2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Structural Entropy Dynamics in CNN Training: A Three-Phase Guided Framework with Applications in Training Optimization

Fengming Dong, Jianghua Lv, Yining Chen and Hexuan Li

Beihang University, Beijing 100191, China
dfmsbc@buaa.edu.cn

Abstract. Recent advances in convolutional neural networks (CNNs) have achieved remarkable success across computer vision domains, yet the inherent complexity and opaque nature of their training processes continue to impede further efficiency improvements. As a quantitative indicator of graph structural complexity, structural entropy offers a novel perspective for analyzing the training dynamics of neural networks. This work proposes a graph structure abstraction-based representation method for CNNs, establishing a quantitative framework for training complexity assessment through the transformation of computational graphs into weighted directed graphs followed by structural entropy calculation. Through systematic monitoring of classical CNN architectures, we identify a three-phase evolution pattern of complexity dynamics: Adjustment Phase, Convergence Phase, and Specialization Phase, thereby formulating a structural entropy-guided characterization framework for CNN training processes. Furthermore, by establishing the correlation between dynamic structural entropy features and model performance, we develop optimization strategies including entropy-aware early stopping criteria and adaptive learning rate scheduling. Experimental results demonstrate that the proposed methodology achieves 27% training acceleration without sacrificing model accuracy, providing a principled approach to enhance CNN training efficiency through complexity-aware optimization.

Keywords: Structural Entropy, Convolutional Neural Networks, Training Process Analysis, Training Strategy Optimization

1 Introduction

The efficacy of neural network training fundamentally determines the extent to which a model's architectural potential can be realized. Conventional training methodologies for convolutional neural networks (CNNs), however, predominantly rely on extrinsic observational metrics such as loss functions and accuracy rates [1, 2]. While these indicators provide surface-level performance assessments, they fail to elucidate the intrinsic characteristics of information processing within neural networks or quantify the profound impact of structural complexity on model capabilities. This critical limitation stems from the traditional paradigm's neglect of internal information dynamics during network evolution.

The Information Bottleneck Theory offers a transformative theoretical framework by examining neural networks through an information-theoretic lens [3, 4]. Its central premise posits that deep learning essentially performs hierarchical information compression through multilayer nonlinear transformations, maximizing task-relevant information retention while minimizing input redundancy. According to this theory, the continuous parameter updates during training optimize the network's information compression capability, inevitably imprinting observable patterns in weight distribution dynamics. Specifically, initial network weights typically follow Gaussian distributions characterized by high entropy and randomness, reflecting an informationally redundant state prior to effective feature representation establishment. As training progresses, backpropagation-driven gradient updates progressively restructure weight distributions into task-specific configurations [5].

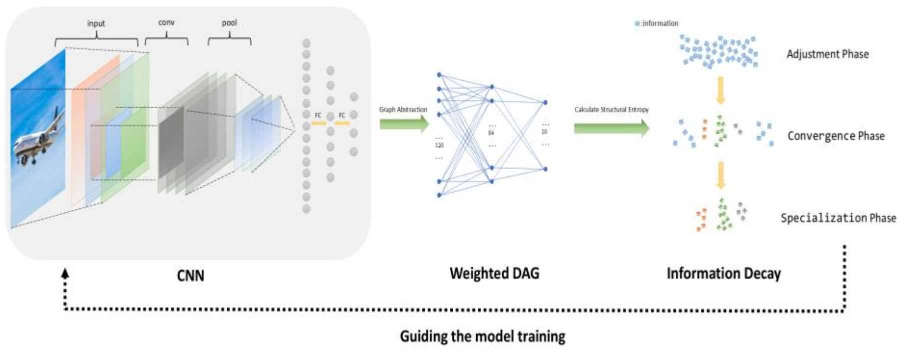

Fig. 1. After abstracting the convolutional neural network model into a graph structure, its structural entropy during the training process is computed.

Structural entropy [6], as an information-theoretic measure of network architecture complexity, provides a novel metric for quantifying information complexity evolution during CNN training. This study consequently aims to establish a complexity quantification framework for CNN training processes, transcending the limitations of conventional external metrics. By adopting structural entropy as an endogenous measure of training dynamics, we systematically analyze the mapping relationships between topological evolution and model performance, ultimately constructing a "monitoring-analysis-control" optimization paradigm. Our principal contributions include:

(1) Graph structure abstraction-based CNN representation: We develop a weighted directed graph transformation method for CNN computational graphs, achieving 99.41% structural correlation preservation with original models. This graphical representation enables subsequent structural entropy computation, establishing a quantitative complexity assessment framework for CNN training processes.

