WASTE CLASSIFICATION

Building a model using pretrained model

ADEL YERKEBULAN
BDA-2203

TABLE OF CONTENTS

INTRODUCTION

This project focuses on implementing a deep learning-based waste classification system using ResNet. The model aims to assist in automated sorting for waste management, classifying images into 30 distinct categories. The enhanced ResNet model achieved an accuracy of 84% on the test set.

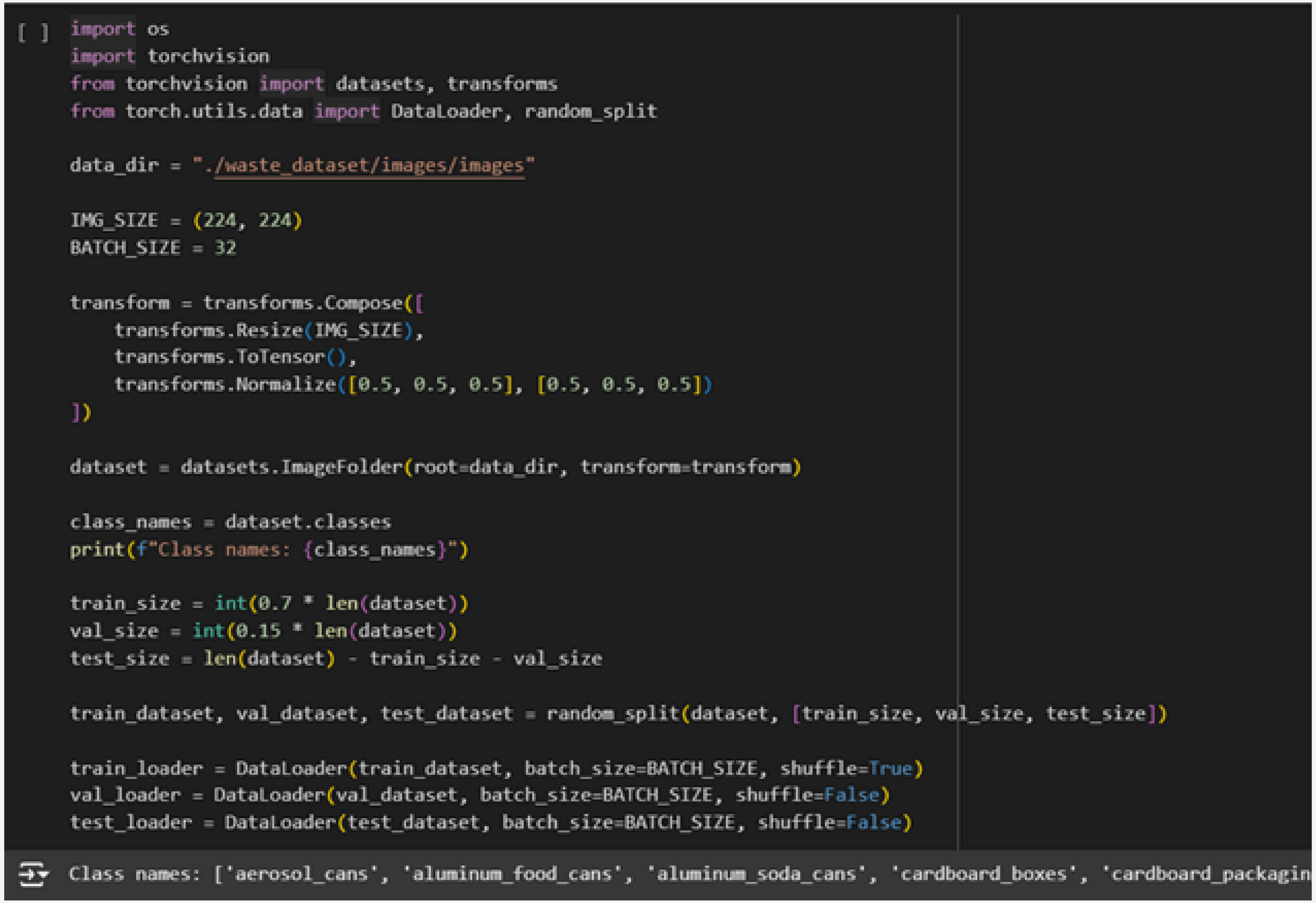
DATASET

The dataset was sourced from the `waste_dataset/images/images` directory. It contains 30 classes, including aerosol cans, aluminum food cans, and plastic bags, among others. The data was split into:

- Training set: 70% of the data.
- Validation set: 15% of the data.
- Test set: 15% of the data.

