

ДОДАТОК А

Основний код підготовки даних та машинного навчання

Лістинг А.1 – Програмний код побудови гістограми розподілу класів датасету.

```
import os
import matplotlib.pyplot as plt
import seaborn as sns

DATASET_PATH = 'dataset/garbage_classification'

classes = []
counts = []

for class_name in sorted(os.listdir(DATASET_PATH)):
    class_path = os.path.join(DATASET_PATH, class_name)
    if os.path.isdir(class_path):
        classes.append(class_name)
        num_files = len([f for f in os.listdir(class_path)
if os.path.isfile(os.path.join(class_path, f))])
        counts.append(num_files)

sns.set_theme(style="whitegrid")
plt.figure(figsize=(14, 8))

bars = sns.barplot(x=counts, y=classes, palette="viridis")

for i, v in enumerate(counts):
    bars.text(v + 50, i, str(v), color='black',
va='center', fontweight='bold')

plt.title('Розподіл кількості зображень за класами
відходів', fontsize=16, fontweight='bold', pad=20)
```

```
plt.xlabel('Кількість зображень', fontsize=14)
plt.ylabel('Клас відходів', fontsize=14)

plt.tight_layout()
plt.savefig('results/figure_3_1_histogram_dynamic.png',
            dpi=300)
plt.show()
```

Лістинг А.2 – Програмний код розширення даних (Data Augmentation) та генерації вибірок

```
# Використовуємо ImageDataGenerator для аугментації
datagen = ImageDataGenerator(
    rescale=1./255,          # Нормалізація пікселів (0-1)
    rotation_range=30,       # Випадкові повороти (до 30
    градусів)
    width_shift_range=0.2,    # Випадкове зміщення
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,          # Випадкове наближення (для
    зім'ятого сміття)
    horizontal_flip=True,
    validation_split=0.2     # 20% на валідацію
)

# Тренувальний генератор
train_generator = datagen.flow_from_directory(
    DATASET_PATH,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)

# Валідаційний генератор
```

```

val_generator = datagen.flow_from_directory(
    DATASET_PATH,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)

```

Лістинг А.3 – Програмний код ініціалізації та конфігурації моделі MobileNetV2.

```

# Беремо базову модель без останнього шару
(include_top=False)
base_model = MobileNetV2(weights='imagenet',
include_top=False, input_shape=(224, 224, 3))

# Заморожуємо базові шари, щоб не зіпсувати попередньо
навчені ваги
base_model.trainable = False

# Додаємо наші власні шари класифікації поверх бази
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x) # Dropout для уникнення перенавчання
predictions = Dense(len(class_names),
activation='softmax')(x)

model = Model(inputs=base_model.input,
outputs=predictions)

model.compile(optimizer=Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

```

Лістинг А.4 – Програмний код процесу навчання та збереження моделі.

```
print("Починаємо навчання...")
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=EPOCHS
)
model.save(f"{RESULTS_DIR}/model_{EXPERIMENT_NAME}.h5")
```

Лістинг А.5 – Програмний код побудови матриці плутанини та текстового звіту.

```
val_generator.reset()
Y_pred = model.predict(val_generator)
y_pred_classes = np.argmax(Y_pred, axis=1)
y_true = val_generator.classes

# Текстовий звіт (Precision, Recall, F1)
report = classification_report(y_true, y_pred_classes,
    target_names=CLASS_NAMES)
with open(f"{RESULTS_DIR}/report_{EXPERIMENT_NAME}.txt",
    "w") as f:
    f.write(report)
print(f"Текстовий звіт збережено як
    report_{EXPERIMENT_NAME}.txt")

# Матриця плутанини
cm = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
    xticklabels=CLASS_NAMES, yticklabels=CLASS_NAMES)
```

```
plt.title(f'Матриця плутанини ({EXPERIMENT_NAME})',
          fontsize=16)
plt.ylabel('Реальний клас', fontsize=12)
plt.xlabel('Передбачений клас', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig(f"{RESULTS_DIR}/confusion_matrix_{EXPERIMENT_NAME}.png", dpi=300)
print(f"Матрицю плутанини збережено як
confusion_matrix_{EXPERIMENT_NAME}.png")
```

