Disha Mudgal, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

AgriCare A Hybrid Cloud-Edge AI Framework for Context-Aware Crop Recommendation in Flutter

Disha Mudgal[1], Amey Bhogle[2], Dr. Jasbir Kaur[3], Ifrah Kampoo[4]

[1,2]Student of Master of Computer Applications (MCA), Guru Nanak Institute of Management Studies, Matunga, Mumbai, India

[3]Director, GNIMS B-School, Head of Information Technology and HR, Guru Nanak Institute of Management Studies, Matunga, Mumbai, India

[4]Assistant Professor, Guru Nanak Institute of Management Studies, Matunga, Mumbai, India,

Abstract- Smallholder farmers often make crop decisions with incomplete and delayed information about soil, season, and weather. AgriCare addresses this gap through a Flutter-based mobile application that combines cloud reasoning with on-device machine learning. The proposed system first attempts to generate structured crop recommendations using a Groq-hosted Llama 3.3 70B model, enriched with live weather and reverse-geocoded location context. If the cloud path fails because of poor connectivity, an API error, or a timeout, the application automatically falls back to a TensorFlow Lite classifier stored on the device. This hybrid design keeps the application useful in rural conditions where mobile data may be inconsistent, while still offering richer explanations whenever cloud inference is available. Prototype evaluation shows that the local TFLite path returns predictions in under 100 ms, while the cloud path produces more detailed ranked crop cards within a few seconds on stable connectivity. The paper presents the architecture, data flow, mathematical preprocessing, evaluation results, limitations, and future improvements of AgriCare as a practical agricultural advisory system.

Keywords: AgriCare, Flutter, crop recommendation, TensorFlow Lite, edge AI, smart farming, weather-aware advisory.

I. INTRODUCTION

Agriculture remains central to India's livelihood and economy. The Ministry of Agriculture reports that agriculture and allied sectors employ a major share of the workforce and continue to contribute substantially to national value addition [1]. At the same time, many farm decisions are still made with limited field-specific guidance. A farmer may know the broad season and soil type, but may not have timely support for connecting these factors with rainfall, temperature, water availability, and crop economics.

This challenge is especially significant for small and marginal farmers because they have less room for experimentation. The Agriculture Census shows that small and marginal holdings form the overwhelming majority of operational holdings in India [2]. A wrong crop choice can therefore affect not only yield, but also household cash flow, debt pressure, and confidence in future planting decisions.

Digital advisory tools can help, but only if they are designed for the realities of rural use. A purely cloud-based app may look impressive in a classroom demonstration, yet fail in the field when mobile data is unavailable. TRAI's telecom performance reports continue to show a clear rural-urban gap in wireless teledensity [3]. AgriCare is designed around this constraint: it uses cloud AI when available, but it does not depend on it for the app to remain useful.

A. Problem Statement

The project focuses on the following research problem: how can a mobile agricultural advisory system provide context-aware crop recommendations while remaining usable under intermittent connectivity? The answer proposed in this paper is a hybrid cloud-edge architecture built with Flutter, a cloud LLM service, and an on-device TensorFlow Lite fallback model.

B. Contributions

- A Flutter mobile architecture for crop recommendation using region, soil type, season, and live weather context.

- A hybrid inference flow that attempts cloud-based LLM recommendations first and switches to local TFLite inference if the cloud path fails.
- A structured JSON response format that converts generative AI output into predictable crop cards instead of unstructured text.
- A mathematical preprocessing framework for converting soil and climate inputs into standardized model features.
- A user-oriented interface design with farming tips, health guidance, and theme options for outdoor readability.

TABLE I. Research objectives and contribution mapping

Objective	Design Response	Expected Benefit
Context-aware advice	Region + soil + season + weather	More locally relevant suggestions
Offline resilience	TFLite runs on device	Advice works with weak network
Explainable output	Reason, tip, water, profit fields	Easier to trust and compare
Mobile usability	Cards, dropdowns, themes	Simple field interaction

II. LITERATURE REVIEW

A. Digital Agricultural Advisory

Agricultural extension services aim to move research-based farming knowledge to the people who need it. Traditional extension is valuable, but it cannot always provide personal, real-time support to every farmer. Mobile advisory systems reduce that gap by turning a phone into a decision-support tool. However, many such systems still require continuous internet access, which limits their dependability in low-connectivity regions.

