```python
"""
EECS 445 - Introduction to Machine Learning
Fall 2022 - Project 2
Train Challenge
    Train a convolutional neural network to classify the heldout images
    Periodically output training information, and saves model checkpoints
    Usage: python train_challenge.py
"""
import torch
import numpy as np
import random
from dataset_challenge import get_train_val_test_loaders
from model.challenge import Challenge
from train_common import *
from utils import config
import utils

learning_rate = 0.00005


def main():
    # Data loaders
    if check_for_augmented_data("./data"):
        tr_loader, va_loader, te_loader, _ = get_train_val_test_loaders(
            task="target",
            batch_size=config("challenge.batch_size"), augment=True
        )
    else:
        tr_loader, va_loader, te_loader, _ = get_train_val_test_loaders(
            task="target",
            batch_size=config("challenge.batch_size"),
        )
    # Model
    model = Challenge()

    # TODO: define loss function, and optimizer
    criterion = torch.nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(
        model.parameters(), lr=learning_rate, weight_decay=0.001)

    #

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

    axes = utils.make_training_plot()

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

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

    # TODO: define 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
        )

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

        # Updates early stopping parameters
        curr_count_to_patience, global_min_loss = early_stopping(
            stats, curr_count_to_patience, global_min_loss
        )
        #
        epoch += 1
    print("Finished Training")
    # Save figure and keep plot open
    utils.save_challenge_training_plot()
    utils.hold_training_plot()


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