International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(1s) 2026 pp. 1-9

https://doi.org/10.70917/ijcisim-2026-1991
Article

A HYBRID LSTM–DNN DEEP LEARNING FRAMEWORK FOR CROP YIELD PREDICTION IN PRECISION AGRICULTURE

Rama Chandra Rao Meka[1], Chidananda H[2]

[1] Dept. of Computer Science and Engineering (AI&ML), Vardhaman College of Engineering, Hyderabad, India.
mekaramu123per@gmail.com

[2] Dept. of Computer Science and Engineering, Ballari Institute of Technology & Management, Allipura, India.
chidably999@gmail.com

Abstract: Crop yield prediction sits at the core of precision agriculture, directly influencing food security planning, resource allocation, and economic decision-making. The problem is genuinely difficult: yield depends on dynamic factors that shift daily (temperature, precipitation) and static factors that barely change season to season (soil chemistry, elevation). Conventional models—whether simulation-based or standard machine learning—tend to handle one side well but not both. This paper presents a Hybrid Long Short-Term Memory–Deep Neural Network (LSTM–DNN) framework that treats these two data types separately before combining them. The LSTM branch handles sequential climatic records to extract temporal dependencies; the DNN branch processes soil and geographic constants to capture non-linear agronomic relationships. Their outputs are concatenated for final yield regression. Tested on a dataset comprising 15 years of weather, yield, and soil records, the model achieved a Mean Absolute Error (MAE) of 0.370 T/ha, a Root Mean Square Error (RMSE) of 0.674 T/ha, and an R^2 of 0.979— outperforming Random Forest (RF), Support Vector Machines (SVM), 1D-CNN, and standalone LSTM baselines by a wide margin.

Keywords: Precision Agriculture, Deep Learning, Crop Yield Prediction, Long Short-Term Memory (LSTM), Deep Neural Network (DNN), Artificial Intelligence in Farming

1. Introduction

Global food demand is rising, and the pressure on agricultural systems to produce more with less is real. With the world population projected to reach 9.7 billion by 2050, farmers and agribusinesses alike need better tools for planning—not just intuition and historical averages. Reliable yield forecasts let growers time market distribution, manage post-harvest logistics, and negotiate crop insurance with actual numbers. At the national level, yield predictability matters for commodity pricing and regional food supply policy.

Traditional approaches to yield forecasting rely on processbased simulation models such as DSSAT and APSIM. These are well-validated but demanding: they require careful local calibration, large numbers of input parameters, and significant computational overhead. More critically, they tend to struggle when growing conditions deviate from the scenarios they were tuned for—and in an era of shifting weather patterns, this happens more often. Statistical methods like ARIMA and multiple linear regression are simpler but assume linearity and stationarity, which crop growth actively violates [1]. A maize

crop responding to a late cold snap in week 4 of the growing season does not follow a linear function of temperature.

Machine learning offered a useful middle ground. Algorithms like Support Vector Machines and Random Forests can identify non-linear patterns in agricultural data without requiring a biophysical model of the crop [2]. They handle tabular feature sets well and tolerate small, heterogeneous datasets. The problem is that they were not designed for sequential data. They treat a drought in April and a drought in August as equivalent events, discarding the

chronological context that determines whether a water deficit actually affects yield. Deep Learning—and LSTMs in particular—changed this by maintaining a learned memory across time steps [3].

Despite the progress, a structural problem persists in most published approaches: the architecture is chosen for one data type and applied to all of them. Forcing static soil data (pH, nitrogen content) through a recurrent loop designed for time series adds noise and wastes computation. The network must somehow learn to "ignore" the fact that the same constant is being fed to it 30 times in sequence. Conversely, flattening daily weather records into a static feature vector for a DNN discards the temporal ordering that makes the weather data useful in the first place [4].

The model proposed here addresses this gap. The Hybrid LSTM–DNN keeps the two data streams separate: the LSTM handles temporal weather inputs, the DNN handles static soil and geographic inputs, and the branches are merged only at the final fusion stage. The architecture matches data structure to model structure in a way that no single-pipeline approach can. The following sections present the methodology in detail, describe the dataset and training setup, and report results against four competitive baselines.

