Accuracy-Computation Trade-offs in Compact VGG and ResNet Architectures for CIFAR-10 Classification

Ammar Sameer Anaz[1*]

[1] Basic Education College, University of Mosul, Mosul, Iraq

***Corresponding email:** ammars.anaz@uomosul.edu.iq

Received 11 July 2025**; Revised** 01 August 2025**; Accepted** 22 August 2025**; Published** 31 August 2025

Abstract: *This research explores the trade-off between classification accuracy and computational efficiency in deep learning by conducting an empirical comparison of compact, optimized versions of the Visual Geometry Group (VGG) and Residual Neural Network (ResNet) architectures using the CIFAR-10 dataset. The main objective was to identify which design approach VGG's straightforward sequential structure or ResNet's skip-connection-based parameter efficiency achieves a more effective balance in resource-limited settings. The results show that while ResNet is more parameter-efficient, requiring 1.23 million parameters compared to VGG's 5.35 million, it exhibited slower inference performance (6.71 ms per image versus 6.04 ms for VGG). Moreover, ResNet demonstrated weaker robustness to noisy inputs, with its accuracy decreasing to 35.28%, whereas VGG's accuracy declined more moderately to 48.16%. Overall, the study highlights that in low-resolution image classification tasks, simpler architectures can outperform more complex models that are theoretically more efficient. These results offer valuable guidance for practitioners seeking to select and optimize convolutional neural networks under strict computational constraints.*

Keywords: convolutional neural networks, model efficiency, VGG, ResNet, CIFAR-10, accuracy-computation tradeoff.

1. Introduction

Deep convolutional neural networks (CNNs) are the foundation of modern computer vision [1], [2], with key designs like VGG and ResNet being used widely. VGG network is known for its simple, uniform structure of 3x3 convolutional layers. This design provides a strong, easy-to-understand baseline that's often used because it works well with different hardware [3]. In contrast, a major breakthrough was made by ResNet with the introduction of skip connections. These connections were introduced to solve vanishing gradient problem, which allowed much deeper networks to be trained. Because of this, ResNet often achieves better efficiency and performance on large-scale datasets like ImageNet [4]. However, efficiency of these networks performs in situations with limited resources—especially in smaller tasks like classifying CIFAR-10 images—isn't fully understood. This lack of information is what prompted our study. While ResNet's benefits are well-documented for large-scale problems, its computational demands from skip connections, as well as its performance and stability with smaller, low-resolution images (32x32 pixels), have not been a thoroughly investigated. This leads to a critical research question: In compact, efficient model design, does architectural simplicity (VGG) or sophisticated complexity (ResNet) provide a better balance of accuracy, speed, and robustness?

To answer this, a systematic empirical comparison is conducted. Inclusion of VGG architecture is essential in this investigation. It serves not as a state-of-the-art benchmark, but as canonical representative of a deep, sequential network. The simplicity of VGG architecture provides a reliable baseline for isolating the specific impact of ResNet's skip connections. In the context of small-scale datasets, the relatively larger number of parameters in VGG appears to act as a built-in safeguard, contributing not only to higher classification accuracy but also to improved stability and overall performance. This assumption is reinforced by the experimental results of the study. The principal contribution of this work is the introduction of a comparative framework that evaluates compact and simplified versions of both architectures under identical conditions. The analysis demonstrates that VGG, despite its greater parameter count, can achieve superior performance in terms of both accuracy and robustness when compared to ResNet on small-scale datasets. These findings challenge the common assumption of ResNet's consistent superiority and highlight that architectural simplicity can, in certain contexts, provide more practical advantages. Moreover, the results offer meaningful guidance for researchers and developers who need to select and optimize convolutional neural networks for deployment in environments constrained by limited computational resources. This research offers clear, evidence-based guidance for deciding between a simple or a more complex network design when accuracy must be balanced against strict computational limitations.

This paper consists of: Section 2, literature review on comparative studies between VGG and ResNet networks. experimental setup and dataset specifications, as well as network architectures with training protocols covered in Section 3. Results and discussion are thoroughly and performance evaluation procedure is explained in Section 4, and conclusions are presented in Section 5.

2. Literature Review

Neural networks are one of the most important developments that have revolutionized science in various fields. [5]–[10] . Many types of neural networks have emerged, differing in various aspects such as architecture and size... etc. to suit the purpose for which they were designed [11]. This necessitates investigating strengths and weaknesses of these networks to choose the best one according to need. And there are many studies that have addressed this topic. In 2021 Macarena's et.al. intend to determine which model of architecture of (CNN) system can be used to solve a real-life problem of product classification to help optimize pricing comparison. they compared accuracy of VGG19, VGG16, and ResNet50 using real-life data set [12]. In 2022 Santos-Bustos et.al provides an exploratory approach utilizing CNNs VGG and ResNet to detect ocular cancer, The study showed that ResNet network outperformed VGG network by 1.8% in accuracy [13]. Shah et.al. in 2023 compared several types of neural networks to identify diseases affecting rice crops. Results indicated that the most accurate network in their research was ResNet 50 [14]. In 2024 Naik A detailed analysis of three neural networks (VGG16, ResNet50, and custom CNN model) to diagnose certain chest diseases using transfer learning techniques was presented, highlighting strengths and weaknesses of each network in field of radiographic image classification considering accuracy and time factors [15]. In the same year, Muhammad and Khalaf presented a study comparing three models (CNN, ResNet, VGG) in deep learning for fingerprint classification. Superior performance was in favor of the CNN network at 96.5%, followed by ResNet network at 94.3%, and thirdly VGG at 92.11%, with a 2.11% difference between ResNet and VGG networks [16]. In 2025, Santoso et.al. conducted a study to evaluate capabilities of three types of neural networks (ResNet-50, EfficientNet-B1, and VGG-16) in classifying visual data, with only two classes to determine normal state of eye and cataract condition. They obtained a result showing that ResNet network outperformed VGG-16 network by 4.79% [17].