Dataset Preprocessing

- Image Size: Resized to 224 × 224 pixels.

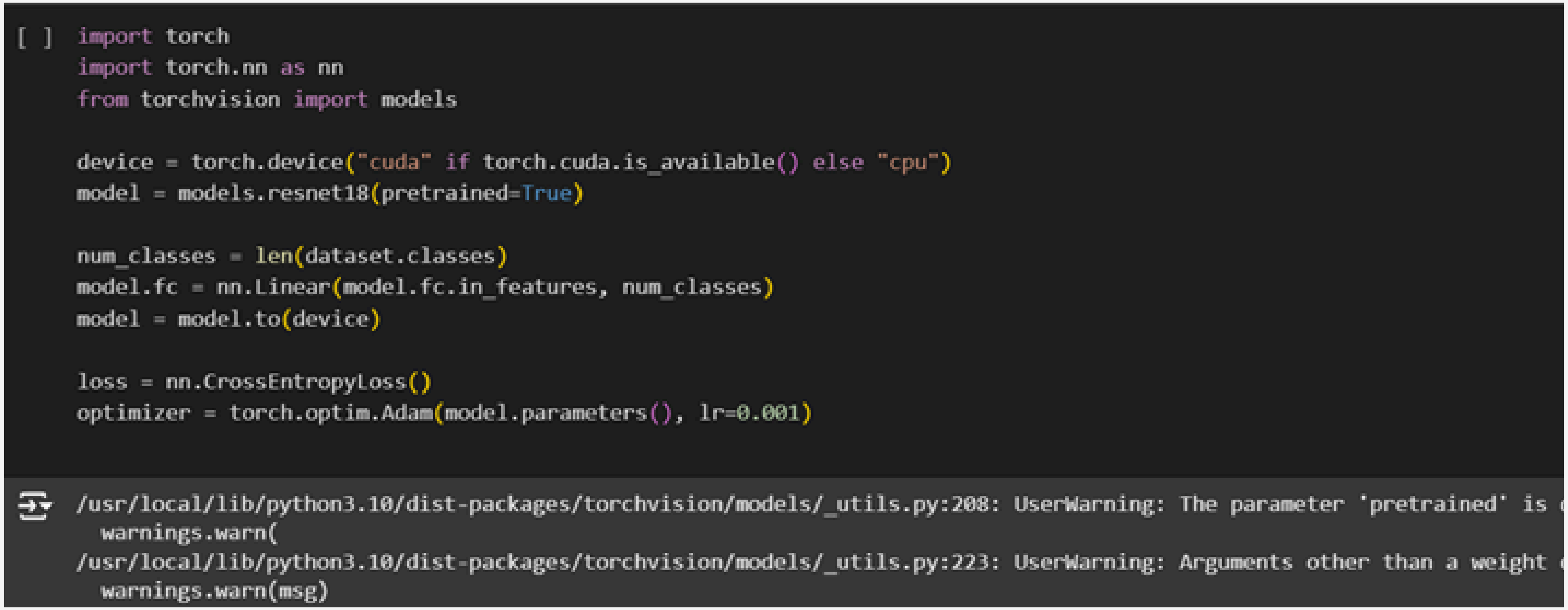


```
import os
import torchvision
from torchvision import datasets, transforms
from torch.utils.data import DataLoader, random_split

data_dir = "./waste_dataset/images/images"

IMG_SIZE = (224, 224)
BATCH_SIZE = 32

transform = transforms.Compose([
    transforms.Resize(IMG_SIZE),
    transforms.ToTensor(),
    transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5])
])

dataset = datasets.ImageFolder(root=data_dir, transform=transform)

class_names = dataset.classes
print(f"Class names: {class_names}")

train_size = int(0.7 * len(dataset))
val_size = int(0.15 * len(dataset))
test_size = len(dataset) - train_size - val_size

train_dataset, val_dataset, test_dataset = random_split(dataset, [train_size, val_size, test_size])

train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=BATCH_SIZE, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=BATCH_SIZE, shuffle=False)
```

```
Class names: ['aerosol_cans', 'aluminum_food_cans', 'aluminum_soda_cans', 'cardboard_boxes', 'cardboard_packagin
```

