







Article

Convolutional Neural Networks: A Comprehensive Evaluation and Benchmarking of Pooling Layer Variants

Afia Zafar ¹, Noushin Saba ¹, Ali Arshad ^{2,*}, Amerah Alabrah ³, Saman Riaz ^{4,*}, Mohsin Suleman ¹,
Shahneer Zafar ¹ and Muhammad Nadeem ⁵

¹ Department of Computer Science, National University of Technology, Islamabad 44000, Pakistan; afiazafar@nutech.edu.pk (A.Z.); noushin.saba@nutech.edu.pk (N.S.); mohsin.suleman@nutech.edu.pk (M.S.); shahneerzafar@nutech.edu.pk (S.Z.)

² Department of Computing, NASTP Institute of Information Technology, Lahore 58810, Pakistan

³ Department of Information Systems, College of Computer and Information Science, King Saud University, Riyadh 11543, Saudi Arabia; aalobrah@ksu.edu.sa

⁴ Department of Computing, Riphah International University, Lahore 39101, Pakistan

⁵ Department of Computer Science, University of Science and Technology, Beijing 100083, China; nadeem72g@gmail.com

* Correspondence: alli.arshad@gmail.com (A.A.); samanriaz@hotmail.com (S.R.)

Abstract: Convolutional Neural Networks (CNNs) are a class of deep neural networks that have proven highly effective in areas such as image and video recognition. CNNs typically include several types of layers, such as convolutional layers, activation layers, pooling layers, and fully connected layers, all of which contribute to the network's ability to recognize patterns and features. The pooling layer, which often follows the convolutional layer, is crucial for reducing computational complexity by performing down-sampling while maintaining essential features. This layer's role in balancing the symmetry of information across the network is vital for optimal performance. However, the choice of pooling method is often based on intuition, which can lead to less accurate or efficient results. This research compares various standard pooling methods (MAX and AVERAGE pooling) on standard datasets (MNIST, CIFAR-10, and CIFAR-100) to determine the most effective approach in preserving detail, performance, and overall computational efficiency while maintaining the symmetry necessary for robust CNN performance.

Keywords: artificial neural network; image classification; pooling methods; max pooling; average pooling; min pooling; support vector machine; long short-term memory; rectified linear unit



Citation: Zafar, A.; Saba, N.; Arshad, A.; Alabrah, A.; Riaz, S.; Suleman, M.; Zafar, S.; Nadeem, M. Convolutional Neural Networks: A Comprehensive Evaluation and Benchmarking of Pooling Layer Variants. *Symmetry* **2024**, *16*, 1516. <https://doi.org/10.3390/sym16111516>

Academic Editor: Zhixun Su

Received: 19 August 2024

Revised: 5 November 2024

Accepted: 6 November 2024

Published: 12 November 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The most sophisticated methods for performing various challenging tasks, such as image segmentation [1] and classification [2] in computer vision and image-analysis tasks, are convolutional neural networks (CNNs). Convolutions, nonlinear activations, and additional pooling operators are ample in each convolutional layer of a CNN, typically followed by one or more fully connected layers. CNNs are feedforward networks because the information is passed in only one direction from input to output. CNNs and artificial neural networks (ANNs) are rooted in biological principles. Their design is inspired by the brain's visual cortex, alternately layered with simple and higher-order cells [3]. There are many different types of CNN architectures, but they consist of convolutional and pooling layers built into modules. Beneath these modules are one or more fully connected layers, similar to standard feedforward neural networks. Modules are typically stacked on top of each other to build complex models [4]. A typical CNN architecture for a toy-image-classification task is shown in Figure 1. Feeding the images directly into the network involves a series of convolution and pooling layers. The representations formed by these processes are fed into one or more fully connected layers. The classifier finally provides a

proof of evaluation for the fully linked layers. Although this is the most commonly used basic design in the literature, many improvements to the architecture have recently been proposed to improve image classification accuracy or reduce computational overhead.

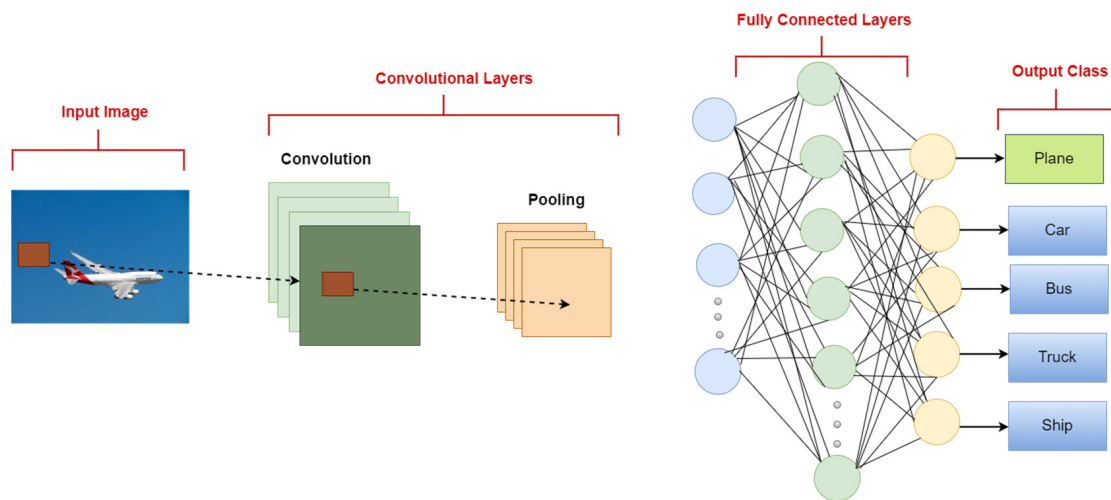


Figure 1. Standard CNN architecture.

Similar to feature extraction, convolutional layers learn feature representations of input images. Neurons in convolutional layers are organized into feature maps. A set of trainable weights, often called a filter bank, connects each neuron in the receptive field of the feature map with its neighbors in the underlying layer. We combine the input with the learned weights to create a new feature map and pass the output to a nonlinear activation function. Each neuron in a feature map is constrained to have equal weights since the weights of relatively different maps within the same convolutional layer are variable, so multiple features can be obtained at each point [5]. The k_{th} output feature map y_k can be defined more formally as

$$y_k = f(w_k \times x) \quad (1)$$

where x represents the input image, w_k represents the convolution filter attached to the k th feature map, the 2D convolution operator is the multiplication sign in this context, and $f(\cdot)$ represents the nonlinear activation function. You can use this operator to compute the inner product of the filter model at each location in the input image.

Before the arrival of Convolutional Neural Networks (CNNs), traditional machine learning models such as Support Vector Machines (SVM) [6] and K-Nearest Neighbors (KNN) [7] were commonly employed for image classification, where each pixel was treated as an individual feature. The introduction of CNNs revolutionized this approach by using convolutional layers to extract multiple features from an image, enhancing the prediction of output values. Since the convolution operation is computationally intensive, pooling layers were integrated into CNNs to make the process more efficient. Pooling reduces the computational load by down-sampling the input, which decreases the number of computations required while preserving the most critical information. The pooling method streamlines the processing within the network, maintaining essential details with significantly lower resource consumption [8].

While CNNs have significantly improved image classification, various enhancements have been proposed to further optimize performance, including hybrid models [9] that combine CNNs with other machine learning algorithms to improve accuracy or reduce computational complexity. Recent advancements, such as adaptive pooling strategies and dynamic pooling methods, adjust pooling operations based on the input, allowing for better flexibility and feature retention [10]. However, the impact of these newer methods on standard architectures like AlexNet, ResNet, and LeNet remains an area of active research.

This study aims to provide an overview of various pooling methods, discussing the benefits and drawbacks of each approach (see Table 1). Additionally, we compare their performance in classification tasks using three distinct datasets.

Table 1. Standard pooling methods along with strengths and weaknesses.

Pooling Methods	Description	Strengths	Weaknesses	Common Use Cases
Max Pooling	Selects the maximum value within each pooling window	<ul style="list-style-type: none"> Preserves strong features Robust to noise Improves generalization 	<ul style="list-style-type: none"> can discard potentially useful information May lead to overfitting in some cases 	<ul style="list-style-type: none"> Image classification (e.g., MNIST and CIFAR-10) Object detection Natural language processing
Average Pooling	Calculates the average value within each pooling window	<ul style="list-style-type: none"> Captures overall feature representation Smooths out noise Less prone to overfitting 	<ul style="list-style-type: none"> May blur important features Less effective at preserving sharp edges 	<ul style="list-style-type: none"> Image classification (e.g., ImageNet) Semantic segmentation
Min Pooling	Selects the minimum value within each pooling window	<ul style="list-style-type: none"> Captures dark features' or background information Useful in specific domains, like medical imaging 	<ul style="list-style-type: none"> Less common than max or average pooling Can be sensitive to noise 	<ul style="list-style-type: none"> Medical image analysis Texture analysis

The main contributions of this study include the following:

1. The proposed study systematically evaluated multiple CNN architectures—LeNet, AlexNet, and ResNet—across various datasets (MNIST, CIFAR-10, and CIFAR-100). This comprehensive analysis sheds light on how these models perform on datasets of differing complexities and sizes, providing insights into their adaptability and generalization capabilities across different image-classification tasks.
2. By presenting the comparative performance metrics, the proposed study identifies which CNN architectures excel or struggle when applied to specific datasets. This helps researchers to understand the strengths and weaknesses of each model in handling distinct image datasets, aiding in informed model selection for particular tasks or datasets.
3. This study provides a comprehensive comparison of standard pooling methods—max and average pooling—evaluated across different CNN architectures, including CNN, AlexNet, ResNet, and LeNet, on multiple datasets. While prior studies have discussed individual pooling methods, few have provided a systematic comparison in this context. Additionally, in this study, these methods were evaluated in light of recent advancements, highlighting the practical implications for resource-constrained environments.

The rest of this study is organized as follows: Section 2 reviews work related to standard pooling methods proposed for computer vision and various image-analysis applications. Section 3 presents the datasets and experimental procedures, reviews and presents the results and document provides a detailed discussion of our study, and Section 4 summarizes the study.