(2) Three-phase evolution pattern identification: Through structural entropy monitoring of classical CNN architectures, we discover a distinct complexity progression comprising Adjustment Phase (initial parameter exploration), Convergence Phase

(stable feature formation), and Specialization Phase (task-specific refinement). This finding provides novel theoretical criteria for training process supervision.

(3) Structural entropy-driven adaptive optimization: By establishing correlations between entropy dynamics and model performance, we design optimization strategies including entropy-aware early stopping criteria and adaptive learning rate adjustment. Experimental validation demonstrates that our methodology reduces training duration by 27% while maintaining model fidelity, offering a complexity-aware paradigm for efficient CNN training.

2 Background Study

2.1 Weight Distribution Dynamics in CNN Training

From the perspective of neural network weight evolution, the parameter updates in stochastic gradient descent (SGD) can be modeled through the stochastic differential equation (SDE) [7, 8]:

$$d\boldsymbol{w} = -\eta \nabla L(\boldsymbol{w})dt + \sqrt{2\eta\sigma^2}d\boldsymbol{W}_t \tag{1}$$

where η denotes the learning rate, $\nabla L(\boldsymbol{w})$ represents the loss function gradient, and the term $\sqrt{2\eta\sigma^2}d\boldsymbol{W}_t$ quantifies the noise induced by minibatch sampling. The magnitude of σ inversely correlates with batch size, where smaller batches yield larger σ values. By coupling this with the Fokker-Planck equation (FPE), which governs the temporal evolution of probability density functions in stochastic systems:

$$\frac{\partial p(\boldsymbol{w},t)}{\partial t} = -\nabla \cdot [\boldsymbol{\mu}(\boldsymbol{w},t)p(\boldsymbol{w},t)] + \frac{1}{2}\nabla^2[D(\boldsymbol{w},t)p(\boldsymbol{w},t)] \tag{2}$$

here, $\boldsymbol{\mu}(\boldsymbol{w},t)$ corresponds to the drift term (deterministic dynamics), while $D(\boldsymbol{w},t)$ characterizes the diffusion coefficient (stochastic noise intensity). The operators ∇ and ∇^2 denote divergence and Laplacian operations, respectively. Substituting Eq.2 into the FPE yields:

$$\frac{\partial p(\boldsymbol{w},t)}{\partial t} = \eta\nabla \cdot [\nabla L(\boldsymbol{w})p(\boldsymbol{w},t)] + \eta^2\sigma^2\nabla^2 p(\boldsymbol{w},t) \tag{3}$$

the first term $\eta\nabla \cdot [\nabla L(\boldsymbol{w})p(\boldsymbol{w},t)]$ captures gradient-driven drift, where weights migrate along loss function descent directions. The second term $\eta^2\sigma^2\nabla^2 p(\boldsymbol{w},t)$ represents noise-induced diffusion. During early training stages, weight initialization positions lie distant from loss minima, resulting in large-magnitude gradients $\nabla L(\boldsymbol{w})$ [9]. This causes drift-dominated dynamics, concentrating weight distributions toward regions of lower loss. As training progresses toward convergence, gradient magnitudes diminish significantly, reducing the drift term to $\mathcal{O}(\eta \parallel \nabla L \parallel p)$, where $\parallel \nabla L \parallel \rightarrow 0$. Concurrently, the diffusion term persists at $\mathcal{O}(\eta^2\sigma^2 \parallel \nabla^2 p \parallel)$, establishing a competitive equilibrium between drift and diffusion that drives the system toward steady-state distributions.

2.2 Structural Entropy

The theoretical foundations of structural entropy trace back to Shannon's seminal 1948 information theory [10], where entropy quantifies uncertainty in discrete random variables:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \, log \, p(x_i) \tag{4}$$

with $p(x_i)$ denoting the probability of event x_i.

Recent advancements by Li introduced structural entropy as a novel metric for evaluating network architecture and dynamical complexity through encoding trees [6]. The core concept involves capturing the separation between regular patterns and stochastic noise via high-dimensional encoding, formally defined as the minimal bit requirement to encode node accessibility during graph random walks. Structural entropy fundamentally characterizes two aspects: K-dimensional structural information (measuring spatial orderliness) and dynamic complexity (quantifying uncertainty in network interactions and evolution).

For general network complexity assessment, one-dimensional structural entropy evaluates node degree distributions:

$$\mathcal{H}^1(G) = -\sum_{i=1}^{n} \frac{d_i}{2m} log_2 \frac{d_i}{2m} \tag{5}$$

where d_i represents node degree and m the total edge count. For weighted graphs, degrees can be substituted with node weights for analogous computations.

