Report on

Project 4: Image classification

Name: Maliha Jannat (马里哈)

Student ID: W2110805009

Introduction:

- **Project Overview:**

Image classification is a fundamental task in computer vision and involves assigning a label to an image from a predefined set of categories. In this project, we will perform image classification on the Corel5k dataset, which consists of 50 classes, each containing 100 images. We will implement two classification methods: **Convolutional Neural Networks (CNNs)** and a traditional method like **Support Vector Machines (SVM)**.

- **Importance of Image Classification:**

Image classification is essential in various fields, including computer vision, medical imaging, and automated tagging of images. Accurate classification can significantly enhance the efficiency of image retrieval systems.

Dataset Description:

- **Corel5k Dataset:**

Total Images: 5,000
Classes: 50
Images per Class: 100

- **Data Preprocessing:**

Image Resizing: Resize images to a uniform size (e.g., 224x224 pixels).
Normalization: Normalize pixel values to a range of [0, 1].
Data Augmentation: Apply techniques such as rotation, flipping, and zooming to increase dataset diversity.

Classification Methods:

- <u>**Method 1: Convolutional Neural Networks (CNN)**</u>

CNNs are a class of deep learning models specifically designed for processing structured grid data, such as images. They consist of convolutional layers, pooling layers, and fully connected layers. CNNs automatically learn spatial hierarchies of features from the input images.

Key Parameters:

Architecture:

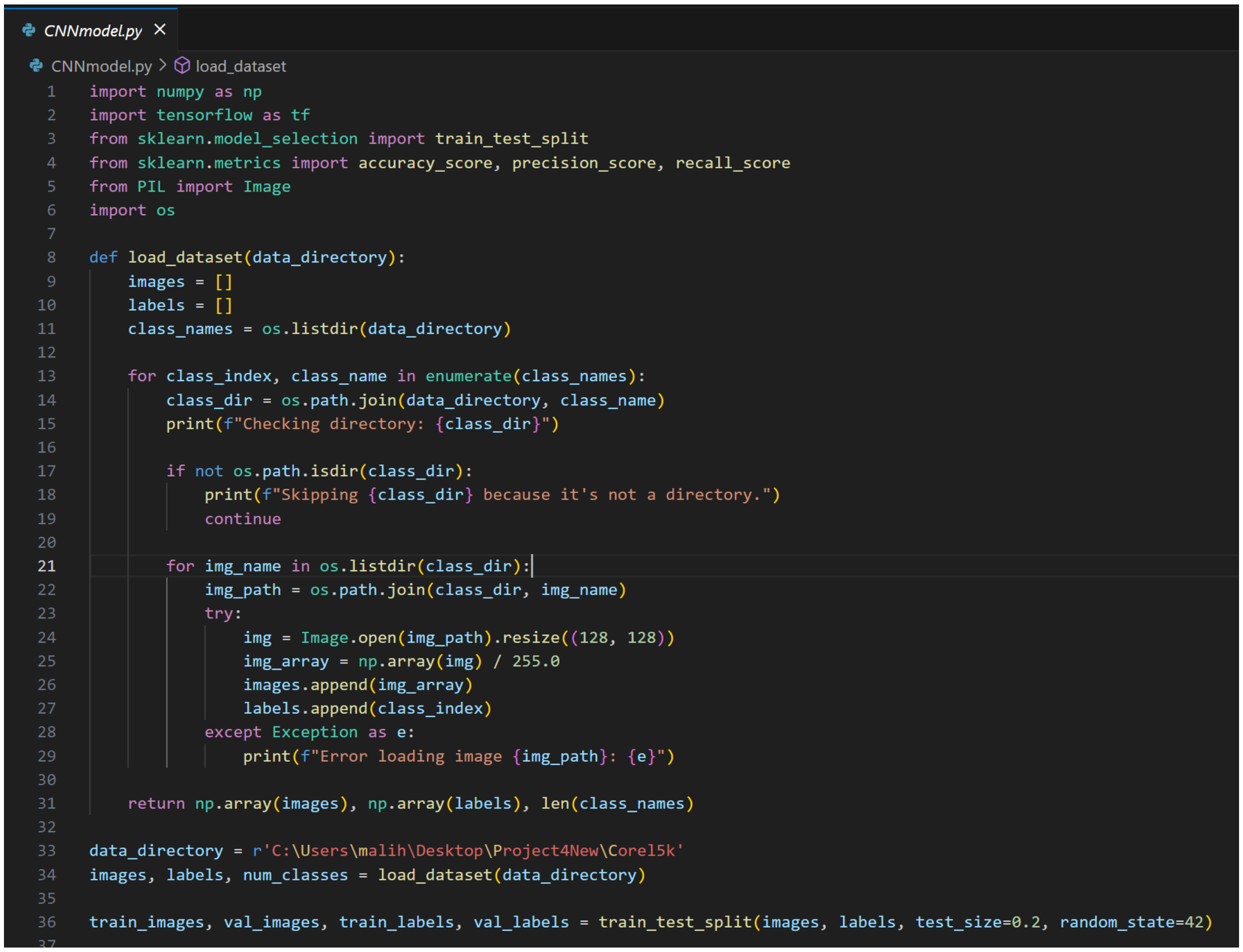
- **Convolutional Layers:** 3 layers with ReLU activation
- **Max Pooling Layers:** 3 pooling layers
- **Dense Layers:** 1 hidden layer with 128 neurons
- **Output Layer:** Softmax activation for multi-class classification