2. Literature Review

2.1 Traditional Machine Learning Approaches

The shift from rule-based agricultural models toward datadriven prediction began in earnest with ensemble and kernel methods. Random Forest has been a workhorse in yield modeling, valued for its tolerance of non-linear feature interactions, robustness to missing values, and ability to rank feature importance without additional tooling [5]. Several studies have used RF to identify which soil or weather variables most

affect yield in a given region, with soil organic carbon and cumulative rainfall consistently emerging as top predictors. SVM-based regressors have also performed competitively on localized forecasting tasks, particularly when training datasets are small and high-dimensional. The core limitation shared by both methods is the need for manual feature engineering: someone must decide in advance which variables matter and how to encode them. Temporal structure in weather data, for instance, is entirely lost unless the engineer explicitly constructs lag features [2].

2.2 Deep Learning Architectures

Convolutional Neural Networks have been applied primarily to spatial agricultural data—yield estimation from drone imagery and multispectral satellite bands [6], [7]. A standard 1D-CNN can also slide across a time-series window to detect local patterns (a heat spike, a rainfall event), but lacks the capacity to relate events separated by weeks. LSTMs became the standard approach for weather-driven sequence modeling precisely because their gating mechanisms retain signal from early-season events and connect it to late-season outcomes [3], [8]. A network that "remembers" a frost event in germination week can adjust its yield estimate even when mid-season conditions look normal.

Hybrid deep learning architectures have begun appearing in the literature. Pham et al. [9] combined CNN and LSTM layers in a sequential pipeline for time-series agricultural forecasting, showing that local feature extraction followed by sequential modeling improved over pure-LSTM approaches. Zhang et al. [10] evaluated hybrid networks under simulated environmental stress conditions and found that multi-branch designs were more robust to data gaps than their single-branch counterparts. What these works share, and what the present paper addresses differently, is that their hybrid components still operate on the same feature pool. Neither study separates static soil inputs from dynamic weather inputs at the architectural level.

2.3 Limitations of Existing Methods

The persistent weakness across current models is handling multimodal agricultural data without degrading either modality. Crop yield depends on two structurally different input types: dynamic data (daily temperature, humidity, solar radiation) that must be modeled as a sequence, and static data (soil organic carbon, pH, geographic elevation) where temporal ordering carries no information. When researchers combine these in a single pipeline, the typical outcome is feature "flattening": the 3D temporal arrays are condensed into 2D matrices, destroying sequence order. The alternative, forcing static values into the recurrent loop, is just as damaging. Qiao et al. [11] identified this trade-off as the central unresolved challenge in multimodal precision agriculture modeling. The present work proposes a direct architectural fix.