When analyzing network partitions (e.g., community structures), two-dimensional structural entropy minimizes intra-module uncertainty:

$$\mathcal{H}^2(G) = \min_{\mathcal{P}} \left(\sum_{j=1}^{L} \frac{V_j}{2m} H(p_j) - \sum_{j=1}^{L} \frac{g_j}{2m} log_2 \frac{V_j}{2m} \right) \tag{6}$$

where V_j denotes module volume and g_j intra-module edge count. This formulation effectively captures community structures, facilitating community detection algorithms.

Higher-dimensional structural entropy extends this framework to hierarchical networks through partition trees:

$$\mathcal{H}^K(G) = \min_{\mathcal{T}} \sum_{\alpha \in \mathcal{T}} \left(-\frac{g_\alpha}{2m} log_2 \frac{V_\alpha}{V_{\alpha^-}} \right) \tag{7}$$

where $\mathcal{T}$ representing a height-K partition tree. K-dimensional entropy proves particularly effective for analyzing multi-level architectures in biological and molecular networks [11].

Through its multidimensional encoding and minimization principles, structural entropy establishes a unified framework for structural analysis and dynamical modeling of complex networks. The one-dimensional variant quantifies global structural complexity, the two-dimensional version identifies mesoscopic community organizations, and higher-dimensional extensions decode hierarchical topologies—applications spanning biological networks, dynamical systems, and network security analyses [10, 11, 12, 13, 14, 15].

3 Approach

Structural entropy, as a metric for quantifying graph structural complexity, necessitates the abstraction of convolutional neural networks (CNNs) into directed weighted graphs. The hierarchical architecture of neural networks inherently aligns with directional edge flows, where weight parameters directly map to edge weights. Sparse adjacency matrices effectively represent localized connectivity patterns. This study therefore establishes a methodology to transform CNNs into directed weighted graphs for structural entropy quantification.

3.1 Graph Abstraction of CNNs

Initial weight normalization maps network weights $w_{ij} \in [w_{min}, w_{max}]$ to a non-negative interval $[\epsilon, 1+\epsilon]$:

$$w'_{ij} = \frac{w_{ij} - w_{min}}{w_{max} - w_{min}} + \epsilon \tag{8}$$

where $\epsilon = 10^{-6}$ prevents zero-weight edges.

To mitigate semantic distortion from normalization, we implement entropy weighting for secondary weight calibration. Given the normalized weight matrix $W' = [w'_{ij}]$, information entropy is computed as:

$$H_j = -\frac{1}{\ln m} \sum_{i=1}^{m} p_{ij} \ln p_{ij} \tag{9}$$

with $p_{ij} = \frac{w'_{ij}}{\sum_{k=1}^{m} w'_{kj}}$, where H_j denotes entropy for the j-th neuron. Lower entropy indicates higher weight dispersion and information content. Weight correction coefficients are derived as:

$$\alpha_j = \frac{1 - H_j}{\sum_{k=1}^{n} (1 - H_k)} \tag{10}$$

where larger α_j signifies greater neuron influence. Final calibrated weights become:

$$\widetilde{w}_{ij} = \alpha_j \cdot w'_{ij} \tag{11}$$

This process transforms neurons into nodes, inter-neuron connections into directed edges, and calibrated weights into edge weights, completing the graph abstraction.

3.2 Structural Entropy Quantification

For directed graphs, one-dimensional structural entropy incorporates in-degree d_v^{in} and out-degree d_v^{out} [6]:

$$\mathcal{H}^1(G) := -\sum_{v\in V} \frac{d_v^{in}}{m} log_2 \frac{d_v^{in}}{m} \tag{12}$$

we can select the in-degree of all nodes in the same time or select the out-degree of all nodes in the same time for calculation.

For weighted directed graphs, layer-wise weight normalization ensures hierarchical consistency. The weights of the first layer have been treated by entropy weight method in advance.

For layer $l > 1$:

$$\widehat{w}_{ij}^{(l)} = \frac{w_{ij}^{(l)}}{\sum_{j\in N_{l+1}} w_{ij}^{(l)}}, \forall i \in N_l, j \in N_{l+1} \tag{13}$$

where $w_{ij}^{(l)}$ denotes weights from layer l to $l+1$. Cross-layer weights are iteratively adjusted:

$$\widehat{w}_{ik}^{(l)} = \widehat{w}_{ij}^{(l-1)} \cdot \widehat{w}_{jk}^{(l)}, \forall i \in N_{l-1}, j \in N_l, k \in N_{l+1} \tag{14}$$

this yields a weighted directed graph encoding inter-layer interactions. Structural entropy is then computed as:

$$\mathcal{H}^1(G) = -\sum_{i=1}^{n} \frac{\widehat{w}_i^{(l)}}{vol(G)} log_2 \frac{\widehat{w}_i^{(l)}}{vol(G)} \tag{15}$$

where $\widehat{w}_i^{(l)} = \sum_j \widehat{w}_{ij}^{(l)}$.