3. Proposed Method

This paper does not introduce a new architecture but instead it presents a new framework to compare the two prevailing architectural paradigms (sequential vs. residual) assuming hard computational limit. The distinctive feature of methodological front is the systematic framework of streamlined, miniature adaptation of familiar VGG and ResNet models, which on small-scale image data creates a reasonable and informative comparison. In all experiments, CIFAR-10 data comprising 60,000 32x32 color images of 10 classes was used [18]. For equality, VGG and ResNet were designed to have the same number of convolutional layers and similar feature map dimensions at all stages which requires analysis to focus on the key differences in design i.e. VGG sequential stacking versus ResNet identity skipping.

The set of methodological steps was the following:

Architecture Design: Reduced versions of VGG and ResNet were implemented for the experiments. The compact VGG model employed fewer filters per layer while retaining the sequential structure and the characteristic 3×3 convolutions. The ResNet variant was designed with compactness in mind, adopting the same narrow configuration and incorporating an identity-mapping residual block consisting of a single residual unit. Training Protocol: Both networks were trained from scratch under identical conditions. The same optimizer (stochastic gradient descent with momentum), learning rate schedule, batch size, and number of epochs were used to ensure a fair comparison and to eliminate performance differences attributable to training settings rather than architectural design. Evaluation Metrics: Model performance was comprehensively assessed across five dimensions. Classification accuracy was measured using Top-1 accuracy on the clean test set. Computational efficiency was evaluated based on the average time per training epoch and the total number of parameters. Robustness was tested by measuring the drop in accuracy when Gaussian noise was introduced into the test data. Reproducibility was ensured by standardizing the experimental setup, with the overall process illustrated in Figure 1. In this systematic process, identification of potential biases in performance can be done systematically due to a controlled study of extent to which inherent biases in application of models to scale them to efficient action.