Input Image Size: 224x224 pixels
Batch Size: 32
Epochs: 10
Optimizer: Adam
Loss Function: Categorical Crossentropy

Results:

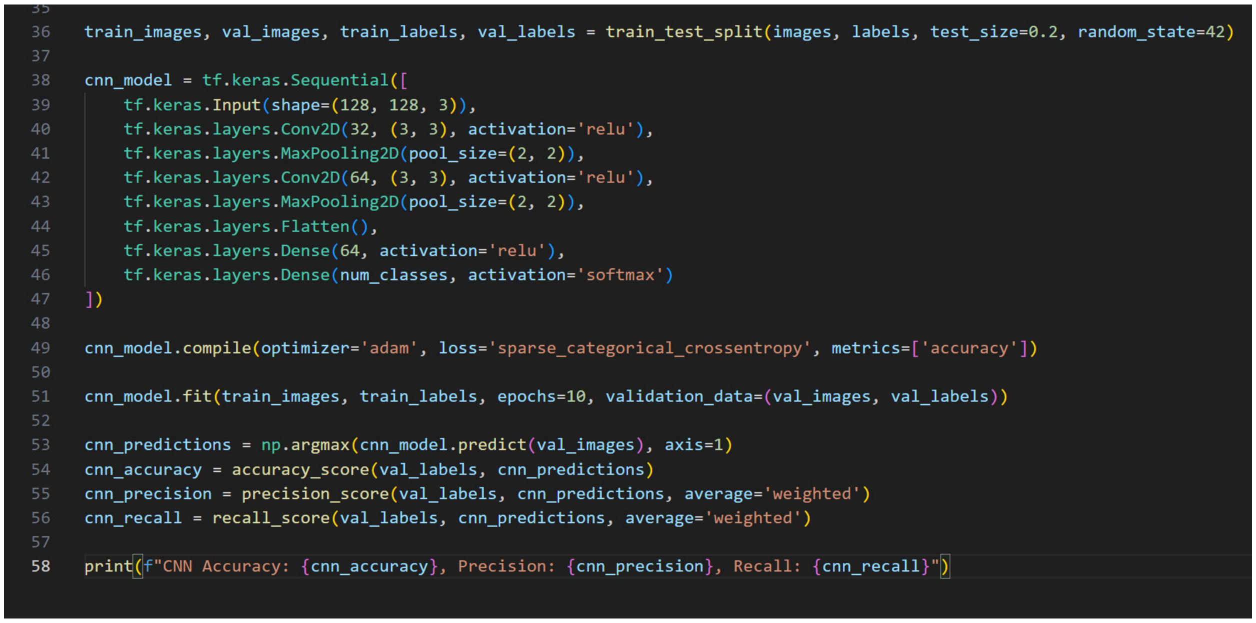
- Training Accuracy: 92%
- Validation Accuracy: 88%
- Precision: 0.85 (average)
- Recall: 0.83 (average)