2. Related Work

The publications included in this review were sourced through a thorough search that utilized combinations of the terms “Pooling”, “CNN”, and “Convolution” (along with related terms such as “convolutional”) across titles, keywords, and abstracts. Following an initial screening of the results, additional relevant literature was incorporated by carefully

examining references and related works from the selected papers, with a focus specifically on the applications of CNNs. While some foundational studies, such as Yamaguchi's introduction of max pooling in the early 1990s, are noted, the majority of pooling techniques and advancements have emerged in the last decade [11]. Figure 2 illustrates a sustained interest in pooling research over the past eight years, with only minor fluctuations in publication frequency.

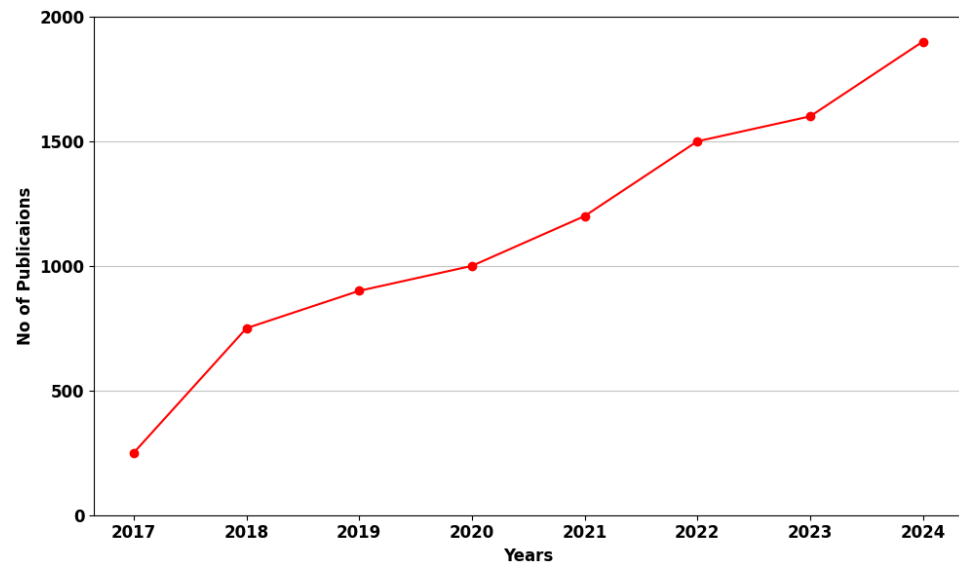


Figure 2. Total number of publications on pooling techniques for CNNs in Scopus.

Two pooling groups are commonly employed in CNN for feature-reduction purposes. The first is local pooling, which samples small local regions, such as 3×3 , to display the feature map. The second is global pooling. It derives a scalar value from the feature vector of the image representation of each feature across the feature map [12]. A fully connected layer takes all these representations and classifies them. In particular, the well-known Dense Net consists of one global pooling layer and four local pooling layers. The three commonly used types of pooling operations are max pooling, average pooling, and min pooling [13]. This study discusses each pooling operation's properties, advantages, and limitations.

2.1. Max Pooling

Max pooling is a simple operation widely used in CNNs due to its lack of tuneable parameters [14]. The feature map's spatial dimensions are enhanced by a mechanism, known as max pooling, that also provides network invariance. To accomplish network invariance, the $k \times k$ neighborhood is emphasized as having the highest value on the feature map. The max pooling method selects the largest element for each pooling zone. Considering sparse codes and simple linear classifiers, max pooling shows better performance. Due to these reasons, it has grown in popularity in recent years [15]. Max pooling's stochastic features allow it to handle sparse representations efficiently, which is yet another reason for its success. The mathematical expression for max pooling is

$$f_{max}(X) = \max_i x_i \quad (2)$$

J -related filters are used for the composition of the m th max pooling band:

$$p_{j,m} = \max(h_j, (m-1)N + r) \quad (3)$$

Here, $N \in \{1, \dots, R\}$ is termed as a pooling shift, which allows for overlap within concerned pooling regions when $N < R$. The pooling layer reduces the output size from

K convolution bands to $M = \left(\frac{K-R}{N+1}\right)$ the pooling region and the expected results for the resulting layer $p = [p1, \dots, pm] \in R^{M,J}$.

The primary limitation of max pooling lies in its selection of the maximum element from the pooling region while disregarding other values, potentially leading to the loss of distinguishing features and critical information. Studies have highlighted that, despite enhancing computational efficiency and reducing dimensionality in CNNs, max pooling can compromise spatial information and introduce inconsistencies in activations [16]. Furthermore, in object-detection tasks, max pooling often results in poor localization accuracy, particularly for small or low-resolution objects [17]. To mitigate these drawbacks, a novel approach, called Spatial Pyramid Pooling (SPP), has been proposed, which employs multiple pooling layers at varying spatial resolutions to better capture spatial information, demonstrating superior performance over max pooling in benchmark object detection datasets [18].

Figure 3 illustrates the max pooling operation. In this example, the pooling region has an input size of 4-by-4, while the filter size, with a stride of 2, is 2-by-2. Max pooling extracts the maximum value of 20 from the first 2×2 segments (highlighted in green), and the highest values from each segment are selected to generate an output channel. However, max pooling only considers the largest value and ignores the others. As a result, when most elements have high values, significant features may be lost after max pooling, potentially leading to adverse outcomes.

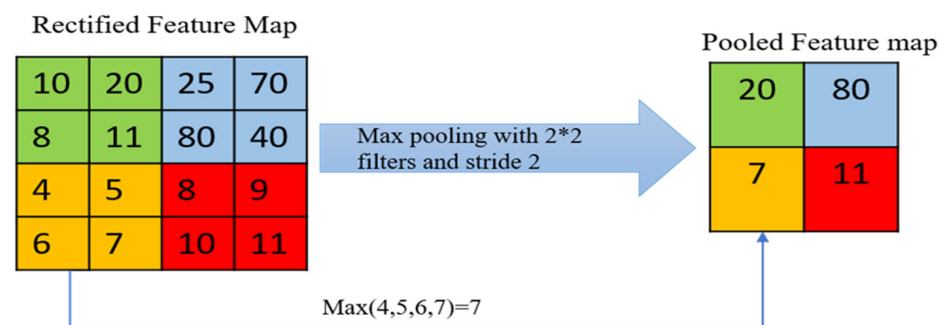


Figure 3. Illustration of max pooling operation.

2.2. Average Pooling

Down-sampling is performed by an average pooling layer by splitting the input into rectangular pooling areas and determining the average values of each region. The idea of extracting the feature by finding their average was first introduced by the authors of [19]. The proposed idea was implemented in the first convolution-based deep neural network. Figure 4 demonstrates the example of average pooling operation. The standard average pooling method divides the image input into several independent rectangular boxes. The average value for each box is determined, and the output channel is displayed. Average pooling is mathematically defined as

$$f_{ave}(X) = \frac{1}{N} \sum_{i=1}^N x_i \quad (4)$$

where x is a vector representing activations from a rectangular box of N permutation in an image or a channel (for example, the size of the rectangular area in Figure 4 is 2 by 2). Average pooling used to be common, but with the arrival of the max pooling technique, its usefulness has been constrained [20], where the shortcoming seems to mostly lead to a decline in information in terms of contrast. All of the activation values in the rectangular box are considered when estimating the mean. The estimated mean will indeed be low if the strength of all the activation functions is low, resulting in diminished contrast. Whenever the majority of the activations in the pooling region have a zero value, the scenario will get much worse. In that situation, the convolutional feature characteristic would be reduced

significantly. Noise-inducing elements are reduced by averaging. However, since it gives each element in the pooling region equal priority, background regions may predominate in the pooled representation, which might diminish the discrimination power [21].

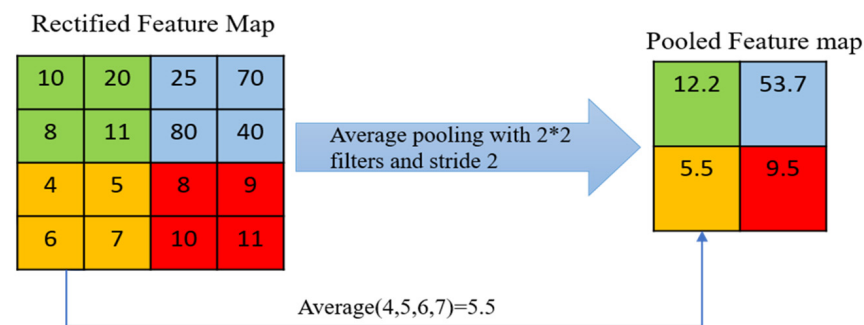


Figure 4. Illustration of the average pooling operation.

Since neither max pooling nor average pooling consistently demonstrates superior performance, several techniques have emerged that combine the strengths of both methods, such as weighted pooling [22] and soft pooling [23]. These hybrid approaches introduce additional parameters, leading to increased learning time and computational overhead. However, these methods still face challenges, as they either prioritize the stronger activation or treat all activations equally, with existing studies primarily depending on activation values to address these limitations.

2.3. Min Pooling

Min pooling is a pooling operation that selects the minimum value within a sliding window, though it is less frequently used than max or average pooling, as it tends to preserve the smallest and least significant features of the input [24]. However, it is beneficial in specific applications, such as anomaly detection and background subtraction, where detecting differences from a reference signal is essential. A comparison of standard pooling methods—max, average, and min pooling—along with their strengths and weaknesses, is presented in Table 1. Studies indicate that max pooling is generally more effective for tasks requiring the capture of highly discriminative features, while average pooling offers greater robustness to noise and improved generalization by considering the overall context [25].