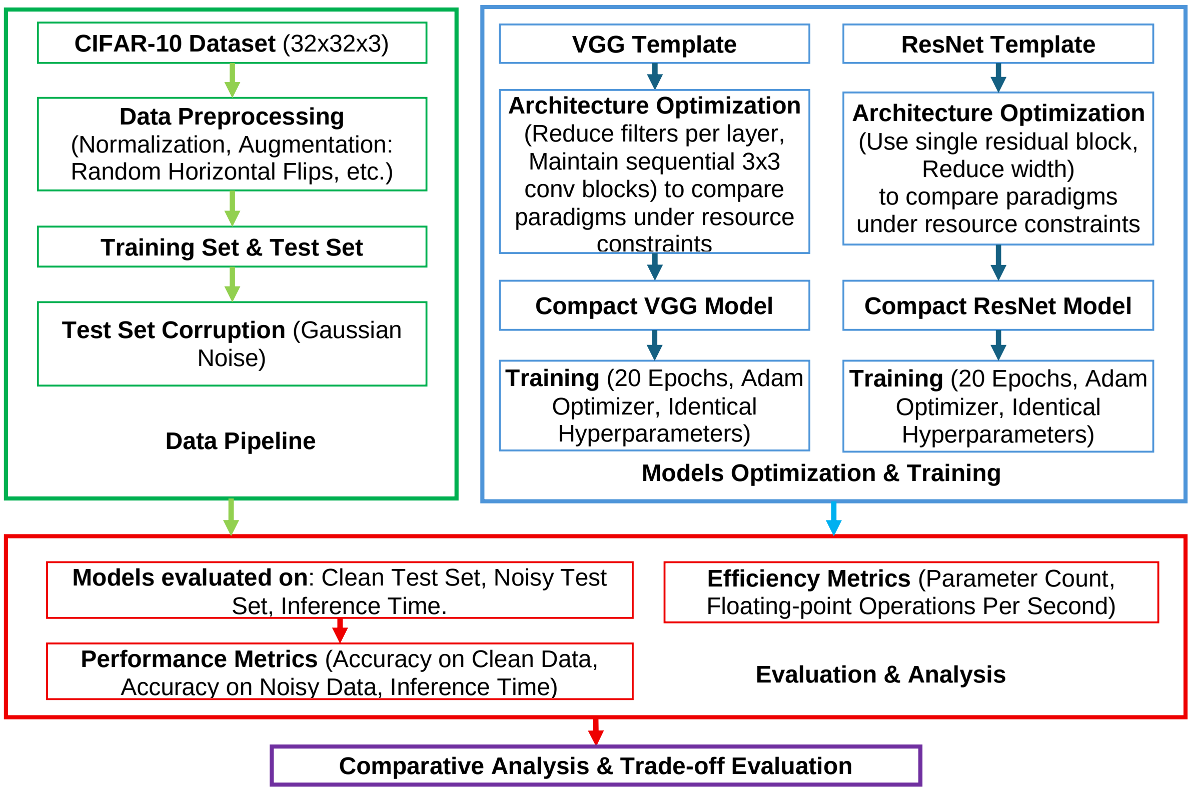

CIFAR-10 Dataset (32x32x3)
Data Preprocessing (Normalization, Augmentation: Random Horizontal Flips, etc.)
Training Set & Test Set
Test Set Corruption (Gaussian Noise)
Data Pipeline
VGG Template
Architecture Optimization (Reduce filters per layer, Maintain sequential 3x3 conv blocks) to compare paradigms under resource constraints
Compact VGG Model
Training (20 Epochs, Adam Optimizer, Identical Hyperparameters)
ResNet Template
Architecture Optimization (Use single residual block, Reduce width) to compare paradigms under resource constraints
Compact ResNet Model
Training (20 Epochs, Adam Optimizer, Identical Hyperparameters)
Models Optimization & Training
Models evaluated on: Clean Test Set, Noisy Test Set, Inference Time.
Performance Metrics (Accuracy on Clean Data, Accuracy on Noisy Data, Inference Time)
Efficiency Metrics (Parameter Count, Floating-point Operations Per Second)
Evaluation & Analysis
Comparative Analysis & Trade-off Evaluation

Figure 1: The custom compact design process

3.1 Experimental setup

Software component used in experiments to compare these neural networks is as follows: Python 3.8 was used, and for AI part, we relied on Keras (version 2.4) and TensorFlow (also 2.4). As for hardware part, we made sure every test ran on the exact same setup so our comparisons were fair. It had an Intel Core i7-1065G7 processor (we only used one part of it, running at 1.3 GHz). It also had 16 GB of memory (RAM). For heavy duty of graphics and AI calculations, we used an NVIDIA GeForce MX230 graphics card with 4 GB of its own memory. Plus, we put everything on a really fast 512GB NVMe storage drive to make sure things loaded quickly. This consistent setup allowed us to get really solid, repeatable results when testing our different network designs

3.2 Dataset specifications

The study relied on the CIFAR-10 dataset [18], which is a pretty standard benchmark in image recognition. It is a collection of 60,000 small (32x32 pixel) color images, all neatly sorted into ten different categories like cars, birds, cats, etc. images in CIFAR-10 are perfectly balanced: each of those ten categories has exactly 6,000 images, so no single type of object is overrepresented. Like most deep learning projects, dataset has been splinted. 50,000 images were used for training models, teaching subjects what to look for, then set aside 10,000 images purely for test to see how well they actually learned. Before any of that, though, pixel values were all scaled to be between 0 and 1, and converted category labels into a 'one-hot' format, which is how neural networks typically like to see them. Crucially, images in their original RGB color were kept to preserve all that valuable visual detail.

3.3 Network architectures

For a systematic and fair comparison, compact yet representative variants of VGG and ResNet architectures were designed and implemented in TensorFlow. The core design principle was defined as maintenance of each architecture's fundamental characteristics—sequential convolution blocks for VGG and residual skip connections for ResNet—rather than artificial equalization of their parameter counts. This approach was adopted to enable each paradigm to be compared in its most characteristic and efficient form. The architectures were designed so that spatial dimensions of their feature maps (32x32, 16x16, 8x8) could be aligned at each stage, ensuring that information was processed by networks at similar scales. Detailed, layer-by-layer specifications for each model, with exact parameter counts calculated from source code, are provided in Table 1 and Table 2.

Table 1: Exact layer-by-layer specification of the compact VGG model.

layer Type	Configuration	Output Shape	Parameters	# Trainable Params
Input	-	(32, 32, 3)	-	0
Block 1				
Conv2D	64 filters, 3x3, same	(32, 32, 64)	(3x3x3x64) + 64	1,792
Batch Normalization	-	(32, 32, 64)	(64x2)	128
ReLU	-	(32, 32, 64)	-	0
Conv2D	64 filters, 3x3, same	(32, 32, 64)	(3x3x64x64) + 64	36,928
Batch Normalization	-	(32, 32, 64)	(64x2)	128
ReLU	-	(32, 32, 64)	-	0
MaxPooling2D	2x2 pool	(16, 16, 64)	-	0
Block 2				
Conv2D	128 filters, 3x3, same	(16, 16, 128)	(3x3x64x128) + 128	73,856
Batch Normalization	-	(16, 16, 128)	(128x2)	256
ReLU	-	(16, 16, 128)	-	0
Conv2D	128 filters, 3x3, same	(16, 16, 128)	(3x3x128x128) + 128	147,584
Batch Normalization	-	(16, 16, 128)	(128x2)	256
ReLU	-	(16, 16, 128)	-	0
MaxPooling2D	2x2 pool	(8, 8, 128)	-	0
Block 3				
Conv2D	256 filters, 3x3, same	(8, 8, 256)	(3x3x128x256) + 256	295,168
Batch Normalization	-	(8, 8, 256)	(256x2)	512
ReLU	-	(8, 8, 256)	-	0
Conv2D	256 filters, 3x3, same	(8, 8, 256)	(3x3x256x256) + 256	590,080
Batch Normalization	-	(8, 8, 256)	(256x2)	512
ReLU	-	(8, 8, 256)	-	0
MaxPooling2D	2x2 pool	(4, 4, 256)	-	0
Classifier				
Flatten	-	(4096)	-	0
Dense	1024 units	(1024)	(4096 * 1024) + 1024	4,198,400
Dropout	0.5	(1024)	-	0
Dense	10 units	(10)	(1024 * 10) + 10	10,250
Total				5,354,570

Table 2: Exact layer-by-layer specification of the compact ResNet model.

