AGRI MAGAZINE
(International E-Magazine for Agricultural Articles)
Volume: 03, Issue: 10 (October, 2026)
Available online at http://www.agrimagazine.in
©Agri Magazine, ISSN: 3048-8656

Deep Learning for Agricultural Prediction: Open-Source Models and Tools

Prabhat Kumar, Santosha Rathod, Prashantkumar. S. Hanjagi, *Sushma. M. Awaji, Suraj Gund, Sudhir Kumar Mishra and R N Singh
ICAR-National Institute of Abiotic Stress Management, Baramati, Maharashtra, India
***Corresponding Author's email: sma1624@gmail.com**

Agriculture generates a lot of data from meteorological stations, soil sensors, satellite imagery, crop experiments, and farm management systems. The data can be used for crop yield prediction, crop growth monitoring, soil moisture estimation and improve irrigation and fertilizer management. Agricultural prediction is typically performed with traditional statistical and machine-learning methods. Agricultural variables, however, often have complex non-linear relationships and change over time. These relationships can be learned from data through deep learning, which is increasingly being applied for agricultural forecasting and monitoring (Oikonomidis et al., 2023; Jabed & Murad, 2024). Various types of agricultural data suit different deep-learning models. Multilayer perceptrons (MLPs) can be used to analyze tabular data, convolutional neural networks (CNNs) can process agricultural images and recurrent neural networks (RNNs) especially Long Short-Term Memory (LSTM) networks can model time-dependent observations. This article describes important deep learning methods for agricultural prediction, popular open source libraries and a simple LSTM implementation for the agricultural time-series data.

Deep Learning for Agricultural Regression

Regression is used to predict continuous numerical values. In agriculture, these values may include crop yield, soil moisture, crop height, temperature, or irrigation requirements.
A deep-learning regression model estimates an output from a set of input variables:

$$\hat{y} = f_{\boldsymbol{\theta}}(\mathbf{x}), \quad (1)$$

where $\mathbf{x}$ represents input features, $\hat{y}$ is the predicted value, f is the neural-network function, and $\boldsymbol{\theta}$ represents the model parameters.
Agricultural input variables may include rainfall, temperature, humidity, soil moisture, soil nutrients, fertilizer application, and vegetation indices. The target variable and input features should be selected according to the prediction objective.
Agricultural data frequently have a temporal structure. For example, crop yield depends not only on weather conditions on a particular day but also on conditions throughout the growing season. Similarly, soil moisture can depend on previous rainfall, irrigation, temperature, and evaporation. Models that learn temporal dependencies can therefore be useful for agricultural forecasting.

Deep-Learning Models Used in Agriculture

Multilayer Perceptron (MLP)

An MLP is a feed-forward neural network that learns relationships between input features and a target variable. It can be applied to agricultural datasets containing soil characteristics, seasonal weather summaries, fertilizer quantities, and crop-management information.
MLPs are useful for tabular regression problems. However, they do not explicitly represent the order of sequential observations unless temporal information is incorporated into their inputs.

Convolutional Neural Network (CNN)

CNNs are designed to learn spatial patterns from image data. In agriculture, they are used to analyze satellite images, drone imagery, and crop photographs.

Typical applications include crop classification, disease identification, weed detection, and vegetation monitoring. CNNs are particularly useful when visual characteristics provide important information about crop conditions.

Recurrent Neural Networks and LSTM

Recurrent neural networks process sequential observations and maintain information from earlier time steps. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network designed to learn temporal dependencies more effectively in many sequence-learning tasks.

LSTMs are suitable for agricultural time-series data, including daily weather observations, soil-moisture measurements, crop-growth records, and vegetation-index sequences.

For example, an LSTM can use rainfall, temperature, humidity, and previous soil-moisture measurements to predict future soil moisture. It can also be adapted for crop-yield forecasting when the input sequences and harvest-level targets are correctly aligned.

Gated Recurrent Units (GRUs) provide an alternative recurrent architecture with a different gating mechanism. The performance of LSTM and GRU models depends on the dataset, prediction objective, and model configuration.

Temporal CNNs and Transformers

Temporal convolutional networks learn patterns across sequences using convolutional operations. Transformers use attention mechanisms to model relationships between observations at different time steps.

Hybrid approaches can combine CNNs for satellite imagery with LSTMs for weather or vegetation-index sequences. Such models may be useful when spatial and temporal information are both important, although they generally require more development and computational resources.

Open-Source Libraries and Tools

Several Python libraries support agricultural data preparation, deep-learning model development, and evaluation.

Library or tool	Purpose
Python	Main programming language for agricultural data analysis and modeling.
Jupyter Notebook	Interactive environment for running code, documenting experiments, and visualizing results.
Google Colab	Cloud-based notebook environment that may provide GPU access.
NumPy	Numerical calculations and array operations.
pandas	Reading CSV files, managing dates, and processing agricultural datasets.
Matplotlib	Plotting weather trends and comparing observed and predicted values.
scikit-learn	Data preprocessing, scaling, baseline models, and evaluation metrics.
PyTorch	Developing and training LSTM and other neural-network models.
TensorFlow/Keras	An alternative framework for implementing deep-learning models.
AgML	An open-source framework for machine learning workflows in agriculture.