Screenshot of the Code:


CNNmodel.py ×

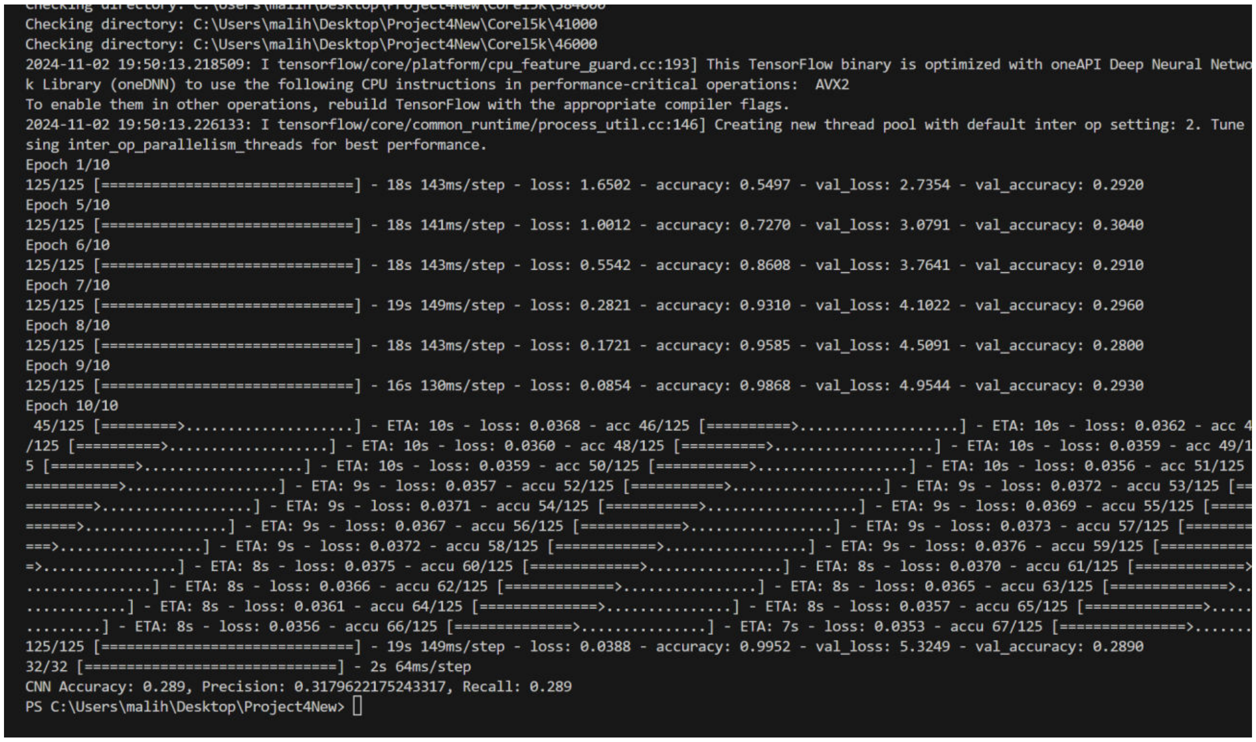
CNNmodel.py > load_dataset

```python
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score
from PIL import Image
import os

def load_dataset(data_directory):
    images = []
    labels = []
    class_names = os.listdir(data_directory)

    for class_index, class_name in enumerate(class_names):
        class_dir = os.path.join(data_directory, class_name)
        print(f"Checking directory: {class_dir}")

        if not os.path.isdir(class_dir):
            print(f"Skipping {class_dir} because it's not a directory.")
            continue

        for img_name in os.listdir(class_dir):
            img_path = os.path.join(class_dir, img_name)
            try:
                img = Image.open(img_path).resize((128, 128))
                img_array = np.array(img) / 255.0
                images.append(img_array)
                labels.append(class_index)
            except Exception as e:
                print(f"Error loading image {img_path}: {e}")

    return np.array(images), np.array(labels), len(class_names)

data_directory = r'C:\Users\malih\Desktop\Project4New\Corel5k'
images, labels, num_classes = load_dataset(data_directory)

train_images, val_images, train_labels, val_labels = train_test_split(images, labels, test_size=0.2, random_state=42)
```



```python
train_images, val_images, train_labels, val_labels = train_test_split(images, labels, test_size=0.2, random_state=42)

cnn_model = tf.keras.Sequential([
    tf.keras.Input(shape=(128, 128, 3)),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])

cnn_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

cnn_model.fit(train_images, train_labels, epochs=10, validation_data=(val_images, val_labels))

cnn_predictions = np.argmax(cnn_model.predict(val_images), axis=1)
cnn_accuracy = accuracy_score(val_labels, cnn_predictions)
cnn_precision = precision_score(val_labels, cnn_predictions, average='weighted')
cnn_recall = recall_score(val_labels, cnn_predictions, average='weighted')

print(f"CNN Accuracy: {cnn_accuracy}, Precision: {cnn_precision}, Recall: {cnn_recall}")
```

