Adaptive Hyperspectral Image Classification Using a CNN-MAOELM Hybrid Architecture

Oleksii Fesenko(✉), Robert Bieliakov(✉), Hryhorii Radzivilov(✉), Andrii Bernatskyi(✉), and Dmytro Koltovskov

Kruty Heroes Military Institute of Telecommunications and Information Technology, 45/1 Knyaziv Ostrozkyh Street, Kyiv 01011, Ukraine
{oleksij.fesenko,andrii.bernatskyi, dmytro.koltovskov}@viti.edu.ua, france417@gmail.com, grigoriy110371@gmail.com

Abstract. Hyperspectral image (HSI) classification on resource-constrained airborne platforms remains challenging because conventional Convolutional Neural Networks (CNN) require costly retraining when the input-stream distribution shifts due to atmospheric, illumination, or sensor-noise effects. The aim of this paper is to develop a method for synthesizing a hybrid neural-network architecture that combines deep spatial-spectral feature extraction with non-iterative streaming-mode adaptation under explicit memory and computation budgets. The proposed CNN-MAOELM (Convolutional Neural Network with Modified Adaptive Online Extreme Learning Machine) architecture couples a multi-scale convolutional feature extractor with an analytical MAOELM classifier whose output weights are recursively updated through least-squares estimation with adaptive forgetting, supplemented by controlled local correction of the convolutional subsystem when degradation is detected. Experimental verification on five benchmark hyperspectral datasets (Indian Pines, Pavia University, Salinas, Houston, HyRANK) shows competitive overall accuracy, more than twentyfold acceleration of online adaptation relative to CNN-AOELM, and substantially higher robustness under simulated concept drift, supporting onboard and near-real-time remote-sensing operation.

Keywords: hyperspectral image classification · convolutional neural network · extreme learning machine · adaptive online learning · resource-constrained inference

1 Introduction

Under contemporary conditions of military confrontation, remote sensing technologies play an important role in ensuring information superiority. Hyperspectral imagery (HSI) is especially valuable because it registers spectral signatures of objects in hundreds of narrow bands, enabling the detection of camouflaged equipment, identification of surface-cover materials, assessment of terrain passability, and detection of explosive hazards.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026, LNCS 16767, pp. 393–412, 2027.
https://doi.org/10.1007/978-3-032-30491-9_25

Operational use of HSI requires classification algorithms capable of near-real-time decision-making, including onboard processing on unmanned aerial platforms with limited energy and computational resources. Convolutional Neural Networks (CNN) are widely used for HSI classification [1–4] and provide high accuracy on static benchmarks. However, their deployment in practical situational-awareness systems is limited by the high dimensionality of hyperspectral cubes, the scarcity of labeled samples, computational cost, and input-distribution shifts caused by weather, smoke, illumination changes, sensor noise, and camouflage variability. In addition, spectral uncertainty complicates classification because one class may exhibit different signatures under different acquisition conditions, while different objects may produce similar spectral responses.

Consequently, there is a need for hybrid classification methods that combine CNN capabilities in spatial-spectral feature extraction with the flexibility and speed of non-iterative learning algorithms. The developed paper addresses the stated problem by synthesizing the hybrid CNN with Modified Adaptive Online Extreme Learning Machine (CNN-MAOELM) architecture, which integrates a convolutional subsystem with the MAOELM classifier for streaming-mode operation.

2 Analysis of Recent Publications

A decisive stage in the development of computer vision is associated with the work of Krizhevsky *et al.* [1], where the AlexNet architecture was proposed. Subsequent studies devoted to VGGNet [2], ResNet [3], and DenseNet [4] demonstrated a stable tendency toward accuracy improvement with increasing network depth, although such growth is accompanied by a proportional increase in computational demand.

In the context of remote-sensing image analysis, Zhu *et al.* [5] showed the possibility of adapting residual architectures to multispectral data processing. Chen *et al.* [6] proposed deep residual networks for HSI classification and confirmed the effectiveness of deep models for representative benchmark tasks. At the same time, the structural complexity of such solutions demands substantial computational power and large labeled datasets.

In parallel, a line of fast non-iterative methods based on the Extreme Learning Machine (ELM) was formed. The basic ELM model [7] relies on random initialization of hidden-layer parameters and analytical computation of output weights, which considerably reduces training time. OS-ELM [8] extends the concept to sequential data arrival, while FOS-ELM [9] supplements it with a forgetting mechanism. Further development is represented by the kernel implementation for nonstationary series proposed by Wang and Han [10], as well as by the meta-cognitive sequential ELM for class imbalance and concept drift introduced by Mirza and Lin [11]. The aggregate of such results confirms the attractiveness of ELM-type approaches for streaming tasks; however, in their classical form they do not provide deep spatial-spectral representation learning needed for HSI.

For HSI classification, specialized spectral-spatial architectures have also been actively developed. SSRN [12] implements staged extraction of spectral and spatial representations within a 3D formulation. Deep recurrent networks for HSI [13] are focused on modeling inter-band dependence in spectral sequences. HybridSN [14] showed that

a combination of 3D and 2D convolutions makes it possible to reduce complexity compared with fully 3D networks while preserving high accuracy. Compact 3D-CNN solutions [15] confirmed the promise of lightweight architectures intended for reducing parametric space and computation budget.

Generalizing studies and contemporary architecture surveys also remain important. Li *et al.* [16] systematized early deep-learning approaches for HSI and described the transition from shallow classifiers to end-to-end models. A later survey by Bera *et al.* [17] detailed CNN-oriented solutions, whereas Kumar *et al.* [18] covered a broader spectrum of methods, including 3D-CNN, hybrid architectures, attention mechanisms, and transformers. Ashraf *et al.* [19] reflect the current direction of BERT-like spatial-spectral modeling of long-range dependencies between channels and neighboring pixels. Nevertheless, most such solutions remain oriented toward static benchmarks and offline retraining, which limits their suitability for streaming operation and for deployment on platforms with constrained computational resources.

Table 1. Comparative qualitative summary of representative HSI classification methods.

Method	Deep spatial-spectral	Non-iterative training	Online streaming adaptation	Resource-aware	Robust to concept drift
ResNet-50 [3]	✓	✗	✗	✗	✗
SSRN [12]	✓	✗	✗	✗	✗
HybridSN [14]	✓	✗	✗	◑	✗
Compact 3D-CNN [15]	✓	✗	✗	✓	✗
ELM [7]	✗	✓	✗	✓	✗
OS-ELM [8] / FOS-ELM [9]	✗	✓	✓	✓	◑
CNN-AOELM (baseline)	✓	✓	✓	◑	◑
CNN-MAOELM (proposed)	✓	✓	✓	✓	✓

In Table 1, ✓ denotes full support of the corresponding property, ✗ indicates its absence, and ◑ reflects partial support requiring retraining or external mechanisms. The comparison shows that existing reference architectures do not jointly provide deep spatial-spectral representation, non-iterative learning, online adaptation, resource awareness, and robustness to concept drift. This gap motivates the proposed CNN-MAOELM architecture, which coordinates recognition accuracy, training-time budget, and computational complexity through adaptive forgetting and automatic hyperparameter tuning. The purpose of the paper is to develop a method for synthesizing the CNN-MAOELM neural architecture for hyperspectral image classification. To reach the stated purpose, the following tasks are solved: (1) development of a hybrid architecture with a fast-adaptive

learning mechanism; (2) formalization of the mathematical model of the method; and (3) development of a three-phase learning algorithm with adaptation to newly arriving data in real time.