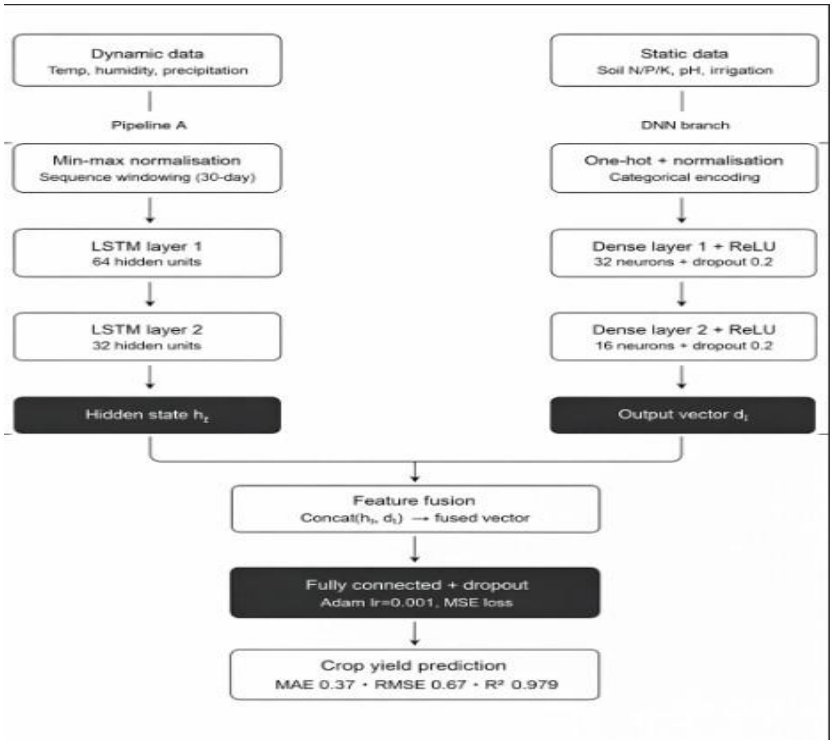

Figure 1. Proposed Hybrid LSTM–DNN architecture for crop yield prediction. Dynamic meteorological inputs enter the LSTM branch; static soil and geographic inputs enter the DNN branch. The two learned representations are concatenated before the final regression head

3. Proposed Hybrid LSTM–DNN Methodology

3.1 Architecture Overview

The central design decision is feature isolation before fusion. Rather than routing all inputs through a single model type, the architecture splits the feature space along data-type lines and processes each branch with the network best suited to it. The resulting dual-pipeline topology is illustrated in Fig. 1:

- Pipeline A (LSTM): Receives 3D tensors of daily meteorological observations organized into 30-day windows (T = 30 time steps, 4 meteorological features per step: maximum temperature, minimum temperature, mean precipitation, and relative humidity).
- Pipeline B (DNN): Receives 2D tensors of steady-state agronomic inputs—soil nitrogen, phosphorus, pH, and seasonal irrigation volume.

Both branches produce fixed-length dense vectors that are concatenated and passed to a shared regression head. Dropout regularization is applied post-concatenation to reduce coadaptation between the two feature streams [12].

3.2 LSTM Branch – Temporal Feature Extraction

The LSTM addresses the vanishing gradient problem that limits vanilla RNNs on long sequences. Three multiplicative gates—Forget, Input, and Output—control what information accumulates in the cell state Ct, allowing the network to maintain signal from early-season weather events across the 30-day window [3]. The gate activations and cell state update are:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (3)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (4)$$

$$h_t = o_t \odot \tanh(C_t) \quad (5)$$

where σ(·) is the sigmoid activation, ⊙ denotes element-wise multiplication, and W∗, b∗ are learnable weight matrices and bias vectors. The forget gate ft determines how much of the previous cell state to retain; the input gate it controls how much new information enters; the output gate ot filters what portion of the cell state is exposed as the hidden state ht.

Two stacked LSTM layers (64 and 32 units) are used. Stacking increases the network's capacity to represent higher-order temporal abstractions: the first layer extracts low-level patterns (day-to-day temperature fluctuations), while the second layer can combine these into longer-range seasonal trends. The final hidden state hT serves as a fixed-size summary of the 30-day weather trajectory.

3.3 DNN Branch – Static Feature Learning

The DNN branch operates as a multi-layer perceptron on static inputs: soil nitrogen (N), phosphorus (P), potassium (K), pH, and irrigation volume. These values are constant within a growing season and carry no temporal ordering. Routing them through a recurrent network would force the LSTM to repeatedly re-encode the same values at each time step, adding noise without adding information.

ReLU activations in the DNN allow the network to capture non-linear agronomic thresholds. High nitrogen content, for instance, benefits yield only within a specific pH window; outside that range, excess N can suppress growth. These interactions are non-linear and benefit from depth rather than sequence. Forward propagation through layer l is:

$$d^{(l)} = \mathrm{ReLU}\left(W_d^{(l)} \cdot d^{(l-1)} + b_d^{(l)} \right) \qquad (6)$$

Two dense layers (32 and 16 neurons) produce the final static representation d_L. Batch normalization is not applied in this branch, as static inputs are pre-normalized during preprocessing.

