IMAGE CLASSIFICATION

Github link to the jupyter notebook:
https://github.com/ashutosh-999/img_cls_jn.git

SCREENSHOTS

1. **Neural Network model:**

```
class TinyVGG(nn.Module):
  def __init__(self, input_shape, hidden_units, output_shape):
    super().__init__()
    self.conv_block_1 = nn.Sequential(
        nn.Conv2d(in_channels=input_shape,
                  out_channels=hidden_units,
                  kernel_size=3,
                  stride=1,
                  padding=1),
        nn.ReLU(),
        nn.Conv2d(in_channels=hidden_units,
                  out_channels=hidden_units,
                  kernel_size=3,
                  stride=1,
                  padding=1),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2)
    )
    self.conv_block_2 = nn.Sequential(
        nn.Conv2d(in_channels=hidden_units,
                  out_channels=hidden_units,
                  kernel_size=3,
                  stride=1,
                  padding=1),
        nn.ReLU(),
        nn.Conv2d(in_channels=hidden_units,
                  out_channels=hidden_units,
                  kernel_size=3,
                  stride=1,
                  padding=1),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2)
    )
    self.classifier = nn.Sequential(
        nn.Flatten()
```

```
    self.classifier = nn.Sequential(
        nn.Flatten(),
        nn.Linear(in_features=hidden_units*16*16,
                  out_features=output_shape))

  def forward(self, x):
    x = self.conv_block_1(x)
    # print(f"Layer 1 shape: {x.shape}")
    x = self.conv_block_2(x)
    # print(f"Layer 2 shape: {x.shape}")
    x = self.classifier(x)
    # print(f"Layer 3 shape: {x.shape}")
    return x
```

2. Downloading the dataset:

```python
# Setup path to data folder
data_path = Path("data/")
image_path = data_path / "pizza_steak_sushi_20_percent"

# If the image folder doesn't exist, download it and prepare it...
if image_path.is_dir():
    print(f"{image_path} directory exists.")
else:
    print(f"Did not find {image_path} directory, creating one...")
    image_path.mkdir(parents=True, exist_ok=True)

# Download pizza, steak, sushi data
with open(data_path / "pizza_steak_sushi_20_percent.zip", "wb") as f:
    request = requests.get("https://github.com/mrdbourke/pytorch-deep-learning/raw/main/data/pizza_steak_sushi_20_percent.zip")
    print("Downloading pizza, steak, sushi 20% data...")
    f.write(request.content)

# Unzip pizza, steak, sushi data
with zipfile.ZipFile(data_path / "pizza_steak_sushi_20_percent.zip", "r") as zip_ref:
    print("Unzipping pizza, steak, sushi 20% data...")
    zip_ref.extractall(image_path)
```

3. Optimizer and Loss Function used:

```python
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model_4.parameters(), lr=0.001)
```

4. Accuracy of the model:

```
Epoch: 34 | train_loss: 0.0039 | train_acc: 1.0000 | test_loss: 2.8498 | test_acc: 0.5159
Epoch: 35 | train_loss: 0.0020 | train_acc: 1.0000 | test_loss: 2.9435 | test_acc: 0.5068
Epoch: 36 | train_loss: 0.0016 | train_acc: 1.0000 | test_loss: 2.9854 | test_acc: 0.5068
Epoch: 37 | train_loss: 0.0013 | train_acc: 1.0000 | test_loss: 3.0201 | test_acc: 0.5159
Epoch: 38 | train_loss: 0.0012 | train_acc: 1.0000 | test_loss: 3.0623 | test_acc: 0.5159
Epoch: 39 | train_loss: 0.0011 | train_acc: 1.0000 | test_loss: 3.1062 | test_acc: 0.5222
Epoch: 40 | train_loss: 0.0010 | train_acc: 1.0000 | test_loss: 3.1454 | test_acc: 0.5222
Epoch: 41 | train_loss: 0.0009 | train_acc: 1.0000 | test_loss: 3.1671 | test_acc: 0.5159
Epoch: 42 | train_loss: 0.0008 | train_acc: 1.0000 | test_loss: 3.1913 | test_acc: 0.5284
Epoch: 43 | train_loss: 0.0007 | train_acc: 1.0000 | test_loss: 3.2148 | test_acc: 0.5222
Epoch: 44 | train_loss: 0.0007 | train_acc: 1.0000 | test_loss: 3.2454 | test_acc: 0.5284
Epoch: 45 | train_loss: 0.0007 | train_acc: 1.0000 | test_loss: 3.2679 | test_acc: 0.5375
Epoch: 46 | train_loss: 0.0006 | train_acc: 1.0000 | test_loss: 3.2939 | test_acc: 0.5312
Epoch: 47 | train_loss: 0.0006 | train_acc: 1.0000 | test_loss: 3.3156 | test_acc: 0.5222
Epoch: 48 | train_loss: 0.0006 | train_acc: 1.0000 | test_loss: 3.3332 | test_acc: 0.5222
Epoch: 49 | train_loss: 0.0005 | train_acc: 1.0000 | test_loss: 3.3536 | test_acc: 0.5159
Epoch: 50 | train_loss: 0.0005 | train_acc: 1.0000 | test_loss: 3.3776 | test_acc: 0.5159
```