ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar1605

Identification of Blood Vessels from Anigography Images Using CNN

Dr. Sajja Suneel[1]; B. Shiva Kumar[2]; G. Siddhartha[3]; V. Suresh[4]

[1]Department of CSE (DS) Institute of Aeronautical Engineering Hyderabad, India
[2]Department of CSE (DS) Institute of Aeronautical Engineering Hyderabad, India
[3]Department of CSE (DS) Institute of Aeronautical Engineering Hyderabad, India
[4]Department of CSE (DS) Institute of Aeronautical Engineering Hyderabad, India

Publication Date: 2026/04/08

Abstract: Accurate and early detection of blood vessel block-ages is crucial for diagnosing cardiovascular diseases. This work presents a deep learning-based approach for identifying stenosis in angiography images using the YOLO (You Only Look Once) object detection algorithm integrated with a Convolutional Neural Network (CNN) framework. The proposed system is trained using a dataset of angiography images obtained from the Mendeley repository. The YOLO-CNN model is designed to detect and classify blood vessel blockades by processing image inputs and predicting bounding boxes around affected areas, along with their classification as "Stenosis" or "No Blockade." A Django-based web application is developed to facilitate user regis-tration, dataset upload, model training, and real-time prediction. The model achieves a mean Average Precision (mAP) exceeding 90%, demonstrating robust performance in detecting various stages of vascular blockages. This automated detection system not only highlights the blockade region with bounding boxes but also calculates the blockage area, aiding medical professionals in evaluating the severity of the condition. The system ensures efficient, accurate, and user-friendly diagnostics through its web interface and can be extended to support clinical decision-making processes.

Keywords: *Convolutional Neural Networks, Image Classification, Deep Learning, Computer Vision, Performance Evaluation.*

How to Cite: Dr. Sajja Suneel; B. Shiva Kumar; G. Siddhartha; V. Suresh (2026) Identification of Blood Vessels from Anigography Images Using CNN. *International Journal of Innovative Science and Research Technology*, 11(3), 3529-3536. https://doi.org/10.38124/ijisrt/26mar1605

I. INTRODUCTION

Cardiovascular diseases (CVDs) remain the leading cause of death worldwide, with stenosis— the narrowing of blood vessels—playing a major role in reducing blood flow and triggering conditions such as heart attacks and strokes. Early and accurate detection of these blockages is essential to prevent severe complications, yet the manual interpretation of coronary angiography images is often slow, subjective, and dependent on expert radiologists.

With recent advancements in artificial intelligence, deep learning models—especially Convolutional Neural Networks Identify applicable funding agency here. If none, delete this (CNNs)—have transformed medical image analysis. CNNs are highly effective at learning spatial patterns directly from images, but traditional architectures struggle with real-time detection tasks. YOLO (You Only Look Once), however, overcomes this limitation by performing object detection and classification in a single pass, making it fast, accurate, and suitable for clinical applications.

In this work, we present a YOLO-based CNN system de-signed to automatically detect and classify stenosis in angiog-raphy images. Trained on a labeled dataset from the Mendeley repository, the model identifies regions of interest, draws bounding boxes around suspected blockages, and classifies them as either "Stenosis" or "No Blockade." The system also calculates the estimated area of the affected region, providing quantitative support for clinical decision-making.

To ensure accessibility, we developed an interactive Django-based web application that allows users to register, load datasets, train the model, and upload angiography images for real-time prediction. This end-to-end platform aims to reduce diagnostic delays, improve accuracy, and support healthcare professionals—especially in settings where expert analysis may not be readily available.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar1605

II. LITERATURE REVIEW

➢ *From Biological Inspiration to Deep Learning Break-Throughs*

The foundations of Convolutional Neural Networks (CNNs) trace back to biological vision research. Hubel and Wiesel's work in 1962 revealed that the human visual cortex processes information hierarchically—an idea that directly inspired the design of early neural vision models. Building on this concept, Fukushima introduced the Neocognitron in 1980, a pioneering architecture featuring local receptive fields and layered feature extraction. This model laid the groundwork for modern CNNs. The first widely recognized practical CNN, LeCun's LeNet-5 (1989), demonstrated impressive performance on handwritten digit recognition using MNIST. It combined convolution, pool-ing, and backpropagation, marking a major shift in machine-learning-based vision.

