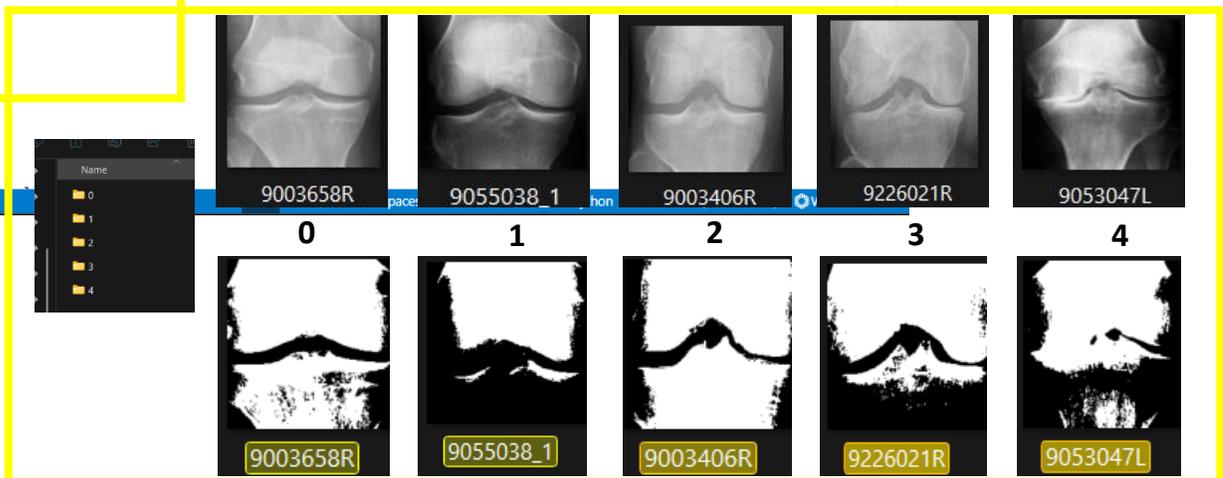
```
File Edit Selection View Go Run Terminal Help ← → KOA
EXPLORER
algorithm.txt
FYPAutoSelect.py
FYPDataPreparation.py 1 x
FYPFeatureExtraction.py
FYPFusionandFFNN.py 5
KOA
best_VGG19+_ResNe...
BMS_grid.py
cm.py
code1.py
code2.py
ComboKOA_fixe...
ComboKOAFFNN_S...
datapreparation...
diagonalconfusio...
div.py
feature_fusion_re...
feature_fusion_v...
FeatureExtraction...
FeaturesFFNN.py
final_bar_chart...
final_heatmap_d...
final_line_chart...
fusionwork.py
fyp.py
FYPAutoSelect.py
FYPDataPreparat...
FYPFeatureExtra...
FYPFusionandFF...
fypoi.py
handcrafted_fea...
handcraftedwork...
heatmap_diagon...
how.py
KOA_Streamlit.py
labeldir.py
labels_with_set...
line_chart_diago...
network_vgg19+_...
requirements.txt
resnet101_model...
valtest.py
OUTLINE
TIMELINE
6
```

```
8 from tensorflow.keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array
9
10 # ----- Configuration -----
11 INPUT_DIR = r"C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\selected_clear_images"
12 IMAGE_SIZE = (224, 224)
13 TRAIN_RATIO = 0.8
14 VAL_RATIO = 0.2
15 SEED = 42
16 # Output paths
17 handcrafted_output_base = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\hand\augSplitAfter'
18 cnn_output_base = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\cnn\augSplitAfter'
19
20 # ===== Preprocessing Functions =====
21 # CNN preprocessing: CLAHE + Avg Filter
22 def preprocess_cnn(img_path):
23     img = cv2.imread(img_path)
24     if img is None:
25         return None
26     h, w = img.shape[:2]
27     crop = img[int(h*0.1):int(h*0.9), int(w*0.1):int(w*0.9)]
28     gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
29     gray_resized = cv2.resize(gray, IMAGE_SIZE)
30     clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
31     clahe_img = clahe.apply(gray_resized)
32     avg_filtered = cv2.blur(clahe_img, (3, 3))
33     final_img = cv2.cvtColor(avg_filtered, cv2.COLOR_GRAY2BGR)
34     return final_img
35
36 # handcrafted preprocessing: ROI + basic threshold segmentation
37 def preprocess_handcrafted(img_path):
38     img = cv2.imread(img_path)
39     if img is None:
40         return None
41     h, w = img.shape[:2]
42     crop = img[int(h*0.1):int(h*0.9), int(w*0.1):int(w*0.9)]
43     gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
44     gray_resized = cv2.resize(gray, IMAGE_SIZE)
45     segmented = cv2.threshold(gray_resized, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
46     final_img = cv2.cvtColor(segmented, cv2.COLOR_GRAY2BGR)
47     return final_img
48
49 # ===== Data Split Functions =====
50 def load_images_by_class():
51     image_info = []
52     for label in ['0', '1', '2', '3', '4']:
```

CNN preprocess



Handcrafted preprocess



Data Preparation – Splitting and Augmentation

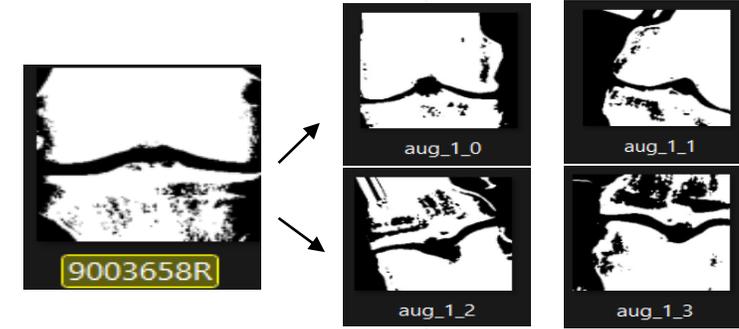
```

11 INPUT_DIR = r"C:\Users\Enggie\Documents\KOA\train_data_dir"
12 IMAGE_SIZE = (224, 224)
13 TRAIN_RATIO = 0.8
14 VAL_RATIO = 0.2
15 SEED = 42
16 # Output paths
17 handcrafted_output_base = r"C:\UTM Degree\y4s2\PSM1"
18 cnn_output_base = r"C:\UTM Degree\y4s2\PSM1 Dr Nies"
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50 def load_images_by_class():
51     image_info = []
52     for label in ['0', '1', '2', '3', '4']:
53         paths = glob(os.path.join(INPUT_DIR, label, "*.jpg")) + glob(os.path.join(INPUT_DIR, label, "*.png"))
54         for p in paths:
55             image_info.append((p, label))
56     return image_info
57
58 def split_data(image_info):
59     train_val, test = train_test_split(image_info, test_size=(1 - TRAIN_RATIO),
60                                     stratify=[lbl for _, lbl in image_info], random_state=SEED)
61     train, val = train_test_split(train_val, test_size=VAL_RATIO,
62                                 stratify=[lbl for _, lbl in train_val], random_state=SEED)
63     return train, val, test
64
65 # ===== Augmentation =====
66 datagen = ImageDataGenerator(
67     rotation_range=30,
68     width_shift_range=0.2,
69     height_shift_range=0.2,
70     shear_range=0.2,
71     zoom_range=0.2,
72     horizontal_flip=True,
73     vertical_flip=True,
74     fill_mode='nearest'
75 )
76
77 def augment_and_save(img_array, label, img_index, output_dir, csv_records, augment_count=2, all_images_dir=None):
78     img_array = img_array.reshape((1,) + img_array.shape)
79     for i, batch in enumerate(datagen.flow(img_array, batch_size=1)):
80         aug_filename = f"aug_{label}_{img_index}_{i}.jpg"
81         aug_path = os.path.join(output_dir, aug_filename)
82         array_to_img(batch[0]).save(aug_path)
83         csv_records.append((aug_filename, label, "train"))
84
85         if all_images_dir:
86             os.makedirs(all_images_dir, exist_ok=True)
87             array_to_img(batch[0]).save(os.path.join(all_images_dir, aug_filename))
88
89         if i + 1 >= augment_count:
90             break
91
92 # ===== Image Saving =====
93 def save_image(img, filename, base_output_dir, set_name, label):

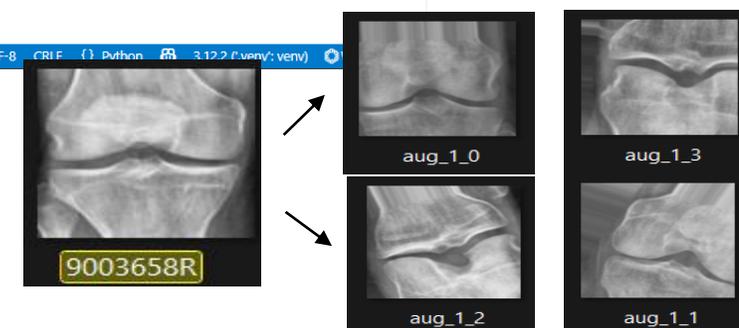
```