To enhance interpretability, we define the final structural entropy metric as $E(G) = 2^{\mathcal{H}^1(G)}$.

3.3 Phase Identification via Structural Entropy Dynamics

The training process of convolutional neural networks (CNNs) is partitioned into three distinct phases based on structural entropy evolution:

Adjustment Phase: Rapid structural entropy fluctuations with unstable parameter updates.

Convergence Phase: Steady structural entropy reduction driven by systematic optimization.

Specialization Phase: Disordered structural entropy resurgence due to task-specific overfitting.

To establish a quantitative identification framework, we formulate phase transition criteria as follows.

Adjustment Phase Termination. Let $C(t) \in \{0,1\}$ denote the convergence detection function, where $C(t) = 1$ indicates convergence criteria satisfaction. The adjustment phase exit condition is defined:

$$E_{\text{exit}}(t) = \begin{cases} 1 & \text{if} \quad \underbrace{t \geq 10}_{\text{Mandatory termination}} \vee \underbrace{(C(t) = 1 \wedge t < 10)}_{\text{Early convergence}} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

where $t \in \mathbb{N}^+$ represents epoch count. Training forcibly exits the adjustment phase at epoch 10, while early termination occurs if convergence is detected beforehand.

Convergence Phase Termination. The convergence phase exhibits monotonic structural entropy reduction. We combine Exponentially Weighted Moving Average (EWMA) [16, 17] and Local Trend Analysis (LTA) for robust identification:

For time-series data $X = \{x_1, x_2, \dots, x_n\}$:

$$\begin{cases} \hat{x}_1 = x_1 \\ \hat{x}_t = \alpha x_t + (1-\alpha)\hat{x}_{t-1} t \geq 2 \end{cases} \tag{17}$$

where $\alpha \in (0,1)$ controls recent observation weighting.

To mitigate EWMA's inherent response lag and spurious declining trends during stabilization [18], we implement sliding-window regression:

For time-series $S = \{s_1, s_2, \dots, s_n\}$ with window size w:

$$W_t = \{s_{t-w+1}, s_{t-w+2}, \dots, s_t\} \tag{18}$$

First-order linear regression on W_t:

$$\min_{a,b} \sum_{i=1}^{w} (y_i - (a \cdot x_i + b))^2 \tag{19}$$

where $x_i = i$ (time index), $y_i = s_{t-w+i}$. The slope a quantifies local trend intensity or direction.

When the model exits the convergence phase, it enters the specialization phase.

4 Experiment

4.1 Validation of CNN Graph Abstraction

To validate the graph abstraction method proposed in Section 3.1, we trained a ResNet18 model [19, 20] on the CIFAR10 [23] dataset, achieving 94.8% classification accuracy. The original weights were transformed using normalization and entropy weighting (Section 3.1), generating a modified ResNet18 model. Functional equivalence was tested on a 10,000-image subset.

As shown in **Fig. 2**, output layer correlations between the original and modified models ranged from 0.9716 (minimum) to 0.9972 (maximum), with a mean of 0.9958. **Fig. 3** presents representative cases of maximum and minimum correlation scenarios. Both models exhibit nearly identical confidence distributions in their output layers, even in the minimal correlation case. Although slight discrepancies exist between the confidence distributions of the original and modified models in the low-correlation sample, both architectures consistently produce high confidence scores for category 2 and category 6, ultimately yielding identical classification decisions ("category 2") despite distributional variations. This observation indicates that the modified model achieves equivalent sensitivity to challenging samples as the original architecture. These experimental findings demonstrate that our graph structural abstraction method for convolutional neural networks effectively preserves both the geometric characteristics of intermediate feature spaces and the inherent semantic propagation patterns of the original network. The successful conversion of CNNs into directed weighted graphs through this methodology ensures that the calculated structural entropy authentically reflects the essential properties of the original neural architecture.