For the practical example in this article, the main libraries are **pandas, NumPy, scikit-learn, Matplotlib, and PyTorch**.

The required Python libraries can be installed using:

```
pip install numpy pandas matplotlib scikit-learn torch
```

Jupyter Notebook can be installed separately if needed:

```
pip install notebook
```

These tools can be used in a local Python environment or in Google Colab. GPU support depends on the available hardware and the installed software configuration.

A Basic LSTM Implementation for Agricultural Data

Consider an agricultural dataset containing daily rainfall, temperature, humidity, and soil-moisture observations. The objective is to use recent observations to predict soil moisture at the next time step.

The following example illustrates the essential components of an LSTM model using PyTorch. It assumes that the dataset is stored in agricultural_data.csv and contains the columns rainfall, temperature, humidity, and soil_moisture.

Data Preparation and Model Training

```
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
from sklearn.preprocessing import StandardScaler

# Load agricultural data
df = pd.read_csv("agricultural_data.csv")
features = [
    "rainfall", "temperature",
    "humidity", "soil_moisture"
]
df = df[features].dropna()

# Scale the data
scaler = StandardScaler()
data = scaler.fit_transform(df).astype(np.float32)

# Create sequences of 14 observations
X, y = [], []
target_idx = features.index("soil_moisture")

for i in range(14, len(data)):
    X.append(data[i-14:i])
    y.append(data[i, target_idx])

X = torch.tensor(np.array(X))
y = torch.tensor(np.array(y)).float().reshape(-1, 1)

# Define a basic LSTM model
class AgriculturalLSTM(nn.Module):
    def __init__(self, input_size):
        super().__init__()
        self.lstm = nn.LSTM(
            input_size, 32, batch_first=True
        )
        self.output = nn.Linear(32, 1)

    def forward(self, x):
        sequence, _ = self.lstm(x)
        return self.output(sequence[:, -1, :])

model = AgriculturalLSTM(len(features))
```

```
optimizer = torch.optim.Adam(
    model.parameters(), lr=0.001
)
loss_function = nn.MSELoss()

# Basic training loop
for epoch in range(50):
    model.train()
    optimizer.zero_grad()
    predictions = model(X)
    loss = loss_function(predictions, y)
    loss.backward()
    optimizer.step()
print("Training completed.")
```

This example demonstrates the basic structure of an LSTM workflow: reading agricultural data, preparing sequences, defining a recurrent neural network, and training it by minimizing mean squared error.

