```
"""
EECS 445 - Introduction to Machine Learning
Fall 2022 - Project 2
Train Source CNN
    Train a convolutional neural network to classify images.
    Periodically output training information, and saves model checkpoints
    Usage: python train_source.py
"""

import torch
import numpy as np
import random
from dataset import get_train_val_test_loaders
from model.source import Source
from model.challenge import Challenge
from train_challenge import *
from utils import config
import utils

torch.manual_seed(42)
np.random.seed(42)
random.seed(42)


def main():
    """Train source model on multiclass data."""
    # Data loaders
    tr_loader, va_loader, te_loader, _ = get_train_val_test_loaders(
        task="source",
        batch_size=config("source.batch_size"),
    )

    # Model
    model = Challenge()

    # TODO: define loss function, and optimizer
    criterion = torch.nn.CrossEntropyLoss()

    optimizer = torch.optim.Adam(
        model.parameters(), lr=0.001, weight_decay=0.001)
    #

    print("Number of float-valued parameters:", count_parameters(model))

    # Attempts to restore the latest checkpoint if exists
    print("Loading source...")
    model, start_epoch, stats = restore_checkpoint(
        model, config("source.checkpoint"))

    axes = utils.make_training_plot("Source Training")

    # Evaluate the randomly initialized model
    evaluate_epoch(
        axes,
        tr_loader,
        va_loader,
        te_loader,
        model,
        criterion,
        start_epoch,
        stats,
        multiclass=True,
    )

    # initial val loss for early stopping
    global_min_loss = stats[0][1]

    # TODO: patience for early stopping
    patience = 5
    curr_count_to_patience = 0
    #

    # Loop over the entire dataset multiple times
    epoch = start_epoch
    while curr_count_to_patience < patience:
```

```
        # Train model
        train_epoch(tr_loader, model, criterion, optimizer)

        # Evaluate model
        evaluate_epoch(
            axes,
            tr_loader,
            va_loader,
            te_loader,
            model,
            criterion,
            epoch + 1,
            stats,
            multiclass=True,
        )

        # Save model parameters
        save_checkpoint(model, epoch + 1, config("source.checkpoint"), stats)

        curr_count_to_patience, global_min_loss = early_stopping(
            stats, curr_count_to_patience, global_min_loss
        )
        epoch += 1

    # Save figure and keep plot open
    print("Finished Training")
    utils.save_source_training_plot()
    utils.hold_training_plot()


if __name__ == "__main__":
    main()
```