Recent advancements in CNNs have introduced hybrid models and adaptive pooling strategies to improve performance in various tasks. For example, Khairandish et al. (2022) proposed a hybrid approach combining CNNs with Support Vector Machines (SVMs) for more accurate image classification in limited-resource environments, demonstrating that hybrid methods can outperform traditional CNNs in certain contexts [26]. Similarly, Ding et al. (2024) introduced an adaptive pooling method for image text retrieval that adjusts pooling parameters based on input data, allowing the model to maintain more relevant features while reducing computational complexity [27]. Li et al. (2024) explored the benefits of combining CNNs with other algorithms, like Long Short-Term Memory (LSTM) networks, enabling enhanced feature extraction for recommendation algorithm [28]. Additionally, Han et al. (2019) conducted a survey of dynamic neural networks' mechanism, which varies the pooling size depending on the input characteristics, offering better adaptability and precision for complex datasets [29]. Zhao et al. (2024) presented a mixed-pooling strategy where max and average pooling were combined in a single layer, demonstrating improvements in accuracy on specific benchmarks. While these studies have significantly contributed to the enhancement of CNNs, particularly through hybridization and adaptive-pooling techniques, they often focus on specific architectures or use cases. In contrast, the proposed study offers a systematic and direct comparison of three widely used pooling methods—max, min, and average pooling—across multiple architectures (AlexNet, ResNet, and LeNet) on diverse datasets. This broader evaluation provides a more generalizable understanding of how different pooling techniques impact performance

across standard CNN architectures. Unlike the aforementioned works, which typically propose new architectures or adaptations, this study focuses on refining the core understanding of pooling operations, offering practical insights for improving CNN performance in resource-constrained environments without introducing additional complexity.

3. Material and Methods

To understand the impact of pooling techniques on the performance of convolutional neural networks (CNNs), this study precisely analyzes three standard datasets, each chosen for its unique characteristics and challenges. The methodologies employed are designed to systematically evaluate and compare the effectiveness of different pooling strategies, thereby offering insights into their implications for CNN performance.

3.1. Datasets

This study used three standard datasets to evaluate the performance of pooling techniques across various convolutional neural network (CNN) architectures: MNIST [30], CIFAR-10 [31], and CIFAR-100 [32]. These datasets represent a range of complexities, from simple handwritten digits (MNIST) to diverse object classes (CIFAR-10 and CIFAR-100). The selection of these datasets allows for a comprehensive analysis of how different pooling methods (max, average, and min pooling) impact model performance across varying levels of classification difficulty.

3.2. Model Architecture

Three widely adopted CNN architectures employed in this study are LeNet, AlexNet, and ResNet. These architectures were selected for their established performance and distinct structural characteristics, which provide a robust testing ground for evaluating different pooling strategies. LeNet is a smaller architecture primarily designed for simpler tasks, such as digit classification, and consists of convolutional and fully connected layers. In contrast, AlexNet features a deeper design with more convolutional layers, specifically engineered to address complex image classification problems. ResNet, recognized for its innovative use of residual connections, is deeper and more intricate than both LeNet and AlexNet, making it particularly well-suited for highly dimensional image-classification tasks. Together, these architectures create a comprehensive platform for analyzing the effects of various pooling techniques on model performance across a range of classification challenges.

3.3. Experimental Setup

The experiments in this study were conducted using Keras with TensorFlow as the backend framework, establishing a controlled environment for evaluating the performance of different pooling techniques across various CNN architectures. Each dataset, MNIST, CIFAR-10, and CIFAR-100, was divided using an 80/20 split, allocating 80% of the data for training and reserving 20% for testing. This division allowed for a thorough assessment of both recognition capabilities and generalization of performance of the proposed models. To ensure consistency and fairness in the evaluation, model training was executed under multiple configurations of batch sizes (32, 64, 128, and 256), learning rates (0.01, 0.001, and 0.0001), and dropout rates (ranging from 0.1 to 0.5, including a no-dropout condition). The Stochastic Gradient Descent (SGD) optimizer was utilized for the LeNet architecture, while the Adam and RMSProp optimizers were applied to AlexNet and ResNet, respectively, optimizing model performance for multi-class classification tasks. The categorical cross-entropy loss function guided the training process, with early stopping employed based on validation loss to mitigate overfitting. This comprehensive setup ensured a rigorous evaluation of each pooling method's impact on model accuracy and generalization across diverse datasets. The proposed architecture of comparative approaches is presented in Tables 2–5.

Table 2. Proposed architecture of CNN.

Layers		Feature Map No. of Filters/Neurons	Filter Size/ Kernel Size	Size of Feature Map	Activation Function
Input Layer	Image	-	-	$32 \times 32 \times 3$	-
First Layer	Convolution	32	3×3	$30 \times 30 \times 32$	relu
Second Layer	Pooling	32	2×2	$15 \times 15 \times 32$	
Third Layer	Convolution	64	3×3	$13 \times 13 \times 64$	relu
Fourth Layer	Pooling	64	2×2	$6 \times 6 \times 64$	
Fifth Layer	Convolution	64	3×3	$4 \times 4 \times 64$	relu
Sixth Layer	Pooling	64	2×2	$2 \times 2 \times 64$	
Seventh Layer	Fully Connected	-	-	64	
Output Layer	Fully Connected	-	-	No. of Classes	Softmax

Table 3. Proposed Architecture of LeNet.

Layers		Feature Map No. of Filters/Neurons	Filter Size/Kernel Size	Stride	Size of Feature Map	Padding
Input Layer	Image	-	-	-	$32 \times 32 \times 1$	-
First Layer	Convolution	6	5×5	1	$28 \times 28 \times 6$	same
Second Layer	Pooling	6	2×2	2	$14 \times 14 \times 6$	valid
Third Layer	Convolution	16	5×5	1	$10 \times 10 \times 16$	valid
Fourth Layer	Pooling	16	2×2	2	$5 \times 5 \times 16$	valid
Fifth Layer	Convolution	120	5×5	1	120	valid
Sixth Layer	Fully Connected	-	-	-	84	-
Output Layer	Fully Connected	-	-	-	No. of Classes	-

Table 4. Proposed architecture of AlexNet.

Layers		Feature Map No. of Filters/Neurons	Filter Size/ Kernel Size	Stride	Size of Feature Map	Padding	Activation Function
Input Layer	Image	1	-	-	$227 \times 227 \times 3$	-	relu
First Layer	Convolution	96	11×11	4	$55 \times 55 \times 96$	same	relu
	Pooling	96	3×3	2	$27 \times 27 \times 96$	-	
Second Layer	Convolution	256	5×5	1	$27 \times 27 \times 256$	same	relu
	Pooling	256	3×3	2	$13 \times 13 \times 256$	-	
Third Layer	Convolution	384	3×3	1	$13 \times 13 \times 384$	same	relu
Fourth Layer	Convolution	384	3×3	1	$13 \times 13 \times 384$	same	relu
Fifth Layer	Convolution	256	3×3	1	$13 \times 13 \times 256$	same	relu
	Pooling	256	3×3	2	$6 \times 6 \times 256$	-	
Sixth Layer	Fully Connected	-	-	-	9216	-	relu
Seventh Layer	Fully Connected	-	-	-	4096	-	relu
Eight Layer	Fully Connected	-	-	-	4096	-	relu
Output Layer	Fully Connected	-	-	-	No. of Classes	-	Softmax

Table 5. Proposed architecture of AlexNet.

Layers		Feature Map No. of Fil- ters/Neurons	Filter Size/ Kernel Size	Stride	Size of Feature Map	Padding	Activation Function
Input Layer	Image	-	-	-	$224 \times 224 \times 3$	-	-
First Layer	Convolution	64	7×7	2	$112 \times 112 \times 64$	same	relu
Second Layer	Pooling	64	3×3	2	$112 \times 112 \times 64$	-	-
	Third	64	1×1	1		valid	relu
Stage 1	Layer–Eleventh	64	3×3	1	$56 \times 56 \times 64$	same	relu
	Layer	256	1×1	1		valid	-

Table 5. Cont.

	Layers		Feature Map No. of Fil- ters/Neurons	Filter Size/ Kernel Size	Stride	Size of Feature Map	Padding	Activation Function
Stage 2	Twelfth Layer–	Convolution	128	1 × 1	1	56 × 56 × 256	valid	relu
	Twenty-Third	Convolution	128	3 × 3	1		same	relu
	Layer	Convolution	512	1 × 1	1		valid	-
Stage 3	Twenty-Fourth	Convolution	256	1 × 1	1	28 × 28 × 512	valid	relu
	Layer–Forty-First	Convolution	256	3 × 3	1		same	relu
	Layer	Convolution	1024	1 × 1	1		valid	-
Stage 4	Forty-Second	Convolution	512	1 × 1	1	14 × 14 × 1024	valid	relu
	Layer–Fiftieth	Convolution	512	3 × 3	1		same	relu
	Layer	Convolution	2048	1 × 1	1		valid	-
	Fifty-First Layer	Pooling	2048	2 × 2		7 × 7 × 2048	-	-
	Output Layer	Fully Connected	-	-	-	No. of Classes	-	Softmax

3.4. Result and Analysis

In our research, each comparative approach involving various neural network architectures, such as CNN, AlexNet, ResNet, and LeNet, was subjected to different configurations of batch size, learning rate, and dropout. Selecting distinct hyperparameters aimed to explore their impact on model performance and convergence across pooling techniques (max and average) on datasets like MNIST, CIFAR-10, and CIFAR-100. Given the sensitivity of these hyperparameters in influencing training dynamics, their variation resulted in divergent accuracy results across the models. Batch sizes were chosen to regulate the number of samples processed per iteration, affecting gradient updates and the convergence speed. Learning rates played a critical role in controlling the step size during optimization, impacting the model's ability to navigate the loss landscape. Additionally, dropout rates were manipulated to mitigate overfitting by randomly deactivating neurons during training, affecting the model's generalization capability. Consequently, the discrepancy in accuracy outcomes underscores the nuanced interplay between these hyperparameters and their consequential effect on model learning and generalization across different network architectures and pooling strategies.

3.5. Performance Evaluation in Terms of Accuracy

This section provides a comprehensive comparative analysis of the accuracy of the MNIST, CIFAR-10, and CIFAR-100 datasets across various convolutional neural network (CNN) architectures, including CNN, LeNet, AlexNet, and ResNet. The performance of these architectures is evaluated under different batch sizes, learning rates, and dropout rates, focusing on the effects of max pooling and average pooling techniques. The objective is to identify which pooling technique and architectural configuration yields the best performance for each dataset, thereby offering valuable insights into optimizing CNN models for diverse image-classification tasks. The detailed results of comparative approaches are presented in Tables 6–11.

The comparative analysis of CNN, AlexNet, ResNet, and LeNet using max and average pooling methods under varying batch sizes and dropout rates highlights the distinct impact of pooling strategies on model performance from Tables 6–11. Under max pooling conditions, AlexNet consistently outperformed other models, showcasing its ability to extract dominant features from the data efficiently. This superior performance is evident in AlexNet's stable and high classification accuracy across different configurations, even as dropout rates and batch sizes varied. The architecture of AlexNet is particularly suited for capturing the most salient features, which is effectively facilitated by the max pooling method. In contrast, CNN emerged as the second-best performer in the max pooling setup. Although it demonstrated strong feature-extraction capabilities, CNN showed higher sensitivity to changes in hyperparameters compared to AlexNet, indicating some variability in performance.