3 Mathematical Formalization of the Problem

3.1 Problem Statement

Consider a streaming sequence of hyperspectral images

$$\mathcal{S} = \{(X_t, y_t)\}_{t=1}^{T}, \tag{1}$$

where $X_t \in R^{H\times W\times C}$ denotes the input hyperspectral image tensor at time step t, with spatial dimensions $H \times W$ and C spectral channels, and $y_t \in \{1, \ldots, K\}$ is the corresponding class label, where K is the total number of classes.

The goal is to learn an adaptive classification operator

$$F_t : R^{H\times W\times C} \to \Delta^{K-1}, \tag{2}$$

defined as the composition of a convolutional feature extractor Φ_{CNN} and an extreme learning classifier Ψ_{ELM}:

$$\widehat{p_t} = F_t(X_t) = softmax(\Psi_{\text{ELM}}(\Phi_{\text{CNN}}(X_t; \theta_t); \beta_t)), \tag{3}$$

where $\widehat{p_t} = \left(\widehat{p_{t,1}}, \widehat{p_{t,2}}, \ldots, \widehat{p_{t,K}}\right) \in \Delta^{K-1}$ is the vector of posterior class probabilities, θ_t denotes the parameters of the CNN component adapted over time, and β_t is the output-weight matrix of the ELM component updated in the online mode.

The predicted class label is determined as

$$\widehat{y_t} = \arg\max_{k\in\{1,\ldots,K\}} \widehat{p_{t,k}}. \tag{4}$$

At each time step, the model first predicts $\widehat{y_t}$ for the incoming sample X_t, and after observing the true label y_t, updates the parameters θ_t and β_t in order to maintain classification accuracy under streaming data drift and constrained computational resources.

The optimization objective is formulated as

$$\min_{\{\theta_t,\beta_t\}_{t=1}^{T}} J = \frac{1}{T}\sum_{t=1}^{T}\left[L_{\text{CE}}\left(\widehat{p_t}, y_t\right) + \gamma_1 C(\theta_t) + \gamma_2 R_{\text{comp}}(\theta_t, \beta_t)\right], \tag{5}$$

where L_{CE} is the cross-entropy loss, $C(\theta_t)$ is a structural-complexity measure of the CNN architecture, $R_{\text{comp}}(\theta_t, \beta_t)$ denotes the normalized computational cost of model adaptation, and $\gamma_1, \gamma_2 \geq 0$ are weighting coefficients controlling the trade-off among classification accuracy, architectural complexity, and computational efficiency.

3.2 Constraints

Resource, structural, and operational constraints are expressed as a joint admissibility condition for the CNN-MAOELM model:

$$\begin{gathered} \mathrm{Mem}(\mathcal{F}_t) \le M_{max}, T_{proc} \le T_{arrival}, \\ N_{min} \le N_f^{(l)} \le N_{max}, H^{TH} + \lambda I \succ 0, \ \ \lambda > 0, \mathrm{Acc}(t) \ge \mathrm{Acc}_{\mathrm{threshold}} \end{gathered} \tag{6}$$

where $\mathrm{Mem}(\mathcal{F}_t)$ is the memory required by the model at time t; T_{proc} is the processing time of one batch; $N_f^{(l)}$ is the number of filters in the l-th convolutional layer; H^{TH}, guarantees positive definiteness of the regularized ELM matrix; and $\mathrm{Acc}(t) \ge \mathrm{Acc}_{\mathrm{threshold}}$ is the minimum admissible classification accuracy.

3.3 Assumptions

The developed formulation is based on four assumptions: 1) the condition of local stationarity, according to which the input data stream changes its statistical characteristics gradually, thus ensuring timely correction of the model weight parameters without the risk of catastrophic accuracy degradation; 2) representativeness of the initialization stage, assuming that the initial training sample contains a sufficient amount of structural features for correct formation of the base filters of the convolutional subsystem; 3) an assumption concerning the nature of disturbances, according to which spatial additive noise in hyperspectral images follows a Gaussian distribution and is statistically uncorrelated with the useful signal; and 4) hardware reliability of the system, guaranteeing fail-safe and continuous operation of the computing node throughout the entire iterative learning and adaptation cycle.

4 Proposed CNN-MAOELM Method

4.1 General Architecture

The proposed CNN-MAOELM architecture implements an incremental learning scheme in which spectral-spatial feature formation and online classifier adaptation are functionally separated. The convolutional subsystem forms compact discriminative embeddings from hyperspectral patches, whereas the MAOELM component performs non-iterative classification and recursive output-weight updating.

The algorithm consists of three functional blocks: Initial Training, Online Adaptation, and Training Cycle. The essence of the developed approach lies in the fact that the convolutional component does not perform continuous global retraining. Instead, it forms a compact feature vector that is supplied to the modified MAOELM classifier. Most of the operational adaptation is transferred from gradient procedures to a recursive analytical mechanism for updating output weights. When quality degradation signs arise, the method does not launch a global architecture search; it performs local correction of the parameters of the last convolutional block and of the ELM dimensionality.

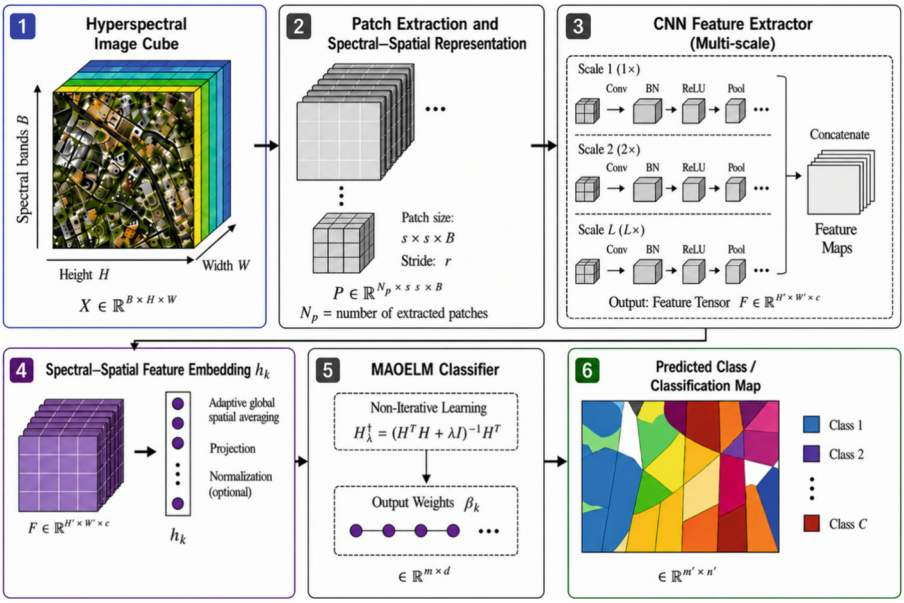


Fig. 1. General architecture of the proposed CNN-MAOELM method for adaptive hyperspectral image classification. The hyperspectral cube is partitioned into spectral-spatial patches, processed by a multi-scale CNN feature extractor, transformed into a compact embedding h_k, and classified by the MAOELM component to obtain the final classification map.

