Optimizing Resource-Constrained Data Classification in Mushroom Cultivation with Lightweight Machine Learning Models

Nguyen Van Minh[1], Khanh Nghiem Dinh[1], Thinh Duc Nguyen[1], Khuu Chi Khang[2], and Vo Minh Huan[1,3]

[1] Ho Chi Minh University of Technology and Education
[2] International School Ho Chi Minh City
[3] Coressponding author: huanvm@hcmute.edu.vn

Abstract. Classifying data in mushroom cultivation and smart agriculture is vital for optimizing resources, improving productivity, and enabling real-time decisions. Deploying AI models on embedded devices enhances this process by offering low latency, reduced bandwidth, better security, and cost efficiency. However, challenges such as limited resources and model complexity persist. This study explores three models for classifying environmental and biological image-encoded data: CNN, BNN, and LSVM+K-Means. CNN provides the highest accuracy (99%) but demands significant computation. BNN, with 85–89% accuracy, is lightweight and ideal for low-power systems. LSVM+K-Means balances speed and efficiency (81% accuracy). A finite-state machine-based framework is also proposed for flexible model deployment, showing that while CNN is ideal for accuracy, BNN and LSVM+K-Means are better suited for embedded smart agriculture systems.

Keywords: mushroom cultivation · Machine learning model · Embedded Device · Smart agriculture.

1 Introduction

Classifying data in mushroom cultivation and smart agriculture is vital for optimizing resources, boosting productivity, and enabling real-time decisions [9]. Environmental data like temperature, humidity, and CO levels directly affect mushroom growth and must be monitored continuously, while substrate and biological data (e.g., pH, growth stage, disease signs) are essential for maintaining healthy cultivation. Economic data such as costs and yields support long-term planning. Structuring these diverse data types into image-like formats such as binary matrices or heatmaps, allows the use of powerful deep learning models like CNNs or BNNs. This transforms complex data classification into image recognition tasks, enabling accurate detection of patterns or anomalies and supporting automated monitoring in resource-constrained environments [4].

Convolutional Neural Networks (CNNs) are widely used in deep learning for their effectiveness in image recognition, extracting local features like edges

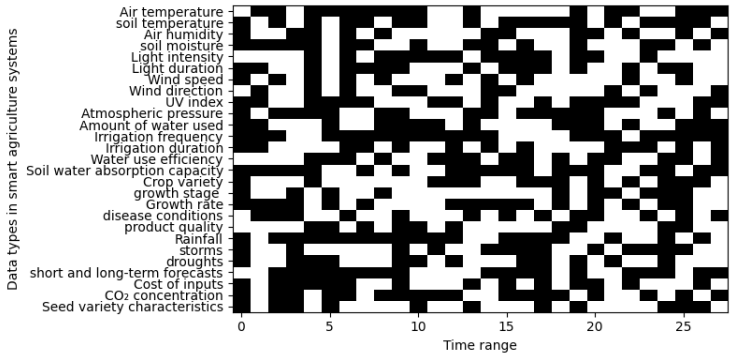

Fig. 1. Mushroom cultivation data classification as a MNIST 28x28 image

and textures [8]. While CNNs deliver high accuracy, they demand substantial computational power and memory, limiting their use in embedded systems [2].

To overcome these limitations, lightweight models such as Binary Neural Networks (BNNs) have been proposed. BNNs use binary weights and activations, significantly reducing model size and computation, making them suitable for resource-constrained devices, albeit with lower accuracy [3], [10]. Another efficient solution combines Support Vector Machine (SVM) with K-Means clustering, enhancing speed and dimensionality reduction while maintaining accuracy, even with smaller datasets [7],[5].

Recent advancements also include XNOR-Net and Ternary Neural Networks (TNNs), which utilize bitwise and ternary operations to balance performance and efficiency [11], [1]. Meanwhile, attention-based models like Vision Transformer (ViT) offer promising accuracy but remain computationally demanding for embedded applications due to heavy matrix operations [12].

This study explores and optimizes machine learning models for image recognition on embedded devices using the MNIST dataset. It evaluates three models: CNN, BNN, and LSVM with K-Means. CNN delivers the highest accuracy but requires significant resources, while BNN is more efficient for low-power devices. LSVM with K-Means strikes a balance between accuracy and efficiency. A deployment framework is also proposed to reduce processing time and enhance resource utilization. The research offers practical solutions for applying AI in resource-constrained environments.