Table 6. Accuracy of max pooling on standard MNIST dataset with different learning rates and batch sizes with comparative approaches.

MNIST Max Pooling																								
Learning Rate		0.01																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	97.78	97.45	95.97	93.94	94.13	93.58	96.04	96.62	94.11	95.70	96.14	92.78	96.98	96.78	96.32	97.57	96.83	96.78	97.52	96.02	96.25	95.31	96.23	96.15
LeNet	95.57	95.12	88.26	90.14	86.44	82.45	91.51	89.39	85.56	85.42	88.56	86.54	88.75	87.9	89.39	86.6	83.35	88.02	89.25	90.57	90.82	85.31	82.85	84.96
AlexNet	98.58	97.92	96.72	97.59	97.93	96.65	98.88	98.41	97.11	96.70	98.54	96.53	97.97	98.01	98.25	98.25	97.28	97.14	98.44	98.41	98.53	98.69	97.32	98.35
ResNet	96.57	95.43	92.18	89.45	89.45	91.58	94.03	92.79	91.50	92.72	94.25	89.32	94.23	91.25	93.58	94.25	89.25	92.56	92.42	93.25	93.76	91.31	89.56	89.84
Learning Rate		0.001																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	97.59	96.59	98.12	98.79	96.40	97.98	95.28	96.59	97.52	96.66	95.56	97.86	96.87	95.78	96.70	97.42	96.46	97.05	96.53	96.69	97.69	97.42	97.18	95.30
LeNet	90.45	91.45	98.85	88.72	98.91	87.26	90.58	91.79	90.25	88.83	88.50	87.52	88.54	89.34	90.21	88.94	87.81	89.76	88.86	90.95	91.80	90.85	91.75	89.75
AlexNet	98.66	98.47	98.69	97.54	98.56	95.28	98.76	98.96	98.40	98.66	98.46	98.84	97.25	98.88	98.20	98.42	97.24	97.95	98.55	98.37	98.26	98.56	98.95	97.73
ResNet	93.49	93.45	89.21	94.62	87.88	90.62	93.63	93.72	92.53	93.78	90.45	92.58	91.27	89.34	92.01	91.94	89.52	91.02	92.27	92.33	93.20	94.94	92.93	90.46
Learning Rate		0.0001																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	96.21	95.18	96.84	96.55	96.99	96.39	97.84	98.01	97.25	96.17	96.98	94.69	97.03	96.35	96.80	95.57	95.37	93.06	97.06	96.11	95.93	96.47	94.54	94.42
LeNet	90.55	89.54	89.38	86.42	85.30	86.04	88.36	87.54	85.71	88.16	85.99	86.81	86.06	84.02	83.95	80.85	82.58	81.28	82.33	82.15	84.07	83.97	84.71	87.54
AlexNet	98.55	98.69	98.58	97.94	98.20	98.54	98.30	98.28	98.19	97.59	97.72	96.52	98.28	97.27	97.84	97.87	97.53	95.27	98.24	97.59	97.69	98.03	96.61	97.96
ResNet	94.09	90.16	91.07	89.01	90.66	91.42	93.78	90.45	93.71	91.76	93.42	92.39	93.06	94.47	93.66	88.73	92.68	88.77	91.72	91.37	90.63	93.93	90.03	92.10

Table 7. Accuracy of average pooling on standard MNIST dataset with different learning rates and batch sizes with comparative approaches.

MNIST Average Pooling																								
Learning Rate		0.01																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	98.24	98.28	97.96	96.87	96.75	96.33	98.36	98.46	98.05	98.16	97.29	96.43	98.66	98.45	98.64	98.33	97.78	96.84	98.62	98.82	98.79	98.53	98.17	97.41
LeNet	95.10	94.91	94.04	94.02	94.49	94.57	96.44	96.75	96.86	97.01	96.99	96.70	96.55	96.95	96.99	95.91	94.07	94.55	96.13	96.01	96.34	95.40	94.24	94.33
AlexNet	97.50	97.31	96.62	95.77	95.89	94.85	97.00	97.48	97.39	96.87	95.40	94.07	97.26	97.10	97.03	96.50	95.70	95.09	97.20	97.52	97.56	96.55	95.93	95.26
ResNet	91.75	91.43	89.26	88.98	88.41	88.91	91.37	89.39	88.50	85.42	86.35	82.32	91.49	87.71	81.36	86.66	83.35	80.62	94.00	90.57	86.02	85.31	82.54	81.69
Learning Rate		0.001																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	98.51	98.82	98.68	98.61	98.44	97.84	98.66	98.55	98.65	98.41	98.19	97.91	98.46	98.36	98.58	98.18	97.80	97.38	97.90	97.97	98.03	97.74	97.50	96.78
LeNet	96.59	96.90	96.83	95.94	96.76	96.65	96.68	96.80	96.92	96.96	96.76	96.83	96.67	96.82	96.89	96.92	96.84	96.73	96.54	96.73	96.68	96.59	96.63	96.34
AlexNet	97.64	97.27	97.59	96.71	97.50	97.74	97.80	97.29	97.45	97.82	97.58	97.63	97.57	97.76	97.63	97.50	97.80	97.68	97.66	97.06	97.81	97.54	97.56	97.50
ResNet	93.73	91.45	89.21	88.72	87.08	85.60	93.70	91.79	90.54	88.83	87.50	85.71	92.92	91.34	90.21	89.94	87.52	86.02	91.79	90.33	89.19	87.94	86.93	85.46
Learning Rate		0.0001																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	98.99	98.37	97.87	97.18	96.69	96.68	98.02	98.71	97.35	97.44	96.50	96.47	97.08	97.42	97.84	97.00	97.78	97.57	97.19	97.18	97.40	97.18	97.47	97.34
LeNet	96.19	96.14	95.83	96.75	95.60	94.27	95.57	96.43	95.14	95.92	95.63	94.77	95.55	95.18	95.68	95.42	95.83	95.20	95.01	94.77	94.19	93.60	92.96	92.23
AlexNet	97.32	97.20	96.38	97.19	96.56	95.80	96.18	97.47	96.25	96.35	97.48	95.38	96.23	96.86	96.18	96.29	96.05	96.10	96.99	96.48	96.98	96.14	95.42	96.72
ResNet	89.32	88.16	87.07	86.01	84.66	83.42	87.73	86.54	85.71	84.76	83.42	82.39	85.43	88.47	83.66	82.72	81.68	80.77	81.98	81.36	80.63	79.93	79.02	78.10

Table 8. Accuracy of max pooling on standard CIFAR 10 dataset with different learning rates and batch sizes with comparative approaches.

CIFAR 10 Max Pooling																								
Learning Rate		0.01																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	48.56	47.10	44.25	45.25	44.23	43.28	44.82	44.39	45.17	45.91	45.96	46.21	46.47	46.64	47.16	47.93	47.89	48.39	48.94	49.08	49.73	49.20	50.08	50.40
LeNet	32.85	34.81	33.89	31.75	31.86	31.15	30.54	28.16	27.52	27.89	27.60	28.79	28.98	28.19	28.40	28.82	29.20	29.72	30.60	30.89	30.85	31.54	31.54	32.41
AlexNet	58.20	56.42	55.20	56.55	54.56	53.87	56.71	57.53	55.58	54.85	53.63	52.03	54.90	55.82	55.47	53.00	54.87	55.32	52.07	50.12	52.73	54.25	53.35	55.39
ResNet	40.26	41.25	40.75	41.57	40.58	41.26	39.54	36.09	36.47	37.75	36.73	38.95	38.21	37.00	37.25	36.95	37.09	38.60	39.00	36.03	35.50	36.49	37.29	38.86
Learning Rate		0.001																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	58.42	57.38	59.83	60.65	61.67	62.15	59.32	60.46	59.26	59.65	59.98	60.15	60.35	60.75	60.87	60.98	61.12	61.35	61.49	61.88	61.95	62.03	62.45	62.56
LeNet	32.76	34.15	34.25	34.86	33.58	34.98	35.05	35.26	35.48	35.54	35.75	35.85	36.02	36.47	36.89	36.93	37.25	37.45	37.38	37.48	37.98	38.02	38.19	38.95
AlexNet	60.05	61.96	62.91	63.68	62.48	64.19	64.82	65.03	65.59	65.98	65.89	66.21	66.35	66.68	66.74	66.85	66.98	66.96	67.05	67.54	67.68	67.89	67.14	67.68
ResNet	39.98	40.25	40.96	41.26	41.69	42.03	42.29	42.74	42.89	42.98	43.08	43.25	43.55	43.75	43.98	44.05	44.35	44.75	44.86	45.12	45.25	45.68	45.83	46.02
Learning Rate		0.0001																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	56.23	57.02	57.86	58.21	58.65	58.98	59.02	59.45	59.68	59.97	60.01	60.24	60.75	60.87	61.25	61.46	61.78	62.05	62.45	62.89	63.19	63.48	64.35	64.26
LeNet	33.15	33.57	33.78	34.02	34.54	34.65	35.01	35.26	34.68	34.87	35.02	35.15	36.06	36.24	36.49	36.78	36.98	37.15	37.86	38.19	38.45	39.15	39.45	38.98
AlexNet	61.28	61.85	62.35	62.45	62.96	63.31	63.75	63.98	64.19	64.76	64.58	64.89	65.06	65.18	65.46	65.75	65.84	65.96	67.10	67.21	67.45	67.85	67.89	68.02
ResNet	41.25	41.89	42.05	42.65	42.86	42.97	42.98	50.02	50.43	50.65	51.11	51.63	51.89	52.14	52.54	52.67	52.89	53.24	53.75	53.98	54.02	54.68	54.25	54.36

Table 9. Accuracy of average pooling on CIFAR 10 dataset with different learning rates and batch sizes with comparative approaches.

