Lake CHLA Prediction Model: Manual Multi-layer LSTM with Lake Embedding

# Overview

This report documents the architecture and rationale for the LSTM-based model used to predict chlorophyll-a (CHLA) concentrations in lakes, using both dynamic environmental variables and static lake parameters. The model is trained using weekly sequences of multi-variable time series data, and it integrates spatial context via learnable lake embeddings.

# Model Architecture

The core model is a manually implemented two-layer LSTM, followed by fully connected layers to process LSTM outputs and lake embeddings. This custom implementation provides better control over temporal processing and offers more flexibility for research.

Key Components:

* • Two manually unrolled LSTM layers (`nn.LSTMCell`) with 64 hidden units each
* • Lake embedding layer (`nn.Embedding`) with 16 dimensions
* • Fully connected layers to fuse temporal and spatial features
* • Final output layer predicts a single CHLA value for each time step

## Model Code Snippet

class LSTMWithLakeEmbeddingFusion(nn.Module):  
 def \_\_init\_\_(self, input\_size, num\_lakes, emb\_dim=16, hidden\_size=64):  
 super().\_\_init\_\_()  
 self.embedding = nn.Embedding(num\_lakes, emb\_dim)  
 self.lstm1 = nn.LSTMCell(input\_size, hidden\_size)  
 self.lstm2 = nn.LSTMCell(hidden\_size, hidden\_size)  
 self.fc\_lstm = nn.Linear(hidden\_size, 32)  
 self.fc\_emb = nn.Linear(emb\_dim, 16)  
 self.final = nn.Linear(48, 1)  
  
 def forward(self, x, lake\_id):  
 batch\_size, seq\_len, \_ = x.size()  
 h\_t1 = torch.zeros(batch\_size, self.lstm1.hidden\_size, device=x.device)  
 c\_t1 = torch.zeros(batch\_size, self.lstm1.hidden\_size, device=x.device)  
 h\_t2 = torch.zeros(batch\_size, self.lstm2.hidden\_size, device=x.device)  
 c\_t2 = torch.zeros(batch\_size, self.lstm2.hidden\_size, device=x.device)  
 for t in range(seq\_len):  
 h\_t1, c\_t1 = self.lstm1(x[:, t, :], (h\_t1, c\_t1))  
 h\_t2, c\_t2 = self.lstm2(h\_t1, (h\_t2, c\_t2))  
 lake\_emb = self.embedding(lake\_id)  
 lstm\_feat = torch.relu(self.fc\_lstm(h\_t2))  
 lake\_feat = torch.relu(self.fc\_emb(lake\_emb))  
 return self.final(torch.cat([lstm\_feat, lake\_feat], dim=1))

# Why Use Lake Embeddings?

Lake embeddings allow the model to learn a latent representation of each lake. This is useful because each lake may have unique spatial or ecological properties that are not fully captured by observable static attributes.

Learned embeddings have been widely used in natural language processing and recommender systems (e.g., Mikolov et al., 2013). In spatial-temporal modeling, entity embeddings allow for encoding categorical inputs such as IDs (e.g., lakes) into a dense feature space which the model can optimize during training.

Reference:

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.

# Training Parameters

* • Input features: 18 (including dynamic variables and seasonal sin/cos)
* • Sequence length: 5 weeks
* • Batch size: 64
* • Optimizer: Adam (lr=0.001)
* • Loss function: MSELoss
* • Training epochs: 20

# Evaluation Metrics

Model evaluation was performed using the test dataset and includes the following metrics:  
- RMSE (Root Mean Squared Error)  
- R² (Coefficient of Determination)  
- rRMSE (Relative RMSE): (RMSE / mean\_true) \* 100

# Output

The model outputs weekly predicted CHLA concentrations. Evaluation metrics such as RMSE, R², and relative RMSE are used to assess performance.