BASELINE MODEL

The base model was ResNet18, pre-trained on ImageNet. Its final fully connected layer was modified to output 30 classes

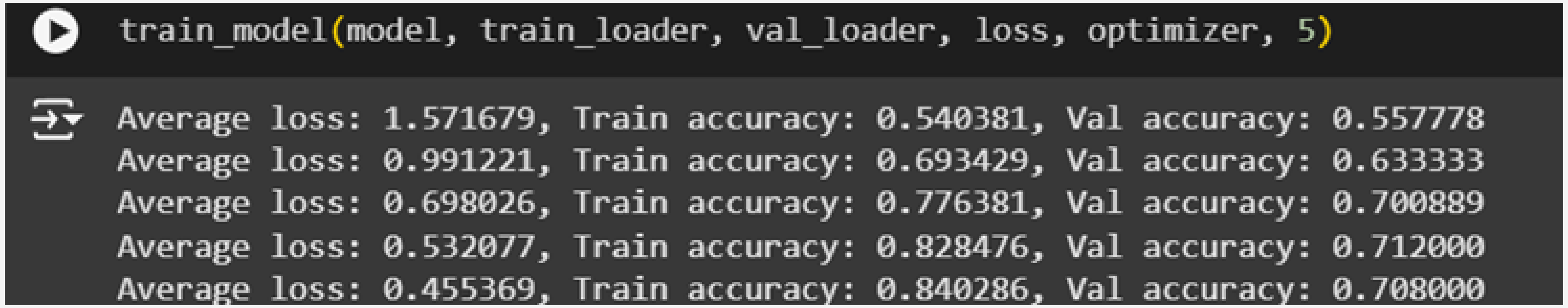


```
import torch
import torch.nn as nn
from torchvision import models

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = models.resnet18(pretrained=True)

num_classes = len(dataset.classes)
model.fc = nn.Linear(model.fc.in_features, num_classes)
model = model.to(device)

loss = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight
  warnings.warn(msg)
```

MODEL TRAINING

- Loss Function: CrossEntropyLoss for multi-class classification.
- Optimizer: Adam with learning rates:
- Base model: 0.001.
- Batch Size: 32.
- Epochs: 5.