2 Proposed LSVM Model Combined with K-Means

The K-Means clustering model processes each label in the dataset by checking sample counts, extracting image features, and applying K-Means to group images with similar traits. This clustering simplifies analysis and supports automated classification, object recognition, and dataset preparation for model training.

The Linear Support Vector Machine (LSVM) is a widely used linear classification method that identifies an optimal hyperplane to separate classes [7]. As illustrated in Figure 2, the combined LSVM+K-Means approach includes preprocessing, training, and classification modules, ensuring an efficient and structured workflow.

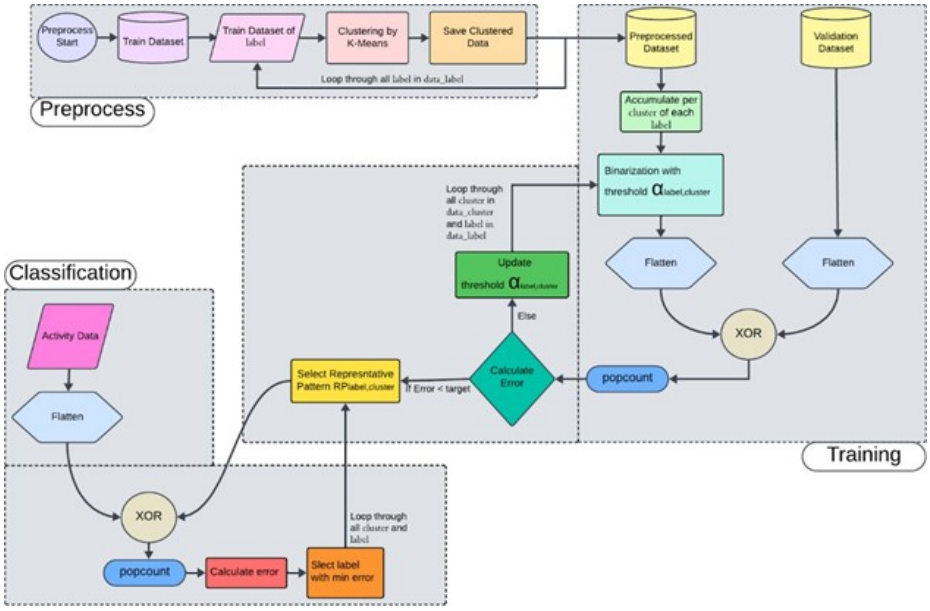

Fig. 2. Block diagram of the proposed LSVM+K-means model

Preprocessing Block: To build effective hyperplanes, training data is binarized to reduce complexity while preserving core features. However, this can remove fine-grained details, such as handwriting variations in MNIST. To address this, K-Means clustering is applied within each label group to retain intra-class diversity. The result is a set of clustered training samples with enhanced representativeness.

Training: Representative samples are generated from each cluster and binarized using a threshold α to suppress outliers. These are flattened into vectors R_P. XOR and popcount operations compare R_P with validation data t_d to compute error, which guides threshold updates until a target error is met. This improves hyperplane quality and classification performance [10].

Classification Processing: Classification uses fast bitwise operations. The `Classify` function compares inputs to R_P via XOR and popcount to determine

similarity. This approach avoids multiplications, making it ideal for low-power devices and enabling fast, accurate classification.

3 Result

The BNN model was trained for 30 epochs on 60,000 MNIST images and evaluated on 10,000, using standard 28x28 pixel inputs. BNN+K-Means used the same data but trained for only 10 epochs, indicating a faster process. LSVM+K-Means trained for 50 epochs on 48,000 images and evaluated on 10,094, requiring the longest training time due to higher complexity. These variations reflect trade-offs between training time and model complexity.

Figure 3 (a) illustrates the BNN model's training process. During the first 7 epochs, the model shows instability with fluctuating accuracy and high loss, likely due to suboptimal weight initialization and an unsuitable learning rate. Gradually reducing the learning rate helped stabilize training. From the 8^{th} epoch onward, accuracy stabilized within $\pm 1\%$ and saturated around epoch 16^{th}. The loss also declined notably, indicating effective feature learning and weight adjustment. Figure 3 (b) shows the training process of the BNN+K-Means model, which uses clustering to enhance learning by adjusting weights for each label cluster. Trained for only 10 epochs to prevent overfitting, the model shows a sharp accuracy increase in the second epoch, then a gradual linear improvement. This suggests clustering accelerates learning and boosts accuracy. The loss function decreases inversely to accuracy.