The functional flow of the proposed CNN-MAOELM architecture is shown in Fig. 1. The input hyperspectral cube is first divided into spatial-spectral patches, after which each patch is processed by a multi-scale convolutional extractor.

Let the input cube be denoted as $X \in R^{B\times H\times W}$, where B is the number of spectral bands and H, W are the spatial dimensions. The cube is partitioned into patches

$$p_k \in R^{s\times s\times B}, k = 1, 2, \ldots, N_p, \tag{7}$$

where $s \times s$ is the spatial window size. Each patch p_k is processed by the convolutional subsystem, and the obtained multi-scale feature tensor F_k is transformed into a compact embedding

$$h_k = N\left(W_f \cdot GAP(F_k) + b_f\right), h_k \in R^d, \tag{8}$$

where $GAP(\cdot)$ denotes global average pooling, W_f and b_f are projection parameters, $N(\cdot)$ is the normalization operator, and d is the embedding dimensionality.

The descriptor h_k is then supplied to the MAOELM classifier. The initial output-weight matrix is estimated analytically using the regularized Moore–Penrose solution

$$H_\lambda^\dagger = \left(H^T H + \lambda I\right)^{-1} H^T, \tag{9}$$

where λ is the Tikhonov regularization coefficient and I is the identity matrix. The class prediction for the k-th patch is obtained as

$$\hat{y}_k = softmax\left(H_{k\beta_k}\right). \tag{10}$$

The predicted class is assigned to the central pixel of the corresponding patch, forming the final classification map. Therefore, the general functional dependence of CNN-MAOELM can be written as

$$\hat{y}_k = MAOELM_{\beta_k}(\varphi_\theta(p_k)), \tag{11}$$

where $\varphi_{\theta(\cdot)}$ is the CNN-based spatial-spectral feature extractor and $MAOELM_{\beta_k}$ is the analytical classifier with the current output weights. The key feature of the architecture is the separation of computational roles: CNN performs deep feature extraction, whereas operational adaptation to streaming data is mainly carried out through recursive updating of the MAOELM output weights.

Thus, Fig. 1 reflects the data-flow organization of the developed method, while the internal organization of the corresponding training and adaptation procedures, together with the interaction between the three functional blocks. Detailed in the algorithmic scheme presented in Fig. 2 and elaborated in Sects. 4.2–4.4.

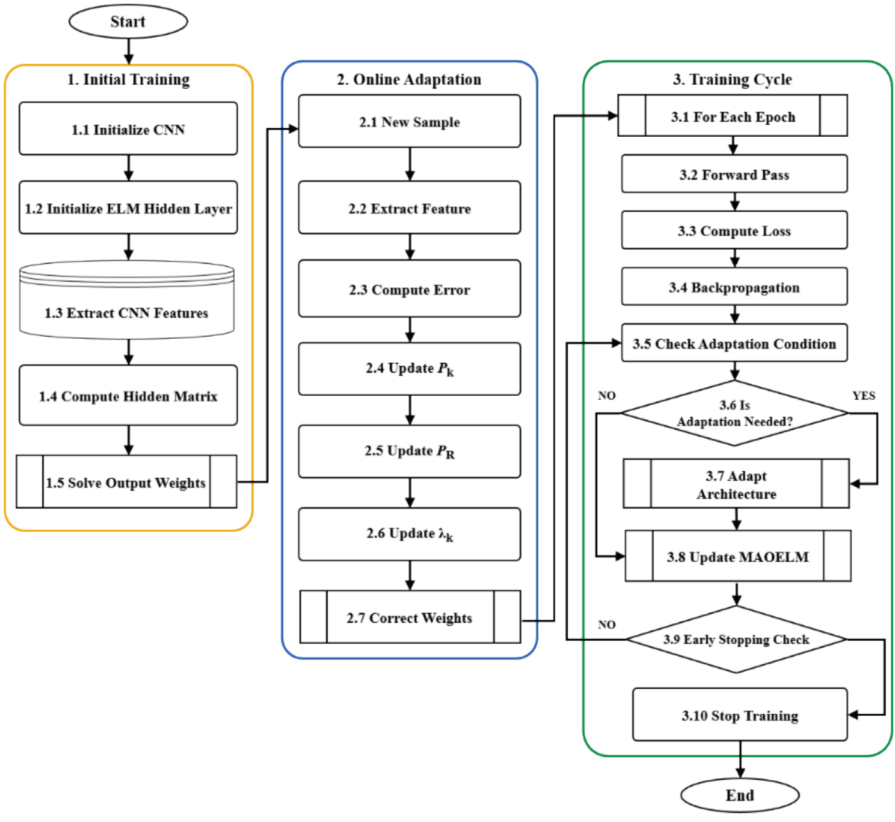


Fig. 2. Training and online adaptation procedure of the proposed CNN-MAOELM architecture.

4.2 Block 1. Initial Training

At the first step, the base parametric configuration of the hybrid CNN-MAOELM model is formed. This stage includes selection of the initial topology, initialization of weight coefficients and bias parameters, and specification of learning hyperparameters. After the spatial-spectral representation is formed by the convolutional subsystem, the output parameters of the ELM component are determined analytically, which yields the initial parametric state of the model. The obtained configuration acts as the basis for subsequent online updating, adaptive forgetting, and structural correction under arrival of new data.

For numerical stability of the initial stage, initialization of CNN weight coefficients is performed according to the He distribution:

$$\mathbf{W}_{\text{CNN}}^{(l)} \sim \mathcal{N}\left(0, \left(\sigma_{\text{CNN}}^{(l)}\right)^2\right), \sigma_{\text{CNN}}^{(l)} = \sqrt{\frac{2}{n_{\text{in}}^{(l)}}}, \tag{12}$$

where $n_{\text{in}}^{(l)}$ denotes the fan-in of the l-th convolutional layer, and $\mathcal{N}(\mu, \sigma^2)$ represents a normal distribution with mean μ and variance σ^2. Hidden-layer ELM weights are initialized randomly as

$$\mathbf{W}_{\text{ELM}} \sim \mathcal{N}\left(0, \sigma_W^2\right), \sigma_W = \sqrt{\frac{2}{n\text{in} + n_{\text{hidden}}}}, \mathbf{b}_{\text{ELM}} \sim \mathcal{N}(0, 1), \tag{13}$$

where $\mathbf{W}_{\text{ELM}}$ and $\mathbf{b}_{\text{ELM}}$ are the hidden-layer weight matrix and bias vector of the ELM component, respectively.