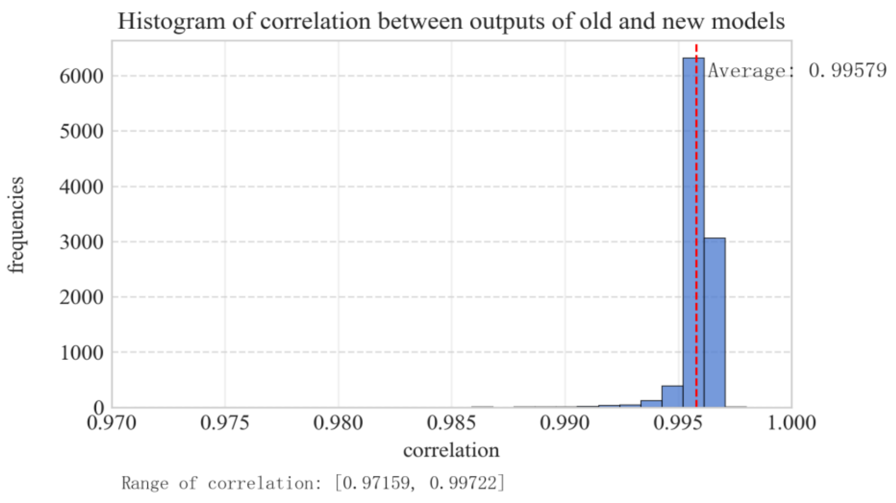

Fig. 2. Correlation between the output results of the original model and the new model on the test set.

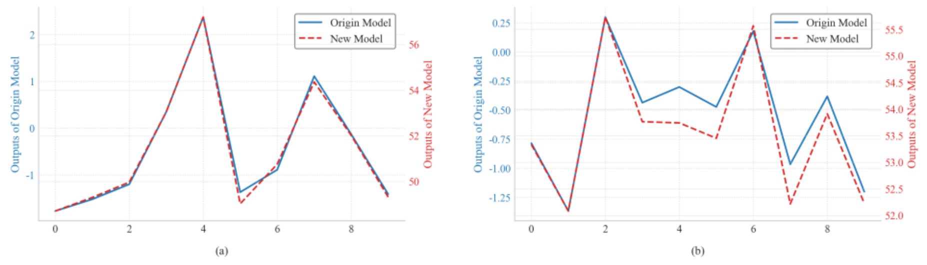

Fig. 3. Subplots a and b show the comparison of the highest and lowest correlation outputs, respectively.

4.2 Structural Entropy Dynamics in CNN Training

Training MobileNetV2 [22] and VGG16 [21] on CIFAR10 [23] (200 epochs, baseline parameters) revealed three-phase structural entropy evolution（**Fig. 4**）:
Adjustment Phase: Short-term entropy fluctuations during initial parameter exploration.

Convergence Phase: Steady entropy reduction driven by systematic optimization.
Specialization Phase: Oscillatory entropy resurgence due to task-specific structural reorganization.

VGG16 exhibited larger entropy variations due to its densely connected architecture and high-dimensional parameter space, whereas MobileNetV2's compact design constrained structural evolution. These observations empirically validate the tri-phase theory.

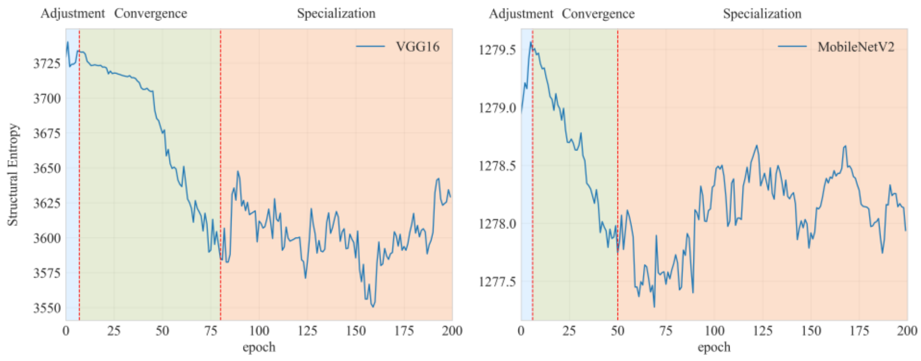

Fig. 4. The left and right figures show the trend of structural entropy change and stage division of VGG16 model and MobileNetV2 model respectively.

4.3 Training Process Characterization

The convergence detection algorithm (Section 3.3) was tested on MobileNetV2 and VGG16. **Fig. 5** demonstrates robust identification of decreasing entropy trends, even under non-monotonic reduction. The algorithm reliably detects stabilization or resurgence, triggering timely phase transitions from convergence to specialization.