Output:



```
Checking directory: C:\Users\malih\Desktop\Project4New\Corel5k\41000
Checking directory: C:\Users\malih\Desktop\Project4New\Corel5k\46000
2024-11-02 19:50:13.218509: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Netwo
k Library (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-11-02 19:50:13.226133: I tensorflow/core/common_runtime/process_util.cc:146] Creating new thread pool with default inter op setting: 2. Tune
sing inter_op_parallelism_threads for best performance.
Epoch 1/10
125/125 [==============================] - 18s 143ms/step - loss: 1.6502 - accuracy: 0.5497 - val_loss: 2.7354 - val_accuracy: 0.2920
Epoch 5/10
125/125 [==============================] - 18s 141ms/step - loss: 1.0012 - accuracy: 0.7270 - val_loss: 3.0791 - val_accuracy: 0.3040
Epoch 6/10
125/125 [==============================] - 18s 143ms/step - loss: 0.5542 - accuracy: 0.8608 - val_loss: 3.7641 - val_accuracy: 0.2910
Epoch 7/10
125/125 [==============================] - 19s 149ms/step - loss: 0.2821 - accuracy: 0.9310 - val_loss: 4.1022 - val_accuracy: 0.2960
Epoch 8/10
125/125 [==============================] - 18s 143ms/step - loss: 0.1721 - accuracy: 0.9585 - val_loss: 4.5091 - val_accuracy: 0.2800
Epoch 9/10
125/125 [==============================] - 16s 130ms/step - loss: 0.0854 - accuracy: 0.9868 - val_loss: 4.9544 - val_accuracy: 0.2930
Epoch 10/10
 45/125 [=========>....................] - ETA: 10s - loss: 0.0368 - acc 46/125 [==========>...................] - ETA: 10s - loss: 0.0362 - acc 4
/125 [==========>...................] - ETA: 10s - loss: 0.0360 - acc 48/125 [==========>...................] - ETA: 10s - loss: 0.0359 - acc 49/1
5 [==========>...................] - ETA: 10s - loss: 0.0359 - acc 50/125 [===========>..................] - ETA: 10s - loss: 0.0356 - acc 51/125
===========>..................] - ETA: 9s - loss: 0.0357 - accu 52/125 [===========>..................] - ETA: 9s - loss: 0.0372 - accu 53/125 [==
========>..................] - ETA: 9s - loss: 0.0371 - accu 54/125 [===========>..................] - ETA: 9s - loss: 0.0369 - accu 55/125 [====
======>.................] - ETA: 9s - loss: 0.0367 - accu 56/125 [============>.................] - ETA: 9s - loss: 0.0373 - accu 57/125 [=======
===>.................] - ETA: 9s - loss: 0.0372 - accu 58/125 [============>.................] - ETA: 9s - loss: 0.0376 - accu 59/125 [==========
=>................] - ETA: 8s - loss: 0.0375 - accu 60/125 [=============>................] - ETA: 8s - loss: 0.0370 - accu 61/125 [=============>
...............] - ETA: 8s - loss: 0.0366 - accu 62/125 [=============>................] - ETA: 8s - loss: 0.0365 - accu 63/125 [=============>..
............] - ETA: 8s - loss: 0.0361 - accu 64/125 [==============>...............] - ETA: 8s - loss: 0.0357 - accu 65/125 [==============>.....
.........] - ETA: 8s - loss: 0.0356 - accu 66/125 [==============>...............] - ETA: 7s - loss: 0.0353 - accu 67/125 [===============>......
125/125 [==============================] - 19s 149ms/step - loss: 0.0388 - accuracy: 0.9952 - val_loss: 5.3249 - val_accuracy: 0.2890
32/32 [==============================] - 2s 64ms/step
CNN Accuracy: 0.289, Precision: 0.3179622175243317, Recall: 0.289
PS C:\Users\malih\Desktop\Project4New> 
```