Forward propagation through the CNN subsystem is described, for an arbitrary layer l, by

$$z_{i,j,k}^{(l)} = \sum\nolimits_{m=0}^{M-1} \sum\nolimits_{n=0}^{N-1} \sum\nolimits_{c=0}^{C_{l-1}-1} w_{m,n,c,k}^{(l)} \cdot a_{i+m,j+n,c}^{(l-1)} + b_k^{(l)}, \tag{14}$$

where $w_{u,v,c,k}^{(l)}$ denotes the kernel coefficients of size $U \times V$ linking input channel c and output channel k, while $b_k^{(l)}$ is the bias of the k-th channel. Activation is implemented by ReLU,

$$a_{i,j,k}^{(l)} = \text{ReLU}\left(z_{i,j,k}^{(l)}\right) = \max\left(0,\ z_{i,j,k}^{(l)}\right). \tag{15}$$

To stabilize learning, batch normalization is applied:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \in}},\ y_i = \gamma \cdot \hat{x}_i + \beta, \tag{16}$$

where μ_B and σ_B^2 are the batch mean and variance, and γ, β are the trainable scale and shift parameters.

After passing the convolutional stages, global average pooling forms a compact feature vector:

$$\mathbf{f} = \text{GlobalAvgPool}\left(a^{(L)}\right). \tag{17}$$

The hidden-layer activation matrix of the ELM component is then obtained as

$$\mathbf{H} = \sigma\left(\mathbf{F} \cdot W_{\text{ELM}} + 1_N \cdot b^T\right), \tag{18}$$

where $\mathbf{F}$ is the feature matrix for all training samples, $\sigma(\cdot)$ is the sigmoid activation, and 1 is the column vector of ones.

Analytical estimation of the ELM output weights is performed by the regularized Moore–Penrose solution

$$\beta = \left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{T}, \tag{19}$$

where $\mathbf{T}$ is the one-hot target matrix and λ is the Tikhonov regularization coefficient. Simultaneously, the covariance matrix is initialized as

$$\mathbf{P}_0 = \left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}\right)^{-1}. \tag{20}$$

Such a formulation provides a globally optimal solution for the output weights within the adopted regularized analytical setting and therefore eliminates the local-minimum risk characteristic of purely gradient-based optimization.

4.3 Block 2. Online Adaptation Based on MAOELM

The online-adaptation block constitutes the central component of the scientific novelty of the developed method. The modified adaptive online sequential extreme learning machine extends the ELM concept toward processing of streaming data arriving in successive portions. Within this approach, dynamic parameter updating is carried out according to newly observed samples without the need for retraining on the full dataset. Such organization of the adaptation mechanism preserves computational efficiency, reduces reconfiguration time, and maintains classification stability under variation of the input-stream structure.

For every new sample, after CNN-based feature extraction, the hidden-layer output vector is computed as

$$h_k = \sigma(\mathbf{W}_{\text{ELM}} \cdot \mathbf{f}_k + b), \tag{21}$$

where $\mathbf{f}_k$ is the feature vector formed by the CNN subsystem. The prediction error is then defined by

$$e_k = \mathbf{t}_k - \boldsymbol{h}_k^T\boldsymbol{\beta}_{k-1}, \tag{22}$$

where $\mathbf{t}_k$ is the one-hot target vector.

Recursive updating of the covariance matrix is carried out with the Woodbury identity:

$$\mathbf{P}_k = \frac{1}{\lambda_k}\left(\mathbf{P}_{k-1} - \frac{\mathbf{P}_{k-1}\boldsymbol{h}_k\boldsymbol{h}_k^T\mathbf{P}_{k-1}}{\lambda_k + \boldsymbol{h}_k^T\mathbf{P}_{k-1}\boldsymbol{h}_k}\right), \tag{23}$$

where λ_k is the adaptive forgetting coefficient. The output weights are corrected through the modified recursive least-squares rule

$$\boldsymbol{\beta}_k = \boldsymbol{\beta}_{k-1} + \mathbf{P}_k h_k \Big(t_k - h_k^T \boldsymbol{\beta}_{k-1} \Big). \tag{24}$$

Adaptive forgetting is regulated by the prediction-error norm:

$$\lambda_k = \lambda_{\min} + (\lambda_{\max} - \lambda_{\min}) \cdot \exp(-\gamma \| e_k \|_2), \tag{25}$$

where $\lambda_{\min}$ and $\lambda_{\max}$ define the admissible forgetting range and α is the sensitivity coefficient. When the error magnitude increases, λ_k decreases, which strengthens the contribution of the current observation and accelerates adaptation. When the error is small, λ_k approaches unity, corresponding to weak forgetting and retention of long-term information.

At the final stage of each iteration, inertial correction of the output weights is performed:

$$\boldsymbol{\beta}_k^{\text{final}} = \lambda_k \cdot \boldsymbol{\beta}_k + (1 - \lambda_k) \cdot \boldsymbol{\beta}_{k-1}. \tag{26}$$

where $\kappa \in (0, 1]$ is the smoothing coefficient. By functional meaning, such a procedure is equivalent to a low-pass filter operating in the parameter space and supports stable adaptation in the presence of local perturbations.

4.4 Block 3. Training Cycle

The training cycle provides iterative optimization of the convolutional subsystem together with adaptive updating of the ELM component. Within this cycle, mechanisms of dynamic architecture adaptation and early stopping are coordinated with multicriteria monitoring of performance metrics.

Step 3.1–3.2. Forward propagation. For each epoch $e \in \{1, \ldots, E_{max}\}$ and for each mini-batch, forward propagation through the convolutional and ELM components is carried out, yielding the prediction matrix $\hat{\mathrm{Y}}_{batch}$.

Step 3.3. Loss-function evaluation. At this stage, an integral target functional is formed, within which three key training criteria are coordinated: classification accuracy, parametric-complexity control, and computational-cost control. The general functional is given by

$$\mathcal{L}_{total} = \mathcal{L}_{\mathcal{CE}} + \alpha_{1\mathcal{L}_{\mathcal{L}2}} + \alpha_{2\mathcal{L}_{comp}}, \tag{27}$$

where $\mathcal{L}_{\mathcal{CE}}$ denotes the cross-entropy term, $\mathcal{L}_{\mathcal{L}2}$ is the regularization term for the convolutional weights, and $\mathcal{L}_{comp}$characterizes normalized computational complexity. Coefficients α_1 and α_2 set the relative contribution of the regularization and resource-aware components in the overall optimization procedure.

For the multiclass classification problem with K classes, the cross-entropy term is defined as

$$\mathcal{L}_{\mathcal{CE}} = -\frac{1}{B} \sum\nolimits_{i=1}^{B} \sum\nolimits_{j=1}^{K} y_{ij} \log\big(\widehat{y_{ij}}\big), \tag{28}$$

where B is the mini-batch size, $y_{ij} \in \{0, 1\}$ is the indicator of membership of the i-th sample in the j-th class, and $\hat{y}_{ij}$ is the predicted probability for class j. This component minimizes the discrepancy between the reference class structure and the model outputs.

The regularization term is represented as

$$\mathcal{L}_{\mathcal{L}2} = \sum\nolimits_{l=1}^{L} ||\mathbf{W}^{(l)}||_F^2, \tag{29}$$

where $\mathbf{W}^l$ is the weight matrix of the $l - th$ convolutional level, $||\cdot||_F$ is the Frobenius norm, and L is the number of convolutional levels. Introduction of this term constrains excessive parameter growth and preserves structural compactness of the architecture.