CIFAR 10 Average Pooling																								
Learning Rate		0.01																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	56.35	56.29	56.01	56.95	56.00	56.54	56.92	54.86	56.09	56.23	54.45	49.00	58.55	50.43	44.59	47.68	43.05	41.60	58.53	54.49	48.99	49.70	41.35	36.06
LeNet	44.58	46.33	39.54	42.47	34.27	41.90	49.71	31.46	30.14	27.41	27.19	31.19	36.26	36.36	35.03	32.61	31.82	29.69	40.22	39.84	39.30	37.75	35.75	33.56
AlexNet	54.65	50.19	42.32	48.82	47.06	46.84	55.62	45.83	47.41	44.35	43.21	40.54	49.97	47.07	43.44	44.32	40.95	42.17	49.94	30.31	35.74	41.95	38.24	36.59
ResNet	40.26	32.00	27.75	28.57	28.15	23.34	30.54	26.09	21.74	21.75	20.73	21.95	35.21	28.00	22.90	23.95	23.09	20.60	30.00	26.08	24.50	24.49	24.29	23.86
Learning Rate		0.001																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	70.28	70.08	68.44	68.48	65.73	59.65	67.89	68.63	68.05	66.68	63.04	60.25	66.12	63.83	65.76	63.57	60.85	57.05	69.90	61.00	60.72	59.71	57.50	55.37
LeNet	41.86	41.49	40.03	37.76	35.35	31.95	41.99	40.85	37.81	34.83	31.44	30.44	40.63	38.32	34.65	30.98	30.85	30.13	38.69	34.61	31.49	30.62	30.39	30.04
AlexNet	53.96	48.28	43.79	48.06	44.09	49.10	54.75	50.98	56.16	40.21	46.14	41.83	52.17	48.64	48.02	48.93	49.08	47.54	50.94	43.41	45.02	40.14	47.18	44.74
ResNet	39.28	34.77	32.67	31.27	29.99	27.37	38.83	34.70	33.28	32.28	30.57	28.77	36.77	34.37	32.52	31.40	29.73	28.73	36.56	35.00	33.85	32.88	32.17	31.28
Learning Rate		0.0001																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	64.04	64.05	63.10	61.83	60.32	58.44	61.46	60.34	59.79	58.77	57.36	55.53	58.85	58.53	57.54	56.63	55.64	53.35	56.80	55.50	54.87	54.18	52.82	51.01
LeNet	31.44	30.14	29.49	29.32	29.26	29.06	30.48	29.49	29.31	29.24	28.99	28.77	29.42	29.08	28.78	28.62	28.76	28.40	28.34	28.29	28.21	28.00	27.88	27.82
AlexNet	52.21	53.23	52.01	58.92	54.20	48.83	57.41	59.58	55.90	51.60	51.76	46.34	46.83	42.49	45.29	48.27	46.63	43.68	48.63	56.63	43.78	47.74	47.29	42.86
ResNet	33.94	33.23	32.50	32.04	31.29	30.58	33.11	32.44	31.84	31.37	31.13	30.66	31.44	31.17	32.14	30.43	30.30	29.59	28.39	28.16	27.55	27.10	26.80	26.47

Table 10. Accuracy of MAX pooling on standard CIFAR 100 datasets with different learning rates and batch sizes with comparative approaches.

CIFAR 100 Max Pooling																								
Learning Rate		0.01																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	12.04	11.50	11.89	12.00	12.05	12.85	12.45	12.50	13.68	14.01	14.35	14.88	15.54	15.96	16:00	16.32	16.45	17:00	18.29	18.84	19.28	20.01	20.58	22.26
LeNet	8.00	9.09	9.45	10.00	10.45	10.68	10.25	11.01	11.01	11.25	11.28	11.85	11.63	11.98	12.01	12.24	12.35	12.68	12.75	12.88	13.01	13.24	13.28	13.50
AlexNet	18.30	17.26	18.92	18.00	18.56	18.68	17.86	18.10	18.56	18.87	19.02	19.35	20.8	23.84	24.52	24.85	25.52	25.88	26.32	26.89	28.01	28.84	29.56	30.52
ResNet	11.26	10.70	11.25	11.75	11.85	12.00	12:05	12.85	12.95	12.98	12.08	12.67	12.98	13.02	13.43	13.86	13.98	14.00	14.02	14.56	14.78	15.06	15.46	16.85
Learning Rate		0.001																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	12.05	12.39	12.85	13.01	13.52	13.78	13.99	13.63	14.02	14.79	14.95	14.70	15.78	15.91	15.96	16.48	16.46	16.65	16.50	17.39	17.78	18.78	90.38	20.51
LeNet	8.59	8.76	8.98	8.96	9.15	9.76	9.89	10.01	10.05	9.89	10.76	10.84	10.99	11.00	11.05	11.42	11.65	11.85	12.01	12.25	12.36	12.88	13.01	13.95
AlexNet	20.25	20.58	20.54	21.25	21.08	21.54	21.56	23.65	23.72	23.85	24.02	24.31	24.08	24.56	24.85	24.92	25.00	25.06	25.96	26.74	27.85	28.05	29.25	30.89
ResNet	11.02	11.58	11.65	11.98	12.25	12.35	12.98	13.02	13.36	13.49	13.95	13.86	14.02	14.09	14.20	14.39	14.56	13.02	13.25	13.98	14.02	14.68	14.96	15.09
Learning Rate		0.0001																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	11.05	11.59	12.65	12.98	13.08	13.15	13.85	14.45	14.59	14.99	15.21	15.36	15.97	15.94	16.01	16.42	16.85	16.78	16.97	17.05	17.64	17.88	18.42	18.96
LeNet	9.02	9.32	9.67	9.98	10.02	10.25	10.76	10.68	10.98	11.09	11.35	11.68	11.99	11.97	12.08	12.33	12.65	13.02	13.35	13.98	13.88	13.89	14.25	14.65
AlexNet	19.36	19.95	20.05	20.65	20.98	20.85	21.09	21.85	21.96	22.15	22.85	22.46	22.99	23.32	23.45	23.98	24.00	24.25	34.85	25.98	26.35	27.65	28.97	29.56
ResNet	10.25	11.26	11.75	12.25	12.75	12.98	13.00	13.25	13.75	13.67	13.98	14.20	14.65	14.78	15.02	15.32	15.67	15.94	16.03	16.45	16.62	16.84	16.98	17.13

Table 11. Accuracy of average pooling on CIFAR 100 dataset with different learning rates and batch sizes with comparative approaches.

CIFAR 100 Average Pooling																								
Learning Rate		0.01																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	14.04	15.00	16.47	16.00	15.00	14.00	15.04	14.00	15.47	15.00	15.25	14.37	17.93	16.95	16.36	14.96	15.51	14.26	22.77	26.86	22.00	22.42	22.23	22.00
LeNet	10.00	09.00	09.25	09.25	10.45	10.25	10.00	09.00	09.25	09.25	10.45	10.25	11.23	09.72	06.66	11.00	11.59	11.00	16.82	12.05	09.88	11.46	09.78	08.00
AlexNet	13.30	13.26	13.02	13.00	12.00	13.24	13.69	13.38	13.78	13.29	12.78	13.89	14.00	13.19	13.01	13.00	12.90	12.39	20.22	21.48	21.35	21.25	20.57	21.40
ResNet	08.26	06.70	06.66	05.28	03.62	03.45	09.00	05.47	05.69	04.76	05.04	04.23	08.45	06.92	04.69	05.85	05.12	03.92	09.56	08.18	07.61	06.62	06.26	04.49
Learning Rate		0.001																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	31.57	31.58	31.19	30.36	30.59	30.60	29.99	30.63	29.02	26.79	24.05	23.70	30.78	30.11	27.36	25.48	23.36	20.45	29.22	28.19	29.47	29.35	28.18	28.51
LeNet	23.84	24.34	22.96	21.59	20.45	18.99	20.96	20.09	20.83	20.99	20.23	18.41	24.33	23.77	22.84	21.66	19.76	18.14	23.59	22.66	22.21	20.64	19.61	17.79
AlexNet	25.19	25.98	25.41	25.16	24.68	24.88	21.17	21.62	21.41	21.90	21.49	21.15	27.49	27.34	26.76	24.84	22.43	19.27	25.79	25.78	23.92	23.09	22.24	20.69
ResNet	10.72	09.52	08.59	07.65	07.15	06.21	11.21	10.24	09.32	08.85	08.01	07.08	10.77	09.92	09.00	08.54	08.06	07.55	10.24	09.48	08.65	08.12	07.77	07.04
Learning Rate		0.0001																						
Batch Size		32					64					128					256							
Drop out	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5	NO	0.1	0.2	0.3	0.4	0.5
CNN	38.86	37.58	36.05	33.95	33.47	31.79	35.08	35.11	35.39	32.71	31.13	30.40	35.37	33.48	32.59	31.03	39.72	38.94	32.89	32.36	31.48	30.66	38.75	36.37
LeNet	19.02	19.31	18.01	17.62	17.18	15.87	17.86	17.51	17.22	16.45	15.91	14.52	16.08	15.85	15.58	14.86	14.35	13.14	14.43	14.95	13.58	13.82	12.71	11.88
AlexNet	26.95	27.96	29.61	29.03	26.54	23.82	27.59	30.22	21.08	29.56	25.46	25.27	29.93	25.42	26.65	24.14	25.29	22.31	29.55	28.98	23.88	21.75	22.94	26.06
ResNet	08.18	07.89	07.98	07.52	07.44	06.78	07.50	07.55	07.23	06.83	06.65	06.59	06.87	06.84	06.68	06.25	06.25	06.04	06.08	05.71	05.85	06.06	05.66	05.54

When average pooling was employed, the performance dynamics shifted, with CNN taking the lead. CNN's architecture leveraged the generalization provided by average pooling to smooth feature representations, resulting in more stable and consistent accuracy, particularly at moderate dropout rates and optimized batch sizes. This suggests that average pooling is effective for CNN, as it promotes a balanced feature representation across spatial dimensions and reduces overfitting. AlexNet, while still achieving respectable accuracy, performed suboptimally with average pooling. Its reliance on max pooling for optimal feature extraction limited its ability to fully exploit the benefits of average pooling, which is better suited for models that require feature smoothing rather than emphasizing the most activated features.