The deep learning revolution gained momentum in the mid-2000s. Hinton and Salakhutdinov (2006) addressed the van-ishing gradient problem, while GPUs made training large net-works feasible. AlexNet (2012) was a turning point—winning the ImageNet challenge through innovations like ReLU acti-vation, dropout, and data augmentation. Successive architec-tures continued this trend: VGG (2014) emphasized deeper networks with simple 3×3 filters, Inception (2015) introduced multi-scale feature extraction, and ResNet (2016) solved deep network degradation using skip connections.

➢ *Modern Improvements, Efficiency Models, and Emerging Challenges*

To make CNNs efficient on mobile and edge devices, architectures like MobileNet (2017) and EfficientNet (2019) focused on lightweight depthwise convolutions and balanced scaling. Training stability also improved due to techniques such as batch normalization (Ioffe & Szegedy, 2015) and extensive data augmentation (Shorten & Khoshgoftaar, 2019). Transfer learning (Yosinski et al., 2014) became a practical standard, allowing models pre-trained on large datasets like ImageNet to be adapted to new tasks with limited data.

While CNNs dominate many vision tasks, challenges such as model interpretability, robustness to noise, and learning from very few examples remain open areas of research. Vision Transformers (Dosovitskiy et al., 2020) have recently emerged as strong competitors, showing promise on large-scale datasets while also complementing CNNs in smaller-data scenarios.

III. METHODOLOGY

The proposed system employs a deep learning–based frame-work for automated detection and classification of stenosis in coronary angiography images. The pipeline consists of dataset acquisition, preprocessing, model development using a YOLO-based Convolutional Neural Network (CNN), training, infer-ence, and integration into a Django-based web application. The objective is to accurately localize vascular blockages and classify them while providing interpretable visual outputs for clinical use.

➢ *Dataset Collection*

The dataset utilized in this study was obtained from the Mendeley Data Repository, comprising 8,135 angiographic images collected from 100 subjects. Each image is manually annotated with bounding boxes identifying regions of interest corresponding to normal vessels or stenotic segments. The dataset includes two diagnostic classes:

- No Blockade (Normal)
- Stenosis (Blocked Vessel)

These annotations serve as ground truth for supervised object detection.

➢ *Preprocessing Pipeline*

For angiography, preprocessing includes resolution stan-dardization, grayscale conversion, pixel normalization, and conversion of bounding box annotations into YOLO format (class index, normalized coordinates). For MNIST, pixel in-tensity normalization to [0,1] ensures stable gradient behavior. Additional augmentation (flip, rotation, shift, zoom, brightness variation) supports robustness when evaluating CNN models on more challenging datasets such as CIFAR-10.

➢ *Model Architectures*

The primary stenosis detection model is based on YOLOv8, using a CSPDarknet backbone, PANet neck, and a detection head capable of simultaneous localization and classification. The system predicts bounding boxes, objectness scores, and stenosis labels.

- *To Benchmark CNN Performance, Multiple Architectures are Implemented:*

✓ Baseline CNN: 3 convolutional layers (32–64–128 fil-ters), pooling, dropout-regularized dense layers.
✓ LeNet-5 Inspired: Two convolutional layers (6–16 filters) followed by pooling and fully connected layers.
✓ Deep CNN: Six convolutional blocks (64–256 filters), batch normalization, global pooling, dropout.
✓ ResNet-Inspired: Four residual blocks (64–512 filters) enabling deeper feature representations.