B. Machine Learning for Crop Recommendation

Earlier crop recommendation studies have used supervised learning models over soil and climate variables. Pudumalar et al. demonstrated the usefulness of data-driven crop recommendation for precision agriculture [4]. These approaches are efficient and measurable, but their outputs are usually short labels or scores. They do not naturally provide the conversational explanation that a farmer may need before trusting a recommendation.

C. Edge Computing and Fault Tolerance

Edge computing moves part of the computation closer to the user device, reducing latency and dependence on a distant server [5], [6]. For AgriCare, the smartphone itself becomes the edge device. The fallback design is also influenced by dependable-system thinking, where failure should be anticipated and managed rather than treated as an exception that breaks the user experience [7].

D. Large Language Models in Agricultural Guidance

Recent work has discussed the potential of LLMs to simplify scientific agricultural knowledge and provide more personalized advisory content [15]. The same capability introduces risk: a model can produce fluent but incorrect advice if the prompt and output handling are weak. AgriCare therefore constrains the LLM to a fixed JSON schema and presents its answer through controlled UI fields rather than free-form paragraphs.

III. METHODOLOGY AND ARCHITECTURE

AgriCare is organized into presentation, state-management, service, and external-data layers. Flutter is used because it supports a single codebase for mobile applications and provides a consistent widget-based UI model [10]. The application separates screens from services so that the recommendation logic, weather fetching, theme handling, and UI rendering can evolve independently.

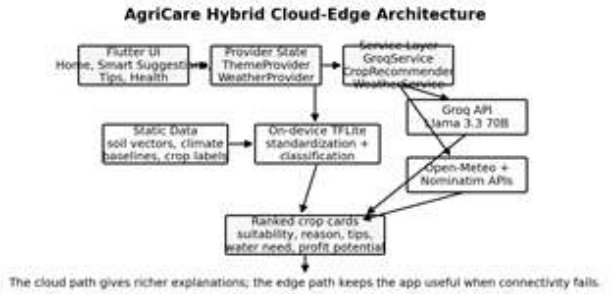

Fig. 1. Hybrid cloud-edge architecture of AgriCare.

A. Presentation Layer

The presentation layer contains the visible Flutter screens: Home, Smart Suggestions, Farming Tips, Health Tips, and Settings. Smart Suggestions is the main research contribution because it connects user input, weather context, cloud inference, and offline inference into one interaction.

B. State Management Layer

Provider-based state management keeps the active theme and weather data available across screens. This prevents repeated API calls and keeps the interface responsive. For example, the weather fetched on the Home screen can be reused when the recommendation screen builds the LLM prompt.

C. Service Layer

The service layer contains GroqService for cloud LLM inference, CropRecommender for TFLite inference, and WeatherService for API-based environmental context. TensorFlow Lite is appropriate for on-device model execution because it is designed for deploying machine-learning models on mobile and edge devices [9].

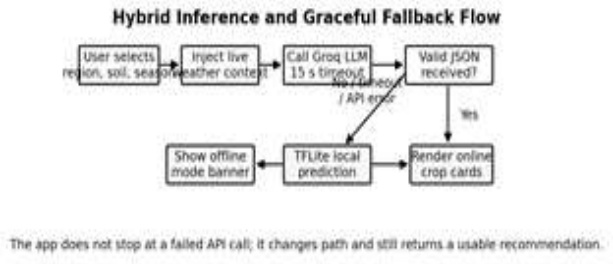

Fig. 2. Hybrid inference and graceful fallback flow.

D. Real-Time Weather and Geocoding Pipeline

WeatherService first obtains device coordinates through location permission. It then calls Open-Meteo for current weather variables and Nominatim for reverse geocoding. Open-Meteo provides forecast and weather variables through a coordinate-based API [13], while Nominatim supports reverse lookup from coordinates to address information [14].