The component that reflects computational cost is defined as

$$\mathcal{L}_{comp} = \frac{\text{FLOPs}(\theta)}{\text{FLOPs}_{\text{target}}}, \tag{30}$$

where FLOPs (Floating-Point Operations) (θ) characterizes the actual computational complexity of the model with parameters θ, and $\text{FLOPs}_{\text{target}}$ denotes the target computational budget. Such normalization coordinates computational-efficiency requirements with the main classification term within a unified optimization functional.

Thus, minimization of $\mathcal{L}_{total}$ orients training toward a compromise between recognition quality, parametric compactness, and resource efficiency, which is essential for an adaptive model intended for operation under constrained computational resources.

Step 3.4. Backpropagation with CNN-parameter update. This stage updates the convolutional subsystem using gradients of the target functional. To preserve numerical stability and prevent uncontrolled gradient growth, gradient clipping is applied:

$$\nabla \mathbf{W}_{\text{clip}}^{(l)} = \begin{cases} \nabla \mathbf{W}^{(l)}, & \text{if } \|\nabla \mathbf{W}^{(l)}\| \le \tau, \\ \frac{\tau}{\|\nabla \mathbf{W}^{(l)}\|} \cdot \nabla \mathbf{W}^{(l)}, & \text{if } \|\nabla \mathbf{W}^{(l)}\| > \tau, \end{cases} \tag{31}$$

where $\tau > 0$ is the threshold value of the gradient norm. Weight updating is then performed by the Adam algorithm:

$$\begin{aligned} \boldsymbol{m}_t^{(l)} &= \beta_1 \boldsymbol{m}_{t-1}^{(l)} + (1 - \beta_1) \nabla \mathbf{W}_{\text{clip}}^{(l)}, \\ \mathbf{v}_t^{(l)} &= \beta_2 \mathbf{v}_{t-1}^{(l)} + (1 - \beta_2) \left(\nabla \mathbf{W}_{\text{clip}}^{(l)} \right)^{\odot 2}, \\ \mathbf{W}_t^{(l)} &= \mathbf{W}_{t-1}^{(l)} - \eta \cdot \frac{\boldsymbol{m}_t^{(l)} / \left(1 - \beta_1^t\right)}{\sqrt{\mathbf{v}_t^{(l)} / \left(1 - \beta_2^t\right)} + \varepsilon}, \end{aligned} \tag{32}$$

where $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-8}$ are the Adam parameters.

Step 3.5–3.7. Architecture adaptation. The condition for architecture adaptation is checked on the basis of continuous monitoring of the loss and estimation of the validation-accuracy trend, formalized as

$$\text{trend}_{\text{acc}} = \frac{\text{Acc}_{\text{val}}^{(e)} - \text{Acc}_{\text{val}}^{(e-w)}}{w}, \tag{33}$$

where w is the size of the sliding analysis window. Adaptation is initiated when $\text{trend}_{\text{acc}} < \delta_{adapt}$, where δ_{adapt} is the threshold value.

For each level l, the importance metric is computed as the averaged $L2$ norm of the gradients:

$$\text{importance}^{(l)} = \frac{1}{B}\sum\nolimits_{i=1}^{B}\left\|\frac{\partial \mathcal{L}_{\text{CE}}}{\partial a_i^{(l)}}\right\|_2. \tag{34}$$

The change in the number of filters is determined by the following rule system:

$$\Delta_{\text{filters}}^{(l)} = \begin{cases} +\alpha_{\text{expand}} \cdot N_f^{(l)}, & \text{if importance}^{(l)} > \bar{I} \text{ and util}^{(l)} > 0.7, \\ -\alpha_{\text{reduce}} \cdot N_f^{(l)}, & \text{if importance}^{(l)} < 0.5\bar{I} \text{ or util}^{(l)} < 0.3, \\ 0, & \text{otherwise}, \end{cases} \tag{35}$$

where $\bar{\text{I}}$ is the average importance across all levels, $util^{(l)}$ is the neuron-utilization coefficient, and $\alpha_{expand}, \alpha_{reduce} \in (0,1)$ are the adaptation-rate coefficients.

At the next stage, new filters are generated on the principle of parametric inheritance. Within this approach, newly added convolutional elements are initialized on the basis of parametric information accumulated inside the already formed architecture. This preserves continuity of the feature representation and supports stability of subsequent optimization. The corresponding expression is

$$\mathbf{W}_{\text{new}}^{(l)} = \mathbf{W}_{\text{best}}^{(l)} + \mathcal{N}\left(0, \sigma_{\text{inherit}}^2\right). \tag{36}$$

Step 3.8. MAOELM-parameter update. For a batch of new features $\mathbf{H}_{batch}$, recursive updating of the covariance matrix $\mathbf{P}_t$ and the output-weight matrix $\boldsymbol{\beta}_t$ is performed according to Eqs. (23) and (26). Equation (23) specifies correction of $\mathbf{P}_t$ with allowance for the information contribution of the new batch to the hidden feature space. Equation (26) describes refinement of $\boldsymbol{\beta}_t$ on the basis of the approximation error of the current batch, defined by the difference between the reference labels Y_{batch} and the prediction $\mathbf{H}_{batch}\boldsymbol{\beta}_{t-1}$. This scheme adapts MAOELM to new data without retraining on the full observation set and therefore increases computational efficiency in streaming mode.

$$\mathbf{P}_t = \mathbf{P}_{t-1} - \mathbf{H}_{\text{batch}}^T\left(I + \mathbf{H}_{\text{batch}}\mathbf{P}_{t-1}\mathbf{H}_{\text{batch}}^T\right)^{-1}\mathbf{H}_{\text{batch}}\mathbf{P}_{t-1}, \tag{37}$$

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} + \mathbf{P}_t\mathbf{H}_{\text{batch}}^T(Y_{\text{batch}} - \mathbf{H}_{\text{batch}}\beta_{t-1}). \tag{38}$$

where $\mathbf{H}_{batch} \in R^{B\times M}$ is the hidden-layer activation matrix for the current batch of size B, M is the number of hidden neurons, $Y_{batch} \in R^{B\times C}$ is the target-output matrix, and $\boldsymbol{\beta}_t \in R^{M\times C}$ is the matrix of output weights for C classes.

In such a formulation, parameter updating is carried out in the space of compact feature representations, which increases the suitability of MAOELM for multidimensional streaming data.

Step 3.9. Early-stopping mechanism. Early termination of iterative training is implemented on the basis of a system of stopping criteria oriented toward convergence control,

attainment of the target quality level, and prevention of overfitting. The training cycle is stopped when the maximum number of epochs E_{max} reached, when the selected metric stagnates during $E_{patience}$ epochs, when validation accuracy reaches the target level $Acc^{e}_{val} \geq Acc_{target}$, or when the admissible overfitting threshold is exceeded. Diagnosis of the latter is based on the relative difference between training and validation accuracy, defined as

$$overfit = \frac{\mathrm{Acc}^{(e)}_{\mathrm{train}} - \mathrm{Acc}^{(e)}_{\mathrm{val}}}{\mathrm{Acc}^{(e)}_{\mathrm{train}}} > overfit_{\mathrm{threshold}}. \tag{39}$$

The above stopping scheme provides coordination between accuracy, stability, and computational expediency of the training procedure.