➢ *Training Configuration*

The YOLO model is trained for 100 epochs using SGD with momentum, batch size 16, and a learning rate of 0.001. For CNN models, a unified training setup is used: batch size 32, Adam optimizer (β_1 = 0.9, β_2 = 0.999), categorical crossentropy loss, and a learning-rate decay schedule (factor 0.5, patience 10, minimum 1e-7). Early stopping prevents overfitting.

➢ *Hardware and Implementation*

Training is conducted on an NVIDIA RTX 3080 GPU with an Intel i7-10700K CPU, 32GB RAM, and NVMe SSD storage, ensuring fast experimentation and reproducible scalability.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar1605

➢ *Inference, Detection, and Deployment*

During inference, YOLO identifies stenosis regions, predicts labels, and calculates bounding-box area to estimate severity. Segmented regions are extracted using OpenCV. A Django web interface enables dataset upload, model training visual-ization, and real-time prediction, providing annotated images with labels and confidence scores.

➢ *Evaluation Metrics*

Performance is assessed using accuracy, loss, Precision, Re-call, F1-Score, mAP, Top-1 error rate, and confusion matrices, offering a detailed breakdown of classification and localization quality.

IV. IMPLEMENTATION

Early and accurate detection of vascular blockages such as stenosis is essential for preventing severe cardiovascular com-plications, yet traditional diagnostic workflows often depend heavily on manual interpretation of angiographic scans. These procedures are prone to human error, require considerable expertise, and are not well suited for real-time decision-making. To overcome these limitations, this study proposes a unified, fully automated, and real-time system that leverages the strengths of YOLO (You Only Look Once) object detection and Convolutional Neural Networks (CNNs) to identify, clas-sify, and localize stenosis directly from angiographic images. Unlike conventional approaches that rely on separate machine-learning modules for detection, segmentation, and classification, the proposed system integrates all stages into a seamless end-to-end pipeline. Once an angiography image is provided, the system instantly analyzes it to determine whether a vascular blockage is present, identifies its location, classifies it as “Stenosis” or “No Blockade,” and overlays intuitive visual markers on the image. To make the output more clinically meaningful, the system also computes the area of the detected stenotic region using geometric calculations based on bounding box coordinates. This quantitative measure assists clinicians in assessing the severity of the narrowing—an important factor when deciding on treatment strategies.

➢ *System Architecture*

The architecture comprises three major components.

- *YOLO-Based CNN Detection Model:* The backbone of the system is the YOLOv8 object detection framework, recog-nized for its high accuracy and low inference latency. YOLOv8 accepts angiography images as input and produces bounding boxes, confidence values, and diagnostic labels. Its single-pass design enables real-time performance, making it suitable for clinical environments where rapid analysis is crucial.
- *Segmentation Module:* After the detection stage, the specific region of interest (ROI) corresponding to the predicted blockage is extracted from the original image. For improved interpretability, the ROI is converted to grayscale, emphasiz-ing vessel structures and enhancing the visibility of stenotic patterns. This step is particularly valuable for clinicians who rely on clear visual cues.
- *Blockage Area Estimation:* To provide a quantitative assessment, the system calculates the area of each detected bounding box using the formula:

$$\text{Area} = (X_{\max} - X_{\min}) \cdot (Y_{\max} - Y_{\min}) \quad (1)$$

By estimating the size of the blockage, the system helps classify stenosis levels as mild, moderate, or severe. This adds an analytical dimension to the detection process, which is beneficial for patient triaging and clinical planning.

➢ *Dataset and Training*

The model is trained on 8,135 manually annotated an-giographic images from the Mendeley dataset. Each sample includes bounding boxes indicating stenosis regions and labels identifying whether the vessel is blocked or normal. Training is performed using the Ultralytics YOLOv8 environment with the following settings:

- Epochs: 100
- Optimizer: Stochastic Gradient Descent (SGD)
- Learning Rate: 0.001
- Loss Functions: Generalized IoU (GIoU), Objectness Loss, and Classification Loss

These hyperparameters were selected based on empirical testing to achieve optimal precision, recall, and mAP. The final model weights are saved as best.pt for deployment.