Layer Type	Configuration	Output Shape	Parameters	# Trainable Params
Input	-	(32, 32, 3)	-	0
Initial Conv				
Conv2D	64 filters, 3x3, same	(32, 32, 64)	(3x3x3x64) + 64	1,792
Batch Normalization	-	(32, 32, 64)	(64x2)	128
ReLU	-	(32, 32, 64)	-	0
Residual Block 1				
Conv2D	64 filters, 3x3, same	(32, 32, 64)	(3x3x64x64) + 64	36,928
Batch Normalization	-	(32, 32, 64)	(64x2)	128
ReLU	-	(32, 32, 64)	-	0
Conv2D	64 filters, 3x3, same	(32, 32, 64)	(3x3x64x64) + 64	36,928
Batch Normalization	-	(32, 32, 64)	(64x2)	128
Add	-	(32, 32, 64)	-	0
ReLU	-	(32, 32, 64)	-	0
Residual Block 2				
Conv2D	128 filters, 3x3, stride 2	(16, 16, 128)	(3x3x64x128) + 128	73,856
Batch Normalization	-	(16, 16, 128)	(128x2)	256
ReLU	-	(16, 16, 128)	-	0
Conv2D	128 filters, 3x3, same	(16, 16, 128)	(3x3x128x128) + 128	147,584
Batch Normalization	-	(16, 16, 128)	(128x2)	256
Shortcut Conv	128 filters, 1x1, stride 2	(16, 16, 128)	(1x1x64x128) + 128	8,320
Add	-	(16, 16, 128)	-	0
ReLU	-	(16, 16, 128)	-	0
Residual Block 3				
Conv2D	256 filters, 3x3, stride 2	(8, 8, 256)	(3x3x128x256) + 256	295,168
Batch Normalization	-	(8, 8, 256)	(256x2)	512
ReLU	-	(8, 8, 256)	-	0
Conv2D	256 filters, 3x3, same	(8, 8, 256)	(3x3x256x256) + 256	590,080
Batch Normalization	-	(8, 8, 256)	(256x2)	512
Shortcut Conv	256 filters, 1x1, stride 2	(8, 8, 256)	(1x1x128x256) + 256	32,896
Add	-	(8, 8, 256)	-	0
ReLU	-	(8, 8, 256)	-	0
Classifier				
GlobalAveragePooling2D	-	(256)	-	0
Dense	10 units	(10)	(256 * 10) + 10	2,570
Total				1,230,090

3.3.1 Compact VGG Variant: The simplified VGG architecture has three convolutional blocks, each characterized by two convolutional layers with normalization and ReLU activation, max-pooling is used at the end of block. It is followed by classical head of classifiers based on flatten layer and fully connected layer of significant size which indicates the design philosophy of original VGG. Architecture places an emphasis in simple, uniform and deep structure with no skip connections. The architecture consists of 5,354,570 parameters described in Figure 2.

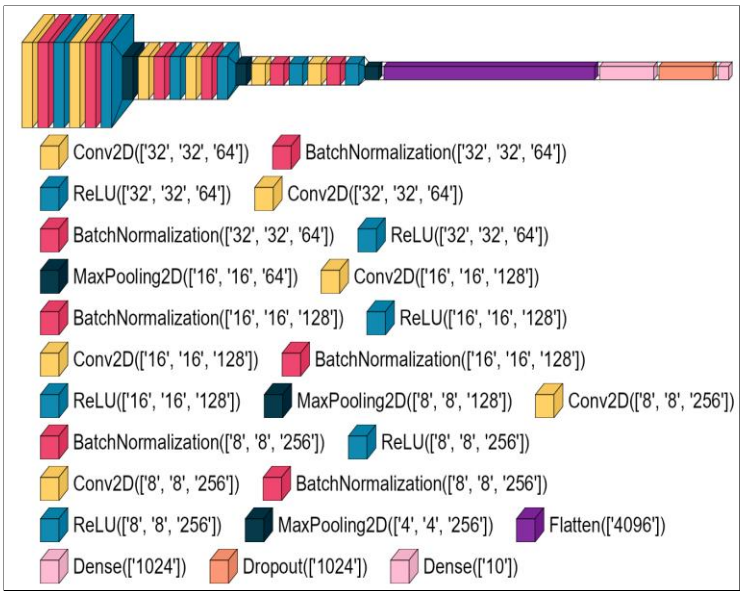

Conv2D(['32', '32', '64'])
BatchNormalization(['32', '32', '64'])
ReLU(['32', '32', '64'])
Conv2D(['32', '32', '64'])
BatchNormalization(['32', '32', '64'])
ReLU(['32', '32', '64'])
MaxPooling2D(['16', '16', '64'])
Conv2D(['16', '16', '128'])
BatchNormalization(['16', '16', '128'])
ReLU(['16', '16', '128'])
Conv2D(['16', '16', '128'])
BatchNormalization(['16', '16', '128'])
ReLU(['16', '16', '128'])
MaxPooling2D(['8', '8', '128'])
Conv2D(['8', '8', '256'])
BatchNormalization(['8', '8', '256'])
ReLU(['8', '8', '256'])
Conv2D(['8', '8', '256'])
BatchNormalization(['8', '8', '256'])
ReLU(['8', '8', '256'])
MaxPooling2D(['4', '4', '256'])
Flatten(['4096'])
Dense(['1024'])
Dropout(['1024'])
Dense(['10'])

Figure 2: VGG structure

3.3.2 Compact ResNet Variant: ResNet variant incorporates three residual blocks. First block uses an identity shortcut, second and third use projection shortcuts to match dimensions when down sampling. Consistent with modern ResNet designs, network uses global average pooling before final classifier, significantly reducing number of parameters in classifier compared to VGG variant. This design emphasizes learning of residual mappings enabled by skip connections and contains 1,230,090 parameters. Architecture is visualized in Figure 3.