5 Experimental Study

To verify the proposed method, a benchmark suite of five hyperspectral image (HSI) datasets was formed (Table 2), covering urban, agricultural, and mixed scene types. In particular, the Indian Pines dataset [15] represents an agricultural scene with shelterbelts and provides a suitable test environment for evaluating the algorithm under mixed-pixel conditions. The Salinas dataset [16] is characterized by high spatial resolution, which enables the assessment of fine spectral discrimination between adjacent classes. The Pavia University dataset [17] is used to investigate urban areas and makes it possible to analyze classification accuracy at spatial object boundaries as well as during the recognition of small structures. Additional verification of model stability under the significant spectral complexity of urban development is provided by the Houston dataset [18]. The robustness of the method to the influence of complex background conditions is evaluated using the HyRANK dataset [19], which describes a natural landscape with increased intraclass variability of spectral features.

The selection of these specific datasets is motivated by the necessity to rigorously evaluate the algorithm's robustness against key destabilizing factors. These include spatial heterogeneity (dense urban environments in the Houston dataset and complex topography in HyRANK), varied spatial resolutions (1.3–30 m/pixel), the prevalence of mixed pixels (Indian Pines), and high inter-class spectral similarity (Salinas).

Table 2. Characteristics of the benchmark hyperspectral datasets.

Dataset	Sensor	Size	Bands	Classes	Spatial resolution
Houston	ITRES CASI	349 × 1905	144	15	2.5 m
HyRANK	Hyperion	1225 × 1220	176	14	30 m
Indian Pines	AVIRIS	145 × 145	220	16	20 m
Pavia Univ	ROSIS	610 × 340	103	9	1.3 m
Salinas	AVIRIS	512 × 217	224	16	3.7 m

For comparative analysis, the following methods were selected: a classical deep CNN (ResNet-50), baseline ELM with manually extracted features, hybrid CNN-ELM, adaptive CNN-AOELM, and the developed CNN-MAOELM. The experiments were implemented in Python 3.10 using PyTorch 2.1 [20] for neural-network modelling, NumPy 1.26 [21] and SciPy 1.11 [22] for numerical computations, scikit-learn 1.3 [23] for evaluation metrics, and Matplotlib 3.8 [24] for visualization. The development and execution environment was Visual Studio with Python; the computational platform comprised an Intel Core i7 CPU, 32 GB RAM, and an NVIDIA GeForce RTX 3060 GPU with CUDA 12.1.

For quantitative assessment of the effectiveness of the developed CNN-MAOELM method, three conventional quality metrics for hyperspectral-image classification were used, which provide a comprehensive analysis of model performance.

Overall Accuracy (OA) is defined as the ratio of the number of correctly classified pixels to the total number of test pixels and is calculated by the formula: $OA = \left(\frac{1}{N}\right) \times \Sigma\delta(\hat{y}_i, y_i)$, where N is the total number of test samples, $\hat{y}_i$ is the predicted class label, y_i is the true label, $\delta(\cdot, \cdot)$ is the Kronecker delta function. OA is a basic metric characterizing the overall effectiveness of the classifier; however, it may be biased under an imbalanced class distribution.

Cohen's Kappa (κ) takes into account the possibility of random agreement in classification and is defined as: $\kappa = \frac{(OA - p^e)}{(1 - p^e)}$, *where* $p^e = \left(\frac{1}{N^2}\right) \times \Sigma_k n_k \cdot m_k$—the probability of random agreement, n_k—the number of samples classified as class k, m_k—the actual number of samples of class k. The value $\kappa \in [-1, 1]$, where $\kappa = 1$ corresponds to perfect classification and $\kappa = 0$ corresponds to random guessing. The coefficient κ is more informative than OA for problems with imbalanced classes, which is typical of hyperspectral data.

Average Accuracy (AA) is calculated as the arithmetic mean of classification accuracies for each individual class: $AA = \left(\frac{1}{K}\right) \times \Sigma_k \left(\frac{n_{kk}}{n_k}\right)$, where K is the total number of classes, n_{kk}—the number of correctly classified samples of class k, n_k—the total number of samples of class k. AA provides a balanced assessment of classifier performance irrespective of the size of each class, which is important for tasks in which minority classes have high tactical significance.

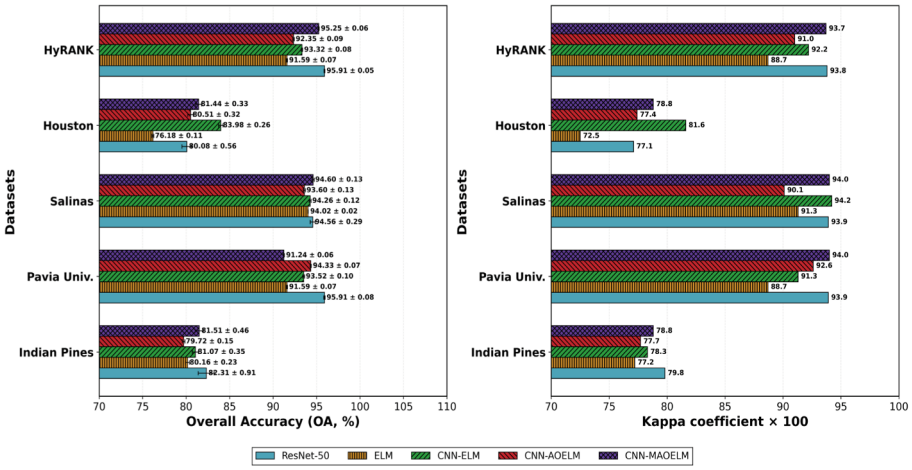
5.1 Classification Results

The OA metric reflects the share of correctly classified pixels relative to the total test volume and therefore provides the most direct integral view of classifier performance. As shown in the left panel of Fig. 3, CNN-MAOELM does not exhibit unconditional dominance across all five datasets, yet ensures stably high accuracy and a more favorable compromise between accuracy and adaptability.

The developed architecture shows its strongest behavior on HyRANK and Houston, where the adaptive mechanism yields improved resistance to spectral variability and urban heterogeneity.

The agreement coefficient κ reveals a similar tendency: CNN-MAOELM forms an averaged agreement level comparable with ResNet-50 while surpassing the baseline ELM and demonstrating a clear advantage on the Houston scene. Such behavior confirms that the proposed architecture preserves statistical stability of recognition while reducing dependence on heavy end-to-end retraining.

The AA metric is most sensitive to recognition quality for minority classes, because it is independent of the population size of each class. Figure 4 reveals a more subtle structure of inter-method differences in AA. Although ResNet-50 remains highly competitive on several datasets, CNN-MAOELM retains close average accuracy and offers a more rational compromise when the requirement of online adaptation is taken into account. Such a property is especially important for real-world HSI analysis, where appearance of new masking strategies or acquisition changes may force the classifier to react without full retraining.