ДОДАТОК Б

Основний код клієнтського мобільного застосунку

Лістинг Б.1 – Програмний код побудови гістограми розподілу класів датасету.

```
Future<void> _uploadAndPredict(File image) async {
  setState(() => _isLoading = true);
  try {
    var request = http.MultipartRequest('POST',
Uri.parse('$baseUrl/predict'));
    request.files.add(await
http.MultipartFile.fromPath('file', image.path));
    request.fields['user_id'] = widget.userId;
    request.fields['lang'] = widget.lang;

    var streamedResponse = await request.send();
    var response = await
http.Response.fromStream(streamedResponse);

    if (response.statusCode == 200) {
      setState(() => _result =
json.decode(utf8.decode(response.bodyBytes)));
    }
  } catch (e) {

ScaffoldMessenger.of(context).showSnackBar(SnackBar(conten
t: Text('Error: $e')));
  } finally {
    setState(() => _isLoading = false);
  }
}
```

Лістинг Б.2 – Програмний код побудови гістограми розподілу класів датасету.

```
Widget _buildResultCard(Map<String, String> langDict) {
  bool isTrash = _result!['is_trash'];
  return Container(
    padding: const EdgeInsets.all(16),
    decoration: BoxDecoration(color: Colors.white,
borderRadius: BorderRadius.circular(20), boxShadow:
[BoxShadow(color: Colors.black.withOpacity(0.1),
blurRadius: 10)]),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      mainAxisAlignment: MainAxisAlignment.min,
      children: [
        Row(
          children: [
            Icon(isTrash ? Icons.recycling :
Icons.warning_amber_rounded, color: isTrash ? const
Color(0xFF1CB06D) : Colors.red, size: 28),
            const SizedBox(width: 10),
            Expanded(child: Text(_result!['item_name'],
style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold, color: isTrash ? const Color(0xFF1B5E20)
: Colors.red.shade800))),
          ],
        ),
        const SizedBox(height: 8),
        Text('${langDict['ai_confidence']}
${(_result!['confidence'] * 100).toStringAsFixed(1)}%',
style: const TextStyle(fontSize: 14, color: Colors.grey)),
        const Divider(height: 24),
        Text(isTrash ? langDict['how_to_sort']! :
langDict['attention']!, style: const TextStyle(fontSize:
16, fontWeight: FontWeight.bold)),
      ],
    ),
  );
}
```

```

        const SizedBox(height: 6),
        Text(isTrash ? _result!['instruction'] :
_result!['message'], style: const TextStyle(fontSize: 14,
height: 1.4)),
      ],
    ),
  );
}

```

Лістинг Б.3 – Програмний код побудови гістограми розподілу класів датасету.

```

Future<void> _fetchStats() async {
  setState(() => _isLoading = true);
  try {
    final response = await
http.get(Uri.parse('$baseUrl/stats/${widget.userId}?period
=$_selectedPeriod'));
    if (response.statusCode == 200) {
      setState(() => _stats =
json.decode(utf8.decode(response.bodyBytes)));
    }
  } catch (e) {
    debugPrint('Error: $e');
  } finally {
    setState(() => _isLoading = false);
  }
}

```

Лістинг Б.4 – Програмний код побудови гістограми розподілу класів датасету.

```

Future<void> _submitForm(Map<String, String> langDict)
async {
  if (_selectedCategory == null) {

ScaffoldMessenger.of(context).showSnackBar(SnackBar(conten
t: Text(langDict['select_category']!)));
    return;
  }

  setState(() => _isSubmitting = true);
  String errorDateTime = "${_selectedDate.year}-
${_selectedDate.month}-${_selectedDate.day}
${_selectedTime.hour}:${_selectedTime.minute}";

  try {
    final response = await http.post(
      Uri.parse('$baseUrl/feedback'),
      headers: {'Content-Type': 'application/json'},
      body: json.encode({
        'user_id': widget.userId,
        'actual_category': _selectedCategory,
        'comment': _commentController.text,
        'clarity_rating': _confidenceRating,
        'error_datetime': errorDateTime,
      })),
    );

    if (response.statusCode == 200) {

ScaffoldMessenger.of(context).showSnackBar(SnackBar(conten
t: Text(langDict['thanks_report']!)));
      _commentController.clear();
    }
  }
}

```

```
        setState(() {
          _selectedCategory = null;
          _confidenceRating = 5.0;
        });
      }
    } catch (e) {

      ScaffoldMessenger.of(context).showSnackBar(SnackBar(conten
t: Text('Error: $e')));
      } finally {
        setState(() => _isSubmitting = false);
      }
    }
  }
```

ДОДАТОК В

Відомість кваліфікаційної роботи

Позначення					Найменування		Дод. відомості	
					Текстові документи			
1.					Пояснювальна записка		100 с.	