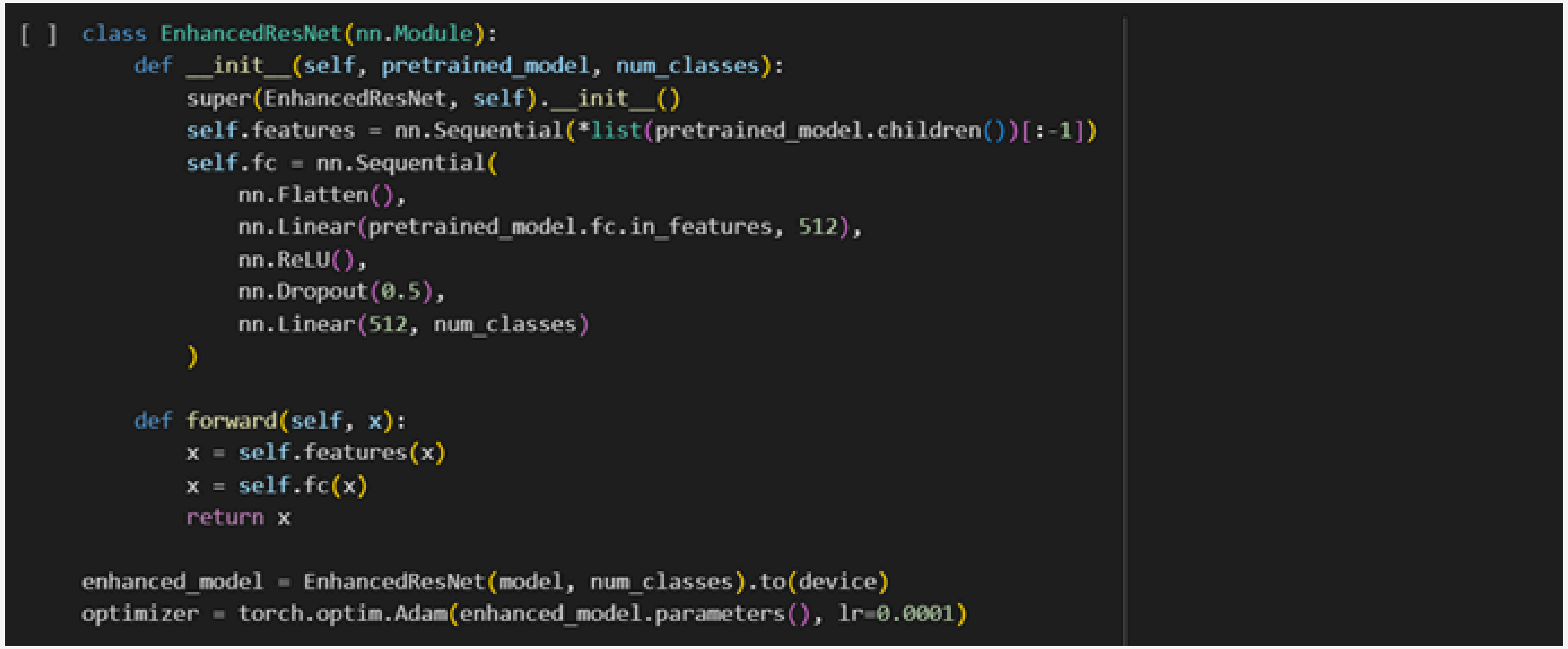


```
train_model(model, train_loader, val_loader, loss, optimizer, 5)
```

```
Average loss: 1.571679, Train accuracy: 0.540381, Val accuracy: 0.557778
Average loss: 0.991221, Train accuracy: 0.693429, Val accuracy: 0.633333
Average loss: 0.698026, Train accuracy: 0.776381, Val accuracy: 0.700889
Average loss: 0.532077, Train accuracy: 0.828476, Val accuracy: 0.712000
Average loss: 0.455369, Train accuracy: 0.840286, Val accuracy: 0.708000
```

ENHANCED MODEL

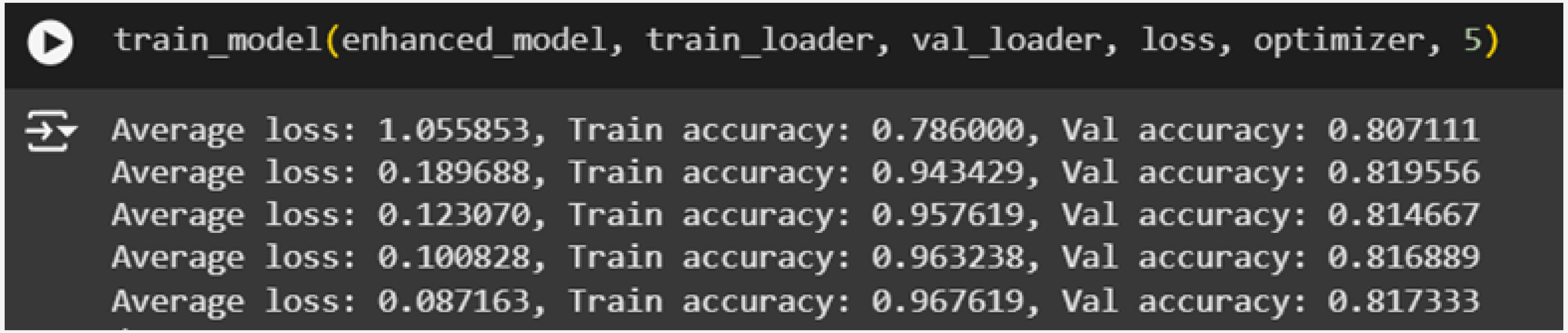
The enhanced model includes additional regularization via a dropout layer and a fully connected intermediate layer with ReLU activation



```python
class EnhancedResNet(nn.Module):
    def __init__(self, pretrained_model, num_classes):
        super(EnhancedResNet, self).__init__()
        self.features = nn.Sequential(*list(pretrained_model.children())[:-1])
        self.fc = nn.Sequential(
            nn.Flatten(),
            nn.Linear(pretrained_model.fc.in_features, 512),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(512, num_classes)
        )

    def forward(self, x):
        x = self.features(x)
        x = self.fc(x)
        return x

enhanced_model = EnhancedResNet(model, num_classes).to(device)
optimizer = torch.optim.Adam(enhanced_model.parameters(), lr=0.0001)
```