The LSVM+K-means model was trained for 50 epochs. It saved parameters that achieved the highest accuracy during training. Figure 3 (c) shows that the XOR popcount error dropped rapidly, reaching the predefined threshold of 34, set to avoid generating unusable all-black or all-white images. For each cluster, the model computes a representative image and binarization threshold. These are XOR popcounted with evaluation clusters, and thresholds are adjusted to minimize error. The model then saves the matrix and threshold yielding the best accuracy.

The threshold value is adjusted based on the following rule: if the average error exceeds the target error and the current threshold value is less than or equal to 0, the threshold is increased by a small amount proportional to the average error. Conversely, if the average error is greater than the target error and the current threshold value is greater than 0, the threshold is decreased by a small amount proportional to the error. Figure 4 (a) shows that each label's cluster has a distinct threshold, which is updated over epochs to optimize accuracy. The highest accuracy was achieved at epoch 20. Due to the randomness of threshold updates, training continued for 50 epochs, and the best-performing model parameters were saved.

Table 1 compares model performance across accuracy, execution time, and memory usage. CNN achieves the highest accuracy (99.2%) but requires the most resources: 15.2 seconds of execution time and 1128 KB of memory. BNN lowers accuracy to 87.2% but drastically improves speed (0.45 seconds) and memory

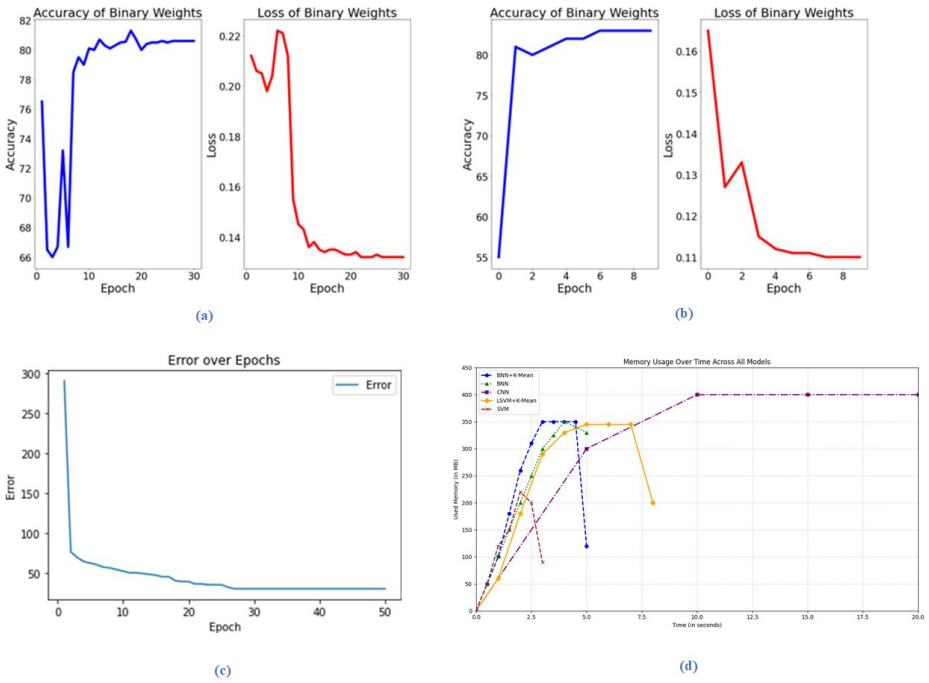

Fig. 3. (a) Accuracy and loss per epoch for the BNN model (b) Accuracy and loss across epochs for the BNN+K-Means model (c) Representation of error over epochs for the LSVM+K-Means model (d) Memory usage over time for all models

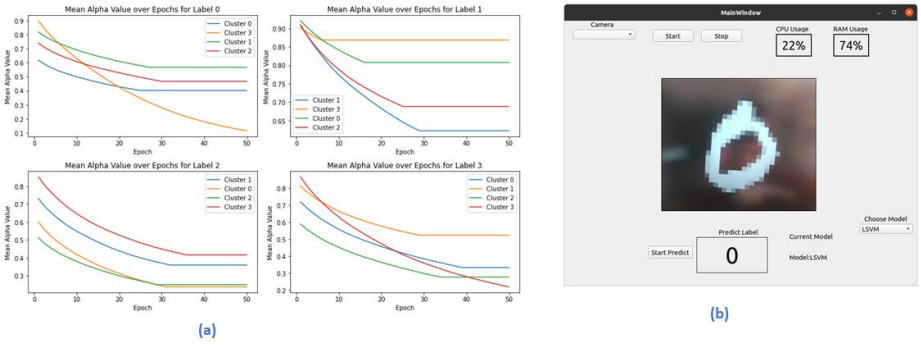

Fig. 4. (a) The threshold value of the LSVM model using the K-Means clustering algorithm over each epoch in lables 0, 1, 2, 3. (b) User interface framework