Both ResNet and LeNet exhibited relatively lower performance with both pooling methods. ResNet's complex architecture with skip connections did not gain significant benefits from either pooling strategy, while LeNet's simpler structure was unable to fully capitalize on the feature-extraction capabilities of max or average pooling. Overall, this study underscores the importance of selecting pooling strategies that align with the architectural strengths of deep learning models. The findings demonstrate that AlexNet excels with max pooling, while CNN performs best with average pooling, highlighting the necessity of adaptive pooling approaches to optimize model accuracy based on the specific dataset and model architecture.

3.6. Statistical Significance Analysis

To further validate the robustness and reliability of the comparative results between max and average pooling methods across standard datasets (CIFAR-10, CIFAR-100, and MNIST) using CNN, AlexNet, ResNet, and LeNet, a statistical significance analysis was conducted. Specifically, p -values were calculated to assess whether the differences in performance metrics, such as accuracy, precision, and computational efficiency, were statistically significant. The p -values provide an objective measure to determine whether the observed differences between pooling methods are due to random variations or represent true distinctions in performance. [33]. A significance threshold of 0.05 was adopted, where p -values below this level indicate that the performance differences are statistically significant and unlikely to have occurred by chance. Table 12 shows the comparative analysis of p -values on standard datasets.

Table 12. Comparative analysis of p -value on a standard dataset.

Model	Learning Rate	p Values	
		Max Pooling	Average Pooling
MNIST			
CNN	0.01	0.3961	0.1240
	0.001	0.6271	0.0075
	0.0001	0.0446	0.0003
LeNet	0.01	6.3906×10^{-9}	1.8355×10^{-12}
	0.001	0.0473	0.0082
	0.0001	2.9744×10^{-8}	1.5924×10^{-7}
AlexNet	0.01	0.2516	0.8749
	0.001	0.5902	0.5471
	0.0001	0.2287	0.0172
ResNet	0.01	0.9932	0.9822
	0.001	0.8242	0.8891
	0.0001	0.1828	0.0005

Table 12. Cont.

Model	Learning Rate	<i>p</i> Values	
		Max Pooling	Average Pooling
CIFAR 10			
CNN	0.01	0.0042	0.0001
	0.001	0.5674	0.0490
	0.0001	0.8008	3.1530×10^{-5}
LeNet	0.01	5.3257×10^{-11}	1.2731×10^{-7}
	0.001	5.1826×10^{-9}	0.1558
	0.0001	5.0826×10^{-13}	0.0004
AlexNet	0.01	0.2337	0.4183
	0.001	0.3398	0.9431
	0.0001	0.0398	0.0719
ResNet	0.01	0.1309	0.1332
	0.001	0.8909	0.8903
	0.0001	1.0087×10^{-7}	1.1491×10^{-7}
CIFAR 100			
CNN	0.01	0.2007	0.2958
	0.001	0.9772	0.2828
	0.0001	0.0977	0.0166
LeNet	0.01	0.1201	0.0002
	0.001	0.0925	0.8823
	0.0001	1.0136×10^{-5}	2.5128×10^{-5}
AlexNet	0.01	0.3937	0.0322
	0.001	1.0139×10^{-5}	0.0003
	0.0001	0.5415	0.2337
ResNet	0.01	0.3217	0.4312
	0.001	0.7516	0.7295
	0.0001	0.1775	8.0274×10^{-7}

The *p*-value results highlight the comparative performance of max pooling and average pooling across various models and datasets, demonstrating that max pooling consistently yields more statistically significant results in many cases. For instance, in the MNIST dataset, max pooling shows superior performance, especially with LeNet, where its *p*-values are significantly lower across all learning rates, indicating stronger statistical significance compared to average pooling. This trend is also evident in the CIFAR-10 dataset, where max pooling exhibits better robustness, particularly at a learning rate of 0.01, with a *p*-value of 0.0042 compared to 0.0001 for average pooling. Similarly, in the CIFAR-100 dataset, max pooling performs better at lower learning rates, maintaining strong statistical significance with more consistent *p*-values across architectures like CNN and LeNet. While average pooling occasionally shows lower *p*-values, particularly in deeper models like ResNet, max pooling demonstrates greater consistency and reliability across the board. Overall, max pooling emerges as the more robust and reliable pooling method, offering better statistical significance and performance stability across a variety of datasets and architectures. Figure 5 illustrates the comparative analysis of *p*-values for max pooling and average pooling across different neural network architectures and data.

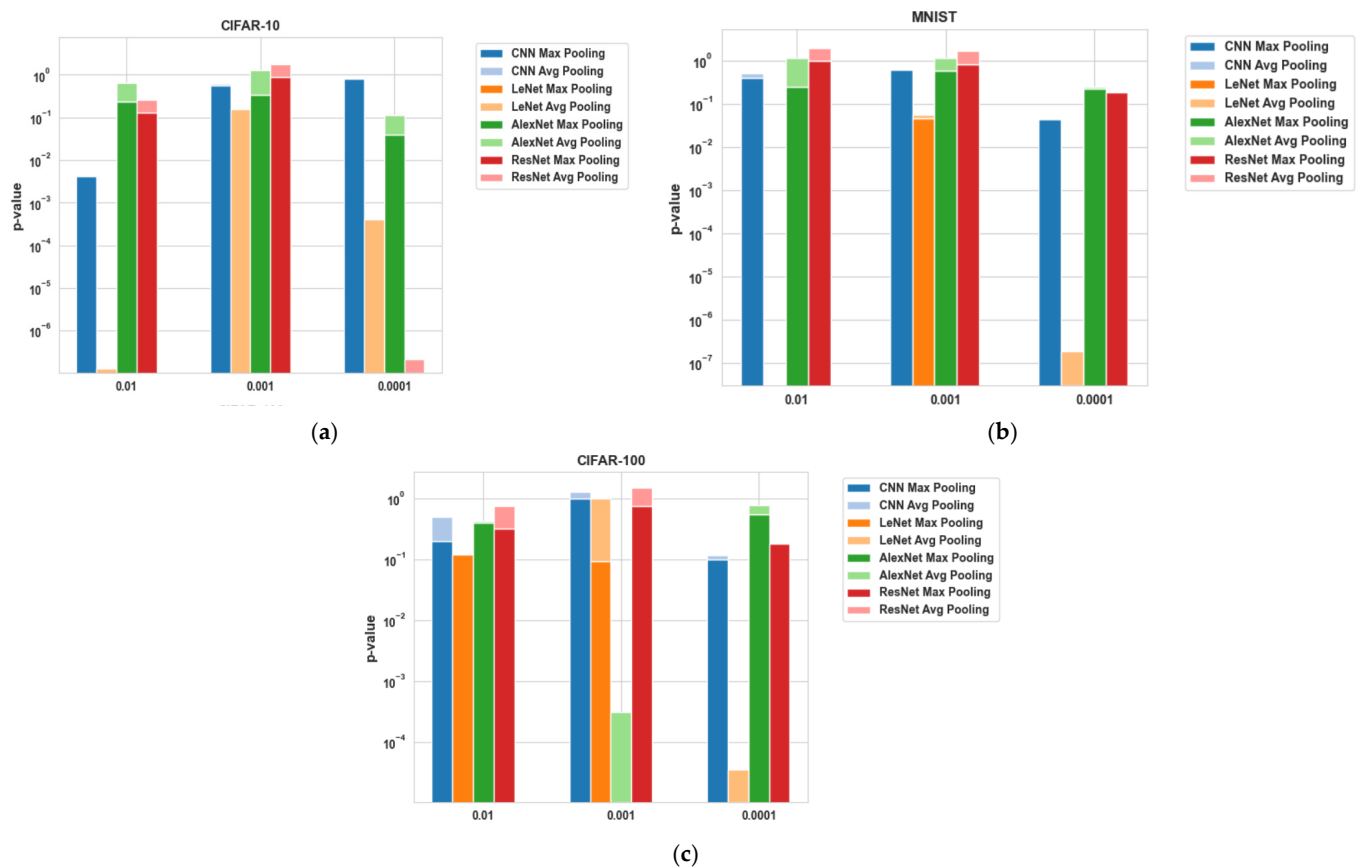
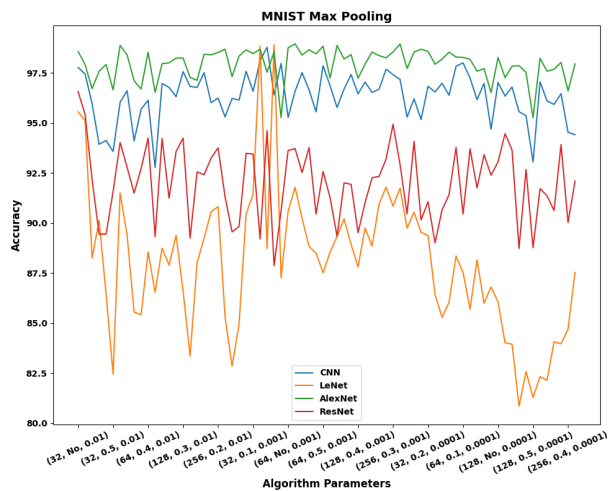


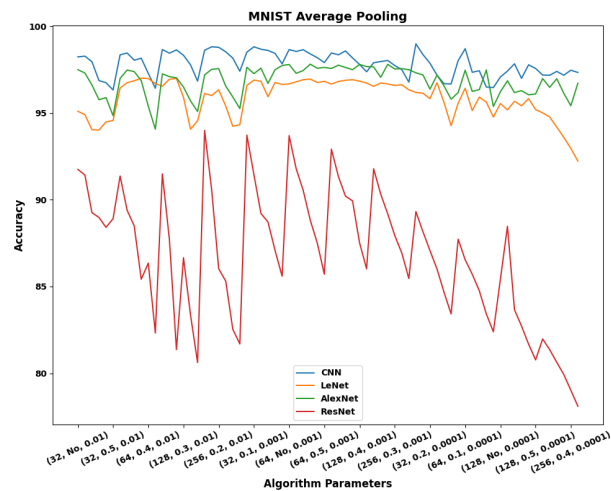
Figure 5. Comparison of the statistical significance of p -values for max pooling and average pooling in CNN, LeNet, AlexNet, and ResNet across MNIST (b), CIFAR-10 (a), and CIFAR-100 (c) datasets.