➢ *Django-Based Web Integration*

To make the system accessible to healthcare professionals, a web-based interface powered by Django has been developed. The platform includes modules for:

- User Registration and Login
- Dataset Loader: Displays dataset statistics and available classes
- Model Training Dashboard: Shows live training metrics including accuracy and loss curves
- Prediction Module: Allows uploading test images and receiving annotated outputs in real time

Outputs include classification labels, bounding boxes, seg-mented ROIs, and computed blockage areas, all displayed in an intuitive, user-friendly interface.

➢ *Output Visualization and Advantages*

Output visuals are generated using OpenCV and matplotlib and returned to the user in base64 format for seamless browser rendering. Stenosis regions are highlighted with green bounding boxes and confidence labels (e.g., “Stenosis: 0.92”), while the segmented grayscale ROI is displayed alongside the original image.

- *This Unified Approach Offers Several Advantages:*

✓ Real-time detection with minimal computational delay
✓ End-to-end processing without the need for separate detection/classification modules

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar1605

- ✓ Clinically relevant insights, including blockage area estimation
- ✓ Web accessibility, enabling remote clinical analysis and integration into hospital systems

Key benefits include real-time performance, a unified pipeline, clinically meaningful outputs, and accessible deploy-ment through the web interface.

V. RESULTS AND EVALUATION

This section presents a comprehensive evaluation of several Convolutional Neural Network (CNN) architectures applied to two widely used benchmark datasets: MNIST and CIFAR-10. All experiments were conducted under controlled and identical conditions to ensure fairness, reproducibility, and a meaningful model comparison. Each model was trained for up to 100 epochs with early stopping using validation accuracy as the stopping criterion. This guarantees that no model overfits excessively or benefits unfairly from extended training.

The goal of this experimentation is to understand how model depth, architectural design, and parameter count influence overall accuracy, generalization ability, computational effi-ciency, and robustness. The results reveal consistent patterns: while deeper and more expressive models generally achieve higher accuracy, smaller networks remain highly parameter-efficient and computationally lightweight, making them attrac-tive for deployment in resource-constrained environments.

➢ *Results on MNIST Dataset*

The MNIST dataset provides a relatively clean and simple benchmark for evaluating core classification ability. Because the dataset consists of grayscale 28×28 digit images, it is not visually complex, but it remains an essential baseline for verifying CNN implementation correctness and training stability.

Across all architectures tested, performance on MNIST was remarkably high, with all models achieving above 98% accuracy. The Deep CNN model performed the best, reaching 99.21% accuracy, followed closely by the ResNet-inspired model at 99.18%. Even the simpler LeNet-5 architecture, de-spite being one of the earliest convolutional designs, achieved an impressive 98.87%, demonstrating that MNIST does not require extremely deep architectures to achieve near-perfect classification performance.

- *Training Time and Parameter Efficiency:* Training times ranged from 4 to 16 minutes, reflecting the relative complexity of each model. LeNet-5, with only 62,006 parameters, trained in just 4.1 minutes, while the Deep CNN required 15.7 minutes due to its 2.8 million parameters. This trade-off highlights an important trend: small networks are ideal for rapid prototyping and deployment on limited hardware, whereas deeper networks achieve marginally higher accuracy at the cost of greater computational demand.
- *Precision, Recall, F1-Score:* All models maintained nearly identical precision, recall, and F1-Score values, with differences only in the third decimal place. This reflects MNIST's low complexity and balanced class distribution. The Deep CNN slightly outperformed others with 0.9922 precision and 0.9921 recall, indicating stronger consistency across all digits.