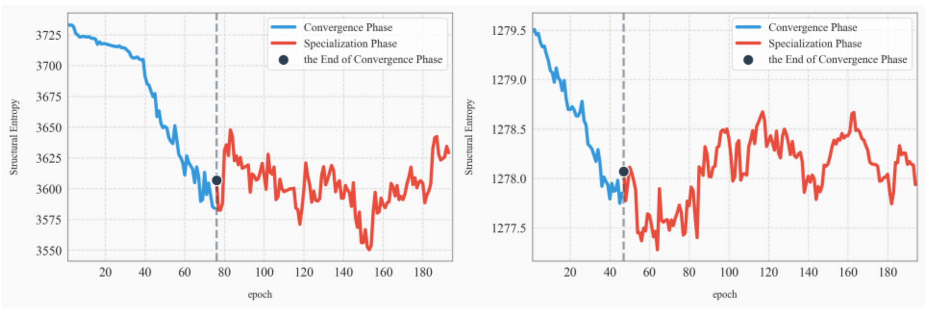

Fig. 5. The left figure and the right figure respectively show the convergence phase judgment results of VGG16 model and MobileNetV2 model under the convergence period monitoring algorithm.

4.4 Structural Entropy-Guided Initial Hyperparameter Configuration

We conducted experiments using the MobileNetV2 model on the CIFAR10 dataset with an initial learning rate of 0.005 and a batch size of 128. By controlling variables, we investigated the evolution of structural entropy under different initial hyperparameter configurations to establish guidelines for hyperparameter tuning.

As shown in **Fig. 6** (a), when the batch size was set to 512, the structural entropy decreased significantly faster than under standard batch sizes, leading the network to enter a low-entropy state prematurely. An excessively large batch size forces the network to prioritize fitting high-frequency features while neglecting low-frequency semantics, thereby reducing its ability to learn non-salient features and resulting in suboptimal model performance. Consequently, if structural entropy declines too rapidly and the network enters a low-entropy state early, the initial batch size should be appropriately reduced.

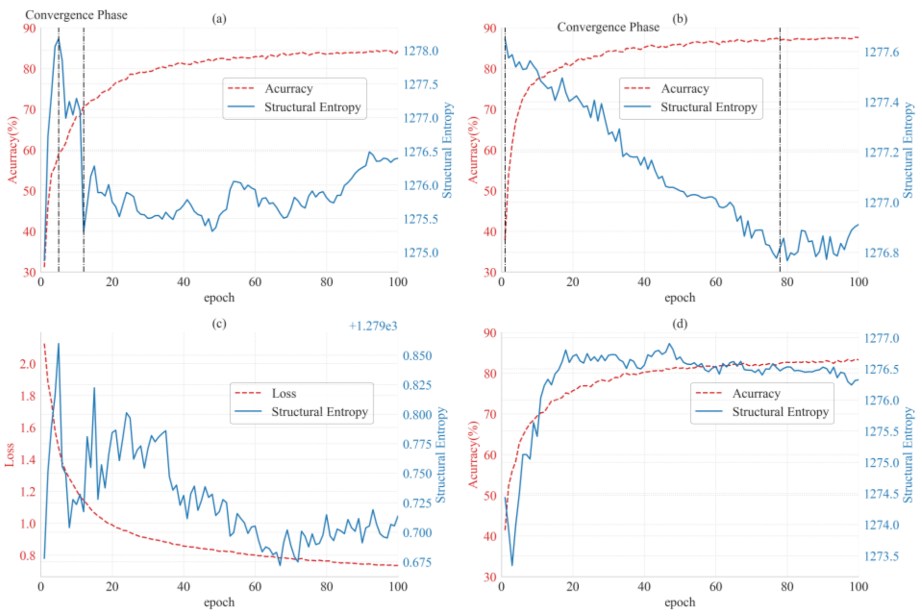

Fig. 6. The change trend of structural entropy and accuracy (Loss) of MobileNetV2 when the Batch size is too large (a) or too small (b), and the learning rate is too high (c) or too low (d).

In **Fig. 6** (b), with a batch size of 32, the convergence phase of structural entropy persisted throughout the entire training process. Under small batch sizes, the continuous decline of structural entropy indicates gradual improvement in the model's representational capacity. Even when validation accuracy stabilizes, extending the training duration may further optimize the model. This contrasts sharply with the rapid entropy reduction observed under large batch sizes. Therefore, starting with a larger batch size and adjusting downward upon detecting abnormal structural entropy trends is a rational strategy.

Fig. 6 (c) illustrates that when the learning rate was set to 0.05 (10× the standard value), the loss curve appeared normal, but structural entropy oscillated within a narrow range without effective reduction. An excessively high initial learning rate causes weight update steps to exceed the smoothness scale of the loss landscape, leading to parameter oscillations. While the average gradient direction still points toward loss reduction, structural entropy—directly reflecting instantaneous layer-wise state changes—sensitively captures such anomalies, enabling early detection and termination to adjust the learning rate.