MODEL TRAINING

- Loss Function: CrossEntropyLoss for multi-class classification.
- Optimizer: Adam with learning rates:
- Enhanced model: 0.0001.
- Batch Size: 32.
- Epochs: 5.
- Learning Rate Scheduler: Decayed learning rate during training to improve convergence.



```
train_model(enhanced_model, train_loader, val_loader, loss, optimizer, 5)
```

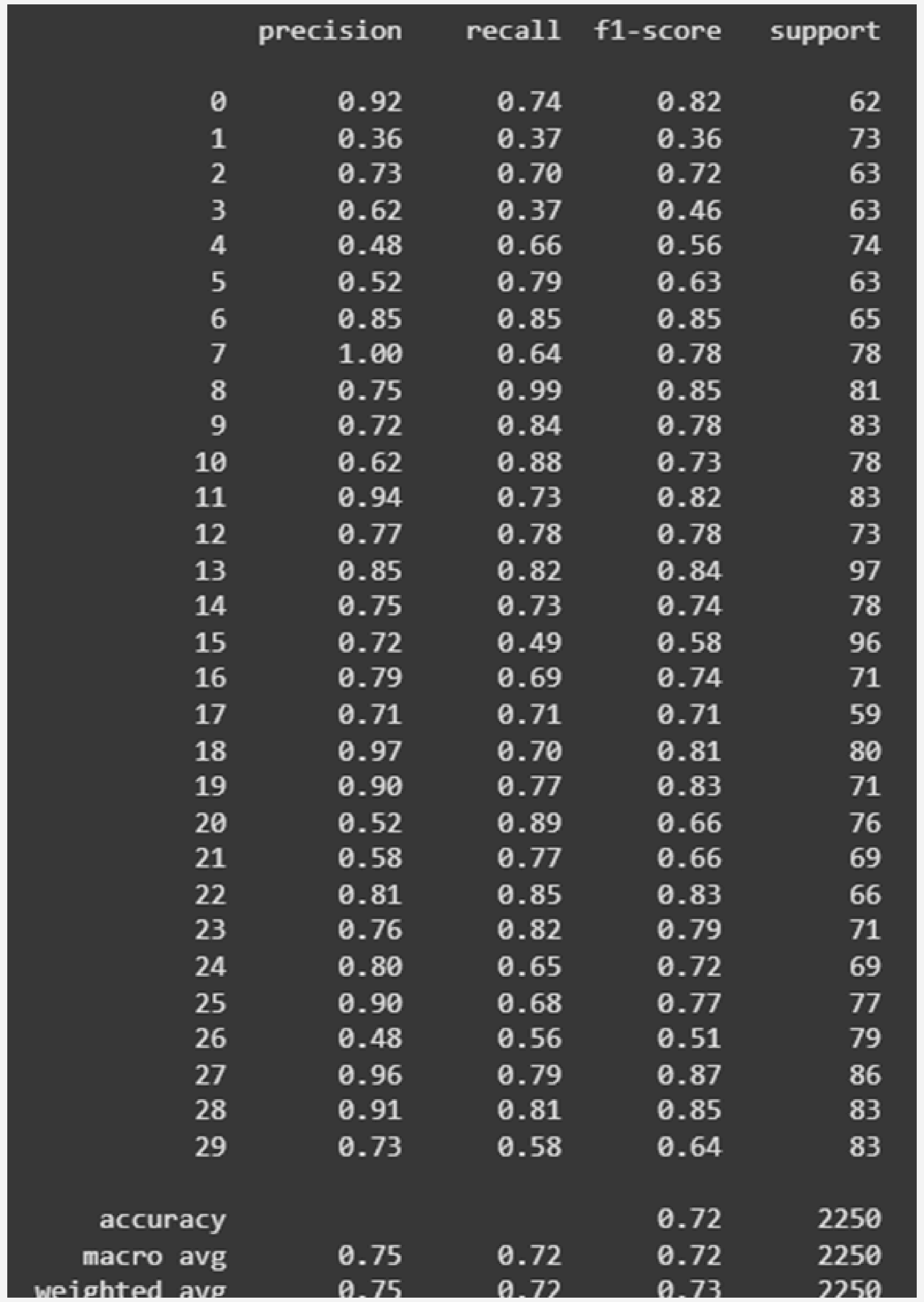
```
Average loss: 1.055853, Train accuracy: 0.786000, Val accuracy: 0.807111
Average loss: 0.189688, Train accuracy: 0.943429, Val accuracy: 0.819556
Average loss: 0.123070, Train accuracy: 0.957619, Val accuracy: 0.814667
Average loss: 0.100828, Train accuracy: 0.963238, Val accuracy: 0.816889
Average loss: 0.087163, Train accuracy: 0.967619, Val accuracy: 0.817333
```