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

Augmentation – Handcrafted Preprocessed Images



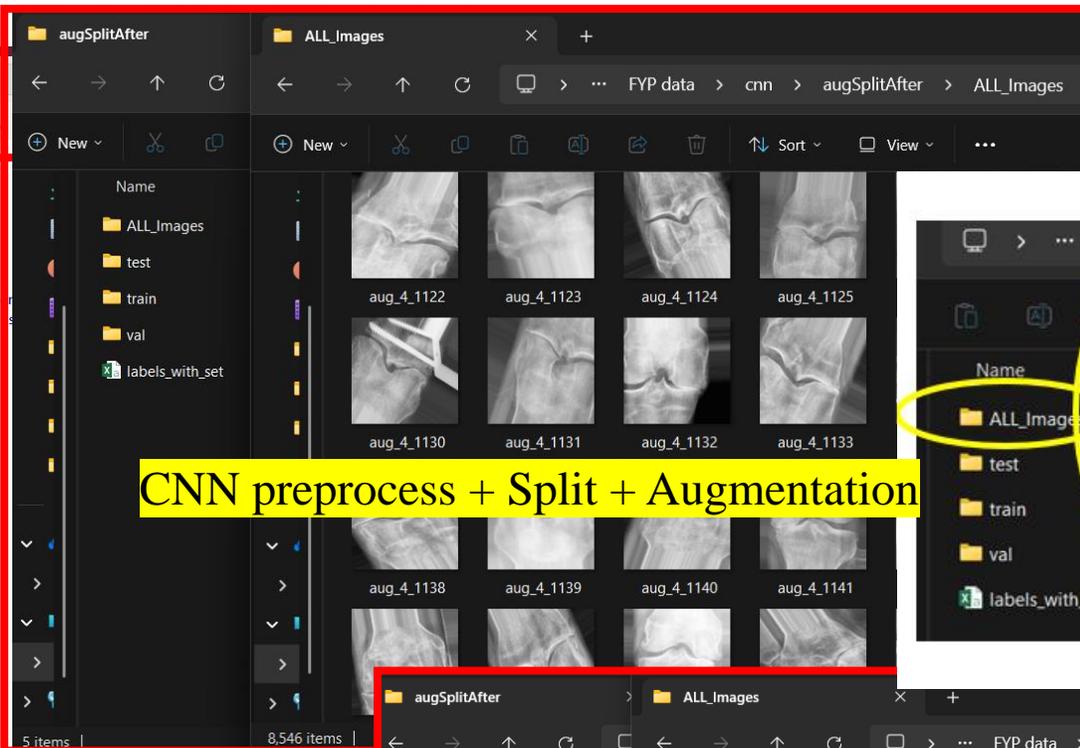
Augmentation – CNN Preprocessed Images



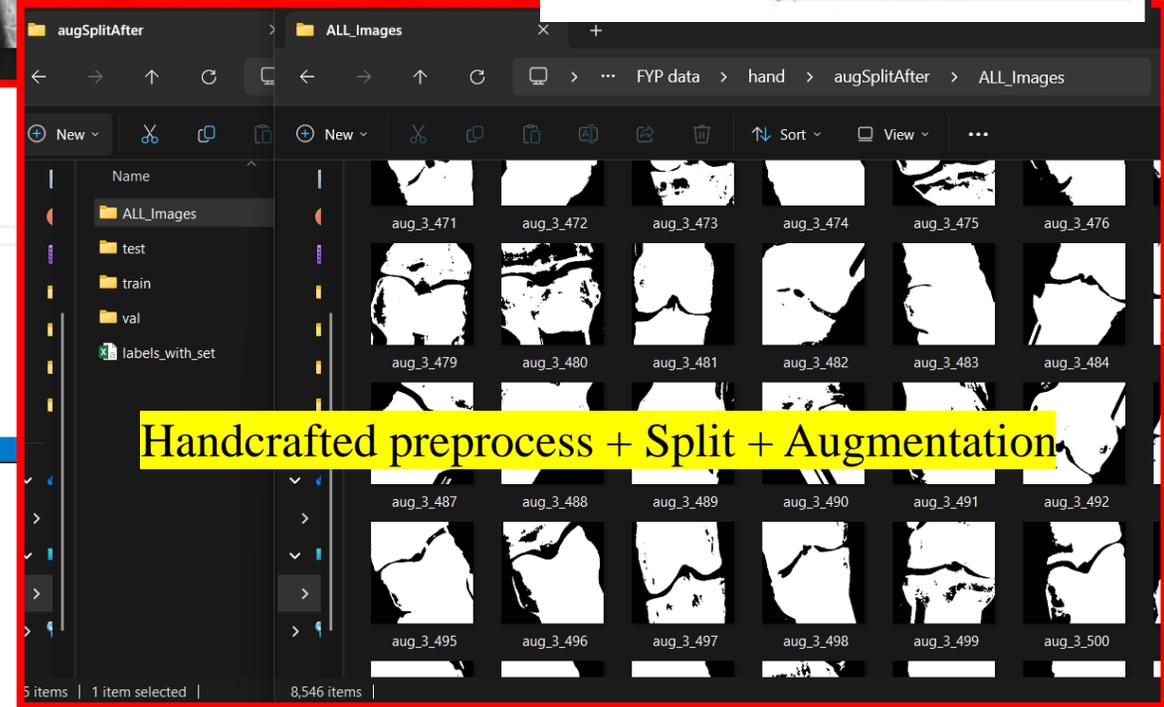
Data Preparation – Result

```
File Edit Selection View Go Run Terminal Help  
EXPLORER  
▼ KOA  
  best_VGG19_+ResNe...  
  BMS_grid.py  
  cm.py  
  code1.py  
  code2.py  
  ComboKOA_fixed.py  
  ComboKOAFFNNLAS...  
  datapreparation.py  
  diagonalconfusionme...  
  div.py  
  feature_fusion_resnet...  
  feature_fusion_vgg19...  
  FeatureExtraction.py  
  FeaturesFFNN.py  
  final_bar_chart_diago...  
  final_heatmap_diagon...  
  final_line_chart_diago...  
  fusionwork.py  
  fyp.py  
  FYPAutoSelect.py  
  FYPDataPreparati... 1  
  FYPFeatureExtraction...  
  FYPFusionandFF... 5  
  fypoi.py  
  handcrafted_features...  
  handcraftedwork.py  
  heatmap_diagonal_va...  
  how.py  
  KOA_Streamlit.py  
  labeldir.py  
  labels_with_set.csv  
  line_chart_diagonal_v...  
  network_vgg19_prefe...  
  requirements.txt  
  resnet101_model.h5  
  valtest.py  
  > OUTLINE  
  > TIMELINE  
  0 6  
  1  
  5  
  es...  
  y  
  va...  
  137  
  138  
  139  
  140  
  141  
  142  
  143  
  144  
  145  
  146  
  147  
  148  
  149
```

```
def save_image(img, filename, base_output_dir, set_name, label):  
    save_path = os.path.join(base_output_dir, set_name, label)  
    os.makedirs(save_path, exist_ok=True)  
    full_path = os.path.join(save_path, filename)  
    cv2.imwrite(full_path, img)  
    return full_path  
  
def process_and_save(data_split, set_name, csv_records, base_output_dir, augment):  
    for i, (img_path, label) in enumerate(tqdm(data_split, desc=f"Processing {set_name}")):  
        base_filename = f"{set_name}_{label}_{i}.jpg"  
        img = preprocess_fn(img_path)  
        if img is None:  
            continue  
        # Save to class subfolder  
        save_image(img, base_filename, base_output_dir, set_name, label)  
        csv_records.append((base_filename, label, set_name))  
        # Save to ALL_Images  
        if all_images_dir:  
            os.makedirs(all_images_dir, exist_ok=True)  
            cv2.imwrite(os.path.join(all_images_dir, base_filename), img)  
        # Augmentation for train only  
        if augment and set_name == "train":  
            img_array = img_to_array(img)  
            augment_and_save(  
                img_array=img_array,  
                label=label,  
                img_index=i,  
                output_dir=os.path.join(base_output_dir, set_name, label),  
                csv_records=csv_records,  
                augment_count=2,  
                all_images_dir=all_images_dir  
            )  
    # ===== Pipeline Runner =====  
def run_pipeline(output_base_dir, csv_output_filename, preprocess_fn):  
    print(f"Running pipeline for: {output_base_dir}")  
    image_info = load_images_by_class()  
    train, val, test = split_data(image_info)  
    all_images_dir = os.path.join(output_base_dir, "ALL_Images")  
    csv_records = []  
    process_and_save(train, "train", csv_records, output_base_dir, augment=True,  
                    preprocess_fn=preprocess_fn, all_images_dir=all_images_dir)  
    process_and_save(val, "val", csv_records, output_base_dir, augment=False,  
                    preprocess_fn=preprocess_fn, all_images_dir=all_images_dir)  
    process_and_save(test, "test", csv_records, output_base_dir, augment=False,  
                    preprocess_fn=preprocess_fn, all_images_dir=all_images_dir)  
  
df = pd.DataFrame(csv_records, columns=["filename", "label", "set"])  
csv_path = os.path.join(output_base_dir, csv_output_filename)  
df.to_csv(csv_path, index=False)  
  
print(f"CSV saved to: {csv_path}")  
print(f"Total processed: {len(csv_records)} images")  
  
# ===== Main =====  
if __name__ == "__main__":  
    run_pipeline(handcrafted_output_base, "labels_with_set.csv", preprocess_fn=preprocess_handcrafted)  
    run_pipeline(cnn_output_base, "labels_with_set.csv", preprocess_fn=preprocess_cnn)
```