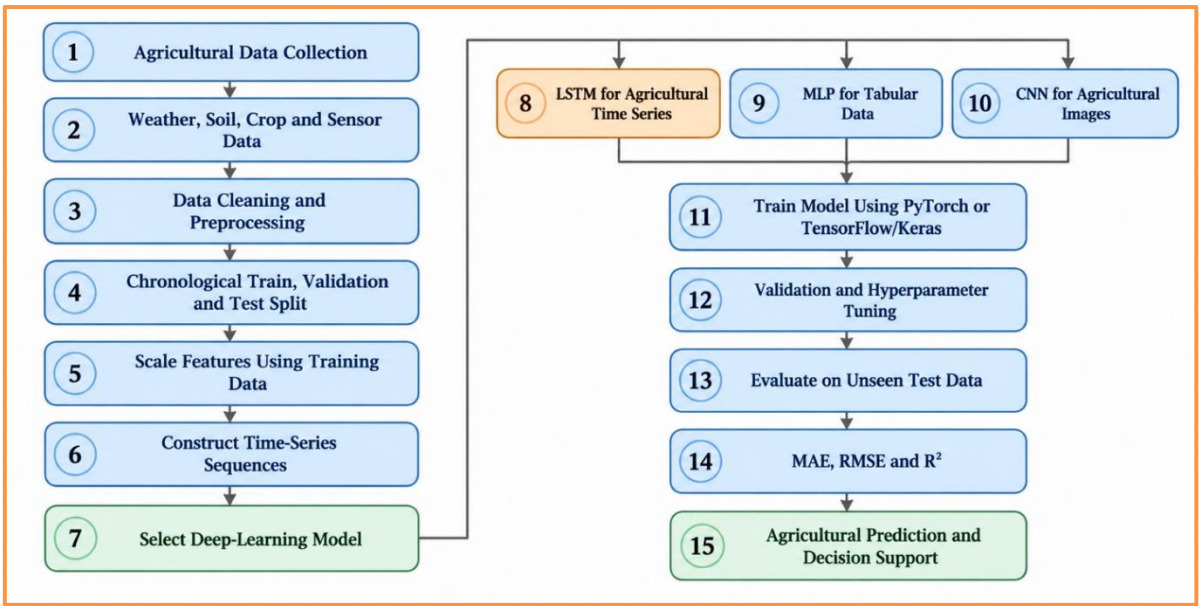


Figure 1. Workflow of Open-Source Deep Learning for Agricultural Prediction

Figure 1 illustrates the workflow of open-source deep learning for agricultural prediction. The process begins with the collection and preprocessing of weather, soil, crop, and sensor data. The dataset is divided chronologically into training, validation, and test sets, after which the features are scaled and time-series sequences are constructed where required. An appropriate model is selected according to the data type: LSTM for agricultural time series, MLP for tabular data, or CNN for agricultural images. The selected model is trained using an open-source deep-learning framework, such as PyTorch or TensorFlow/Keras, and evaluated using unseen test data. Performance metrics, including MAE, RMSE, and R^2, help assess predictive accuracy before the model is considered for agricultural forecasting and decision support.

Note: The code is an introductory illustration rather than a complete evaluation pipeline. For a research application, the data should be split chronologically into training, validation, and test sets before fitting the scaler or training the model. Predictions should then be evaluated on unseen observations.

The example uses the previous 14 observations to predict the next soil-moisture value. For daily measurements, this represents a 14-day input window. The sequence length should be adjusted to suit the data frequency and agricultural process.

If crop yield is the target, the data must be organized according to the crop season and harvest records. Seasonal yield should not be treated as a daily target unless the study design supports that interpretation.

Model Evaluation

Model evaluation is necessary to determine whether a deep-learning approach provides useful predictions. Common regression metrics include mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and the coefficient of determination (R^2).

MSE measures the average squared difference between observed and predicted values:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{2}$$

where y_i is the observed value, $\hat{y}_i$ is the predicted value, and n is the number of evaluated observations.

RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \tag{3}$$

MAE measures the average absolute prediction error:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{4}$$

The coefficient of determination is

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{5}$$

where $\bar{y}$ is the mean of the observed values.

The metrics should be interpreted in relation to the agricultural problem. For instance, soil-moisture errors should be interpreted in the target's original units, while crop-yield errors may be reported in tonnes per hectare. A negative R^2 indicates that the model performed worse than a mean-based reference on the evaluated data.

Challenges and Practical Considerations

Agricultural deep learning presents several challenges.

Data quality: Agricultural datasets may contain missing observations, sensor errors, irregular sampling, and inconsistent measurement units. These issues should be checked before model development.

Data leakage: For time-series prediction, random splitting can allow future observations to influence training. Data should generally be divided chronologically, and preprocessing parameters should be estimated using training data only.

Generalization: Weather, soil, crop varieties, and management practices differ across locations and seasons. Testing on a separate season, field, or region can help assess whether the model generalizes to new conditions.

Baseline comparison: An LSTM should be compared with suitable alternatives, such as persistence forecasting, linear regression, random forests, or an MLP. A more complex model is not necessarily more accurate.

Interpretation: Agricultural decisions can have economic and environmental consequences. Prediction errors, uncertainty, and the intended use of the model should be considered before deploying it in operational farming systems.

Conclusion

Deep learning offers useful approaches for agriculture prediction including MLPs for tabular data, CNNs for imagery and LSTM networks for sequential observations . LSTMs are especially applicable when past weather, soil or crop measurements include information about future agricultural conditions. In this article, we presented the main model types, reviewed open source libraries, and provided a simple PyTorch LSTM example for agricultural time-series data. A comprehensive agricultural forecasting study should also include chronological data splitting, appropriate preprocessing, comparison to baseline methods, and evaluation on unseen observations. Deep learning models, with sufficient data and careful validation, could help in agricultural monitoring, forecasting and data-driven management decisions.

References

1. Jabed, M. A., & Murad, M. A. (2024). Crop yield prediction in agriculture: A comprehensive review of machine learning and deep learning approaches, with insights for future research and sustainability. *Heliyon, 10*(24), e40836. https://doi.org/10.1016/j.heliyon.2024.e40836
2. Oikonomidis, A., Catal, C., & Kassahun, A. (2023). Deep learning for crop yield prediction: A systematic literature review. *New Zealand Journal of Crop and Horticultural Science, 51*(1), 1–26. https://doi.org/10.1080/01140671.2022.2032213
3. PyTorch. (n.d.). *PyTorch tutorials*. https://docs.pytorch.org/tutorials/
4. scikit-learn developers. (n.d.). *Model evaluation: Quantifying the quality of predictions*. https://scikit-learn.org/stable/modules/model_evaluation.html
5. TensorFlow. (n.d.). *TensorFlow tutorials*. https://www.tensorflow.org/tutorials
6. van Klompenburg, T., Kassahun, A., & Catal, C. (2020). Crop yield prediction using machine learning: A systematic literature review. *Computers and Electronics in Agriculture, 177*, 105709. https://doi.org/10.1016/j.compag.2020.105709