Comparative Analysis of Avg Top-K pooling in Convolutional Neural Networks for Enhanced Image Classification

BY

NISHANT OMHARE (21/11/EE/034)

under the supervision of

Pushpendra Singh, School of Engineering JNU, Delhi

in the partial fulfilment of the requirements
for the award of the degree of

Bachelor of Technology

School of Engineering
Jawaharlal Nehru University, Delhi
May, 2025

JAWAHARLAL NEHRU UNIVERSITY
SCHOOL OF ENGINEERING

DECLARATION

I declare that the project work entitled "**Comparative Analysis of Avg Top-K pooling in Convolutional Neural Networks for Enhanced Image Classification**" which is submitted by me in partial fulfillment of the requirement for the award of degree B.Tech. (a part of Dual-Degree Programme) to the School of Engineering, Jawaharlal Nehru University, New Delhi comprises only my original work and due acknowledgement has been made in the text to all other material used.

NISHANT OMHARE
21/11/EE/034

JAWAHARLAL NEHRU UNIVERSITY
SCHOOL OF ENGINEERING

CERTIFICATE

This is to certify that the project work entitled "**Comparative Analysis of Avg Top-K pooling in Convolutional Neural Networks for Enhanced Image Classification**" being submitted by Mr. Nishant Omhare (21/11/EE/034) in fulfilment of the requirements for the award of the Bachelor of Technology in Electronics & Communication Engineering will be carried out by him under my supervision.

In my opinion, this work fulfils all the requirements of an Engineering Degree in respective streams as per the regulations of the School of Engineering, Jawaharlal Nehru University, Delhi. This thesis does not contain any work, which has been previously submitted for the award of any other degree.

Pushpendra Singh
(Supervisor)
Associate Professor
School of engineering
Jawaharlal Nehru University, Delhi
New Delhi

JAWAHARLAL NEHRU UNIVERSITY
SCHOOL OF ENGINEERING

ACKNOWLEDGEMENT

I am sincerely thankful to my project supervisor, Pushpendra Singh, for his invaluable guidance, continuous support, and insightful feedback throughout the course of this project. His expertise and encouragement have been instrumental in the successful completion of this work.
I would also like to express my appreciation to the faculty and staff of the School of Engineering department, Jawaharlal Nehru University, for providing a strong academic foundation and the necessary resources. Special thanks to the lab assistants and technical staff for their assistance and cooperation.

A heartfelt thank you to my friends and peers for their motivation, shared knowledge, and encouragement during this journey.

Most importantly, I am deeply grateful to my family for their constant support, understanding, and belief in me throughout my academic life.

Nishant Omhare

21/11/EE/034

Abstract

Convolutional neural networks (CNNs) have revolutionized image classification, achieving state-of-the-art performance on a variety of tasks [1][2]. A critical design choice in CNNs is the pooling operation, which controls how spatial information is down-sampled. This report investigates three pooling methods – Max Pooling, Average Pooling, and the novel Avg-TopK pooling[3][4]–in two CNN architectures: a VGG- style network with Batch Normalization and Dropout, and a WideResNet-mini variant. We conduct experiments on the CIFAR-10 and CIFAR-100 datasets[5][6] to evaluate how different pooling strategies affect classification accuracy. We also build an ensemble model that averages the outputs of models using different pooling layers. In summary, Avg-TopK pooling (for optimal K) yields improved accuracy over conventional pooling, and the ensemble of pooling-variant models achieves the best results. On CIFAR-10, the ensemble of VGG-style networks reached accuracies of roughly 88.0%–88.76%, while the WideResNet ensemble achieved about 90.0%–91.9%. On CIFAR-100, the WideResNet ensemble attained around 67%–69.8% accuracy, significantly outperforming the VGG-style models (36%–38%). These findings highlight the importance of pooling choice and ensembling in CNN design. The report is organized as follows: we first review CNN fundamentals and pooling methods, then survey related work on CNN architectures and pooling variations, describe our proposed approach and methodology, and conclude with findings and future work.