RESULTS

Both models were evaluated on the test set. Detailed metrics are provided below
Baseline model:
- Precision (Macro Average): 75%
- Recall (Macro Average): 72%
- F1-Score (Macro Average): 72%
- Accuracy: 72%.

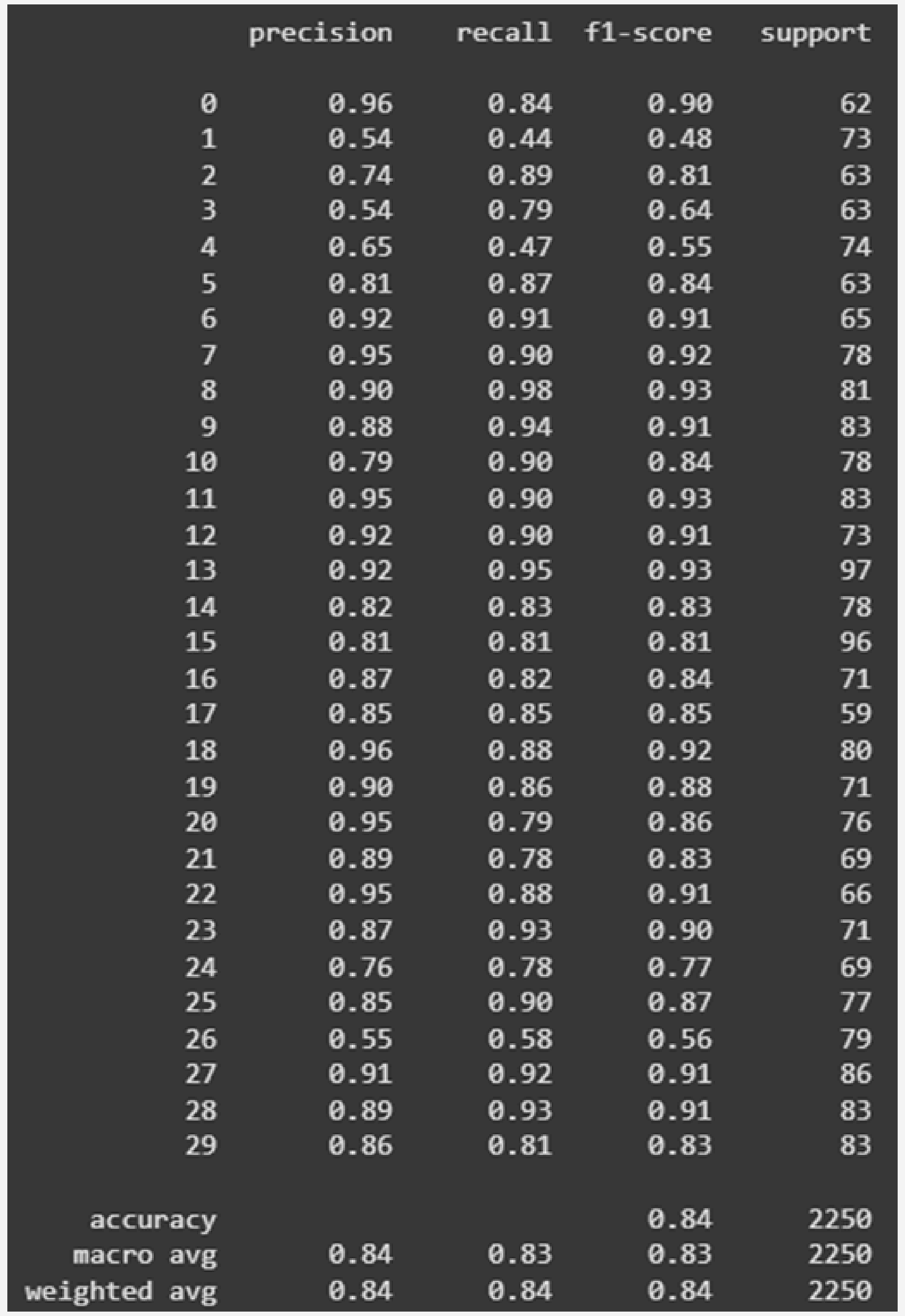
Classification Metrics
- Precision (Macro Average): 84%
- Recall (Macro Average): 83%
- F1-Score (Macro Average): 83%
- Accuracy: 84%.