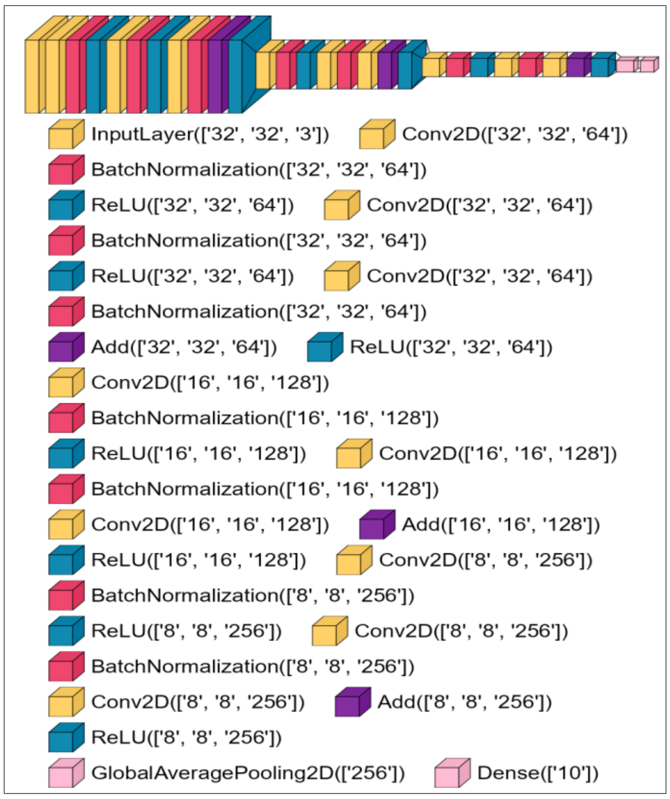

InputLayer(['32', '32', '3'])
Conv2D(['32', '32', '64'])
BatchNormalization(['32', '32', '64'])
ReLU(['32', '32', '64'])
Conv2D(['32', '32', '64'])
BatchNormalization(['32', '32', '64'])
ReLU(['32', '32', '64'])
Conv2D(['32', '32', '64'])
BatchNormalization(['32', '32', '64'])
Add(['32', '32', '64'])
ReLU(['32', '32', '64'])
Conv2D(['16', '16', '128'])
BatchNormalization(['16', '16', '128'])
ReLU(['16', '16', '128'])
Conv2D(['16', '16', '128'])
BatchNormalization(['16', '16', '128'])
Conv2D(['16', '16', '128'])
Add(['16', '16', '128'])
ReLU(['16', '16', '128'])
Conv2D(['8', '8', '256'])
BatchNormalization(['8', '8', '256'])
ReLU(['8', '8', '256'])
Conv2D(['8', '8', '256'])
BatchNormalization(['8', '8', '256'])
Conv2D(['8', '8', '256'])
Add(['8', '8', '256'])
ReLU(['8', '8', '256'])
GlobalAveragePooling2D(['256'])
Dense(['10'])

Figure 3: ResNet structure

3.4 Training Protocol

Training was performed using Adam optimizer and its default momentum parameters (β_1 = 0.9 and β_2 = 0.999). The initial learning rate is 0.001, but then it was decreased as a cosine function over the 20-epoch training session. The decision to train for 20 epochs was carefully made after preliminary experiments; these showed that models generally reached stable convergence around epochs 15 to 18, a point where validation loss values typically flattened out. To both guard against overfitting and ensure efficiency, an early stopping mechanism was put into place, allowing for a patience of 3 epochs. A mini-batch size of 64 samples was consistently employed across all experiments, as this value provided an effective balance between memory utilization and the quality of gradient estimates during the learning process. Since generalization is a critical requirement for deep models, several regularization strategies were integrated into the training protocol. Specifically, L2 weight decay (λ = 0.0001) was applied to prevent model parameters from growing excessively large. To improve the calibration of predicted probabilities and reduce overconfidence, label smoothing with α = 0.1 was introduced. Furthermore, to ensure stability during the initial training stages—when gradients tend to be at their largest—gradient clipping was applied at a global norm of 1.0, which proved particularly effective. Data augmentation beyond the original training set was also incorporated to enhance generalization.

This included random horizontal flipping with a probability of 0.5, as well as controlled variations in pixel intensities of up to ±15% within the augmentation channel. These transformations were selected to maintain semantic integrity while increasing variability in the training data. Additionally, all inputs were normalized to have zero mean and unit variance prior to being fed into the networks.

All hyperparameters and training procedures were kept identical for both architectures, except for those components inherently tied to the design (e.g., residual connections in the ResNet variant). After each epoch, 10% of the training set was reserved as a validation set for monitoring performance, while final evaluations were carried out on the standard CIFAR-10 test set to ensure comparability with established benchmarks. This rigorous training strategy was designed to maximize the use of available computational resources while ensuring reproducibility and reliability of the results. By carefully balancing optimization settings, regularization methods, and data augmentation techniques, a robust training framework was established. This ensured that observed performance differences between the two architectures could be attributed with confidence to their fundamental design characteristics rather than to differences in training conditions.