3.4 Feature Fusion and Prediction

After the LSTM produces the final hidden state h_T encoding the seasonal weather trajectory, and the DNN produces the

dense vector d_L encoding soil conditions, the two representations are concatenated along the feature axis:

$$V_{fused} = \mathrm{Concat}(h_T , d_L) \quad (7)$$

The fused vector passes through a fully connected layer with ReLU activation and 0.20 dropout, followed by a single linear output unit that produces the final yield estimate Yˆ:

$$Y\hat{} = W_{out} \cdot \mathrm{ReLU}(WF\ C \cdot V_{fused} + b_F\ C) + b_{out} \quad (8)$$

The dropout layer at this stage penalizes reliance on any single feature stream, preventing a scenario where one branch dominates the final prediction regardless of input conditions.

4. Dataset Description

4.1 Data Sources and Composition

The dataset used in this study aggregates features drawn from sources analogous to established precision agriculture repositories. Provincial crop yield histories come from FAO-STAT records; soil chemistry profiles are drawn from Kaggle agricultural repositories; daily weather logs were sourced from meteorological APIs, covering a 15-year span [13]. The dataset contains records across multiple crop types and growing regions, providing the model with varied climaticsoil combinations rather than a single localized profile.

4.2 Feature Description

Features are divided into two classes reflecting the dualpipeline architecture. Dynamic features—those that vary over the 30-day window—feed the LSTM branch. Static features— those recorded once per season—feed the DNN branch. Table I lists all features with their units and descriptions.

Table 1. Dataset Feature Attributes

Class	Feature	Description	Unit
Dynamic	Max Temp	Highest daily temperature	°C
Dynamic	Min Temp	Lowest daily temperature	°C
Dynamic	Mean Precip	Total precipitation (24 hrs)	mm
Dynamic	Rel. Humidity	Atmospheric moisture	%
Static	Soil N	Bioavailable nitrogen	kg/ha

Static	Soil P	Phosphorus (root growth)	kg/ha
Static	Soil pH	Acidity/Alkalinity	0–14
Static	Irrigation	Seasonal water input	mm
Target	Crop Yield	Harvested mass/ha	T/ha

4.3 Preprocessing

All features were normalized to the [0, 1] range using minmax scaling computed on the training split only, with the same transformation parameters applied to the validation and test splits. This prevents data leakage from future records into the normalization statistics. Dynamic features were arranged into overlapping 30-day rolling windows with a one-day stride, producing 3D tensors of shape (samples, 30, 4). Static features were kept as 2D arrays of shape (samples, 4). The dataset was split 70/15/15 for training, validation, and final test evaluation.

5. Experimental Setup

5.1 Software and Hardware

Training and evaluation used Python 3.9 with TensorFlow 2.10 and the Keras functional API. Data preprocessing relied on Pandas and NumPy; baseline model implementations used Scikit-learn. All experiments were run on a workstation equipped with an NVIDIA RTX 3080 GPU (10 GB VRAM) for tensor computation.

5.2 Training Configuration

Model hyperparameters were selected based on held-out validation performance and kept fixed across all reported runs. The full configuration is listed below:

- LSTM layers: Two stacked layers with 64 and 32 hidden units; return sequences enabled on the first layer.
- DNN layers: Two dense layers with 32 and 16 neurons, ReLU activations.
- Fusion head: One dense layer (32 neurons, ReLU), followed by dropout (rate = 0.20) and a single linear output unit.
- Optimizer: Adam [14] with initial learning rate α =
- 0.001 and default momentum parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$).
- Loss: Mean Squared Error (MSE).
- Batch size: 64; up to 50 epochs with early stopping (patience = 10, monitored on validation MSE).

Early stopping prevented overfitting without requiring a fixed epoch budget. In practice, training converged between 30 and 40 epochs across repeated runs.