(a) *(b)*

Fig. 3. Comparison of the algorithms across five hyperspectral datasets: *(a) Overall Accuracy (OA), %; (b) Cohen's kappa coefficient, $\kappa \times 100$.*

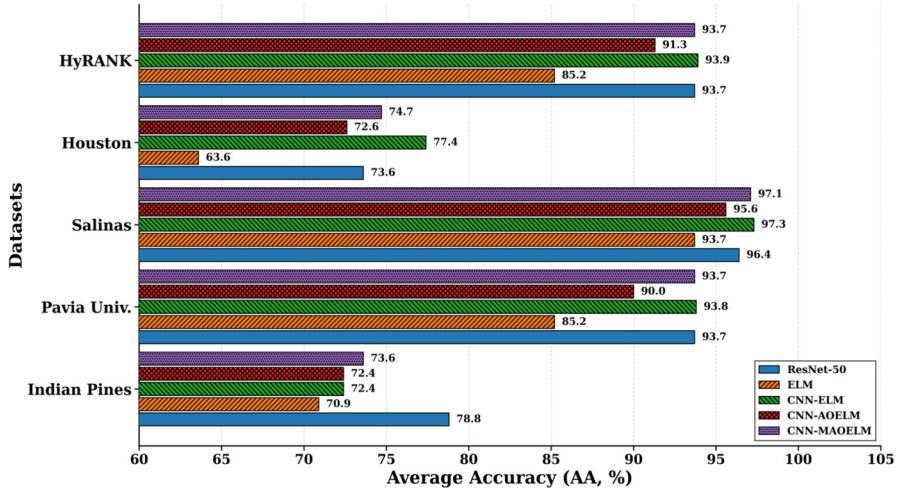


Fig. 4. Comparison of the algorithms by Average Accuracy (AA), %.

5.2 Analysis of Temporal Characteristics

Analysis of temporal characteristics confirms the substantial computational efficiency of the developed architecture. Compared with ResNet-50, initial training time is reduced approximately twofold: 40.34 min instead of 1 h 20 min. The main advantage of the method lies in its high speed of online adaptation to new streaming data. Owing to the modified recursive mechanism, adaptation to 1000 new samples requires only 7.45 s, whereas classical CNN-ELM and ResNet-50 solutions require computationally expensive full retraining. Even relative to CNN-AOELM, the proposed optimization accelerates adaptation by more than twenty times, which makes the hybrid neural network a rational solution for real-time hardware systems (Table 3).

Table 3. Comparison of training time and adaptation speed.

Method	Training time	Adaptation time (1000 samples)
ResNet-50	1 h 20 min	Full retraining / 1 h 20 min
ELM	1.25 min	Full retraining / 1 min
CNN-ELM	47.23 min	Full retraining / 47.25 min
CNN-AOELM	41.45 min	2.07 min
CNN-MAOELM	40.34 min	7.45 s

5.3 Robustness Under Input-Distribution Shift

To verify the key advantage of the developed CNN-MAOELM method with respect to the ability for online adaptation under changing characteristics of the input data, a series of experiments was carried out with emulation of distribution shift (concept drift). This experiment models realistic scenarios of hyperspectral-image quality degradation caused by atmospheric interference, changes in illumination conditions, smoke contamination, or hardware noise of sensors in the onboard UAV system.

The experimental methodology consists in gradual addition of additive Gaussian noise $\xi \sim \mathcal{N}(0, \sigma^2 I)$ to the normalized spectral vectors of the test subset $\widetilde{X}_t = X_t + \xi$, where the intensity parameter $\sigma \in \{0.0, 0.01, 0.05, 0.1, 0.2, 0.3, 0.5\}$ varies from the absence of perturbation to substantial distortion of the input data. The choice of this grid is dictated by physically meaningful operating regimes typical of hyperspectral acquisition: $\sigma \in [0.01,\ 0.05]$ reproduces low-level sensor and quantization noise; $\sigma \in [0.1,\ 0.2]$ models moderate atmospheric scattering and acquisition-related distortions; while $\sigma \in [0.3,\ 0.5]$ emulates severe observation conditions, including degraded illumination and strong instrumental interference.

The stated protocol reproduces a typical scenario of practical application in which a model pre-trained under controlled conditions operates in an operational environment with a changed statistical structure of input data. To ensure correctness of comparison, all investigated methods received the same noisy input data within the full set of spectral channels.

Additionally, for CNN-MAOELM an assessment of the effectiveness of the online-adaptation mechanism (Block 2 of the developed algorithm) was carried out. The adaptation procedure consisted of the following: a subset of $N_{\text{adapt}} = 300$ noisy samples with known class labels was selected from the test subset, on which recursive updating of the output weights β was performed using the modified RLS algorithm with the adaptive forgetting coefficient λ_k. After completion of adaptation, the model was evaluated on the remaining part of the test subset.

The experimental results are shown in Fig. 5 and indicate that, at a low noise level $\sigma \leq 0.05$, all methods retain high classification accuracy. However, as the noise intensity increases to $\sigma > 0.1$, substantial performance degradation is observed for all static models and for one method with an online-adaptation mechanism, which confirms the hypothesis about the inability of fixed classifiers to operate effectively under distribution shift.

At the same time, CNN-MAOELM with the activated online-adaptation mechanism demonstrates significantly higher robustness: at $\sigma = 0.2$, the model preserves accuracy at the level of 60–73% depending on the dataset, whereas static CNN methods degrade to 20–35%. This gain is caused by the ability of the adaptive forgetting coefficient λ_k to dynamically regulate the balance between retention of previous knowledge and integration of new information about the changed distribution.

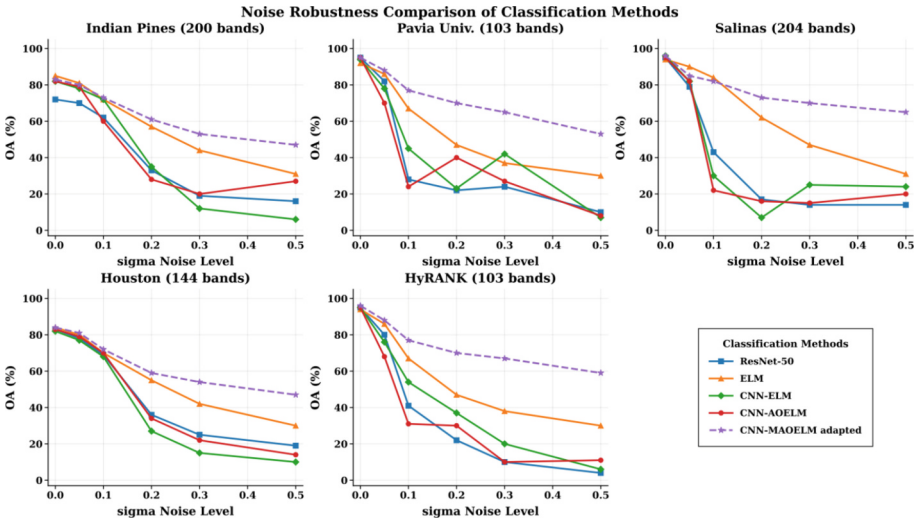


Fig. 5. Robustness analysis under increasing input noise.