3.5 Performance Metrics and Equations

The models were evaluated using the following formal metrics, calculated with the equations below:

3.5.1. Classification Accuracy: This metric serves as the primary indicator of modeling success, defined as the ratio of correctly predicted samples to the total number of samples in the test set.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} 1(\hat{y}_i = y_i) \qquad (1)$$

where N is the total number of samples in test set, $\hat{y}_i$ is the predicted class for the i -th sample, y_i is true classification, and 1 is indicator variable that returns 1 when classification and prediction are matching and zero in the case of classification error.

3.5.2. Robustness to Noise: To evaluate the degradation in performance under adversarial conditions, the relative decrease in accuracy was measured after introducing Gaussian noise into the test set.

$$Relative\ Accuracy\ Drop = \frac{Accuracy_{clean} - Accuracy_{noisy}}{Accuracy_{clean}} \times 100\% \qquad (2)$$

where $Accuracy_{clean}$ *is the standard test accuracy and* $Accuracy_{noisy}$ is the accuracy on test set corrupted with Gaussian noise (μ =0, \ σ =0.1).

3.5.3. Computational Efficiency: Inference Time. Although measured empirically in seconds, the inference time of a model F for processing a batch of data X is defined as the execution duration of its forward pass.

$$T_{inference} = Time(F(X)) \qquad (3)$$

The mean inference time per sample was reported to facilitate a fair comparison between models.

3.5.4. Model Size (Parameter Count): The overall number of trainable parametersθ in each model was calculated as:

$$P = \sum_{l=1}^{L} |\theta_l| \tag{4}$$

where L is number of layers and θ_l is parameters of the layer l. This is a direct count of memory footprint of model.

4. Result and Discussion

A systematic empirical analysis of compact VGG and ResNet architectures was done on various performance dimensions. The results presented in Table 3 reveal a nuanced trade-off that challenges conventional assumptions regarding architectural efficiency. Contrary to the prevailing trend in computer vision research, where increasing network complexity is often associated with improved performance, the findings demonstrate that VGG architecture can still offer an effective balance between high accuracy, fast inference, reduced training time, and robustness when applied to the resource-constrained CIFAR-10 benchmark.

Table 3: Performance metrics for CIFAR-10 dataset.

Model	Test Accuracy	Inference Time (ms/image)	Total Training Time (20 Epochs)	Parameters	Robustness (Noisy Data)
VGG	84.50%	6.04	342 min	5,35M	48.95%
ResNet	74.36%	6.71	466 min	1,23M	34.99%

4.1 Accuracy and the Parameter Efficiency Paradox

The compact VGG model demonstrated a substantial accuracy advantage over the corresponding ResNet model, with a difference of 10.14%, as illustrated in Figure 4a. This result appears paradoxical, as it contradicts the common expectation that ResNet's skip connections which facilitate gradient flow and feature reuse should lead to superior performance. A plausible explanation for this paradox lies in the scale of the problem. For the relatively simple CIFAR-10 dataset, issues such as vanishing gradients in very deep networks are less critical than representational capacity. The VGG model, with its larger layers, was better able to capture and retain a wide variety of features across the ten classes. In contrast, while more parameter-efficient, the ResNet variant may have been under-parameterized for this particular task, limiting its ability to achieve comparable performance.