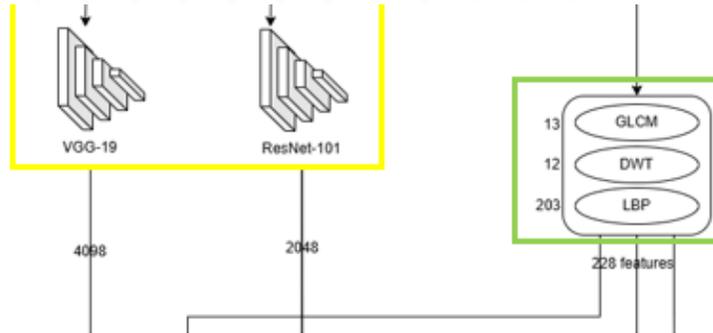


	A	B	D	E	F
1	file name	label	set		
8527	9997869R.png	4	test		
8528	9998089L.png	4	test		
8529	9998089R.png	4	test		
8530	9998089_1.png	4	test		
8531	9998089_2.png	4	test		
8532	9998184L.png	4	test		
8533	9998184R.png	4	test		
8534	9998184_1.png	4	test		
8535	9999295L.png	4	test		
8536	9999295R.png	4	test		
8537	9999365L.png	4	test		
8538	9999365R.png	4	test		
8539	9999510L.png	4	test		
8540	9999510R.png	4	test		
8541	9999862L.png	4	test		
8542	9999862R.png	4	test		
8543	9999862_1.png	4	test		
8544	9999865L.png	4	test		
8545	9999865R.png	4	test		
8546	9999878L.png	4	test		
8547	9999878R.png	4	test		
8548					
8549					
8550					



3

Step 4: Extract CNN Features on Preprocessed CNN Images



Step 5: Extract Handcrafted Features on Preprocessed Hand Images

Feature Extraction – VGG19 and Resnet101

Vgg19 Model Summary

```

File Edit Selection View Go Run Terminal Help ← → KOA
EXPLORER
algorithm.txt FYPAutoSelect.py FYPDataPreparation.py 1 FYPFeatureExtraction.py 4 X FYPFusionandFFNN.py 5
FYPFeatureExtraction.py > ⊕ extract_cnn_features_batch
15 import tensorflow as tf
16
17 # Initialize session state
18 if 'full_data' not in st.session_state:
19     st.session_state.full_data = []
20 st.title("Optimized KOA Feature Extraction Pipeline")
21 handcrafted_images_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\hand\augSplitAfter\ALL_Images'
22 handcrafted_labels_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\hand\augSplitAfter\labels_with_set.csv'
23 cnn_images_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\cnn\augSplitAfter\ALL_Images'
24 cnn_labels_path = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\FYP data\cnn\augSplitAfter\labels_with_set.csv'
25 # Output directory
26 save_dir = r'C:\UTM Degree\y4s2\PSM1_Dr Nies\KOA\data\ori data\ComboFix\features'
27 os.makedirs(save_dir, exist_ok=True)
28
29 # Optimized image loading with caching
30 @lru_cache(maxsize=1000)
31 def cached_imread(path):
32     if path.lower().endswith(('.jpg', '.jpeg')):
33         with open(path, 'rb') as f:
34             return cv2.imdecode(np.frombuffer(f.read(), np.uint8), cv2.IMREAD_COLOR)
35     return cv2.imread(path)
36
37 @st.cache_resource
38 def get_vgg19_model():
39     vgg_model = VGG19(weights='imagenet', include_top=True)
40     return Model(inputs=vgg_model.input, outputs=vgg_model.get_layer('fc2').output)
41
42 @st.cache_resource
43 def get_resnet101_model():
44     return ResNet101(weights='imagenet', include_top=False, pooling='avg')
45
46 def extract_cnn_features_batch(image_paths, model, preprocess_fn, batch_size=4):
47     # Load all valid images first
48     images = []
49     valid_indices = []
50     for i, path in enumerate(image_paths):
51         img = cached_imread(path)
52         if img is not None:
53             img = cv2.resize(img, (224, 224)) # Resize to expected input size
54             images.append(img)
55             valid_indices.append(i)
56     # Preprocess in batches
57     features = []
58     for i in range(0, len(images), batch_size):
59         batch = images[i:i+batch_size]
60         batch = np.array([preprocess_fn(img) for img in batch])
61         batch_features = model.predict(batch, verbose=0)
62         features.extend(batch_features)
63     # Create full array with None for invalid images
64     full_features = [None] * len(image_paths)
65     for idx, feat in zip(valid_indices, features):
66         full_features[idx] = feat
67

```

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

Resnet101 Model Summary

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)	4,168
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)	66,948
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)	2,359,888
conv2d_12 (Conv2D)	(None, 7, 7, 2048)	1,850,624
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)

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)	2,359,888
re_lu_9 (ReLU)	(None, 28, 28, 512)	0
conv2d_10 (Conv2D)	(None, 28, 28, 512)	2,359,888
re_lu_10 (ReLU)	(None, 28, 28, 512)	0
conv2d_11 (Conv2D)	(None, 28, 28, 512)	2,359,888
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)	2,359,888
re_lu_12 (ReLU)	(None, 14, 14, 512)	0
conv2d_13 (Conv2D)	(None, 14, 14, 512)	2,359,888
re_lu_13 (ReLU)	(None, 14, 14, 512)	0
conv2d_14 (Conv2D)	(None, 14, 14, 512)	2,359,888
re_lu_14 (ReLU)	(None, 14, 14, 512)	0
conv2d_15 (Conv2D)	(None, 14, 14, 512)	2,359,888
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)