Fig. 6 (d), with an initial learning rate of 0.001 (1/5 of the standard value), structural entropy rose rapidly from a low baseline and stabilized, while model accuracy converged to a suboptimal level. An overly small learning rate severely restricts parameter updates, manifesting as an early transition into the specialization phase characterized by structural entropy rebound and noise feature learning. Once structural entropy stabilizes at a threshold, neuron response patterns reach maximum disorder under current constraints, resulting in poor model performance. Thus, monitoring abnormal entropy increases during early training necessitates early stopping and increasing the initial learning rate.

4.5 Structural Entropy-Monitored Adaptive Learning Rate Scheduling

When the learning rate is too high, structural entropy becomes trapped in low-amplitude oscillations. We hypothesize that once structural entropy enters a slow-decline or stabilization phase during convergence, reducing the learning rate can prevent premature specialization. On MobileNetV2, we compared three strategies: (1) exponential decay, (2) structural entropy-guided adjustment (lowering the rate upon entropy stabilization or rebound), and (3) a hybrid approach combining early exponential decay with entropy-guided adjustments.

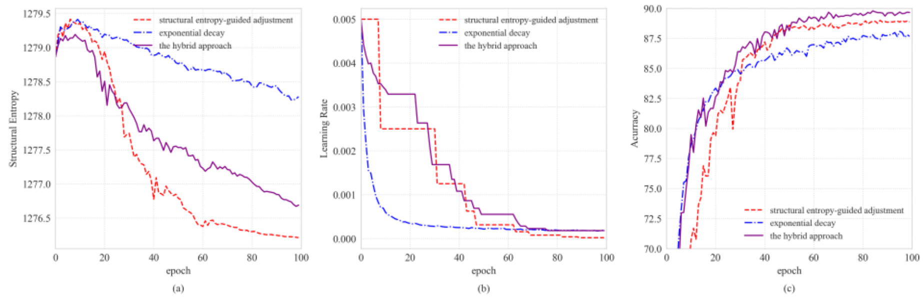

Fig. 7. Figure a, Figure b and Figure c respectively show the change trend of structural entropy, learning rate and accuracy under the three learning rate regulation strategies used in the training process of MobileNetV2 model.

As shown in **Fig. 7.** Figure (a) and (b), the exponential decay strategy caused rapid early learning rate reduction, leading to slow structural entropy decline and inefficient feature learning. In contrast, the other two strategies exhibited pronounced entropy reduction, highlighting structural entropy's sensitivity to local parameter dynamics. **Fig. 7.** Figure (c) compares accuracy across strategies: exponential decay and hybrid strategies achieved faster early accuracy gains, while entropy-guided scheduling surpassed others in later phases. The hybrid strategy combined both advantages, yielding the highest final accuracy.

4.6 Structural Entropy-Driven Optimizer Switching Strategy

We propose an adaptive optimizer switching strategy based on structural entropy evolution [24, 25]. During the adjustment and convergence phases, Adam or AdamW optimizers accelerate parameter exploration. Upon entering the specialization phase (marked by entropy oscillations and flat loss regions), switching to SGD helps escape local optima.

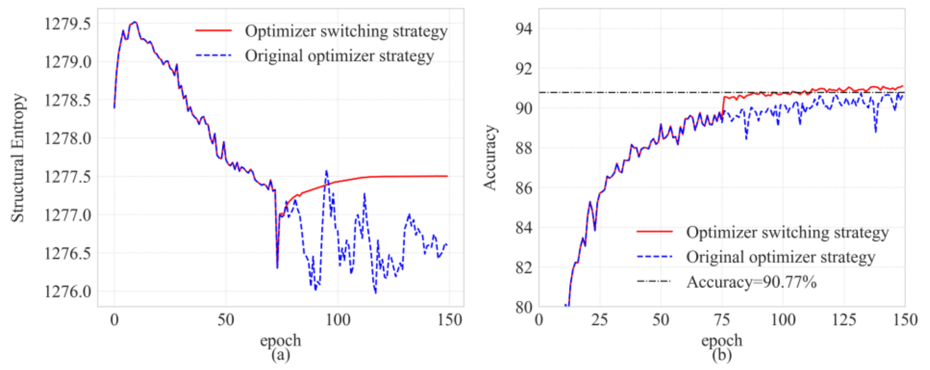

Fig. 8. Figure a and Figure b respectively show the changes in structural entropy and accuracy before and after adopting the adaptive switching optimizer strategy.

In experiments on MobileNetV2 trained on CIFAR10 for 150 epochs (fixed random seed), the control group used AdamW throughout, while the experimental group switched to SGD upon detecting specialization. As shown in **Fig. 8.** Figure (a) and (b), after switching, the experimental group stabilized parameter exploration in new low-loss regions, with structural entropy rising steadily before stabilizing. Stagnant accuracy resumed improvement, achieving a historical best of 90.8% at epoch 100 (vs. 90.03% for the control group). The control group required 149 epochs to reach 90.77% accuracy, whereas the experimental group achieved equivalent performance by epoch 90—a 40% reduction in training time.

