"""

Script to run the hybrid physics-guided neural network wheat model (HybridWheatModel)

Demonstrates the PyTorch implementation with the same physics as BaseWheatModel

"""

import pandas as pd

import numpy as np

import torch

import matplotlib.pyplot as plt

from models import HybridWheatModel

# Set random seed for reproducibility

torch.manual\_seed(42)

np.random.seed(42)

# Import data

print("Loading weather data...")

df = pd.read\_csv('./KS\_Manhattan\_6\_SSW.csv', na\_values=[-99, -9999])

# Convert LST\_DATE to Pandas datetime format

df.insert(2, "DATES", pd.to\_datetime(df["LST\_DATE"], format="%Y%m%d"))

# Replace missing values using interpolation

print("Interpolating missing values...")

df['T\_DAILY\_AVG'].interpolate(method="pchip", inplace=True)

df['SOLARAD\_DAILY'].interpolate(method="pchip", inplace=True)

# Define crop parameters

planting\_date = "2007-10-01"

planting\_date = pd.to\_datetime(planting\_date)

season\_length = 250 # days

harvest\_date = planting\_date + pd.to\_timedelta(season\_length, unit='days')

# Select records for growing season

print(f"Simulating growing season from {planting\_date.date()} to {harvest\_date.date()}...")

idx\_growing\_season = (df["DATES"] >= planting\_date) & (df["DATES"] <= harvest\_date)

df\_season = df.loc[idx\_growing\_season, :].reset\_index(drop=True)

# Extract weather inputs

T\_avg = df\_season['T\_DAILY\_AVG'].values

PAR = df\_season['SOLARAD\_DAILY'].values \* 0.48 # Convert solar radiation to PAR

# Convert to PyTorch tensors

T\_tensor = torch.tensor(T\_avg, dtype=torch.float32)

PAR\_tensor = torch.tensor(PAR, dtype=torch.float32)

# Initialize hybrid model with same parameters as base model

print("\nInitializing HybridWheatModel...")

base\_params = {

'Tbase': 4.0,

'Eb': 1.85,

'Eimax': 0.94,

'K': 0.7,

'Lmax': 7.0,

'T1': 900.0,

'alpha': 0.005,

'beta': 0.002

}

hybrid\_model = HybridWheatModel(base\_model\_params=base\_params, hidden\_size=32)

# Run predictions (no training - using initialized weights)

print("Running HybridWheatModel (untrained - should match base model closely)...")

hybrid\_model.eval()

with torch.no\_grad():

pred\_LAI, pred\_biomass, physics\_LAI, physics\_biomass = hybrid\_model(T\_tensor, PAR\_tensor)

# Convert back to numpy

pred\_LAI = pred\_LAI.numpy()

pred\_biomass = pred\_biomass.numpy()

physics\_LAI = physics\_LAI.numpy()

physics\_biomass = physics\_biomass.numpy()

# Calculate final grain yield

HI = 0.45 # Harvest index

grain\_yield\_gm2 = pred\_biomass[-1] \* HI

grain\_yield\_kg\_ha = grain\_yield\_gm2 \* 10

# Calculate GDD for reporting (using numpy for consistency)

GDD = np.maximum(T\_avg - base\_params['Tbase'], 0)

# Add results to dataframe

df\_season['LAI'] = pred\_LAI

df\_season['Biomass'] = pred\_biomass

df\_season['GDD'] = GDD

df\_season['LAI\_physics'] = physics\_LAI

df\_season['Biomass\_physics'] = physics\_biomass

# Calculate corrections (difference between hybrid and pure physics)

LAI\_corrections = pred\_LAI - physics\_LAI

biomass\_corrections = pred\_biomass - physics\_biomass

# Print results

print(f"\n{'='\*60}")

print(f"HYBRID MODEL RESULTS (PyTorch Implementation)")

print(f"{'='\*60}")

print(f"Final above-ground biomass: {pred\_biomass[-1]:.1f} g/m²")

print(f"Grain yield: {grain\_yield\_gm2:.1f} g/m²")

print(f"Grain yield: {grain\_yield\_kg\_ha:.1f} kg/ha")

print(f"Maximum LAI: {pred\_LAI.max():.2f}")

print(f"Total GDD: {GDD.sum():.1f}")

print(f"\nNeural network corrections (untrained):")

print(f" Mean LAI correction: {LAI\_corrections.mean():.6f} (should be ~0)")

print(f" Mean Biomass correction: {biomass\_corrections.mean():.6f} (should be ~0)")

print(f"{'='\*60}\n")

# Save results

df\_season.to\_csv('hybrid\_model\_results.csv', index=False)

print("Results saved to 'hybrid\_model\_results.csv'")

# Generate visualization

print("Generating visualization...")

fig, axes = plt.subplots(4, 1, figsize=(12, 10))

# LAI

axes[0].plot(df\_season['DATES'], pred\_LAI, '-g', linewidth=2, label='Hybrid output')

axes[0].plot(df\_season['DATES'], physics\_LAI, '--', color='gray', linewidth=1, alpha=0.7, label='Physics component')

axes[0].set\_ylabel('Leaf Area Index', fontsize=12)

axes[0].set\_title('Hybrid Model (PyTorch): Physics + NN Corrections', fontsize=14, fontweight='bold')

axes[0].legend(loc='upper right')

axes[0].grid(True, alpha=0.3)

# Biomass

axes[1].plot(df\_season['DATES'], pred\_biomass, '-b', linewidth=2, label='Hybrid output')

axes[1].plot(df\_season['DATES'], physics\_biomass, '--', color='gray', linewidth=1, alpha=0.7, label='Physics component')

axes[1].set\_ylabel('Biomass (g/m²)', fontsize=12)

axes[1].legend(loc='upper left')

axes[1].grid(True, alpha=0.3)

# LAI Corrections (should be near zero for untrained model)

axes[2].plot(df\_season['DATES'], LAI\_corrections, '-', color='orange', linewidth=1)

axes[2].axhline(y=0, color='k', linestyle='--', linewidth=0.5)

axes[2].set\_ylabel('LAI Correction', fontsize=12)

axes[2].set\_title('Neural Network Corrections (untrained ≈ 0)', fontsize=11)

axes[2].grid(True, alpha=0.3)

# Biomass Corrections

axes[3].plot(df\_season['DATES'], biomass\_corrections, '-', color='purple', linewidth=1)

axes[3].axhline(y=0, color='k', linestyle='--', linewidth=0.5)

axes[3].set\_ylabel('Biomass Correction (g/m²)', fontsize=12)

axes[3].set\_xlabel('Date', fontsize=12)

axes[3].grid(True, alpha=0.3)

plt.tight\_layout()

plt.savefig('hybrid\_model\_visualization.png', dpi=300, bbox\_inches='tight')

print("Visualization saved to 'hybrid\_model\_visualization.png'")

plt.close()

print("\nHybrid model simulation complete!")