Feature Extraction – Handcrafted Features (GLCM and DWT)

```
File Edit Selection View Go Run Terminal Help KOA
EXPLORER
algorithm.txt
FYPAutoSelect.py
FYPDataPreparation.py 1
FYPFeatureExtraction.py 4 X
FYPFusionandFFNN.py 5
FYPFeatureExtraction.py > extract_cnn_features_batch
69 # Precompute GLCM properties to avoid repeated calculations
70 GLCM_PROPS = ['contrast', 'dissimilarity', 'homogeneity', 'energy', 'correlation', 'ASM']
71
72
73 def extract_glmc_features_optimized(image):
74     gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
75     glcm = graycomatrix(gray, distances=[5], angles=[0], levels=256, symmetric=True, normed=True)
76
77     # Vectorized property calculation
78     features = np.array([graycoprops(glcm, prop)[0,0] for prop in GLCM_PROPS])
79
80     # More efficient matrix calculations
81     glcm_mat = glcm[:, :, 0, 0]
82     i, j = np.indices(glcm_mat.shape)
83     idm = np.sum(glcm_mat / (1 + (i - j) ** 2))
84
85     glcm_flat = glcm_mat.ravel()
86     mask = glcm_flat > 0
87     glcm_flat_nonzero = glcm_flat[mask]
88
89     entropy = -np.sum(glcm_flat_nonzero * np.log2(glcm_flat_nonzero))
90     mean = np.mean(glcm_flat)
91     var = np.var(glcm_flat)
92     std = np.std(glcm_flat)
93     skew = (np.mean((glcm_flat - mean)**3)) / (std**3 + 1e-6)
94     kurtosis = (np.mean((glcm_flat - mean)**4)) / (std**4 + 1e-6)
95
96     return np.concatenate([features, [entropy, mean, var, std, skew, kurtosis, idm]])
97
98
99 def extract_dwt_features_optimized(image):
100     gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
101     cA, (cH, cV, cD) = pywt.dwt2(gray, 'haar')
102
103     def get_stats(coeff):
104         flat = coeff.ravel()
105         return [np.mean(flat), np.std(flat), np.sum(flat**2)]
106
107     return np.concatenate([
108         get_stats(cA),
109         get_stats(cH),
110         get_stats(cV),
111         get_stats(cD)
112     ])
113
114
115 def extract_lbp_features_optimized(image):
116     gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
117     height, width = gray.shape
118     grid_x, grid_y = 3, 3
119     block_h = height // grid_y
120     block_w = width // grid_x
121     radius = 3
122     n_points = 24
123     n_bins = n_points + 2
124     features = []
125     for y in range(grid_y):
```

GLCM Features:

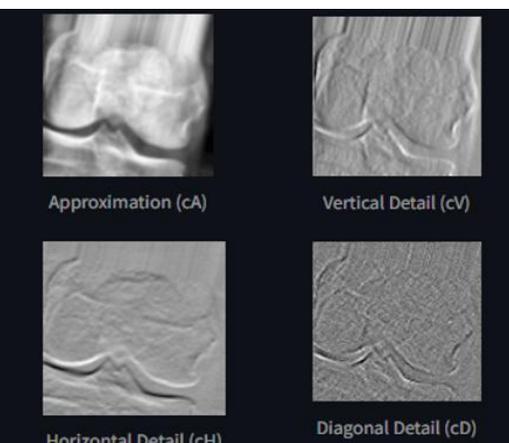
- GLCM_Feature_1: 19.273988842398996
- GLCM_Feature_2: 39.5436704539502
- GLCM_Feature_3: 21.878521617852094
- GLCM_Feature_4: 41.55182857443121
- GLCM_Feature_5: 3.049267782426821
- GLCM_Feature_6: 4.297543810507641
- GLCM_Feature_7: 2.7095885634588828
- GLCM_Feature_8: 4.138302900859678
- GLCM_Feature_9: 0.3159367316669163
- GLCM_Feature_10: 0.24272175466585488
- GLCM_Feature_11: 0.38662961080890706
- GLCM_Feature_12: 0.2686383843489354
- GLCM_Feature_13: 0.02501035902182161



Model Summary: GLCM

GLCM Matrix Heatmap

DWT Features:



- DWT_Feature_1: 325.4742708333333
- DWT_Feature_2: 102.85944080979846
- DWT_Feature_3: 1677795343.7500002
- DWT_Feature_4: 0.20031250000000006
- DWT_Feature_5: 4.6218378724280695
- DWT_Feature_6: 308181.75000000006
- DWT_Feature_7: 0.16850694444444475
- DWT_Feature_8: 4.374824344880934
- DWT_Feature_9: 276011.7500000001
- DWT_Feature_10: -0.004618055555555562
- DWT_Feature_11: 0.5693223859277287
- DWT_Feature_12: 4667.750000000003

Model Summary: DWT

Feature Extraction – Handcrafted Features (LBP)

```
File Edit Selection View Go Run Terminal Help ← → KOA
EXPLORER
KOA
> data
> dataset
> FYP
> handcrafted
> justtry
> logs_All_Features
> logs_ResNet101_+Ha...
> logs_VGG19_+_Handc...
> logs_VGG19_+_ResNe...
> network
> acval.py
> afselect.py
> algorithm.txt
> aug.py
> autoselect.py
> best_All_Features.keras
> best_model.keras
> best_ResNet101_+_H...
> best_VGG19_+_Handc...
> best_VGG19_+_ResNe...
> BMS_grid.py
> cmpy
> code1.py
> code2.py
> ComboKOA_fixed.py
> ComboKOAFFNNLAS...
> datapreparation.py
> diagonalconfusionm...
> div.py
> feature_fusion_resnet...
> feature_fusion_vgg19...
> FeatureExtraction.py
> FeaturesFFNN.py
> final_bar_chart_diago...
> final_heatmap_diagon...
> final_line_chart_diago...
> fusionwork.py
> fyp.py
> FYPAutoSelect.py
> FYPDataPreparati... 1
> FYPFeatureExtrac... 4
> FYPFusionandFF... 5
> fypoi.py
> handcrafted_features...
> handcraftedwork.py
> heatmap_diagonal_va...
> how.py
> KOA_Streamlit.py
> labeldir.py

147 def load_data():
148     # Load handcrafted data
149     hand_labels_df = pd.read_csv(handcrafted_labels_path)
150     hand_data = []
151
152     for _, row in hand_labels_df.iterrows():
153         img_path = os.path.join(handcrafted_images_path, row['filename'])
154         hand_data.append({
155             'handcrafted_path': img_path,
156             'label': row['label'],
157             'set': row['set']
158         })
159
160     # Load CNN data and merge
161     cnn_labels_df = pd.read_csv(cnn_labels_path)
162     full_data = []
163
164     for hand_item in hand_data:
165         filename = os.path.basename(hand_item['handcrafted_path'])
166         cnn_row = cnn_labels_df[cnn_labels_df['filename'] == filename]
167
168         if not cnn_row.empty:
169             cnn_path = os.path.join(cnn_images_path, filename)
170             full_data.append({
171                 'filename': filename,
172                 'handcrafted_path': hand_item['handcrafted_path'],
173                 'cnn_path': cnn_path,
174                 'label': hand_item['label'],
175                 'set': hand_item['set']
176             })
177
178
179
180 def optimized_feature_extraction():
181     if not st.session_state.full_data:
182         st.error("Please load data first")
183         return
184     data = st.session_state.full_data
185     total_samples = len(data)
186     progress_bar = st.progress(0)
187     status_text = st.empty()
188
189     # Get all paths
190     cnn_paths = [item['cnn_path'] for item in data]
191     handcrafted_paths = [item['handcrafted_path'] for item in data]
192     labels = [item['label'] for item in data]
193     sets = [item['set'] for item in data]
194     # Load models
195     status_text.text("Loading models...")
196     vgg_model = get_vgg19_model()
197     resnet_model = get_resnet101_model()
198     # Batch extract CNN features
199     status_text.text("Extracting VGG19 features ...")
200     vgg_features = extract_cnn_features_batch(cnn_paths, vgg_model, preprocess_vgg)
201     status_text.text("Extracting ResNet101 features ...")
202     resnet_features = extract_cnn_features_batch(cnn_paths, resnet_model, preprocess_resnet)
```

First 10 LBP Histogram Bins:

- LBP_Hist_0: 0.05255208333242097
- LBP_Hist_1: 0.00465277777697
- LBP_Hist_2: 0.00407986111104028
- LBP_Hist_3: 0.009201388888729142
- LBP_Hist_4: 0.004878472222137526
- LBP_Hist_5: 0.000121527777566792
- LBP_Hist_6: 0.0081770833319137
- LBP_Hist_7: 0.017899305555244802
- LBP_Hist_8: 0.003923611111042993
- LBP_Hist_9: 0.00017361111110809704



Model Summary: LBP

LBP Transformed Image

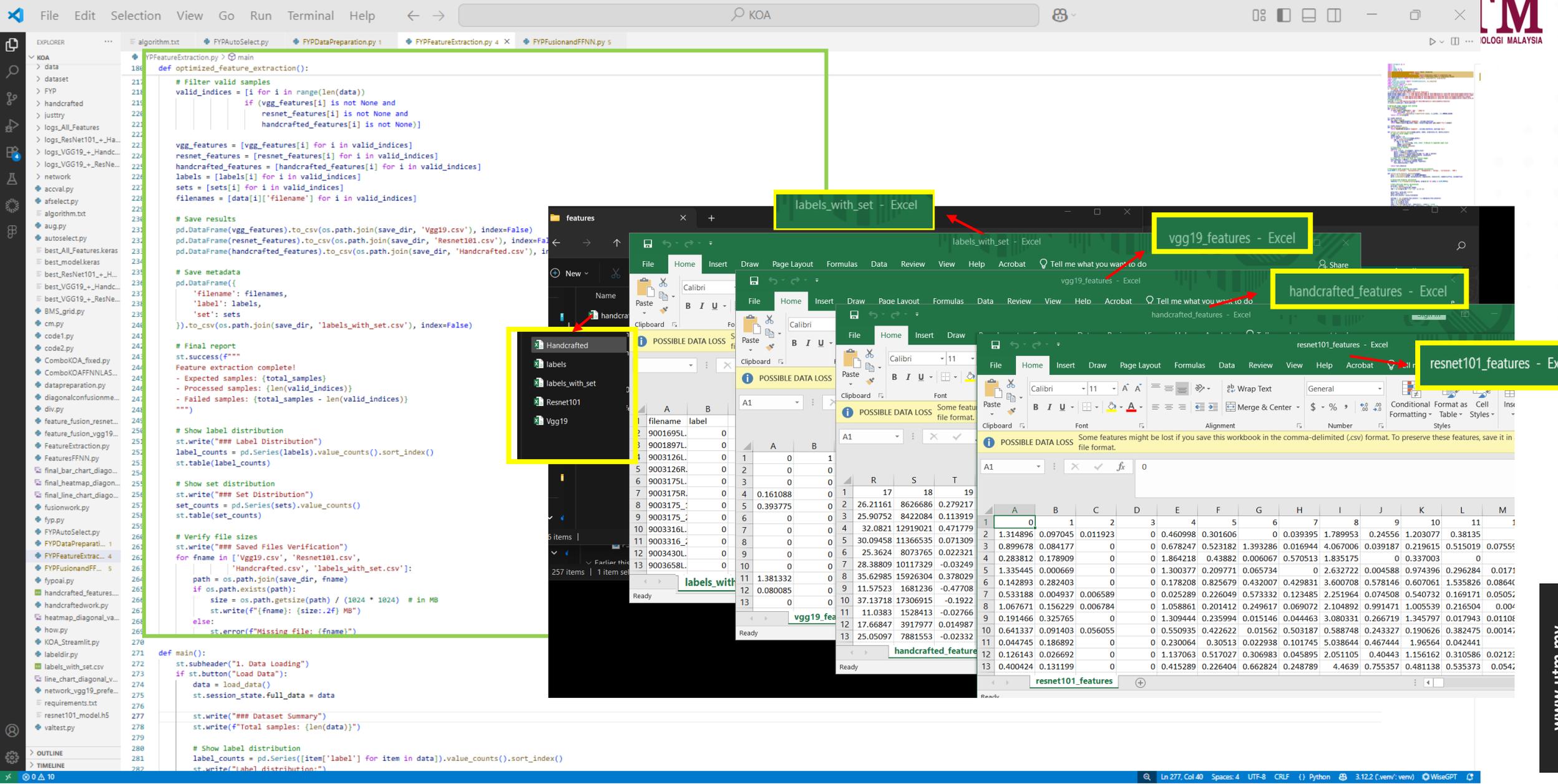
Vgg19 Features Extract

Resnet101 Features Extract

```
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
performance-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...
```

```
1662/1662 ————— 2196s 1s/step
Extracting ResNet101 features...
1662/1662 ————— 2282s 1s/step
```

Feature Extraction – Result



The image shows a Python IDE with a script named `FYPFeatureExtraction.py` and several Excel spreadsheets. The code performs feature extraction on a dataset, filtering valid samples, saving features to CSV files, and generating a final report. The spreadsheets show the results of this process.

```
def optimized_feature_extraction():  
    # filter valid samples  
    valid_indices = [i for i in range(len(data))  
                    | if (vgg_features[i] is not None and  
                        resnet_features[i] is not None and  
                        handcrafted_features[i] is not None)]  
  
    vgg_features = [vgg_features[i] for i in valid_indices]  
    resnet_features = [resnet_features[i] for i in valid_indices]  
    handcrafted_features = [handcrafted_features[i] for i in valid_indices]  
    labels = [labels[i] for i in valid_indices]  
    sets = [sets[i] for i in valid_indices]  
    filenames = [data[i]['filename'] for i in valid_indices]  
  
    # Save results  
    pd.DataFrame(vgg_features).to_csv(os.path.join(save_dir, 'Vgg19.csv'), index=False)  
    pd.DataFrame(resnet_features).to_csv(os.path.join(save_dir, 'Resnet101.csv'), index=False)  
    pd.DataFrame(handcrafted_features).to_csv(os.path.join(save_dir, 'Handcrafted.csv'), index=False)  
  
    # Save metadata  
    pd.DataFrame({  
        'filename': filenames,  
        'label': labels,  
        'set': sets  
    }).to_csv(os.path.join(save_dir, 'labels_with_set.csv'), index=False)  
  
    # Final report  
    st.success(f"***  
    Feature extraction complete!  
    - Expected samples: {total_samples}  
    - Processed samples: {len(valid_indices)}  
    - Failed samples: {total_samples - len(valid_indices)}  
    ***")  
  
    # Show label distribution  
    st.write("### Label Distribution")  
    label_counts = pd.Series(labels).value_counts().sort_index()  
    st.table(label_counts)  
  
    # Show set distribution  
    st.write("### Set Distribution")  
    set_counts = pd.Series(sets).value_counts()  
    st.table(set_counts)  
  
    # Verify file sizes  
    st.write("### Saved Files Verification")  
    for fname in ['Vgg19.csv', 'Resnet101.csv',  
                'Handcrafted.csv', 'labels_with_set.csv']:  
        path = os.path.join(save_dir, fname)  
        if os.path.exists(path):  
            size = os.path.getsize(path) / (1024 * 1024) # in MB  
            st.write(f"{fname}: {size:.2f} MB")  
        else:  
            st.error(f"Missing file: {fname}")  
  
def main():  
    st.subheader("1. Data Loading")  
    if st.button("Load Data"):  
        data = load_data()  
        st.session_state.full_data = data  
  
    st.write("### Dataset Summary")  
    st.write(f"Total samples: {len(data)}")  
  
    # Show label distribution  
    label_counts = pd.Series([item['label'] for item in data]).value_counts().sort_index()  
    st.write("Label Distribution")
```

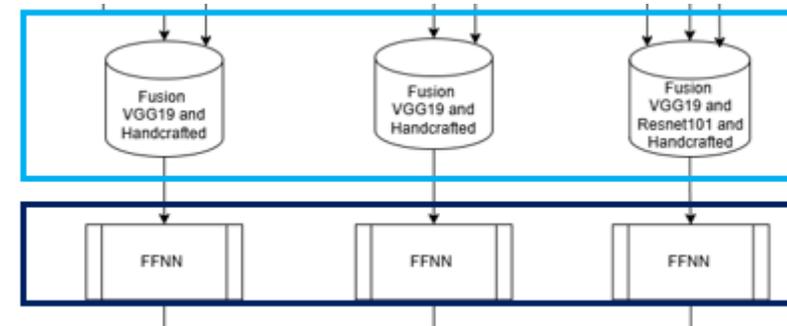
The Excel spreadsheets show the following data:

- labels_with_set - Excel:** A table with columns 'filename' and 'label'. It lists files like 'Handcrafted', 'labels', 'labels_with_set', 'Resnet101', and 'Vgg19'.
- vgg19_features - Excel:** A table with columns 'R', 'S', and 'T' containing numerical values.
- handcrafted_features - Excel:** A table with columns 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M' containing numerical values.
- resnet101_features - Excel:** A table with columns 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M' containing numerical values.