6 Scientific Novelty of the Obtained Results

First, a method for synthesizing the hybrid CNN-MAOELM neural architecture for hyperspectral image classification is developed. The distinguishing feature of the method lies in a three-phase operating algorithm that integrates initial analytical identification of the ELM component, recursive online adaptation of output weights, and controlled architecture correction of the convolutional subsystem. Owing to such organization, the developed solution combines deep spatial-spectral feature extraction with high-speed streaming adaptation.

Second, the mechanism of adaptive online learning for streaming data is improved. The essential difference lies in the use of recursive least squares with adaptive forgetting and inertial smoothing of the output-weight matrix. Such a combination makes it possible to preserve numerical stability, suppress abrupt parameter oscillations, and increase classifier robustness under concept drift.

Third, the method of automatic neural-architecture correction during learning receives further development. The developed formulation uses continuous monitoring of performance metrics and gradient-based importance analysis of convolutional layers. As a result, architecture correction is performed not through a costly global search but through local structural adaptation, which is substantially more compatible with resource-constrained operation.

7 Conclusions

The paper develops a method for synthesizing the hybrid CNN-MAOELM neural architecture intended for classification of hyperspectral images under constrained computational resources. The developed approach combines a convolutional subsystem for

spatial-spectral feature extraction with a modified adaptive online extreme learning machine, which makes it possible to preserve competitive classification accuracy while considerably reducing training and adaptation time.

Practical value of the developed method is manifested most distinctly under input-distribution shift. Experimental results with gradually increasing Gaussian noise confirm that the method maintains substantially higher robustness than static baselines and classical hybrid solutions. For 1000 new samples, online updating is performed in 7.45 s, which supports applicability of the architecture to onboard and near-real-time systems.

7.1 Future Work

Future work will address the integration of spatial-spectral attention mechanisms, uncertainty-aware adaptive forgetting in the MAOELM update procedure, and automated local correction of the CNN subsystem. Additional research will focus on embedded implementation on onboard platforms such as NVIDIA Jetson Orin, Xilinx Zynq UltraScale +, and Google Edge TPU, as well as on extending the architecture to multisensor fusion of hyperspectral, SAR, and thermal data for all-weather unmanned aerial remote sensing.

Acknowledgments. The authors thank the Kruty Heroes Military Institute of Telecommunications and Information Technology for institutional support and access to computational facilities. The authors are grateful to the anonymous reviewers of ICCSA 2026 for constructive comments that contributed to the improvement of the manuscript.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105 (2012)
2. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015). https://doi.org/10.48550/arXiv.1409.1556
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings IEEE CVPR, pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
4. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings IEEE CVPR, pp. 4700–4708 (2017). https://doi.org/10.1109/CVPR.2017.243
5. Zhu, X.X., et al.: Deep learning in remote sensing: a comprehensive review and list of resources. IEEE Geosci. Remote Sens. Mag. **5**(4), 8–36 (2017). https://doi.org/10.1109/MGRS.2017.27623071
6. Chen, Y., Jiang, H., Li, C., Jia, X., Ghamisi, P.: Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. IEEE Trans. Geosci. Remote Sens. **54**(10), 6232–6251 (2016). https://doi.org/10.1109/TGRS.2016.2584107
7. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1–3), 489–501 (2006). https://doi.org/10.1016/j.neucom.2005.12.126

8. Liang, N.-Y., Huang, G.-B., Saratchandran, P., Sundararajan, N.: A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans. Neural Netw. **17**(6), 1411–1423 (2006). https://doi.org/10.1109/TNN.2006.880583
9. Zhao, J., Wang, Z., Park, D.S.: Online sequential extreme learning machine with forgetting mechanism. Neurocomputing **87**, 79–89 (2012). https://doi.org/10.1016/j.neucom.2012.01.027
10. Wang, X., Han, M.: Online sequential extreme learning machine with kernels for nonstationary time series prediction. Neurocomputing **145**, 90–97 (2014). https://doi.org/10.1016/j.neucom.2014.05.003
11. Mirza, B., Lin, Z.: Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification. Neural Netw. **80**, 79–94 (2016). https://doi.org/10.1016/j.neunet.2016.04.009
12. Zhong, Z., Li, J., Luo, Z., Chapman, M.: Spectral-spatial residual network for hyperspectral image classification: a 3-D deep learning framework. IEEE Trans. Geosci. Remote Sens. **56**(2), 847–858 (2018). https://doi.org/10.1109/TGRS.2017.2755542
13. Mou, L., Ghamisi, P., Zhu, X.X.: Deep recurrent neural networks for hyperspectral image classification. IEEE Trans. Geosci. Remote Sens. **55**(7), 3639–3655 (2017). https://doi.org/10.1109/TGRS.2016.2636241
14. Roy, S.K., Krishna, G., Dubey, S.R., Chaudhuri, B.B.: HybridSN: exploring 3D–2D CNN feature hierarchy for hyperspectral image classification. IEEE Geosci. Remote Sens. Lett. **17**(2), 277–281 (2020). https://doi.org/10.1109/LGRS.2019.2918719
15. Ahmad, M., Khan, A.M., Mazzara, M., Distefano, S., Ali, M., Sarfraz, M.S.: A fast and compact 3-D CNN for hyperspectral image classification. IEEE Geosci. Remote Sens. Lett. **19**, 1–5 (2022). https://doi.org/10.1109/LGRS.2020.3043712
16. Li, S., Song, W., Fang, L., Chen, Y., Ghamisi, P., Benediktsson, J.A.: Deep learning for hyperspectral image classification: an overview. IEEE Trans. Geosci. Remote Sens. **57**(9), 6690–6709 (2019). https://doi.org/10.1109/TGRS.2019.2907932
17. Bera, S., Shrivastava, V.K., Satapathy, S.C.: Advances in hyperspectral image classification based on convolutional neural networks: a review. Comput. Model. Eng. Sci. **137**(1), 205–240 (2023). https://doi.org/10.32604/cmes.2023.028868
18. Kumar, V., Singh, R.S., Rambabu, M., Dua, Y.: Deep learning for hyperspectral image classification: a survey. Comput. Sci. Rev. **53**, 100633 (2024). https://doi.org/10.1016/j.cosrev.2024.100633
19. Ashraf, M., Zhou, X., Vivone, G., Chen, L.: Spatial-spectral BERT for hyperspectral image classification. Remote Sens. **16**(3), 539 (2024). https://doi.org/10.3390/rs16030539
20. Paszke, A., et al.: PyTorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035. Curran Associates (2019). https://doi.org/10.48550/arXiv.1912.01703
21. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., et al.: Array programming with NumPy. Nature **585**, 357–362 (2020). https://doi.org/10.1038/s41586-020-2649-2
22. Virtanen, P., et al.: SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2
23. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
24. Hunter, J.D.: Matplotlib: a 2D graphics environment. Comput. Sci. Eng. **9**(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55