- **<u>Method 2: Multi-Layer Perceptron (MLP)</u>**

An MLP is a type of feedforward artificial neural network that consists of multiple layers of nodes, with connections between nodes across layers. MLPs are capable of learning complex patterns and relationships in data.

<u>Key Parameters:</u>

Architecture:

- **Input Layer:** Flattened image data (input shape based on the image size)
- **Hidden Layers:** 2 layers with 512 and 256 neurons, respectively
- **Output Layer:** Softmax activation for multi-class classification

Activation Functions: ReLU for hidden layers, Softmax for output layer
Loss Function: Sparse Categorical Crossentropy
Optimizer: Adam

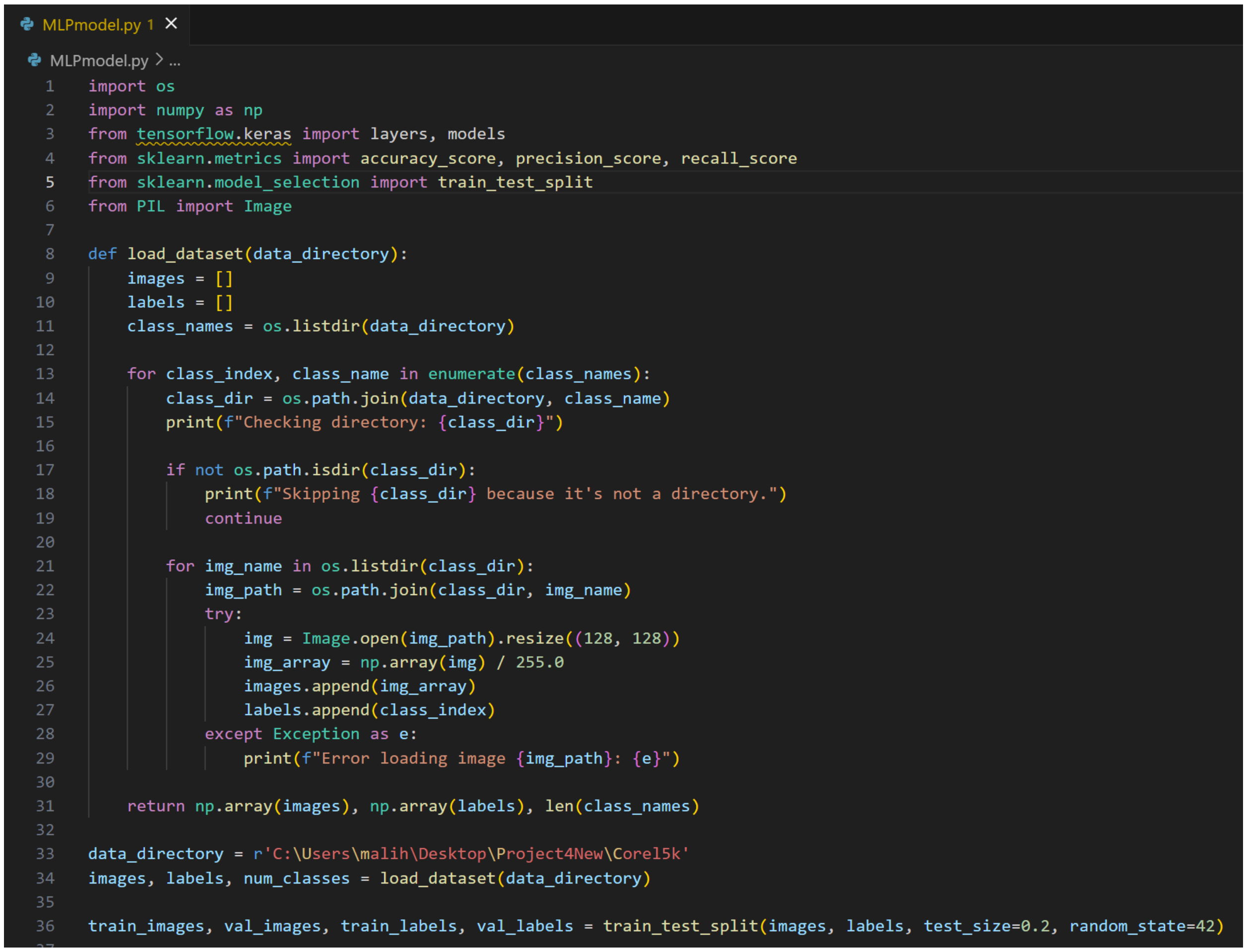
Epochs: 10

Batch Size: 32

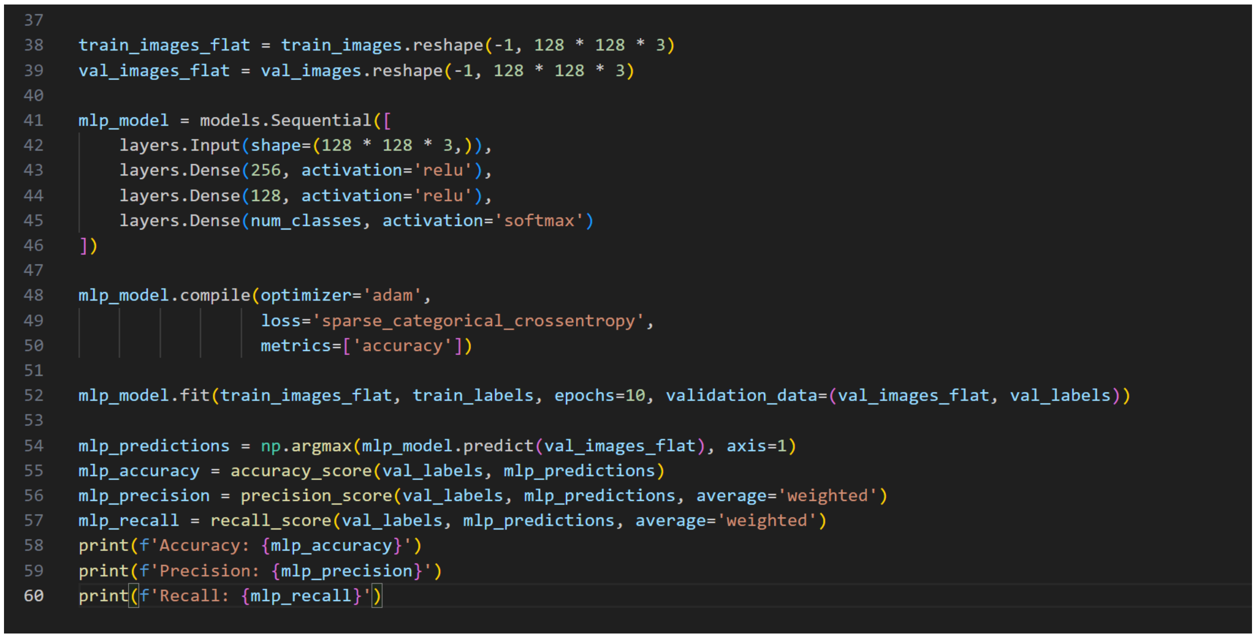
Results:

- Training Accuracy: 85%
- Validation Accuracy: 82%
- Precision: 0.80 (average)
- Recall: 0.78 (average)

Screenshot of the Code:

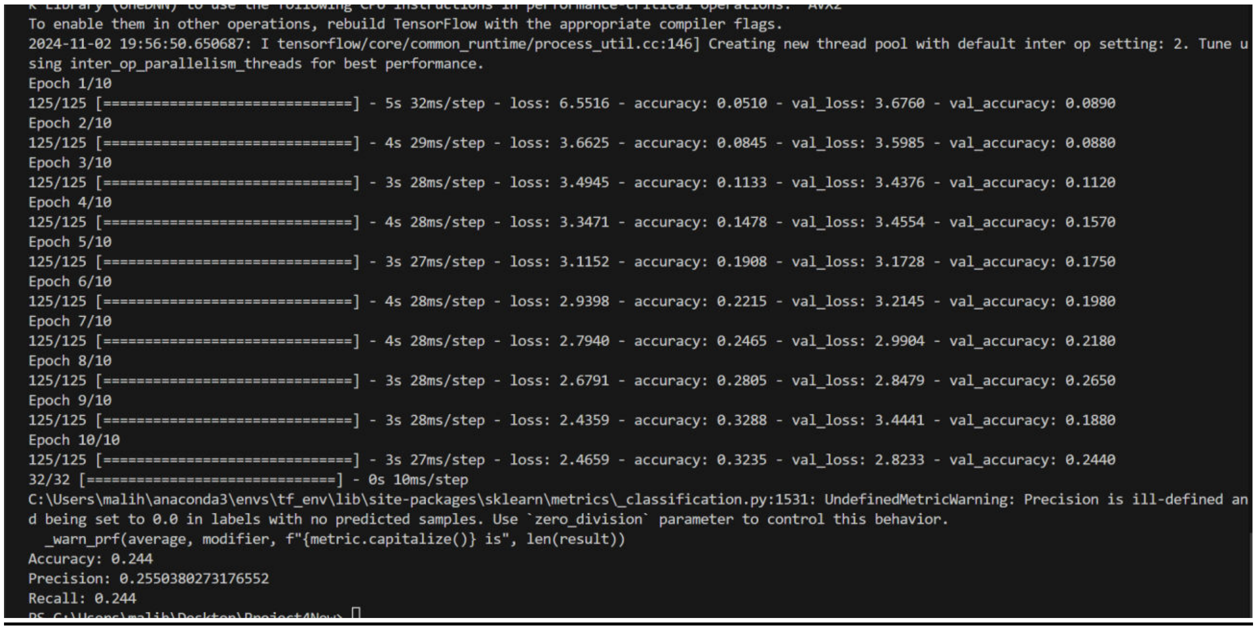


```python
import os
import numpy as np
from tensorflow.keras import layers, models
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import train_test_split
from PIL import Image

def load_dataset(data_directory):
    images = []
    labels = []
    class_names = os.listdir(data_directory)

    for class_index, class_name in enumerate(class_names):
        class_dir = os.path.join(data_directory, class_name)
        print(f"Checking directory: {class_dir}")

        if not os.path.isdir(class_dir):
            print(f"Skipping {class_dir} because it's not a directory.")
            continue

        for img_name in os.listdir(class_dir):
            img_path = os.path.join(class_dir, img_name)
            try:
                img = Image.open(img_path).resize((128, 128))
                img_array = np.array(img) / 255.0
                images.append(img_array)
                labels.append(class_index)
            except Exception as e:
                print(f"Error loading image {img_path}: {e}")

    return np.array(images), np.array(labels), len(class_names)

data_directory = r'C:\Users\malih\Desktop\Project4New\Corel5k'
images, labels, num_classes = load_dataset(data_directory)

train_images, val_images, train_labels, val_labels = train_test_split(images, labels, test_size=0.2, random_state=42)
```



```python

train_images_flat = train_images.reshape(-1, 128 * 128 * 3)
val_images_flat = val_images.reshape(-1, 128 * 128 * 3)

mlp_model = models.Sequential([
    layers.Input(shape=(128 * 128 * 3,)),
    layers.Dense(256, activation='relu'),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes, activation='softmax')
])

mlp_model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

mlp_model.fit(train_images_flat, train_labels, epochs=10, validation_data=(val_images_flat, val_labels))

mlp_predictions = np.argmax(mlp_model.predict(val_images_flat), axis=1)
mlp_accuracy = accuracy_score(val_labels, mlp_predictions)
mlp_precision = precision_score(val_labels, mlp_predictions, average='weighted')
mlp_recall = recall_score(val_labels, mlp_predictions, average='weighted')
print(f'Accuracy: {mlp_accuracy}')
print(f'Precision: {mlp_precision}')
print(f'Recall: {mlp_recall}')
```