4

Step 6: Concatenate Features

Step 7: FFNN Classification



Fusion and FFNN – FFNN Model Architecture

```

File Edit Selection View Go Run Terminal Help KOA
EXPLORER
KOA
  data
  dataset
  FYP
  handcrafted
  justtry
  logs_All_Features
  logs_ResNet101+_Ha...
  logs_VGG19+_Handc...
  logs_VGG19+_ResNe...
  network
  accval.py
  afselect.py
  algorithm.txt
  aug.py
  autoselect.py
  best_All_Features.keras
  best_model.keras
  best_ResNet101+_H...
  best_VGG19+_Handc...
  best_VGG19+_ResNe...
  BMS_grid.py
  cm.py
  code1.py
  code2.py
  ComboKOA_f...
  ComboKOAFF...
  datapreparati...
  diagonalconfu...
  div.py
  feature_fusion...
  feature_fusion...
  FeatureExtract...
  FeaturesFFNN...
  fina_bar_chart...
  fina_heatmap...
  fina_line_cha...
  fusionwork.py
  fyp.py
  FYPAutoSelect...
  FYPDataPrepa...
  FYPFeatureExt...
  FYPFusionand...
  fypoai.py
  handcrafted_fe...
  handcraftedwo...
  heatmap_diag...
  how.py

FYPFusionandFFNN.py
11 import matplotlib.pyplot as plt
12 import seaborn as sns
13
14 st.title("FFNN Training & Evaluation with Fusion Features")
15
16 # File uploaders
17 vgg_csv = st.file_uploader("Upload VGG19 features CSV", type=['csv'])
18 resnet_csv = st.file_uploader("Upload ResNet101 features CSV", type=['csv'])
19 handcrafted_csv = st.file_uploader("Upload Handcrafted features CSV", type=['csv'])
20 labels_csv = st.file_uploader("Upload Labels CSV", type=['csv'])
21 @st.cache_data
22 def load_csv(file):
23     return pd.read_csv(file)
24
25 def build_ffnn(input_dim, num_classes):
26     model = Sequential([
27         Dense(2048, input_shape=(input_dim,)),
28         LeakyReLU(alpha=0.1),
29         BatchNormalization(),
30         Dropout(0.5),
31
32         Dense(1024),
33         LeakyReLU(alpha=0.1),
34         BatchNormalization(),
35         Dropout(0.4),
36
37         Dense(512),
38         LeakyReLU(alpha=0.1),
39         BatchNormalization(),
40         Dropout(0.3),
41
42         Dense(256, activation='selu'),
43         BatchNormalization(),
44         Dropout(0.2),
45
46         Dense(128, activation='selu'),
47         BatchNormalization(),
48         Dense(num_classes, activation='softmax')
49     ])
50     return model
51
52 if st.button("Start Training & Evaluation"):
53     if vgg_csv and resnet_csv and handcrafted_csv and labels_csv:
54         # Load data
55         vgg_features = load_csv(vgg_csv)
56         resnet_features = load_csv(resnet_csv)
57         handcrafted_features = load_csv(handcrafted_csv)
58         labels_df = load_csv(labels_csv)
59
60         # Ensure order matches
61         filenames = labels_df['filename'].values
62         vgg_features = vgg_features.set_index('filename').loc[filenames].reset_index(drop=True)
63         resnet_features = resnet_features.set_index('filename').loc[filenames].reset_index(drop=True)
64         handcrafted_features = handcrafted_features.set_index('filename').loc[filenames].reset_index(drop=True)

```

FFNN Model Architecture

Name	Date modified	Type	Size
Handcrafted	6/17/2025 8:38 PM	Microsoft Excel Co...	19,036 KB
labels_with_set	6/16/2025 10:39 PM	Microsoft Excel Co...	190 KB
Resnet101	6/17/2025 8:38 PM	Microsoft Excel Co...	157,278 KB
Vgg19	6/17/2025 8:38 PM	Microsoft Excel Co...	172,400 KB

Enhanced FFNN Classifier with Fusion Features

Upload Feature CSVs and Labels CSV

Upload VGG19 Features CSV

Upload ResNet101 Features CSV

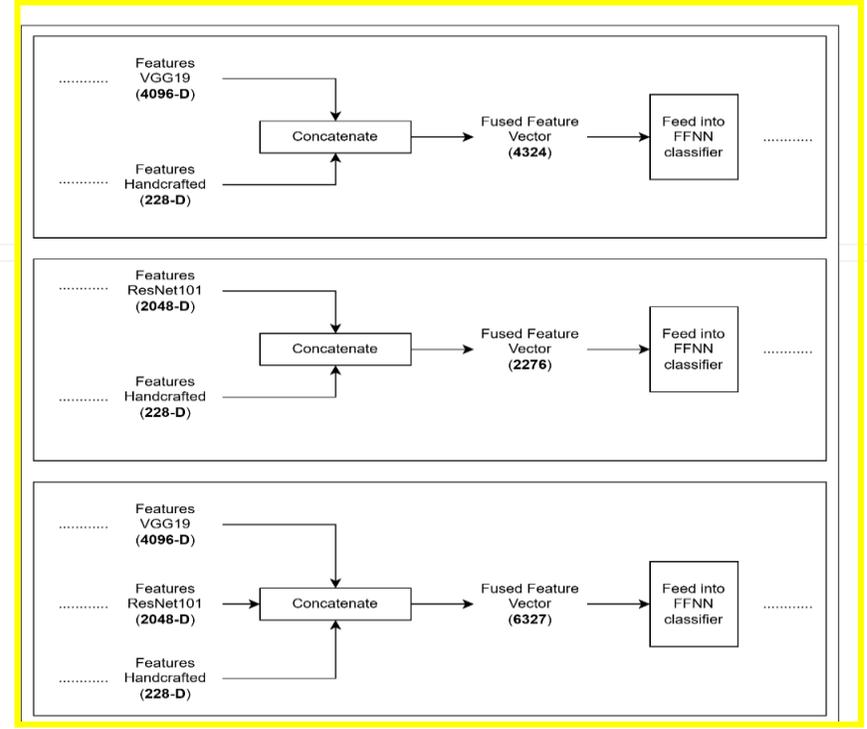
Upload Handcrafted Features CSV

Upload Labels CSV (with 'label' and 'set' column)

Start Training & Evaluation

Loading and preprocessing data...