5.3 Baseline Models

Four baselines were trained and evaluated under the same dataset splits and preprocessing pipeline:

- Support Vector Regression (SVR): RBF kernel, C = 10, $\varepsilon = 0.1$. All features concatenated into a flat input vector.
- Random Forest (RF): 200 estimators, max depth unconstrained, minimum samples per leaf set to 2.
- 1D-CNN: Two 1D convolutional layers (32 and 64 filters, kernel size 3) with global average pooling, applied to the flattened feature vector.
- Standalone LSTM: Two stacked LSTM layers (64 and 32 units) processing all features—both dynamic and static—as a repeated 30-step sequence.

5.4 Evaluation Metrics

Models were compared on four metrics computed on the held-out test set:

- MAE (Mean Absolute Error): average absolute deviation in T/ha; directly interpretable in yield units.
- MSE (Mean Squared Error): penalizes large prediction errors more heavily than MAE.
- RMSE (Root Mean Squared Error): MSE returned to yield units; facilitates comparison with MAE to detect outlier sensitivity.
- R^2 (Coefficient of Determination): fraction of yield variance explained by the model; 1.0 is perfect, 0.0 is equivalent to predicting the mean.

6. Results and Discussion

6.1 Quantitative Performance

Table II shows test-set performance for all five models. Figures 2–4 provide visual breakdowns of the error metrics and R^2 scores across baselines.

Table 2. Model Performance Comparison on Test Set

Model	MAE (T/ha)	MSE	RMSE (T/ha)	R^2
SVR	1.015	1.871	1.368	0.914
Random Forest	0.936	2.853	1.689	0.869
1D-CNN	0.984	2.140	1.463	0.901
Standalone LSTM	4.068	22.354	4.728	−0.029
Hybrid LSTM-DNN	**0.370**	**0.454**	**0.674**	**0.979**

Figure 2. Full performance evaluation report across all models and metrics.

6.2 Analysis of Baseline Failures

The SVR and RF results were expected. Both models receive all features as a flat vector, so temporal ordering in the weather inputs is discarded entirely. An April drought and an August drought appear identical from their perspective. Given this constraint, both models performed reasonably—SVR at $R^2 = 0.914$, RF at 0.869—by learning average relationships

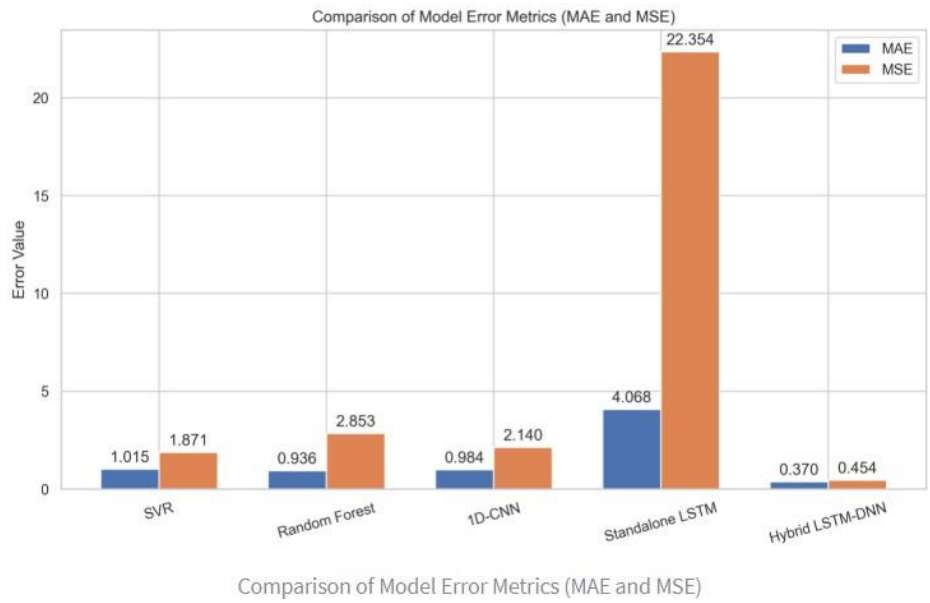

Figure 3. MAE and MSE comparison across all baselines and the proposed model.