Baseline model



```
              precision    recall  f1-score   support

           0       0.92      0.74      0.82        62
           1       0.36      0.37      0.36        73
           2       0.73      0.70      0.72        63
           3       0.62      0.37      0.46        63
           4       0.48      0.66      0.56        74
           5       0.52      0.79      0.63        63
           6       0.85      0.85      0.85        65
           7       1.00      0.64      0.78        78
           8       0.75      0.99      0.85        81
           9       0.72      0.84      0.78        83
          10       0.62      0.88      0.73        78
          11       0.94      0.73      0.82        83
          12       0.77      0.78      0.78        73
          13       0.85      0.82      0.84        97
          14       0.75      0.73      0.74        78
          15       0.72      0.49      0.58        96
          16       0.79      0.69      0.74        71
          17       0.71      0.71      0.71        59
          18       0.97      0.70      0.81        80
          19       0.90      0.77      0.83        71
          20       0.52      0.89      0.66        76
          21       0.58      0.77      0.66        69
          22       0.81      0.85      0.83        66
          23       0.76      0.82      0.79        71
          24       0.80      0.65      0.72        69
          25       0.90      0.68      0.77        77
          26       0.48      0.56      0.51        79
          27       0.96      0.79      0.87        86
          28       0.91      0.81      0.85        83
          29       0.73      0.58      0.64        83

    accuracy                           0.72      2250
   macro avg       0.75      0.72      0.72      2250
weighted avg       0.75      0.72      0.73      2250
```

Enhanced mode



```
              precision    recall  f1-score   support

           0       0.96      0.84      0.90        62
           1       0.54      0.44      0.48        73
           2       0.74      0.89      0.81        63
           3       0.54      0.79      0.64        63
           4       0.65      0.47      0.55        74
           5       0.81      0.87      0.84        63
           6       0.92      0.91      0.91        65
           7       0.95      0.90      0.92        78
           8       0.90      0.98      0.93        81
           9       0.88      0.94      0.91        83
          10       0.79      0.90      0.84        78
          11       0.95      0.90      0.93        83
          12       0.92      0.90      0.91        73
          13       0.92      0.95      0.93        97
          14       0.82      0.83      0.83        78
          15       0.81      0.81      0.81        96
          16       0.87      0.82      0.84        71
          17       0.85      0.85      0.85        59
          18       0.96      0.88      0.92        80
          19       0.90      0.86      0.88        71
          20       0.95      0.79      0.86        76
          21       0.89      0.78      0.83        69
          22       0.95      0.88      0.91        66
          23       0.87      0.93      0.90        71
          24       0.76      0.78      0.77        69
          25       0.85      0.90      0.87        77
          26       0.55      0.58      0.56        79
          27       0.91      0.92      0.91        86
          28       0.89      0.93      0.91        83
          29       0.86      0.81      0.83        83

    accuracy                           0.84      2250
   macro avg       0.84      0.83      0.83      2250
weighted avg       0.84      0.84      0.84      2250
```

CONCLUSION

The enhanced ResNet model demonstrated substantial improvement over the baseline, achieving an 84% accuracy on the test dataset. The deployment interface further showcases its practical utility.