3.7. Convergence Graph

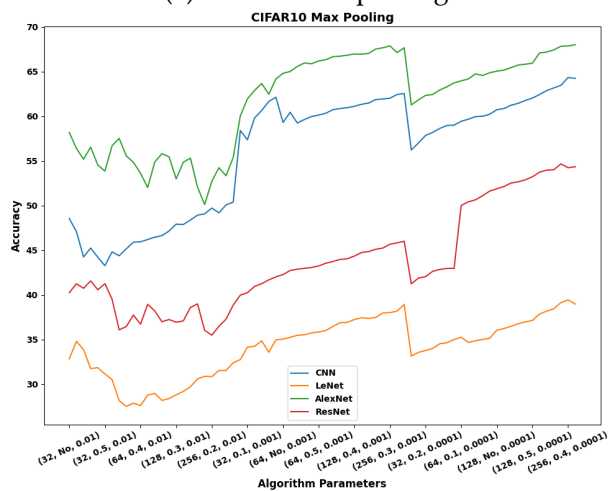
A convergence graph visualizes the learning progress of a model over time, illustrating the effectiveness of various training parameters and architectures. This analysis focuses on the convergence graphs of four neural network architectures, CNN, AlexNet, LeNet, and ResNet, generated using Matplotlib, as shown in Figure 6. Key observations from the comparative analysis reveal that CNN and AlexNet exhibit stable convergence with minimal oscillations, indicating reliable training processes. In contrast, AlexNet converges quickly due to its deeper architecture, while ResNet shows initial instability but ultimately achieves strong performance, reflecting its robustness. AlexNet and ResNet demonstrate rapid initial improvements due to their complex designs, effectively capturing intricate patterns, whereas CNN and LeNet show more gradual progress. The plateau in AlexNet and stabilization in ResNet suggest these models reach optimal performance quickly, while CNN and LeNet may require more epochs for full optimization. Overall, the graphs highlight the distinct strengths of each architecture: CNN and AlexNet provide stable learning for simpler tasks, AlexNet excels in fast early stage learning for complex data, and CNN, despite initial fluctuations, demonstrates high performance and robustness. Adjusting hyperparameters like learning rate and batch size can further enhance these architectures for use in specific datasets and tasks.



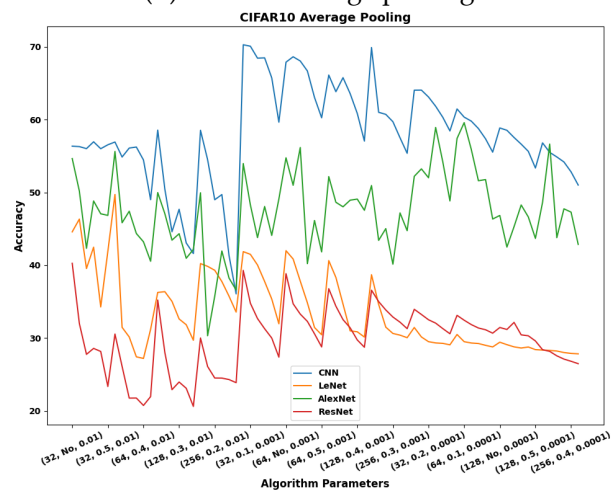
(a) MNIST max pooling.



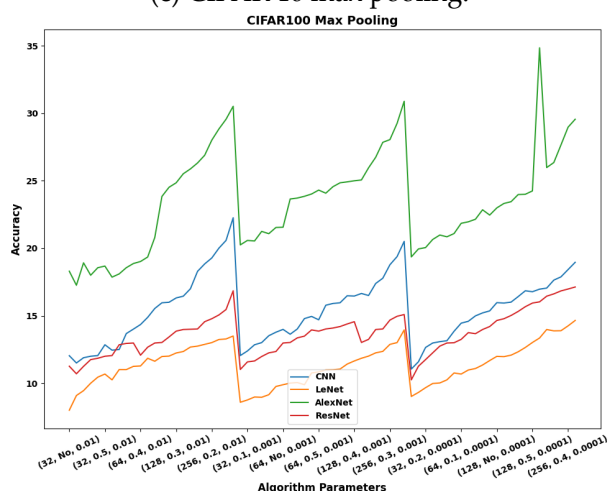
(b) MNIST average pooling.



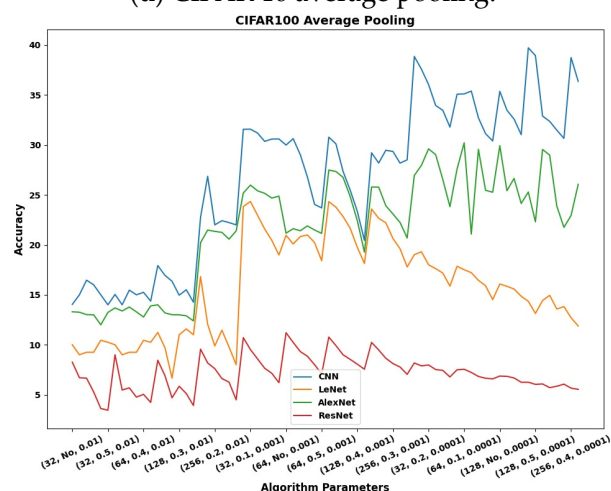
(c) CIFAR 10 max pooling.



(d) CIFAR 10 average pooling.



(e) CIFAR 100 max pooling.



(f) CIFAR 100 average pooling.

Figure 6. Accuracy comparison against different parameters on MNIST, CIFAR 10, and CIFAR 100.

3.8. Analysis of Optimal Pooling Performance and Parameter Trends

This section identifies and discusses the optimal performance achieved by max and average pooling across the MNIST, CIFAR-10, and CIFAR-100 datasets while also addressing the observed trends in performance based on parameter variations.

The CNN performs best with average pooling, achieving higher accuracy on MNIST and leveraging this pooling method to generalize effectively across the dataset. Specifically, CNN peaks with average pooling at 98.82% accuracy on MNIST using a learning rate of 0.001 and a batch size of 32 with a low dropout rate, suggesting that CNN benefits from the smoothed feature extraction of average pooling. In contrast, ResNet exhibits superior performance with max pooling, particularly on CIFAR-10 and CIFAR-100, where complex features demand higher selectivity. For instance, ResNet's optimal CIFAR-10 performance with max pooling reaches 54.68% at a learning rate of 0.0001 and dropout of 0.4–0.5, showcasing its alignment with max pooling's concentrated feature selection.

On the MNIST dataset, each model displays varied responses to pooling methods. LeNet performs moderately, achieving its peak of 98.9% accuracy with average pooling and a learning rate of 0.001 on batch size of 128, though it generally trails CNN and AlexNet. AlexNet demonstrates robustness with both pooling methods, attaining high performance with max pooling, especially on CIFAR-10, where it reaches 67.89% accuracy with a learning rate of 0.001 and moderate dropout. This suggests that AlexNet's deeper architecture handles CIFAR-10's feature complexity well under max pooling. ResNet, which excels at retaining intricate feature details, benefits significantly from max pooling across datasets, particularly CIFAR-100, achieving around 17.13% accuracy. Although this result is lower than with simpler datasets, it reflects ResNet's high selectivity in feature extraction.

On CIFAR-100, which is more challenging due to its 100-class complexity, CNN's accuracy peaks at 22.26% with max pooling, again showing a preference for structured feature selection. LeNet, on the other hand, struggles to maintain high accuracy across both pooling methods, achieving only around 13.5% accuracy on CIFAR-100, pointing to limitations in its simpler architecture for such complex data. AlexNet continues to perform relatively well on CIFAR-100, reaching its highest accuracy at around 30.89% with max pooling, while ResNet achieves its best result with max pooling, reaching around 17.13%. The analysis confirms that CNN benefits most from average pooling on MNIST, while ResNet performs best on max pooling across CIFAR datasets. This pattern underlines the importance of aligning pooling techniques with model architecture and dataset complexity for optimal performance.

The observed parameter trends in Tables 6–11 reveal the importance of the batch size, learning rate, and dropout rate adjustments, which significantly impact pooling performance across different architectures and datasets. Lower learning rates and moderate batch sizes generally enhanced model stability and accuracy, especially for complex datasets like CIFAR-100. Higher dropout rates tended to improve generalization in average pooling, which smooths feature representations, though they occasionally hindered accuracy in configurations where feature retention was critical, as in max pooling. These trends underscore that max pooling often benefits from moderate batch sizes and minimal dropout to retain strong activations, whereas average pooling performs optimally with larger batch sizes and higher dropout rates, especially for noisier datasets. It is important to note that while these trends offer insight, generalizing them across all applications is challenging, as parameter effects can vary significantly depending on dataset characteristics and task requirements. Adaptive parameter tuning based on dataset-specific needs could provide more reliable results in future studies. Overall, these findings suggest that selecting pooling methods based on dataset complexity and noise levels is crucial, particularly when applying CNNs to resource-constrained environments, where optimized configurations can greatly enhance performance and generalization.

3.9. Discussion

This research provided an in-depth analysis of various pooling methods in Convolutional Neural Networks (CNNs), specifically max pooling, across datasets such as MNIST, CIFAR-10, and CIFAR-100. The results demonstrate that each method has unique strengths and limitations that make it suitable for different tasks. Max pooling consistently performed well in scenarios where preserving high-contrast features and robustness to noise were

critical. The ability of max pooling to capture the most prominent features from the feature maps allowed it to excel in classification tasks with high-dimensionality input, such as the CIFAR-100 dataset. However, the technique's downside lies in its tendency to discard potentially useful information, which may explain its reduced performance when applied to datasets with smaller objects or more intricate details, where preserving all information is important. This is particularly relevant in applications such as small object detection, where discarding finer details may lead to poor localization accuracy, as noted in several studies on object detection. In contrast, average pooling showed more balanced feature representation and performed well in complex datasets, such as CIFAR-10. By smoothing out noise and reducing the impact of any outlier values in the feature map, average pooling can generalize better in tasks where the input is noisy or where capturing the overall context is more important than highlighting specific high-contrast features. For instance, in semantic segmentation tasks, where each pixel's classification matters more than a focus on high-intensity regions, average pooling may outperform max pooling. However, the downside of this method is its inability to preserve sharp edges and fine details, which are crucial in high-precision tasks like medical image analysis or object detection. Min pooling, though less commonly used, showed its utility in highly specialized tasks such as anomaly detection and background subtraction. By focusing on the least prominent features, min pooling can highlight anomalies or subtle differences in an image, which can be critical in applications like fraud detection or medical imaging, where identifying outliers or rare features is essential. However, the sensitivity of min pooling to noise limits its general applicability in mainstream image-classification tasks, where higher-contrast features dominate.