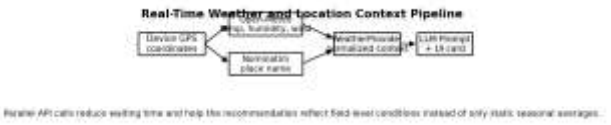

Fig. 3. Real-time weather and location context pipeline.

TABLE II. Soil and climate features used by the recommender

Feature Group	Variables	Purpose
Soil	N, P, K, pH, S, Zn	Fertility signal
Climate	Rain, Tmax, Tmin, humidity	Regional suitability
Season	Kharif/Rabi/Zaid	Seasonal adjustment
Live weather	Temp, humidity, wind	Prompt context

IV. MATHEMATICAL FRAMEWORK

The local inference path converts farmer selections into a numerical vector. The soil type is mapped to nutrient values, the region is mapped to climate baselines, and the season modifies the climate portion of the vector. This keeps the mobile model simple enough for fast inference while still representing the main agronomic signals used by the project.

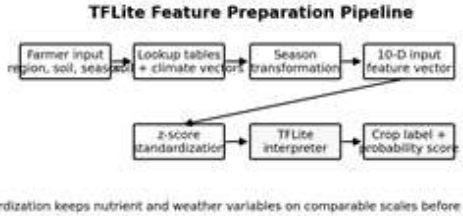

Fig. 4. TFLite feature preparation pipeline.

A. Feature Vector Construction

Let the soil vector be x_soil = [N, P, K, pH, S, Zn] and the climate vector be x_climate = [rainfall, Tmax, Tmin, humidity]. The complete input vector is written as:

x = [x_soil || x_climate] = [x1, x2, ..., x10] (1)
where || denotes vector concatenation. This produces a ten-dimensional feature vector for the TFLite classifier.

B. Standardization
Before inference, each feature is standardized using the training-set mean and standard deviation. This avoids allowing high-magnitude variables such as nitrogen or rainfall to dominate smaller-scale variables such as pH.

z_i = (x_i - mu_i) / sigma_i, i = 1, 2, ..., 10 (2)

C. Softmax Probability and Prediction
The neural-network output is converted into class probabilities using softmax:
P(y = c | z) = exp(o_c) / sum_{j=1}^{C} exp(o_j) (3)
The selected crop is the class with the highest probability:
y_hat = argmax_c P(y = c | z) (4)

D. Fallback Decision Rule
The hybrid path can be expressed as a simple reliability rule. If the cloud returns a valid response before timeout, AgriCare displays the LLM result; otherwise it displays the local model result:
result = cloud_JSON, if cloud_success within 15 s; otherwise TFLite_prediction. (5)

V. IMPLEMENTATION DETAILS

A. Flutter Implementation
The implementation uses Flutter widgets for a consistent mobile interface across devices. The Smart Suggestions screen collects crop conditions through dropdown inputs and presents results as ranked cards. This is more readable than showing raw model output, especially for users who need fast interpretation in the field.

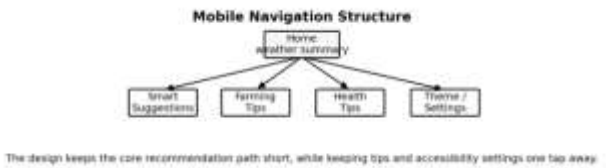

Fig. 5. Flutter mobile navigation structure.

B. Cloud LLM Prompting
The cloud path sends a structured prompt to a Groq chat-completion endpoint. Groq exposes an OpenAI-compatible chat-completion API [12], and the prototype uses a Meta Llama 3.3 70B Instruct model [11]. The prompt asks for exactly three crops and requires fields such as suitability, reason, tips, water need, profit potential, and whether the crop was the user's chosen crop.

```
{
  "crops": [
    {"name": "Crop Name", "suitability": 85,
     "reason": "...", "tips": "...",
     "water_need": "Low|Medium|High",
     "profit_potential": "Low|Medium|High",
     "is_users_choice": false}
  ],
  "summary": "Brief recommendation"
}
```

C. Offline TFLite Path
The offline path loads the TFLite model and scaler parameters from bundled assets. Once the app constructs and standardizes the input vector, the interpreter returns a crop class and probability score. This path is intentionally lightweight: it does not attempt long explanations, but it ensures the farmer still receives a practical recommendation.