(51 KB). BNN+K-Means raises accuracy to 89% while keeping resource usage minimal, showing clustering adds value with little overhead. SVM offers 94% accuracy and moderate resource use (1.1 seconds, 64.874 KB). LSVM+K-Means has the lowest accuracy (81%) but is the fastest (0.24 seconds) and lightweight (70 KB), making it ideal for real-time or constrained systems despite reduced precision.

Table 1. Performance comparison of models on I7-11800H (16GB RAM) and Jetson Nano (270 images)

Model	I7-11800H (Desktop)			Jetson Nano		
	Accuracy (%)	Time (s)	Storage (KB)	Accuracy (%)	Time (s)	Used Memory (MB)
CNN	99.2	15.2	1128	99.5	72	400
BNN	87.2	0.45	51	87.1	1.8	350
BNN + K-Means	89.0	0.5	51	89.0	1.9	350
SVM	94.0	1.1	64.874	94.0	4.6	225
LSVM + K-Means	81.0	0.24	70	81.0	1.5	340

As shown in Table 1, based on predictions for 270 images, the BNN+K-Means model improves accuracy over the BNN while maintaining the same compact model size. The added step of calculating weights per cluster slightly increases execution time from 0.45 to 0.5 seconds, but the model remains well-suited for real-world applications requiring fast and lightweight classification. The table 1 also shows that the LSVM+K-Means model has a significantly smaller size than the 32-bit SVM and only slightly larger than BNN+K-Means, due to storing additional thresholds for cluster-based binarization. It is the fastest model in the study, though its accuracy is moderate, making it ideal for time-sensitive or resource-limited environments where speed outweighs precision. Results on the Jetson Nano board [6] show that CNN consumes the most memory due to its reliance on floating-point operations. In contrast, BNN and LSVM+K-Means use significantly less memory thanks to binary operations. The SVM model requires the least memory, aided by the efficiency of the scikit-learn library, benefiting from strong support provided by the scikit-learn library.

Figure 3 (d) illustrates the memory usage patterns of various models on Jetson Nano. The CNN model shows the highest memory consumption, rapidly reaching 400 MB due to intensive floating-point operations and convolutional layers, then stabilizing around 380 MB. BNN and BNN+K-Means both peak at 350 MB, with BNN stabilizing faster while BNN+K-Means sustains usage longer due to clustering overhead, though both remain more efficient than CNN. LSVM+K-Means starts at 60 MB and peaks at 340MB by 7.5 seconds, then drops to 200 MB as temporary buffers are released. Its memory usage is moderate and comparable to BNN, making it suitable for embedded deployment despite slightly higher overhead than SVM. The SVM model demonstrates the most efficient memory usage, peaking at 225 MB and remaining stable. Optimized via scikit-learn, it avoids heavy computation, making it ideal for memory-constrained devices.

Figure 4 (b) shows the FSM-based model test interface. Users can select a camera, start or stop it, and view the live feed. CPU and RAM usage are displayed to monitor system load. A model can be chosen via dropdown, and the FSM updates the model if conditions are met. The "Start Prediction" button triggers image prediction, displaying results in the output frame. Figure 5 shows

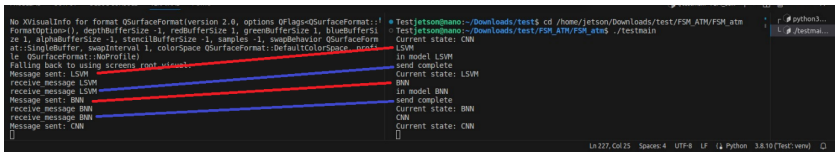

Fig. 5. State transition in FSM

the application's state transition, with the application terminal on the right and the FSM on the left. The system has three states: CNN, BNN, and LSVM. It starts in the CNN state by default. When the user selects LSVM (LSVM+K-Means), the application sends a message to the FSM, which validates it and, if valid, transitions to the LSVM state and sends a confirmation. The application then runs the LSVM model for prediction, and the FSM updates its state accordingly.