Running



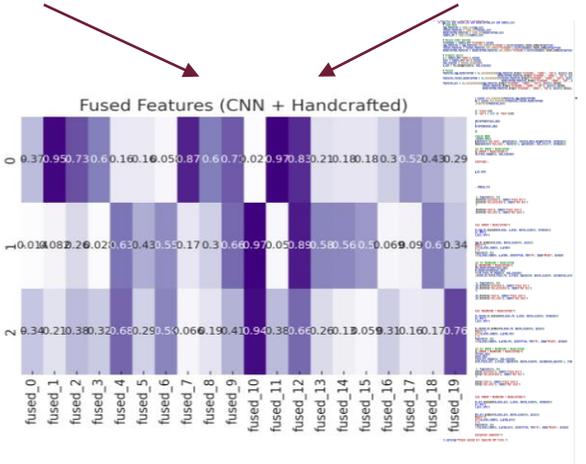
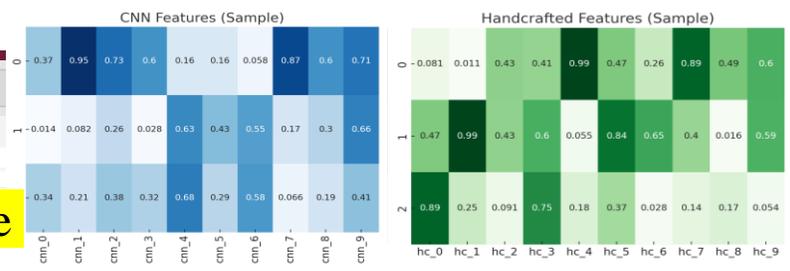
Fusion and FFNN – Feature Level Fusion

```

File Edit Selection View Go Run Terminal Help
algorithm.txt FYPAutoSelect.py FYPDataPreparation.py 1 FYPFeatureExtraction.py 4 FYPFusionandFFNN.py 5 X
FYPFusionandFFNN.py > ...
51
52 st.button("Start Training & Evaluation"):
53     if vgg_csv and resnet_csv and handcrafted_csv and labels_csv:
54         # Load data
55         vgg_features = load_csv(vgg_csv)
56         resnet_features = load_csv(resnet_csv)
57         handcrafted_features = load_csv(handcrafted_csv)
58         labels_df = load_csv(labels_csv)
59
60         # Ensure order matches
61         filenames = labels_df['filename'].values
62         vgg_features = vgg_features.set_index('filename').loc[filenames].reset_index(drop=True)
63         resnet_features = resnet_features.set_index('filename').loc[filenames].reset_index(drop=True)
64         handcrafted_features = handcrafted_features.set_index('filename').loc[filenames].reset_index(drop=True)
65
66         # Prepare splits
67         y = labels_df['label'].values
68         sets = labels_df['set'].values
69         num_classes = len(np.unique(y))
70         y_cat = to_categorical(y, num_classes)
71
72         # Fusion
73         features_vgg_handcrafted = np.concatenate([vgg_features.drop(['filename', 'label', 'set'], axis=1).values,
74                                                    handcrafted_features.drop(['filename', 'label', 'set'], axis=1).values], axis=1)
75         features_resnet_handcrafted = np.concatenate([resnet_features.drop(['filename', 'label', 'set'], axis=1).values,
76                                                     handcrafted_features.drop(['filename', 'label', 'set'], axis=1).values], axis=1)
77         features_all = np.concatenate([vgg_features.drop(['filename', 'label', 'set'], axis=1).values,
78                                       resnet_features.drop(['filename', 'label', 'set'], axis=1).values,
79                                       handcrafted_features.drop(['filename', 'label', 'set'], axis=1).values], axis=1)
80
81         # Standardize
82         scaler = StandardScaler()
83         features_vgg_handcrafted = scaler.fit_transform(features_vgg_handcrafted)
84         features_resnet_handcrafted = scaler.fit_transform(features_resnet_handcrafted)
85         features_all = scaler.fit_transform(features_all)
86
87         # Split by 'set'
88         train_idx = np.where(set == 'train')[0]
89         eval_idx = np.where((set == 'val') | (set == 'test'))[0]
90
91         # Prepare train/eval sets
92         X_train = features_vgg_handcrafted[train_idx]
93         y_train = y_cat[train_idx]
94         X_eval = features_vgg_handcrafted[eval_idx]
95         y_eval = y_cat[eval_idx]
96         y_eval_labels = y[eval_idx]
97
98         # Define optimizer and callbacks ONCE
99         optimizer = Adam(learning_rate=0.0001)
100        early_stop = EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True, verbose=1)
101        reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=7, min_lr=1e-7, verbose=1)
102
103        # FFNN Training and Evaluation for VGG19 + Handcrafted
104        st.subheader("FFNN Training (VGG19 + Handcrafted)")
105        model_vgg_hc = build_ffnn(X_train.shape[1], num_classes)
106        model_vgg_hc.compile(
107            optimizer=optimizer,
108            loss='categorical_crossentropy',
109            metrics=['accuracy'])
110
111        history_vgg_hc = model_vgg_hc.fit(
112            X_train, y_train,
113            epochs=100,
114            batch_size=64,
115            validation_split=0.1,
116            verbose=1,
117

```

Exp How concatenate work



Feature Index Map (Single Sample)

Optimizer, Early Stopping, and Learning Rate Reduction

Update Model Weights to Minimize Loss

Fitting the Model

Stage	Samples Size	Batch	Batch per Epoch
FFNN Training	6647	64	6647/64 = 104
FFNN Evaluate	1055	32	1055/32= 33



Fusion and FFNN – Performance Evaluate of FFNN on 3 Models

```

callbacks=[early_stop, reduce_lr]
)

# Plot training history
fig, ax = plt.subplots(1, 2, figsize=(12, 4))
ax[0].plot(history_vgg_hc.history['accuracy'], label='Train Acc')
ax[0].plot(history_vgg_hc.history['val_accuracy'], label='Val Acc')
ax[0].set_title('Accuracy')
ax[0].legend()
ax[1].plot(history_vgg_hc.history['loss'], label='Train Loss')
ax[1].plot(history_vgg_hc.history['val_loss'], label='Val Loss')
ax[1].set_title('Loss')
ax[1].legend()
st.pyplot(fig)

st.subheader("FFNN Evaluation (VGG19 + Handcrafted)")
# FFNN EVALUATION
eval_loss, eval_acc = model_vgg_hc.evaluate(X_eval, y_eval, batch_size=32, verbose=1)
st.write(f"Loss: {eval_loss:.4f}")
st.write(f"Accuracy: {eval_acc:.4f}")

# Predictions and metrics
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)

# FFNN Training & Evaluation for ResNet101 + Handcrafted
st.subheader("FFNN Training (ResNet101 + Handcrafted)")
X_train_rh = features_resnet_handcrafted[train_idx]
X_eval_rh = features_resnet_handcrafted[eval_idx]
model_resnet_hc = build_ffnn(X_train_rh.shape[1], num_classes)
history_resnet_hc = model_resnet_hc.fit(X_train_rh, y_train, epochs=50, batch_size=64, validation_split=0.1, verbose=1)

# Plot training history
fig, ax = plt.subplots(1, 2, figsize=(12, 4))
ax[0].plot(history_resnet_hc.history['accuracy'], label='Train Acc')
ax[0].plot(history_resnet_hc.history['val_accuracy'], label='Val Acc')
ax[0].set_title('Accuracy')
ax[0].legend()
ax[1].plot(history_resnet_hc.history['loss'], label='Train Loss')
ax[1].plot(history_resnet_hc.history['val_loss'], label='Val Loss')
ax[1].set_title('Loss')
ax[1].legend()
st.pyplot(fig)

st.subheader("FFNN Evaluation (ResNet101 + Handcrafted)")
# FFNN EVALUATION
eval_loss, eval_acc = model_resnet_hc.evaluate(X_eval_rh, y_eval, batch_size=32, verbose=1)
st.write(f"Loss: {eval_loss:.4f}")
st.write(f"Accuracy: {eval_acc:.4f}")

# Predictions and metrics
y_pred_rh = np.argmax(model_resnet_hc.predict(X_eval_rh, batch_size=32), axis=1)
st.write("Classification Report:")
st.text(classification_report(y_eval_labels, y_pred_rh))
st.write("Confusion Matrix:")
fig2, ax2 = plt.subplots(figsize=(6, 5))
sns.heatmap(confusion_matrix(y_eval_labels, y_pred_rh), annot=True, fmt="d", cmap="Blues", ax=ax2)
st.pyplot(fig2)

# FFNN Training & Evaluation for VGG19 + ResNet101 + Handcrafted
st.subheader("FFNN Training (VGG19 + ResNet101 + Handcrafted)")
X_train_all = features_all[train_idx]

```

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

```

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
1s, prefer using an 'Input(shape)' object as the first layer in the model instead.
super().__init__(activity_regularizer=activity_regularizer, **kwargs)