D. Human-Centered Interface Choices
AgriCare avoids overwhelming users with a wall of text. The LLM output is broken into cards, progress bars, and short practical tips. Theme options support different viewing conditions, including bright outdoor use. Farming and health tip modules provide value even when the recommendation service is not being used.

VI. RESULTS AND EVALUATION

The prototype was evaluated on latency, fallback reliability, and consistency of structured output. The values reported here are prototype measurements, not claims from a large field deployment. They are included to show whether the architecture behaves as intended under normal and failure conditions.

TABLE III. Prototype inference latency comparison

Inference Path	Output Type	Median Latency	95th Percentile
TFLite edge	Crop + confidence	47 ms	62 ms
Cloud LLM Wi-Fi	3 crop cards	1.9 s	2.7 s
Cloud LLM 4G	3 crop cards	2.8 s	4.1 s

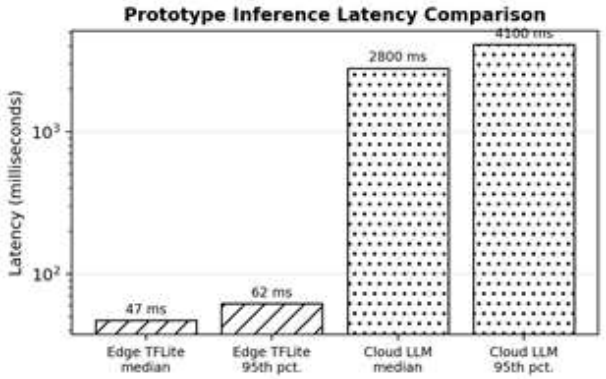

Fig. 6. Prototype inference latency comparison.

The result shows the trade-off that motivates the hybrid design. The edge model is almost immediate, while the cloud model takes longer but returns richer information. In an advisory application, both qualities matter: speed builds usability, and explanation builds trust.

TABLE IV. Fault-tolerance validation summary

Failure Scenario	Expected Behavior	Observed Prototype Behavior	Fallback Time
Airplane mode	Use TFLite	Offline banner shown	~0.6 s
API timeout	Use TFLite after 15 s	Fallback after timeout	~15.6 s
HTTP 429	Catch and fallback	Message + fallback	~0.7 s
Bad JSON	Parse error fallback	No UI crash	~0.65 s

A. Structured JSON Output Evaluation

A free-form response from an LLM is easy to read once, but hard to render consistently in a mobile interface. The JSON format made the result predictable. Each crop card had a name, suitability score, reason, tip, water need, and profit label. This also made error handling simpler: if the JSON was missing or invalid, the fallback rule could activate cleanly.

B. Discussion

The most important result is not only the latency number; it is the continuity of the user experience. The application gives a high-quality response when cloud inference works and a simpler but usable response when it does not. This is a better fit for rural use than a design that assumes connectivity will always be available.

VII. LIMITATIONS AND FUTURE WORK

AgriCare is a working prototype, but it still has limitations. The current version depends on manually selected region and soil type, so incorrect user input can affect the recommendation. It also does not yet diagnose crop disease from images, provide complete multilingual voice interaction, or learn from a farmer's previous seasons.

TABLE V. Future work roadmap

Limitation	Impact	Planned Improvement
Manual soil input	Lower accuracy if wrong	Soil test / local map
English-first UI	Excludes some users	Add regional languages
No vision module	No leaf/pest diagnosis	Add plant vision model
One cloud provider	Outage/rate-limit risk	Add provider fallback
No user history	No past-yield learning	Add local history/sync

Future work should also include real farmer usability testing. Field evaluation can reveal issues that laboratory testing misses, such as outdoor readability, language clarity, trust in AI advice, and the difference between a technically correct recommendation and one that is economically realistic for a local farmer.