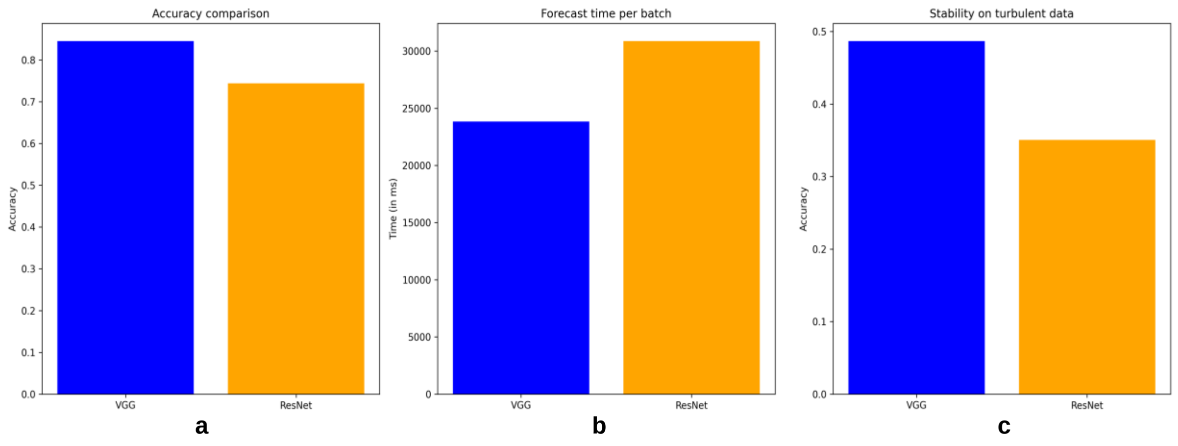


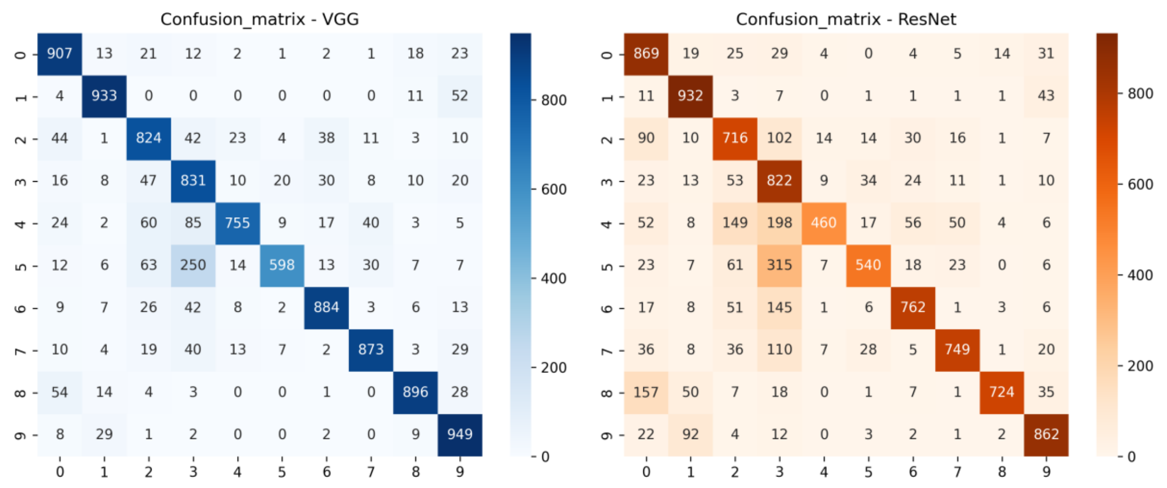
Figure 4: Comprehensive comparative performance analysis

4.2 Computational Performance: Overhead vs. Throughput

Obtained results on computational efficiency showed a critical insight. As shown in Figure 4b, the model with 78% fewer parameters was slightly slower at over all inference time (30500.10 ms vs. 22913.90 ms). This result is given by intrinsic overhead in the design of ResNet. Additional operations to perform element-wise addition were observed to be required in each skip connection. Besides, projection shortcuts of second and third blocks were identified to include additional 1x1 convolutional layers, which contribute to complexity of computational graph. The cumulative effect of these operations on a compact network was determined to be significant, negating ResNet's theoretical efficiency advantage.

4.3 Robustness: The Stability Advantage of a Larger Model

From Figure 4c during robustness evaluation under Gaussian noise corruption, substantially greater resilience was exhibited by VGG model compared to the ResNet model. An accuracy of 48.16% was retained by VGG model, whereas an accuracy of 35.28% was retained by the ResNet model. The clean accuracy of VGG was dropped by 42.9%, versus a drop of 52.5% for ResNet. This finding supports the hypothesis that a form of implicit regularization is provided by VGG's larger parameter count. The model is interpreted as having learned more redundant and robust features, whereas a more "brittle" set of features was observed to have been learned by the compact ResNet model, which disintegrated rapidly under perturbation. This makes the VGG model a markedly more reliable choice for applications where input data cannot be perfectly curated.



Confusion_matrix - VGG

	0	1	2	3	4	5	6	7	8	9
0	907	13	21	12	2	1	2	1	18	23
1	4	933	0	0	0	0	0	0	11	52
2	44	1	824	42	23	4	38	11	3	10
3	16	8	47	831	10	20	30	8	10	20
4	24	2	60	85	755	9	17	40	3	5
5	12	6	63	250	14	598	13	30	7	7
6	9	7	26	42	8	2	884	3	6	13
7	10	4	19	40	13	7	2	873	3	29
8	54	14	4	3	0	0	1	0	896	28
9	8	29	1	2	0	0	2	0	9	949

Confusion_matrix - ResNet

	0	1	2	3	4	5	6	7	8	9
0	869	19	25	29	4	0	4	5	14	31
1	11	932	3	7	0	1	1	1	1	43
2	90	10	716	102	14	14	30	16	1	7
3	23	13	53	822	9	34	24	11	1	10
4	52	8	149	198	460	17	56	50	4	6
5	23	7	61	315	7	540	18	23	0	6
6	17	8	51	145	1	6	762	1	3	6
7	36	8	36	110	7	28	5	749	1	20
8	157	50	7	18	0	1	7	1	724	35
9	22	92	4	12	0	3	2	1	2	862

Figure 5: Confusion matrices

Figure 5 shows confusion matrix for both VGG and ResNet, and provides numerical evidence of actual number of images that were accurately classified and images that networks cannot classify them, and it appears relatively noticeable that VGG network outperformed, as previously mentioned.

5. Conclusion

A comparison between VGG and ResNet has shown some important performance, efficiency, and robustness trade-offs. Although architectural advantages are provided by ResNet’s skip connections, an accuracy of 10.14% higher on small-scale (32×32) datasets was achieved by VGG, suggesting that a simpler sequential structure may be more effective for such tasks. However, lower computational efficiency was spotted in ResNet, with 36.3% more training period required despite presence of less parameters, probable because of the overhead introduced by skip connections. In robustness test, superior noise resilience was established by VGG, with only a 42.1% accuracy drop under Gaussian noise, compared to a 52.9% drop observed in ResNet, possibly because VGG’s larger parameter count acts as a form of regularization. To make further comparisons in the future, it is proposed that the diversity of the datasets should be improved or model tuning should be conducted to validate these results to a greater extent. Overall, while strong performance in deeper architectures is provided by ResNet, VGG may be preferred for small-scale, noise-prone applications where accuracy and robustness are prioritized over training efficiency.