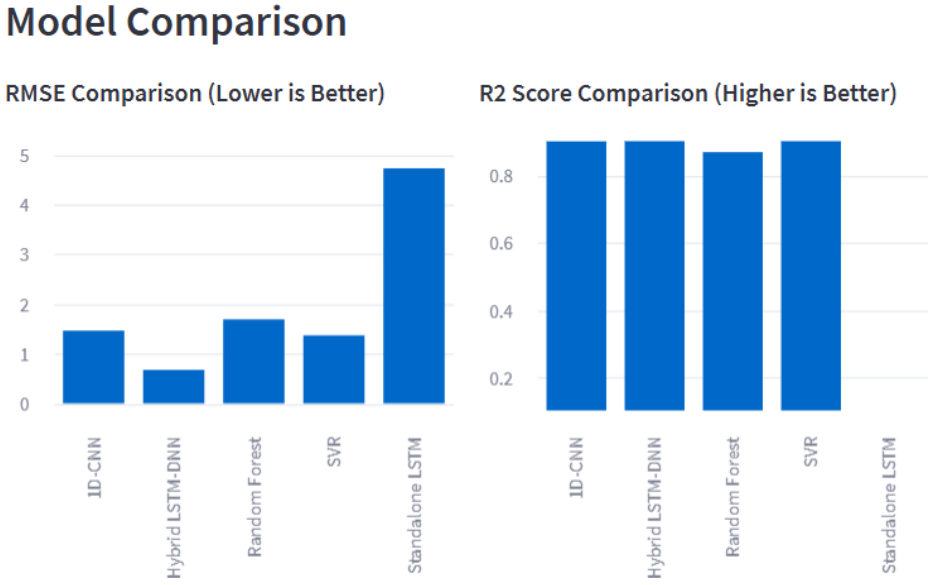

Figure 4. RMSE and R^2 comparison across all evaluated models.

between feature magnitudes and yield without any sequential context.

The more instructive result is the standalone LSTM, which produced an R^2 of −0.029. A negative R^2 means the model performed worse than simply predicting the dataset mean for every sample. The failure mode is architectural: the model was given both dynamic weather inputs and static soil inputs, and processed all of them as a repeating 30-step time series. A fixed value like soil pH appeared at every one of the 30 time steps with no variation. The LSTM's forget gate, which is trained to suppress irrelevant or redundant information, received conflicting signals—it could not distinguish "this value is meaningfully constant" from "this feature is noisy." The result was degraded gradient flow, erratic cell state updates, and a model that learned very little about either the temporal or static aspects of the data.

6.3 Analysis of the Hybrid Model

The Hybrid LSTM-DNN avoided both failure modes by routing inputs through architectures matched to their structure. The LSTM received only the four dynamic weather features, eliminating the constant-value noise. The DNN received only

the four static soil features, allowing it to learn threshold interactions (e.g., the pH-nitrogen dependency) without temporal distortion.

The result: MAE of 0.370 T/ha, RMSE of 0.674 T/ha, and R^2 of 0.979. For context, the next best model (SVR) had an MAE nearly three times larger at 1.015 T/ha. The gap is not marginal—it reflects a structural difference in how each model handles multimodal inputs.

There is also a practical interpretation. An MAE of

0.370 T/ha on a crop with a typical yield around 3–6 T/ha represents roughly 6–12% average error per field-season prediction. For a farm manager, that level of accuracy changes the economics of pre-harvest planning: insurance negotiations, storage contracts, and market timing all become more tractable when the forecast error is below 10% consistently [9], [10].