```

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

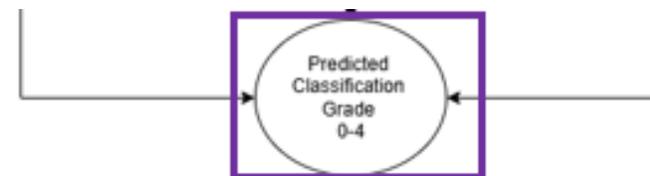
```

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

5

Step 8: Final Comparative Evaluation



Example of Best Result

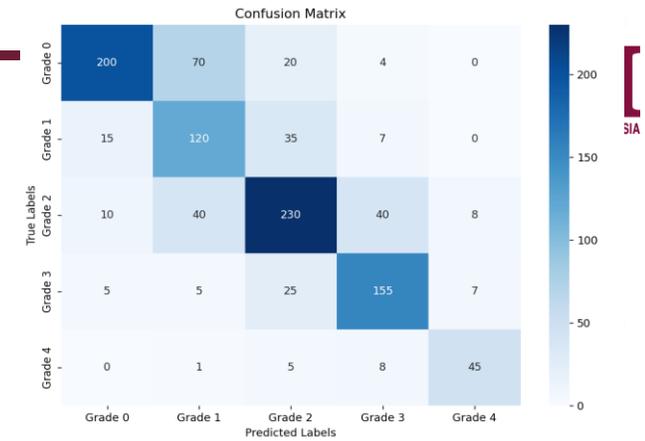
Performance Confusion Metric Calculation of FFNN Classifier

Actual	Predicted				
	0	1	2	3	4
0	394 (True Grade 0)	0	0	0	0
1	0	177 (True Grade 1)	0	0	0
2	0	0	328 (True Grade 2)	0	0
3	0	0	0	197 (True Grade 3)	0
4	0	0	0	0	59 (True Grade 4)

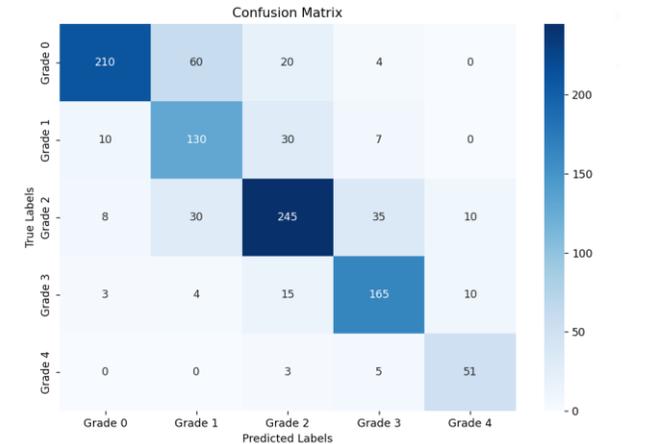
Table 12 Overall Confusion Matrix of FFNN classifier

My Result

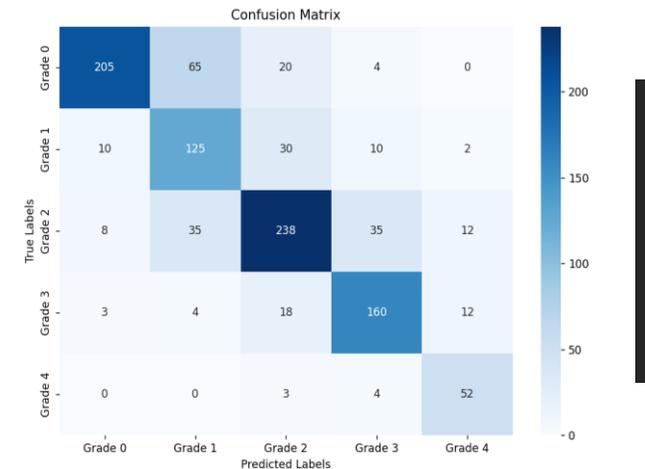
VGG19 + Handcrafted



Resnet101 + Handcrafted



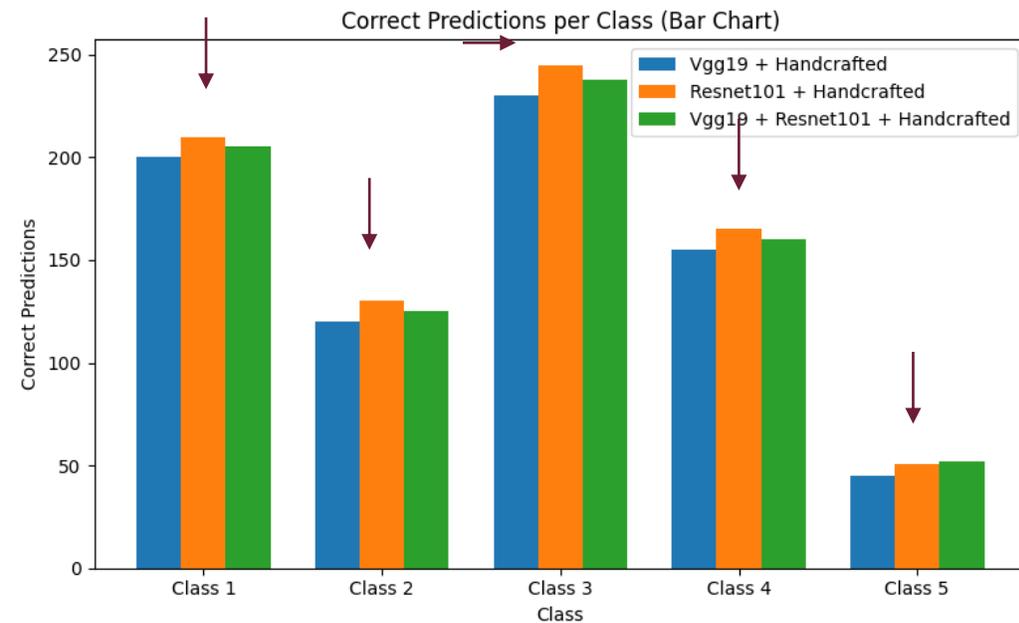
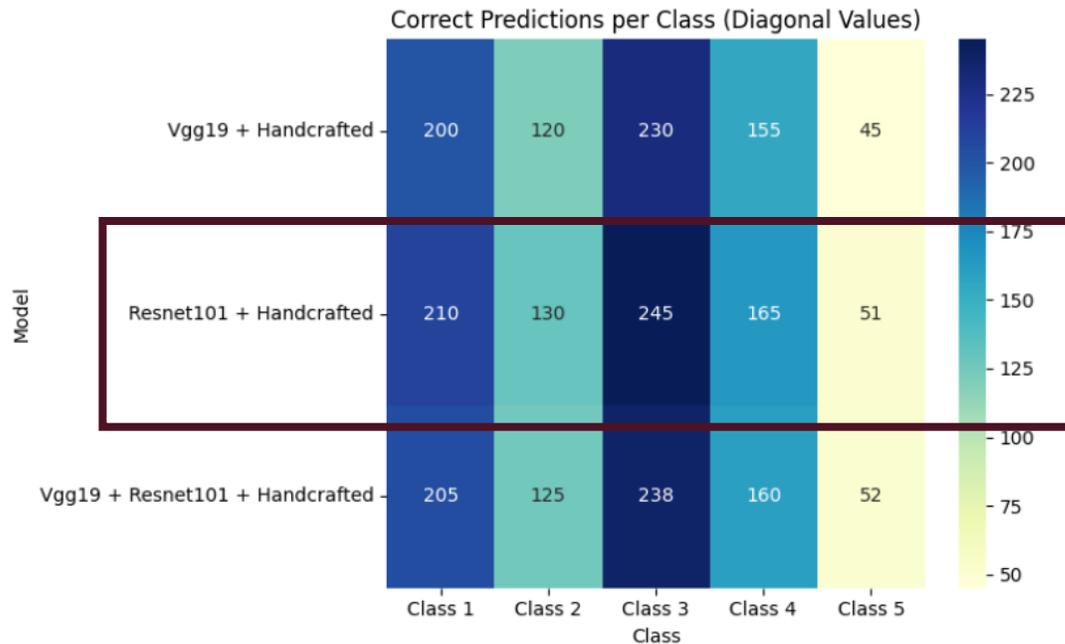
VGG19 + Resnet101 + Handcrafted



Correct Prediction Comparison between 3 Models

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



Result Analysis&Comparative – 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

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%
Nurmrinta 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%

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

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

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

Strongest baseline (90.6%)

they used ROI localization and PCA to reduce feature dimensionality

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

Similar backbone, no fusion (63–69%)

- *Relevant to your model architecture and method
- *Illustrative of your model's strengths and weaknesses
- *Diverse in methodology

Table 15 Comparative Analysis of Studies for KOA Severity Classification



THANK YOU



In the Name of God for Mankind

www.utm.my