Table 1 MNIST Performance Results

Model	Accuracy (%)	Precision	Recall	F1-Score	Train Time (min)	Parameters
Baseline CNN	99.12 ± 0.08	0.9913	0.9912	0.9912	8.2 ± 0.5	1,663,370
LeNet-5	98.87 ± 0.12	0.9888	0.9887	0.9887	4.1 ± 0.3	62,006
Deep CNN	99.21 ± 0.06	0.9922	0.9921	0.9921	15.7 ± 1.2	2,847,562
ResNet	99.18 ± 0.09	0.9919	0.9918	0.9918	12.3 ± 0.8	2,156,874

- *MNIST Summary:* MNIST results demonstrate that although deeper architectures provide small improvements, lightweight networks such as LeNet-5 achieve extremely strong results with far fewer parameters. For educational, embedded, or low-resource applications, such models remain practical and highly effective.

➢ *Results on CIFAR-10 Dataset*

Unlike MNIST, the CIFAR-10 dataset represents a signif-icantly more difficult benchmark. It contains 60,000 color images across ten diverse classes, including animals and vehicles, with considerable intra-class variability and low resolution. CIFAR-10 is therefore more representative of real-world classification challenges.

- *Accuracy Comparison:* The deepest model again achieved the best performance:

- ✓ Deep CNN: 94.23%
- ✓ ResNet-inspired: 92.87%
- ✓ Baseline CNN: 87.34%
- ✓ LeNet-5: 76.22%

The substantial drop in performance for LeNet-5 indicates that earlier architectures lack the representational capacity needed for complex natural images. Meanwhile, the ResNet design, with skip connections allowing better gradient flow, brings significant improvements compared to the Baseline CNN.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar1605

- *Confusion Patterns and Hard Classes:* CIFAR-10's animal classes (especially cat, dog, and deer) produced the highest misclassification rates. This is consistent with existing literature, as these classes contain subtle visual differences and often share similar textures or poses. Vehicle classes such as airplane and automobile achieved near-perfect classification in deeper models, benefiting from stronger edge and shape consistency.
- *Training Efficiency and GPU Usage:* Further analysis revealed that deeper models consume substantially more GPU memory and training time. The Deep CNN required 4.3GB VRAM and 45.7 minutes, roughly ten times longer than LeNet-5 (0.8GB VRAM and 4.1 minutes). This suggests that model depth must be carefully selected based on the deployment environment.
- *Parameter Efficiency:* A key insight is that parameter count does not linearly translate to accuracy. The Deep CNN achieved 94.23% accuracy with 4.2 million parameters, whereas the ResNet-inspired model achieved 92.87% accuracy with 3.7 million parameters — a relatively small difference considering the reduced parameter load.

In contrast, LeNet-5 achieved 76.22% accuracy with only 84,726 parameters, making it exceptionally parameter-efficient despite its lower performance ceiling.

Table 2 Cifar-10 Performance Results

Model	Accuracy (%)	Precision	Recall	F1-Score
LeNet-5	87.34 ± 0.31	0.8751	0.8734	0.8739
Baseline CNN	76.22 ± 0.48	0.7658	0.7622	0.7634
Deep CNN	94.23 ± 0.18	0.9428	0.9423	0.9425
ResNet	92.87 ± 0.22	0.9295	0.9287	0.9291

➢ *Comparison With Existing Literature*

The performance of the proposed Deep CNN model is com-petitive with many established architectures, such as ResNet-20 (91.25% on CIFAR-10). Achieving 94.23% with a simpler design demonstrates the effectiveness of optimization tech-niques and well-chosen hyperparameters.

➢ *Error Analysis and Robustness*

The most common misclassifications arise from visually similar classes. However, robustness tests showed promising stability: introducing noise, brightness shifts, and small rota-tions only reduced accuracy by 2–3%, demonstrating strong generalization despite dataset complexity.

Statistical tests (t-tests, p¡0.001) confirm that differences between models are statistically significant.