ACKNOWLEDGMENT

The author would like to express their gratitude to the Basic Education at the University of Mosul, Iraq, for the positive support.

REFERENCES

[1] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, “A review of convolutional neural networks in computer vision,” *Artif. Intell. Rev.*, vol. 57, no. 4, p. 99, Mar. 2024, doi: 10.1007/s10462-024-10721-6.

[2] M. Krichen, “Convolutional Neural Networks: A Survey,” *Computers*, vol. 12, no. 8, p. 151, Jul. 2023, doi: 10.3390/computers12080151.

[3] X. Zhang, N. Han, and J. Zhang, “Comparative analysis of VGG, ResNet, and GoogLeNet architectures evaluating performance, computational efficiency, and convergence rates,” *Appl. Comput. Eng.*, vol. 44, no. 1, pp. 172–181, Mar. 2024, doi: 10.54254/2755-2721/44/20230676.

[4] M. Swapna, D. Y. K. Sharma, and D. B. Prasad, “CNN Architectures: Alex Net, Le Net, VGG, Google Net, Res Net,” *Int. J. Recent Technol. Eng.*, vol. 8, no. 6, pp. 953–959, Mar. 2020, doi: 10.35940/ijrte.F9532.038620.

[5] S. A. Majeed, A. M. H. Darghaoth, N. M. Z. Hamed, Y. A. Yahya, S. Raed, and Y. S. Dawood, “Detection of COVID-19 from X-Ray Images Using Transfer Learning Neural Networks,” in *2021 2nd Information Technology To Enhance e-learning and Other Application (IT-ELA)*, IEEE, Dec. 2021, pp. 58–63. doi: 10.1109/IT-ELA52201.2021.9773657.

[6] A. S. Anaz, M. Y. Al-Ridha, and R. R. O. Al-Nima, “Signal multiple encodings by using autoencoder deep learning,” *Bull. Electr. Eng. Informatics*, vol. 12, no. 1, pp. 435–440, Feb. 2023, doi: 10.11591/eei.v12i1.4229.

[7] N. A. I. Al-Obaidy, B. S. Mahmood, and A. M. F. Alkababji, “Finger Veins Verification by Exploiting the Deep Learning Technique,” *Ingénierie des systèmes d Inf.*, vol. 27, no. 6, pp. 923–931, Dec. 2022, doi: 10.18280/isi.270608.

[8] A. Al-Saegh, S. A. Dawwd, and J. M. Abdul-Jabbar, “Towards Efficient Motor Imagery EEG-based BCI Systems using DCNN,” in *2024 Arab ICT Conference (AICTC)*, IEEE, Feb. 2024, pp. 59–66. doi: 10.1109/AICTC58357.2024.10735028.

[9] A. S. Abdulaziz, A. Dawood, and A. Daood, “Speaker Identification and Verification Using Convolutional Neural Network CNN,” *Tikrit J. Eng. Sci.*, vol. 32, no. 2, pp. 1–13, May 2025, doi: 10.25130/tjes.32.2.19.

[10] Z. T. Al Mokhtar and S. A. Dawwd, “Deep Learning Video Prediction Based on Enhanced Skip Connection,” *Iraqi J. Electr. Electron. Eng.*, vol. 20, no. 1, pp. 195–205, Jun. 2024, doi: 10.37917/ijeee.20.1.19.

[11] L. Alzubaidi *et al.*, “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” *J. Big Data*, vol. 8, no. 1, p. 53, Mar. 2021, doi: 10.1186/s40537-021-00444-8.

[12] S. Mascarenhas and M. Agarwal, “A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification,” in *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, IEEE, Nov. 2021, pp. 96–99. doi: 10.1109/CENTCON52345.2021.9687944.

[13] D. F. Santos-Bustos, B. M. Nguyen, and H. E. Espitia, “Towards automated eye cancer classification via VGG and ResNet networks using transfer learning,” *Eng. Sci. Technol. an Int. J.*, vol. 35, p. 101214, Nov. 2022, doi: 10.1016/j.jestch.2022.101214.

[14] S. R. Shah, S. Qadri, H. Bibi, S. M. W. Shah, M. I. Sharif, and F. Marinello, “Comparing Inception V3, VGG 16, VGG 19, CNN, and ResNet 50: A Case Study on Early Detection of a Rice Disease,” *Agronomy*, vol. 13, no. 6, p. 1633, Jun. 2023, doi: 10.3390/agronomy13061633.

[15] A. Naik, “Comparative Analysis of VGG16, RESNET50, AND CNN Models for Lung Disease Prediction: A Deep Learning Approach,” *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 12, no. 5, pp. 875–896, May 2024, doi: 10.22214/ijraset.2024.61703.

[16] H. Muhammad and Z. Khalaf, “Fingerprint Identification System based on VGG, CNN, and ResNet Techniques,” *Basrah Res. Sci.*, vol. 50, no. 1, pp. 14--14, 2024, doi: doi.org/10.56714/bjrs.50.1.14.

[17] I. Santoso, A. M. Manurung, and E. R. Subhiyakto, “Comparison of ResNet-50, EfficientNet-B1, and VGG-16 Algorithms for Cataract Eye Image Classification,” *J. Appl. Informatics Comput.*, vol. 9, no. 2, pp. 284–294, Mar. 2025, doi: 10.30871/jaic.v9i2.8968.

[18] A. Krizhevsky, “Learning multiple layers of features from tiny images,” 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html