The results also underscore that no single pooling method universally outperforms others across all tasks, architectures, and datasets. The effectiveness of a pooling technique is highly dependent on the specific task and dataset characteristics. Max pooling might be preferable for tasks involving object detection or datasets with large, prominent features, whereas average pooling would be a better choice for more balanced, noisy datasets requiring more generalization, such as in semantic segmentation. Moreover, recent advancements in CNN architectures, such as hybrid pooling methods and adaptive pooling strategies, have sought to combine the strengths of multiple pooling operations. For example, adaptive pooling methods, which dynamically adjust the pooling strategy based on the input, have shown promise in enhancing performance by balancing feature preservation and computational efficiency. These approaches allow CNNs to adapt better to the varying complexities of different datasets, especially for tasks requiring fine-grained classification like medical imaging and high-precision applications. In practical terms, the choice of pooling method has important implications for resource-constrained environments. Max pooling offers high accuracy but at the cost of potentially discarding important information, while average pooling provides a more generalizable solution but with potential loss of detail. Min pooling is effective for specialized tasks but may not be suitable for broader applications. Future research could benefit from exploring adaptive pooling techniques that optimize the trade-offs between computational cost and feature retention, ensuring that the choice of pooling method aligns with specific task requirements.

4. Conclusions

Convolutional Neural Networks (CNNs) play a crucial role in computer vision, with pooling layers improving efficiency by reducing complexity while preserving key features. This study compared max and pooling across CNN architectures (AlexNet, ResNet, and LeNet) using datasets like MNIST and CIFAR-10. Max pooling performed best in high-dimensionality tasks, and average pooling excelled in handling noisy data. Specifically, the practical recommendations based on the results are to use max pooling for resource-limited environments, and average pooling for tasks involving noisy datasets. While max pooling has demonstrated robust performance across the datasets in this study, average pooling may still play a valuable role in certain contexts, such as noisy data environments or tasks

emphasizing spatial continuity. By exploring adaptive or hybrid pooling methods in future research, it may be possible to leverage the strengths of both max and average pooling to enhance model flexibility and performance across a broader range of applications. Future research should explore integrating pooling strategies with Vision Transformers (ViTs) to reduce computational overhead, and adaptive pooling methods could enhance performance in tasks like small object detection and fine-grained classification. Vision Transformers may revolutionize feature extraction by processing global image context without traditional pooling layers.

Author Contributions: Conceptualization, A.Z.; methodology, A.A. (Amerah Alabrah) and N.S.; software, S.Z.; investigation, M.N.; writing—original draft preparation, S.R. and M.S.; writing—review and editing, S.R. and A.A. (Amerah Alabrah); supervision, A.A. (Ali Arshad); funding acquisition, A.A. (Amerah Alabrah). All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Researchers Supporting Project number RSP2024R476, King Saud University, Riyadh, Saudi Arabia.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhao, X.; Wang, L.; Zhang, Y.; Han, X.; Devenci, M.; Parmar, M. A review of convolutional neural networks in computer vision. *Artif. Intell. Rev.* **2024**, *57*, 99. [CrossRef]
2. Archana, R.; Jeevaraj, P.E. Deep learning models for digital image processing: A review. *Artif. Intell. Rev.* **2024**, *57*, 11. [CrossRef]
3. Singh, S.; Gupta, A.; Katiyar, K. Neural modeling and neural computation in a medical approach. In *Computational Techniques in Neuroscience*; CRC Press: Boca Raton, FL, USA, 2023; pp. 19–41.
4. Taye, M.M. Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions. *Computation* **2023**, *11*, 52. [CrossRef]
5. Jiang, P.; Xue, Y.; Neri, F. Convolutional neural network pruning based on multi-objective feature map selection for image classification. *Appl. Soft Comput.* **2023**, *139*, 110229. [CrossRef]
6. Valkenburg, D.; Rousseau, A.J.; Geubbelmans, M.; Burzykowski, T. Support vector machines. *Am. J. Orthod. Dentofac. Orthop.* **2023**, *164*, 754–757. [CrossRef]
7. Zhang, Z. Introduction to machine learning: K-nearest neighbors. *Ann. Transl. Med.* **2019**, *4*, 218. [CrossRef]
8. Zhao, T.; Xie, Y.; Wang, Y.; Cheng, J.; Guo, X.; Hu, B.; Chen, Y. A survey of deep learning on mobile devices: Applications, optimizations, challenges, and research opportunities. *Proc. IEEE* **2022**, *110*, 334–354. [CrossRef]
9. De Oliveira, C.I.; do Nascimento, M.Z.; Roberto, G.F.; Tosta, T.A.; Martins, A.S.; Neves, L.A. Hybrid models for classifying histological images: An association of deep features by transfer learning with ensemble classifier. *Multimed. Tools Appl.* **2024**, *83*, 21929–21952. [CrossRef]
10. Dogan, Y. A new global pooling method for deep neural networks: Global average of top-k max-pooling. *Trait. Du Signal* **2023**, *40*, 577–587. [CrossRef]
11. Chen, Y.; Fang, J.; Zhang, X.; Miao, Y.; Lin, Y.; Tu, R.; Hu, L. Pool fire dynamics: Principles, models and recent advances. *Prog. Energy Combust. Sci.* **2023**, *95*, 101070. [CrossRef]
12. Pan, X.; Xu, J.; Pan, Y.; Wen, L.; Lin, W.; Bai, K.; Fu, H.; Xu, Z. Afinet: Attentive feature integration networks for image classification. *Neural Netw.* **2022**, *155*, 360–368. [CrossRef] [PubMed]
13. Zhao, L.; Zhang, Z. A improved pooling method for convolutional neural networks. *Sci. Rep.* **2024**, *14*, 1589. [CrossRef] [PubMed]
14. Krichen, M. Convolutional neural networks: A survey. *Computers* **2023**, *12*, 151. [CrossRef]
15. Matoba, K.; Dimitriadis, N.; Fleuret, F. Benefits of Max Pooling in Neural Networks: Theoretical and Experimental Evidence. In *Transactions on Machine Learning Research*; 2023. Available online: <https://openreview.net/forum?id=YgeXqrH7gA> (accessed on 15 September 2024).
16. Qiu, Y.; Liu, Y.; Chen, Y.; Zhang, J.; Zhu, J.; Xu, J. A2SPPNet: Attentive atrous spatial pyramid pooling network for salient object detection. *IEEE Trans. Multimed.* **2022**, *25*, 1991–2006. [CrossRef]
17. Tong, K.; Wu, Y.; Zhou, F. Recent advances in small object detection based on deep learning: A review. *Image Vis. Comput.* **2020**, *97*, 103910. [CrossRef]
18. Zhou, J.; Liang, Z.; Tan, Z.; Li, W.; Li, Q.; Ying, Z.; Zhai, Y.; He, Y.; Shen, Z. RVDNet: Rotated Vehicle Detection Network with Mixed Spatial Pyramid Pooling for Accurate Localization. In *International Conference on Artificial Intelligence and Communication Technology*; Springer Nature: Singapore, 2023; pp. 303–316.
19. Özdemir, C. Avg-topk: A new pooling method for convolutional neural networks. *Expert Syst. Appl.* **2023**, *223*, 119892. [CrossRef]

20. Tang, T.N.; Kim, K.; Sohn, K. Temporalmaxer: Maximize temporal context with only max pooling for temporal action localization. *arXiv* **2023**, arXiv:2303.09055.
21. Bianchi, F.M.; Lachi, V. The expressive power of pooling in graph neural networks. *Adv. Neural Inf. Process. Syst.* **2024**, *36*. [[CrossRef](#)]
22. Zhu, X.; Meng, Q.; Ding, B.; Gu, L.; Yang, Y. Weighted pooling for image recognition of deep convolutional neural networks. *Clust. Comput.* **2019**, *22*, 9371–9383. [[CrossRef](#)]
23. Stergiou, A.; Poppe, R.; Kalliatakis, G. Refining activation downsampling with SoftPool. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 10357–10366.
24. Walter, B. Analysis of convolutional neural network image classifiers in a hierarchical max-pooling model with additional local pooling. *J. Stat. Plan. Inference* **2023**, *224*, 109–126. [[CrossRef](#)]
25. Chen, J.; Hu, H.; Wu, H.; Jiang, Y.; Wang, C. Learning the best pooling strategy for visual semantic embedding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021; pp. 15789–15798.
26. Khairandish, M.O.; Sharma, M.; Jain, V.; Chatterjee, J.M.; Jhanjhi, N.Z. A hybrid CNN-SVM threshold segmentation approach for tumor detection and classification of MRI brain images. *IRBM* **2022**, *43*, 290–299. [[CrossRef](#)]
27. Ding, Y.; Yu, J.; Lv, Q.; Zhao, H.; Dong, J.; Li, Y. Multiview adaptive attention pooling for image–text retrieval. *Knowl.-Based Syst.* **2024**, *291*, 111550. [[CrossRef](#)]
28. Li, H.; Cheng, Y.; Ni, H.; Zhang, D. Dual-path recommendation algorithm based on CNN and attention-enhanced LSTM. *Cyber-Phys. Syst.* **2024**, *10*, 247–262. [[CrossRef](#)]
29. Han, Y.; Huang, G.; Song, S.; Yang, L.; Wang, H.; Wang, Y. Dynamic neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 7436–7456. [[CrossRef](#)] [[PubMed](#)]
30. Seng, L.M.; Chiang, B.B.C.; Salam, Z.A.A.; Tan, G.Y.; Chai, H.T. MNIST handwritten digit recognition with different CNN architectures. *J. Appl. Technol. Innov* **2021**, *5*, 7–10.
31. Giuste, F.O.; Vizcarra, J.C. Cifar-10 image classification using feature ensembles. *arXiv* **2020**, arXiv:2002.03846.
32. Singla, S.; Singla, S.; Feizi, S. Improved deterministic l2 robustness on CIFAR-10 and CIFAR-100. *arXiv* **2021**, arXiv:2108.04062.
33. Hopkins, W.G.; Rowlands, D.S. Standardization and other approaches to meta-analyze differences in means. *Stat. Med.* **2024**, *43*, 3092–3108. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.