Table 3 Model Efficiency and Parameter Comparison

Model	Parameters	Accuracy (%)	Param Efficiency
LeNet-5	84,726	76.22	899.4
Baseline CNN	1,686,090	87.34	51.8
ResNet	3,742,186	92.87	24.8
Deep CNN	4,263,498	94.23	22.1

➢ *Overall Findings*

- Deep models consistently outperform shallow ones on complex datasets.
- Smaller networks remain attractive for lightweight, real-time, or embedded systems.
- Augmentation and normalization significantly boost accuracy.
- MNIST is nearly saturated (¿99%) for all models, while CIFAR-10 remains a meaningful benchmark for evaluating architectural improvements.

VI. DISCUSSION

The most prominent finding emerging from this study is the strong and consistent correlation between architectural complexity and classification performance, particularly when models are evaluated on visually challenging, high-variability datasets. While all evaluated CNN architectures achieved exceptional results on the MNIST dataset—each surpassing 98.8% accuracy—the performance landscape shifted drasti-cally when the same models were applied to the more complex CIFAR-10 dataset. This widened performance gap highlights the limitations of shallow and early-generation architectures and emphasizes the critical importance of depth, representational capacity, and modern architectural enhancements when operating on real-world images.

MNIST, a grayscale handwritten digit dataset, contains relatively low intra-class variance and simple structural features. As a result, even lightweight models such as LeNet-5, originally designed in the 1990s, continue to perform remark-ably well. Since the dataset relies heavily on straightforward pixel-level patterns (strokes, curves, edges), deeper feature hierarchies are not strictly necessary to achieve high-accuracy classification. This is precisely why the performance difference between LeNet-5 (98.87%) and

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar1605

the Deep CNN (99.21%) is minimal—MNIST is effectively "solved," and architectural improvements yield marginal gains.

However, CIFAR-10 presents significantly more complex-ity. It includes natural color images of animals and vehi-cles, with substantial variation in lighting, pose, orientation, and background clutter. These inherent complexities expose the limitations of shallow architectures like LeNet-5, which achieved only 76.22% accuracy on CIFAR-10. In contrast, the Deep CNN, equipped with six convolutional layers, batch normalization, and advanced regularization, achieved a far superior accuracy of 94.23%. The ResNet-inspired model also performed strongly at 92.87%, demonstrating how skip connections help preserve gradient flow and enable deeper networks to learn stable representations.

The core reason deeper architectures excel lies in their ability to develop hierarchical feature representations. Early convolutional layers typically learn simple features such as edges and textures. As depth increases, the network progres-sively learns more abstract representations—curves, contours, object parts, and finally class-specific patterns. This hier-archical progression is particularly vital when dealing with objects that share similar color distributions or textures, such as cats and dogs in CIFAR-10. The Deep CNN and ResNet-inspired models can discriminate subtle differences due to their enhanced feature extraction depth, whereas shallow networks struggle to capture these nuances.

Another major contributor to the performance advantage of deeper architectures is the use of batch normalization and data augmentation. Ablation studies revealed that augmentation improved performance by +4.89% and batch normalization contributed another +2.78% on CIFAR-10. Augmentation in-creases the diversity of training samples, allowing the model to generalize better to real-world image variations such as shifts, rotations, and brightness changes. Batch normalization stabilizes training by reducing internal covariate shift, enabling faster convergence and allowing higher learning rates without destabilizing gradients. Together, these components form the backbone of modern deep learning pipelines and are essential when training on complex visual datasets.

However, the advantages of deeper networks come with computational trade-offs. Training the Deep CNN required approximately 4.3 GB of GPU memory and 45.7 minutes, compared to only 0.8 GB and 4.1 minutes for LeNet-5. This highlights the practical challenge: while deeper networks offer superior accuracy, they demand significantly more com-putational resources. In real-world scenarios where models must be deployed on edge devices, mobile platforms, or embedded biomedical systems, memory, power consumption, and inference latency become critical constraints. Therefore, the decision to use a deeper model must be guided by the target deployment environment, not only accuracy considerations.