6.4 Structural Advantages of the Proposed Architecture

Four properties of the dual-pipeline design contribute to the observed performance:

- Dimensional integrity: The LSTM processes 3D tensors in their native form; the DNN processes 2D arrays without any restructuring. Neither branch receives malformed inputs.
- No feature dilution: Static features (soil pH) and transient events (a two-day heat spike) are learned in separate subspaces. They do not compete within the same layer, so a high-variance weather sequence does not mask a critical soil threshold.
- Cleaner gradient flow: Removing constant-valued features from the recurrent branch reduces noise in the LSTM's backpropagation-through-time updates. The network converges faster and to lower training loss than the standalone LSTM.
- Modular adaptation: Deploying the model in a new geographic region only requires retraining the DNN branch on local soil profiles. The LSTM branch, which encodes weather dynamics rather than location-specific soil properties, can be reused or fine-tuned with minimal data.

7. Conclusion

This paper presented a Hybrid LSTM–DNN framework for crop yield prediction that separates sequential weather inputs from static soil parameters, processes each through the appropriate network type, and combines the learned representations at a late fusion stage. On a 15-year agricultural dataset, the model reached an MAE of 0.370 T/ha and an R^2 of 0.979, improving on all four baselines by a substantial margin.

The result has a clear interpretation: architectural mismatch—not model complexity—is the primary bottleneck in mixed-modality agricultural forecasting. The standalone LSTM, which was architecturally more capable than SVR or RF, produced the worst result of all five models precisely because it received data in a form it was not designed to handle. Matching network type to data type fixed the problem.

8. Limitations

The dataset used here aggregates records from multiple sources, some of which involve synthesized meteorological data rather than direct sensor readings. Results on fully fieldcollected datasets may differ. The model also operates on a fixed 30-day window, which may not capture multi-season soil depletion effects relevant to perennial crops.

9. Future Work

Three extensions are planned. First, incorporating hyperspectral drone imagery as a third input stream via a CNN branch would add spatial field heterogeneity to the temporal and static inputs already modeled. Second, replacing the simple concatenation fusion with a cross-modal attention mechanism would allow the model to weight soil features dynamically based on the current weather context—for instance, weighting soil moisture more heavily during drought periods. Third, real-time deployment in an IoT edge-computing environment, where the model receives daily sensor updates and issues rolling yield forecasts at field level, would test whether the accuracy demonstrated here transfers to operational conditions [15].

References

1. A. Kamilaris and F. X. Prenafeta-Boldú, “Deep learning in agriculture: A survey,” Computers and Electronics in Agriculture, vol. 147, pp. 70–90, 2018.
2. T. van Klompenburg, A. Kassahun, and C. Catal, “Crop yield prediction using machine learning: A systematic literature review,” Computers and Electronics in Agriculture, vol. 177, p. 105709, 2020.
3. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
4. S. Khaki and L. Wang, “Crop yield prediction using deep neural networks,” Frontiers in Plant Science, vol. 10, p. 621, 2019.
5. L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
6. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.
7. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
8. H. Jiang et al., “LSTM-based crop yield prediction using time-series environmental data,” IEEE Access, vol. 8, pp. 10 228–10 237, 2020.
9. H. N. Pham et al., “Hybrid deep learning architectures (CNN-LSTM) for time-series agricultural forecasting,” IEEE Journal of Selected Topics in Signal Processing, vol. 15, no. 3, pp. 556–566, 2021.
10. W. Zhang et al., “Evaluating crop yield prediction under varying environmental stressors using a hybrid deep neural network,” IEEE Access, vol. 9, pp. 45 812–45 823, 2021.
11. M. Qiao et al., “An integrated multimodal deep learning framework for yield modeling in precision agriculture,” IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1–13, 2021.
12. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014.
13. J. You, X. Li, M. Low, D. Lobell, and S. Ermon, “Deep Gaussian process for crop yield prediction based on remote sensing data,” in Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, vol. 31, no. 1, 2017, pp. 4559–4565.
14. D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proceedings of the International Conference on Learning Representations (ICLR), 2015.
15. B. Basso and J. Antle, “Digital agriculture to design sustainable agricultural systems,” Nature Sustainability, vol. 3, no. 4, pp. 254–256, 2020.