4 Conclusion and discussion

The study evaluated several models-BNN, BNN + K-Means, and LSVM + K-Means-against CNN and SVM benchmarks on the MNIST dataset, comparing accuracy, execution time, storage, and memory usage. CNN achieved the highest accuracy (>99%) but required significant computational resources, taking 72 seconds on Jetson Nano and using 400 MB of memory. BNN, with binary weights and activations, ran over 40 times faster (1.8 s) and reduced model size to 51 KB, though with lower accuracy (87.1%). Combining BNN with K-Means improved accuracy to 89% with similar speed and memory efficiency. LSVM + K-Means prioritized speed (0.24 s) and ran faster than SVM, but at the cost of lower accuracy (81%) and higher integration complexity. Despite moderate memory use (340 MB), it suited real-time tasks. In summary, CNN is best for high-accuracy applications, while BNN and LSVM variants offer lightweight, fast alternatives for embedded or real-time systems.

References

1. Alemdar, H., Leroy, V., Prost-Boucle, A., Pétrot, F.: Ternary neural networks for resource-efficient ai applications (2016), https://arxiv.org/abs/1609.00222, arXiv:1609.00222

2. Amami, R., Amami, R., Trabelsi, C., Mabrouk, S., Khalil, H.: A robust voice pathology detection system based on the combined bilstm–cnn architecture. MENDEL **29**(2), 202–210 (2023). https://doi.org/10.13164/mendel.2023.2.202
3. Courbariaux, M., Bengio, Y., David, J.P.: Binaryconnect: Training deep neural networks with binary weights during propagations (2015). https://doi.org/10.48550/ARXIV.1511.00363, accepted at NIPS 2015
4. Kamilaris, A., Prenafeta-Boldú, F.X.: Deep learning in agriculture: A survey. Computers and Electronics in Agriculture **147**, 70–90 (2018). https://doi.org/https://doi.org/10.1016/j.compag.2018.02.016
5. Kumar, G.K., Bangare, M.L., Bangare, P.M., Kumar, C.R., Raj, R., Arias-Gonzáles, J.L., Omarov, B., Mia, M.S.: Internet of things sensors and support vector machine integrated intelligent irrigation system for agriculture industry. Discover Sustainability **5**(1), 6 (jan 2024). https://doi.org/10.1007/s43621-024-00179-5
6. Madrin, F.P., Rosenberger, M., Nestler, R., Dittrich, P.G., Notni, G.: The evaluation of CUDA performance on the Jetson Nano board for an image binarization task. In: Kehtarnavaz, N., Carlsohn, M.F. (eds.) Real-Time Image Processing and Deep Learning 2021. p. 15. SPIE, Online Only, United States (Apr 2021). https://doi.org/10.1117/12.2586650
7. Milod, T., Almhdie Aboubaker, Llahm Omar Faraj Ben Dalla: Diabetes Prediction Using a Support Vector Machine (SVM) and visualize the results by using the K-means algorithm (2024). https://doi.org/10.13140/RG.2.2.35171.16165, publisher: Unpublished
8. Mirshahi, S., Brandtner, P., Oplatková, Z.: Intermittent time series demand forecasting using dual convolutional neural networks. MENDEL **30**(1), 51–59 (Jun 2024). https://doi.org/10.13164/mendel.2024.1.051
9. Moysiadis, V., Kokkonis, G., Bibi, S., Moscholios, I., Maropoulos, N., Sarigiannidis, P.: Monitoring mushroom growth with machine learning. Agriculture **13**(1) (2023). https://doi.org/10.3390/agriculture13010223, https://www.mdpi.com/2077-0472/13/1/223
10. Raha, A., Thirumala, S.K., Gupta, S.K., Raghunathan, V.: Interaxnn: A Reconfigurable and Approximate In-Memory Processing Accelerator for Ultra-Low-Power Binary Neural Network Inference in Intermittently Powered Systems (2024). https://doi.org/10.2139/ssrn.4782177
11. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks (2016). https://doi.org/10.48550/ARXIV.1603.05279, version Number: 4
12. Zhou, S., Meng, S., Tian, H., Yu, J., Wang, K.: Edge-BiT: Software-Hardware Co-design for Optimizing Binarized Transformer Networks Inference on Edge FPGA. In: Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design. pp. 1–9. ACM, Newark Liberty International Airport Marriott New York NY USA (Oct 2024). https://doi.org/10.1145/3676536.3676667