5 Conclusion

This study investigates the training dynamics of CNNs through the lens of information complexity by modeling CNN architectures as directed weighted graphs. We employ structural entropy as a quantitative measure to characterize the phased evolution during CNN training. Three distinct stages of structural entropy variation are identified through empirical analysis, leading to the development of a stage discrimination algorithm. Based on real-time monitoring of structural entropy patterns, two adaptive optimization strategies are proposed: 1) an early stopping mechanism for initial hyperparameter adjustment during the preliminary training phase, and 2) dynamic learning rate adaptation combined with optimizer switching during sustained training. Experimental results demonstrate that these entropy-guided strategies effectively enhance model convergence efficiency. Future research directions include extending structural entropy analysis to other deep neural architectures and investigating its potential implications for neural network interpretability.

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

References

1. Heaton, J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. Genet Program Evolvable Mach 19, 305–307 (2018). https://doi.org/10.1007/s10710-017-9314-z
2. Krizhevsky, A., Sutskever, I., & Hinton, G. E. ImageNet classification with deep convolutional neural networks. Communications of the ACM 60, 84–90 (2012).
3. Tishby, N., & Zaslavsky, N. Deep learning and the information bottleneck principle. In 2015 IEEE Information Theory Workshop (ITW) 1–5 (IEEE, 2015).
4. Saxe, A. M. et al. On the information bottleneck theory of deep learning. Journal of Statistical Mechanics: Theory and Experiment 2019, 124020 (2019).
5. Bianchini, M., & Scarselli, F. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. IEEE Transactions on Neural Networks and Learning Systems 25, 1553–1565 (2014).
6. Li, A., & Pan, Y. Structural information and dynamical complexity of networks. IEEE Transactions on Information Theory 62, 3290–3339 (2016).
7. Welling, M., & Teh, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. In Proceedings of the 28th International Conference on Machine Learning (ICML) 681–688 (2011).
8. Li, Q., Tai, C., & Weinan, E. Stochastic modified equations and adaptive stochastic gradient algorithms. In Proceedings of the 32nd International Conference on Machine Learning (ICML) 1–9 (2015).
9. Mandt, S., Hoffman, M. D., & Blei, D. M. Stochastic gradient descent as approximate Bayesian inference. Journal of Machine Learning Research 18, 1–35 (2017).
10. Shannon, C. E. A mathematical theory of communication. The Bell System Technical Journal 27, 623–656 (1948).
11. Li, A. et al. Decoding topologically associating domains with ultra-low resolution Hi-C data by graph structural entropy. Nature Communications 9, 3265 (2018).

12. Wu, J., Chen, X., Xu, K., & Li, S. Structural entropy guided graph hierarchical pooling. In Proceedings of the 39th International Conference on Machine Learning (ICML) 1–15 (2022).
13. Liu, Y. et al. REM: From structural entropy to community structure deception. In Advances in Neural Information Processing Systems 32 (NeurIPS) 1–10 (2019).
14. Wu, J. et al. A simple yet effective method for graph classification. In Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI) 1–7 (2022).
15. Zeng, X., Peng, H.-L., & Li, A. Effective and stable role-based multi-agent collaboration by structural information principles. In Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI) 1–9 (2023).
16. Hunter, J. S. The exponentially weighted moving average. Journal of Quality Technology 18, 203–210 (1986).
17. Lucas, J. M., & Saccucci, M. S. Exponentially weighted moving average control schemes: Properties and enhancements. Quality Engineering 36, 31–32 (1990).
18. Xie, Y. et al. Local trend analysis method of hydrological time series based on piecewise linear representation and hypothesis test. Journal of Cleaner Production 380, 134891 (2022).
19. He, K. et al. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 770–778 (IEEE, 2016).
20. He, K. et al. Identity mappings in deep residual networks. In Computer Vision – ECCV 2016 630–645 (Springer, 2016).
21. Simonyan, K., & Zisserman, A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).
22. Sandler, M. et al. MobileNetV2: Inverted residuals and linear bottlenecks. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 4510–4520 (IEEE, 2018).
23. Krizhevsky, A. Learning multiple layers of features from tiny images. Technical Report (University of Toronto, 2009).
24. Reddi, S. J., Kale, S., & Kumar, S. On the convergence of Adam and beyond. arXiv preprint arXiv:1904.09237 (2018).
25. Keskar, N. S., & Socher, R. Improving generalization performance by switching from Adam to SGD. arXiv preprint arXiv:1712.07628 (2017).