The results also underscore the importance of under-standing model efficiency, particularly when optimizing for performance-per-parameter. LeNet-5, despite its lower accu-racy on CIFAR-10, remains extremely parameter-efficient, offering 76.22% accuracy with only 84,726 parameters. In contrast, the Deep CNN's 4.26 million parameters achieve 94.23% accuracy. This stark difference highlights how efficiency metrics can inform deployment decisions: what is optimal for a high-performance server may not be optimal for a low-power embedded system.

Beyond architecture and efficiency, the experiments also revealed important challenges and limitations. The hyperpa-rameter search space—learning rate, batch size, optimizer choice, regularization, and augmentation settings—remains vast and sensitive. Even small deviations from optimal learning rate led to noticeable decreases in accuracy, revealing that successful training depends heavily on careful tuning. Addi-tionally, variance across random seeds suggests the importance of training multiple runs to ensure robust, reproducible find-ings. Datasets such as MNIST and CIFAR-10, while widely used as benchmarks, remain idealized and do not capture the full challenges of real-world image distributions, where noise, occlusion, domain shifts, and labeling inconsistencies are common.

The limitations of this study include the focus on only a sub-set of CNN architectures; more advanced models such as Efficient Net, MobileNetV3, DenseNet, and Vision Transformers could provide additional insight into scaling laws and archi-tectural trade-offs. Furthermore, the evaluation mostly centers on classification metrics such as accuracy, precision, recall, and F1-score, which, although informative, may not fully capture robustness or calibration aspects needed in safety-critical applications like medical imaging or autonomous sys-tems. Hardware constraints also played a role; models were optimized on a single GPU configuration, meaning results could vary under different hardware or distributed setups.

In practical terms, the study provides clear guidelines for researchers and practitioners. For resource-constrained environments such as embedded systems, IoT devices, and mobile diagnostic tools, shallow or moderately deep architectures (e.g., LeNet-5 variants or compact CNNs) remain attractive due to their low memory footprint and fast inference. When ac-curacy is the top priority—such as in offline processing, cloud-based services, or high-stakes applications—deeper architec-tures like the Deep CNN or ResNet variants are preferable. The combination of Adam optimization, batch normalization, strong augmentation, and adaptive learning rate schedules emerged as a powerful recipe for achieving high accuracy while maintaining training stability.

For future research, several promising directions emerge naturally from the findings. First, Vision Transformers (ViTs) offer a new paradigm in image recognition that could out-perform CNNs on larger datasets. Second, improving robust-ness through adversarial training, test-time augmentation, and noise-resilient architectures

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar1605

would make models better suited for real-world deployment. Third, exploring efficiency optimizations such as pruning, quantization, distillation, mixed-precision training, and neural architecture search (NAS) could help achieve a better accuracy-efficiency balance. Finally, evaluating these models in real-world applications—such as medical image analysis, anomaly detection, or real-time surveillance—would validate their performance beyond curated benchmarks.

Overall, the expanded findings reinforce a fundamental insight in deep learning research: architectural depth and modern training techniques significantly boost performance on complex datasets, but this improvement must be balanced against computational efficiency, hardware constraints, and application-specific requirements.

VII. CONCLUSION

Convolutional Neural Networks (CNNs) continue to demon-strate exceptional capability in image classification, and the results of this study reinforce the strong link between architec-tural depth and performance. When evaluated on the CIFAR-10 dataset, deeper and more expressive models achieved significantly higher accuracy, with the Deep CNN reaching 94.23%, compared to only 76.22% achieved by simpler archi-tectures such as the LeNet-5–inspired model. This substantial performance gap highlights the critical role of hierarchical feature extraction, where deeper networks learn progressively richer representations capable of distinguishing subtle visual differences in natural images.

However, superior performance comes with a trade-off. Deeper models incur higher computational cost, require more parameters, and consume considerably more GPU memory and training time. For example, the Deep CNN consumes over 4 GB VRAM and requires more than 45 minutes to train, whereas smaller models train in just a few minutes with minimal memory usage. This underscores the importance of parameter efficiency, especially for real-world deployment where hardware resources, latency requirements, and power constraints are major considerations. In such cases, simpler CNNs—while less accurate—can be far more practical.