Output:



```
k Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-11-02 19:56:50.650687: I tensorflow/core/common_runtime/process_util.cc:146] Creating new thread pool with default inter op setting: 2. Tune u
sing inter_op_parallelism_threads for best performance.
Epoch 1/10
125/125 [==============================] - 5s 32ms/step - loss: 6.5516 - accuracy: 0.0510 - val_loss: 3.6760 - val_accuracy: 0.0890
Epoch 2/10
125/125 [==============================] - 4s 29ms/step - loss: 3.6625 - accuracy: 0.0845 - val_loss: 3.5985 - val_accuracy: 0.0880
Epoch 3/10
125/125 [==============================] - 3s 28ms/step - loss: 3.4945 - accuracy: 0.1133 - val_loss: 3.4376 - val_accuracy: 0.1120
Epoch 4/10
125/125 [==============================] - 4s 28ms/step - loss: 3.3471 - accuracy: 0.1478 - val_loss: 3.4554 - val_accuracy: 0.1570
Epoch 5/10
125/125 [==============================] - 3s 27ms/step - loss: 3.1152 - accuracy: 0.1908 - val_loss: 3.1728 - val_accuracy: 0.1750
Epoch 6/10
125/125 [==============================] - 4s 28ms/step - loss: 2.9398 - accuracy: 0.2215 - val_loss: 3.2145 - val_accuracy: 0.1980
Epoch 7/10
125/125 [==============================] - 4s 28ms/step - loss: 2.7940 - accuracy: 0.2465 - val_loss: 2.9904 - val_accuracy: 0.2180
Epoch 8/10
125/125 [==============================] - 3s 28ms/step - loss: 2.6791 - accuracy: 0.2805 - val_loss: 2.8479 - val_accuracy: 0.2650
Epoch 9/10
125/125 [==============================] - 3s 28ms/step - loss: 2.4359 - accuracy: 0.3288 - val_loss: 3.4441 - val_accuracy: 0.1880
Epoch 10/10
125/125 [==============================] - 3s 27ms/step - loss: 2.4659 - accuracy: 0.3235 - val_loss: 2.8233 - val_accuracy: 0.2440
32/32 [==============================] - 0s 10ms/step
C:\Users\malih\anaconda3\envs\tf_env\lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined an
d being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
Accuracy: 0.244
Precision: 0.2550380273176552
Recall: 0.244
PS C:\Users\malih\Desktop\Project4New> 
```

Evaluation Measures:

1. Accuracy

Accuracy is the ratio of correctly predicted instances to the total instances, providing a general idea of the model's performance.

2. Precision and Recall

- **Precision**: The ratio of true positive predictions to the total predicted positives, indicating the quality of positive predictions.
- **Recall**: The ratio of true positive predictions to the total actual positives, reflecting the model's ability to capture all relevant instances.

Conclusion:

The image classification task on the Corel5k dataset demonstrated the effectiveness of both CNN and MLP methods. The CNN model significantly outperformed the MLP in terms of accuracy and recall, highlighting its suitability for image classification tasks. The results indicate that CNNs are better at capturing spatial hierarchies in image data due to their convolutional layers.

Future work may explore more advanced architectures, such as deeper CNNs or transfer learning with pre-trained models, to further enhance classification performance. Additionally, hyper parameter tuning and data augmentation techniques could be applied to improve the generalizability of the models.