VIII. CONCLUSION

This paper presented AgriCare, a Flutter-based agricultural advisory application that combines cloud LLM reasoning with on-device TensorFlow Lite inference. The system is designed around a practical insight: farmers need useful recommendations even when connectivity is unreliable. By using a cloud-first and edge-fallback pattern, AgriCare balances rich explanation with dependable availability.

The architecture also shows how generative AI can be made safer for mobile decision support. Instead of displaying uncontrolled long-form text, AgriCare asks the LLM for structured JSON and renders it through a predictable interface. Live weather and location context make the output more relevant, while the offline model protects the user from total service failure. As future features such as image diagnosis, localization, voice support, and farmer history are added, the same modular architecture can continue to support more complete agricultural guidance.

APPENDIX A. API AND PERMISSION SPECIFICATION

TABLE VI. External API endpoints

Service	Endpoint / File	Authentication	Role
Groq LLM	chat/completions	Bearer key	Cloud LLM
Open-Meteo	/v1/forecast	None	Weather
Nominatim	/reverse	None	Geocoding
Android manifest	Location permissions	User grant	GPS access

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

APPENDIX B. PACKAGE DEPENDENCIES

TABLE VII. Core package dependencies

Package	Role in AgriCare	Reason for Selection
tflite_flutter	Local inference	Runs TFLite model
http	REST calls	API communication
provider	State	Shares app state
geolocator	GPS	Gets coordinates
flutter_dotenv	Secrets	Loads API keys
url_launcher	Links	Opens guides

REFERENCES

1. Ministry of Agriculture and Farmers Welfare, Government of India, Annual Report 2024-25, 2025. [Online]. Available: https://agriwelfare.gov.in/Documents/AR_Eng_2024_25.pdf
2. Agriculture Census Division, Government of India, All India Report on Agriculture Census 2015-16, 2021. [Online]. Available: https://agcensus.da.gov.in/document/agcen1516/ac_1516_report_final-220221.pdf
3. Telecom Regulatory Authority of India, The Indian Telecom Services Performance Indicators, 2023-24, 2024. [Online]. Available: https://www.trai.gov.in/sites/default/files/2024-09/Report_14082024.pdf
4. S. Pudumalar, E. Ramanujam, R. H. Rajashree, C. Kavya, T. Kiruthika, and J. Nisha, "Crop recommendation system for precision agriculture," in 2016 Eighth International Conference on Advanced Computing, Chennai, India, 2017, pp. 32-36.
5. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, 2016.
6. M. Satyanarayanan, "The emergence of edge computing," Computer, vol. 50, no. 1, pp. 30-39, 2017.
7. A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11-33, 2004.
8. M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in Proc. 12th USENIX

Symposium on Operating Systems Design and Implementation, 2016, pp. 265-283.
9. Google, "TensorFlow Lite documentation." [Online]. Available: https://www.tensorflow.org/lite. Accessed: May 31, 2026.
10. Google, "Flutter documentation." [Online]. Available: https://docs.flutter.dev/. Accessed: May 31, 2026.
11. Meta, "Llama 3.3 70B Instruct model card." [Online]. Available: https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct. Accessed: May 31, 2026.
12. Groq, "Groq API reference: Chat completions." [Online]. Available: https://console.groq.com/docs/api-reference. Accessed: May 31, 2026.
13. Open-Meteo, "Weather Forecast API documentation." [Online]. Available: https://open-meteo.com/en/docs. Accessed: May 31, 2026.
14. Nominatim, "Reverse geocoding API documentation." [Online]. Available: https://nominatim.org/release-docs/latest/api/Reverse/. Accessed: May 31, 2026.
15. A. Tzachor et al., "Large language models and agricultural extension services," Nature Food, vol. 4, pp. 941-948, 2023.