Modern training components also proved essential for achieving competitive performance. Batch normalization con-tributed approximately +2.78%, improving training stability and enabling deeper networks to converge more effectively. Data augmentation provided an even larger boost of +4.89%, helping models generalize better by artificially increasing the diversity of the training set. Together, these techniques are foundational for modern deep learning pipelines.

Overall, this work contributes to systematic CNN evaluation by highlighting the importance of balancing accuracy with computational efficiency. The results provide clear guidance for selecting architectures based on deployment needs rather than accuracy alone. Future research can extend this by ex-ploring hybrid architectures, transformer-CNN combinations, and validating models on real-world datasets with greater variability.

REFERENCES

[1]. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ”ImageNet: A Large-Scale Hierarchical Image Database,” in *Proc. IEEE CVPR*, 2009.

[2]. A. Dosovitskiy *et al.*, ”An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” *arXiv:2010.11929*, 2020.

[3]. K. Fukushima, ”Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position,” *Biol. Cybern.*, 1980.

[4]. K. He, X. Zhang, S. Ren and J. Sun, ”Deep Residual Learning for Image Recognition,” in *Proc. IEEE CVPR*, 2016.

[5]. G. Hinton and R. Salakhutdinov, ”Reducing the Dimensionality of Data with Neural Networks,” *Science*, 2006.

[6]. A. Howard *et al.*, ”MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” *arXiv:1704.04861*, 2017.

[7]. G. Huang, Z. Liu, L. van der Maaten and K. Weinberger, ”Densely Connected Convolutional Networks,” in *Proc. IEEE CVPR*, 2017.

[8]. D. Hubel and T. Wiesel, ”Receptive Fields, Binocular Interaction and Functional Architecture in the Cat’s Visual Cortex,” *J. Physiol.*, 1962.

[9]. S. Ioffe and C. Szegedy, ”Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in *Proc. ICML*, 2015.

[10]. A. Krizhevsky, I. Sutskever and G. Hinton, ”ImageNet Classification with Deep Convolutional Neural Networks,” in *Proc. NeurIPS*, 2012.

[11]. Y. LeCun, B. Boser, J. Denker *et al.*, ”Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Computation*, 1989.

[12]. Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, ”Gradient-Based Learning Applied to Document Recognition,” *Proc. IEEE*, 1998.

[13]. M. Lin, Q. Chen and S. Yan, ”Network in Network,” *arXiv:1312.4400*, 2013.

[14]. O. Russakovsky *et al.*, ”ImageNet Large Scale Visual Recognition Challenge,” *IJCV*, 2015.

[15]. T. Shorten and T. Khoshgoftaar, ”A Survey on Image Data Augmentation for Deep Learning,” *J. Big Data*, 2019.

[16]. K. Simonyan and A. Zisserman, ”Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv:1409.1556*, 2014.

[17]. C. Szegedy *et al.*, ”Going Deeper with Convolutions,” in *Proc. IEEE CVPR*, 2015.

[18]. M. Tan and Q. Le, ”EfficientNet: Rethinking Model Scaling for Convo-lutional Neural Networks,” in *Proc. ICML*, pp. 6105–6114, 2019.

[19]. J. Yosinski, J. Clune, Y. Bengio and H. Lipson, ”How Transferable Are Features in Deep Neural Networks?” in *Advances in NeurIPS*, 2014.

[20]. S. Zagoruyko and N. Komodakis, ”Wide

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar1605

Residual Networks," *arXiv:1605.07146*, 2016.
[21]. C. Zhang, S. Bengio, M. Hardt, B. Recht and O. Vinyals, "Understanding Deep Learning Requires Rethinking Generalization," *arXiv:1611.03530*, 2016.
[22]. I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.
[23]. F. Chollet, *Deep Learning with Python*, Manning Publications, 2017.
[24]. C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
[25]. K. Murphy, *Probabilistic Machine Learning: An Introduction*, MIT Press, 2022.