Helper Library Setup Guide for Neural Network Projects

As part of this activity we will create a helper library that encapsulates common functionalities for data loading, model training, and evaluation, reducing duplication across various neural network projects we will work on this semester.

You will move appropriate code from Module4-FCNN and Module4-CNN notebooks to the newly created module.

1. Directory Structure

Set up your project structure as follows:

```
helper_lib/
├── __init__.py
├── data_loader.py
├── trainer.py
├── evaluator.py
├── model.py
└── utils.py
```

2. Data Loader Module

In `data_loader.py`, encapsulate data loading logic:

```python
import torch
from torchvision import datasets, transforms

def get_data_loader(data_dir, batch_size=32, train=True):
    # TODO: create the data loader
    return loader
```

3. Model Module

In `model.py`, define your neural network model:

```
import torch.nn as nn
from torchvision.models import resnet18

def get_model(model_name):
    # TODO: define and return the appropriate model_name - one of: FCNN, CNN,
EnhancedCNN
    return model
```

4. Trainer Module

In `trainer.py`, abstract the training loop:

```
import torch

def train_model(model, data_loader, criterion, optimizer, device='cpu',
epochs=10):
    # TODO: run several iterations of the training loop (based on epochs
parameter) and return the model
    return model
```

5. Evaluator Module

In `evaluator.py`, encapsulate evaluation metrics:

```
import torch

def evaluate_model(model, data_loader, criterion, device='cpu'):
    # TODO: calculate average loss and accuracy on the test dataset
    return avg_loss, accuracy
```

6. Usage Example

Here's how to use your helper library:

```python
from helper_lib.data_loader import get_data_loader
from helper_lib.trainer import train_model
from helper_lib.evaluator import evaluate_model
from helper_lib.model import get_model
from helper_lib.utils import save_model
import torch.nn as nn
import torch.optim as optim

train_loader = get_data_loader('data/train', batch_size=64)
test_loader = get_data_loader('data/test', batch_size=64, train=False)

model = get_model("CNN")
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

trained_model = train_model(model, train_loader, criterion, optimizer, epochs=5)
evaluate_model(trained_model, test_loader, criterion)
```