**🌦️ Weather Forecasting using ConvLSTM with Spatial Attention**

**Aim:**

**To build a deep learning model using ConvLSTM and an optional spatial attention mechanism to forecast future weather conditions (e.g., temperature) based on historical gridded climate data.**

**Description:**

**This project focuses on spatiotemporal weather forecasting using a deep learning approach. The model takes a sequence of 2D weather data frames (e.g., temperature over a geographic region) and predicts the next frame in the sequence. It leverages ConvLSTM layers to handle both spatial and temporal dependencies. Additionally, a custom Spatial Attention layer is optionally included to enhance important regions during training. The dataset is sourced from NetCDF files like ERA5 or NOAA climate reanalysis datasets.**

**Python Program:**

**python**

**CopyEdit**

**import numpy as np**

**import xarray as xr**

**import tensorflow as tf**

**from tensorflow.keras.models import Model**

**from tensorflow.keras.layers import (Input, ConvLSTM2D, BatchNormalization, Conv3D,**

**Layer, Multiply)**

**import matplotlib.pyplot as plt**

**# 1. Load and preprocess data**

**def load\_data(file\_path, var\_name='t2m', time\_steps=10):**

**ds = xr.open\_dataset(file\_path)**

**data = ds[var\_name].values**

**data = (data - np.mean(data)) / np.std(data) # normalize**

**sequences = [data[i:i + time\_steps] for i in range(len(data) - time\_steps)]**

**return np.expand\_dims(np.array(sequences), axis=-1)**

**# 2. Spatial Attention Layer**

**class SpatialAttention(Layer):**

**def \_\_init\_\_(self):**

**super(SpatialAttention, self).\_\_init\_\_()**

**def call(self, inputs):**

**avg\_out = tf.reduce\_mean(inputs, axis=-1, keepdims=True)**

**max\_out = tf.reduce\_max(inputs, axis=-1, keepdims=True)**

**concat = tf.concat([avg\_out, max\_out], axis=-1)**

**score = tf.keras.layers.Conv3D(1, (1, 1, 1), padding='same', activation='sigmoid')(concat)**

**return Multiply()([inputs, score])**

**# 3. Model Architecture**

**def build\_model(input\_shape, use\_attention=False):**

**inputs = Input(shape=input\_shape)**

**x = ConvLSTM2D(32, (3, 3), padding='same', return\_sequences=True)(inputs)**

**x = BatchNormalization()(x)**

**if use\_attention:**

**x = SpatialAttention()(x)**

**x = ConvLSTM2D(16, (3, 3), padding='same', return\_sequences=False)(x)**

**x = BatchNormalization()(x)**

**output = tf.keras.layers.Conv2D(1, (1, 1), activation='linear', padding='same')(x)**

**model = Model(inputs, output)**

**model.compile(optimizer='adam', loss='mse', metrics=['mae'])**

**return model**

**# 4. Train the model**

**def train\_model(model, X, y, epochs=10, batch\_size=4):**

**return model.fit(X, y, epochs=epochs, batch\_size=batch\_size, validation\_split=0.1)**

**# 5. Evaluate the model**

**def evaluate\_model(model, X\_test, y\_test):**

**pred = model.predict(X\_test)**

**mse = np.mean((pred - y\_test)\*\*2)**

**print("MSE on test set:", mse)**

**plt.subplot(1, 2, 1)**

**plt.imshow(y\_test[0, :, :, 0], cmap='coolwarm')**

**plt.title("True")**

**plt.subplot(1, 2, 2)**

**plt.imshow(pred[0, :, :, 0], cmap='coolwarm')**

**plt.title("Predicted")**

**plt.show()**

**# 6. Run the pipeline**

**if \_\_name\_\_ == '\_\_main\_\_':**

**data\_path = 'your\_era5\_or\_noaa\_file.nc'**

**var\_name = 't2m'**

**time\_steps = 10**

**print("Loading data...")**

**X = load\_data(data\_path, var\_name, time\_steps)**

**y = X[:, -1, :, :, :]**

**X = X[:, :-1, :, :, :]**

**print("Building model...")**

**model = build\_model(input\_shape=X.shape[1:], use\_attention=True)**

**print("Training model...")**

**train\_model(model, X, y, epochs=5, batch\_size=2)**

**print("Evaluating model...")**

**evaluate\_model(model, X[-5:], y[-5:])**

**Dataset:**

* **Source: NetCDF climate datasets such as ERA5 or NOAA NCEP Reanalysis**
* **Format: .nc (NetCDF)**
* **Variable used: t2m (2-meter air temperature) or similar**
* **Dimensions: (time, latitude, longitude)**
* **Preprocessing: Standard normalization and sliding window sequence generation**

**Input / Output:**

**Input:**

* **A sequence of 2D weather maps (e.g., past 9 time steps of temperature data) with shape:**

**arduino**

**CopyEdit**

**(batch\_size, time\_steps-1, height, width, 1)**

**Output:**

* **Predicted 2D weather map for the next time step:**

**arduino**

**CopyEdit**

**(batch\_size, height, width, 1)**

**Result:**

**The ConvLSTM model with spatial attention was successfully trained to predict future temperature maps based on past observations. The final Mean Squared Error (MSE) on the test set is printed, and predicted vs. true weather maps are visualized using heatmaps.**