List of Contents

1. Introduction and Thesis Overview

Convolutional Neural Networks (CNNs) have become the dominant paradigm for image classification, achieving remarkable accuracy in tasks ranging from digit recognition to large-scale object detection[1][2].

For example, very deep architectures using small (3×3) convolutional filters demonstrated significant gains in image classification benchmarks[1] . Similarly, the introduction of residual learning enabled networks with over 100 layers to be trained effectively, achieving new state-of-the-art results[2] . These successes underline the flexibility and power of CNNs in modeling visual data. However, many design choices affect performance. In particular, the type of pooling operation – the mechanism by which a network reduces spatial resolution of feature maps – can influence feature retention and network generalization[7][8] .

This report examines how pooling strategies impact CNN performance, and proposes an ensemble approach that leverages multiple pooling methods. We implement two distinct CNN architectures. The first is a VGG-style network with Batch Normalization (BN) and Dropout for regularization, inspired by the classic VGG models[1]. The second is a miniaturized Wide Residual Network (WideResNet-mini) based on the Wide ResNet concept [9] [9] but with reduced depth, suitable for CIFAR-scale images. We train and evaluate these networks on the CIFAR-10 and CIFAR-100 datasets. We experiment with three types of pooling layers: traditional MaxPooling and AveragePooling, and a newer method called **Avg-TopK** pooling. In Avg- TopK pooling, only the top *k* highest activations in each pooling window are averaged, rather than taking a single max or the full average[4][10]. We vary both the window size (2×2 up to 4×4) and the parameter *k*. We also build an **ensemble model** that averages the softmax outputs of models trained with different pooling types, in order to combine their strengths.

Our experiments demonstrate that Avg-TopK pooling can improve accuracy under

certain settings. The ensemble of models using mixed pooling often yields the highest overall accuracy. For instance, on CIFAR-10 our ensemble of VGG-style networks achieved about 88.0–88.6% accuracy, and the WideResNet ensemble reached 90.0–91.8%. On the more challenging CIFAR-100 task, the WideResNet ensemble attained roughly 67–69.8%, a substantial improvement over the VGG-style networks which achieved only 36–42% with Avg-TopK pooling. These results indicate that careful selection of pooling and the use of ensembles can significantly boost classification performance.

The remainder of this report is structured as follows: Section 2 provides background on CNNs and pooling in image classification; Section 3 surveys relevant literature on CNN architectures and pooling methods; Section 4 describes our proposed CNN models and pooling strategy; Section 5 details the experimental methodology; Section 6 will present and discuss results; finally, Sections 7 and 8 conclude the report and outline future work.

2. Background on Image Classification with CNNs

CNNs combine convolutional layers, non-linear activations, and pooling layers to extract hierarchical features from images. Early work by LeCun *et al.* on the LeNet-5 network illustrated the basic structure: alternating convolutional and subsampling (pooling) layers produce progressively abstract feature maps[7]. Each convolutional layer applies learnable filters that respond to local patterns, while pooling layers down sample the feature maps to reduce spatial resolution and introduce invariance to small translations[7][11].

For example, as LeCun *et al.* discuss, a subsampling layer performs local averaging to blur and shrink the feature map, reducing the "sensitivity of the output to shifts and distortions"[7]. Modern CNNs typically use Rectified Linear Unit (ReLU) activations after convolutions and Batch Normalization to stabilize training. After several convolution + pooling blocks, the final feature maps are usually flattened and fed into fully-connected layers or a global pooling layer before the output classification layer Pooling operations come in various forms. **Max pooling** is the most common: it divides the feature map into non-overlapping (or sometimes overlapping) windows and takes the

maximum activation in each window. This tends to preserve the strongest features while discarding other information. For instance, in a 3×3 pooling region, only the largest value (e.g. 0.95) is retained (see Figure 1).

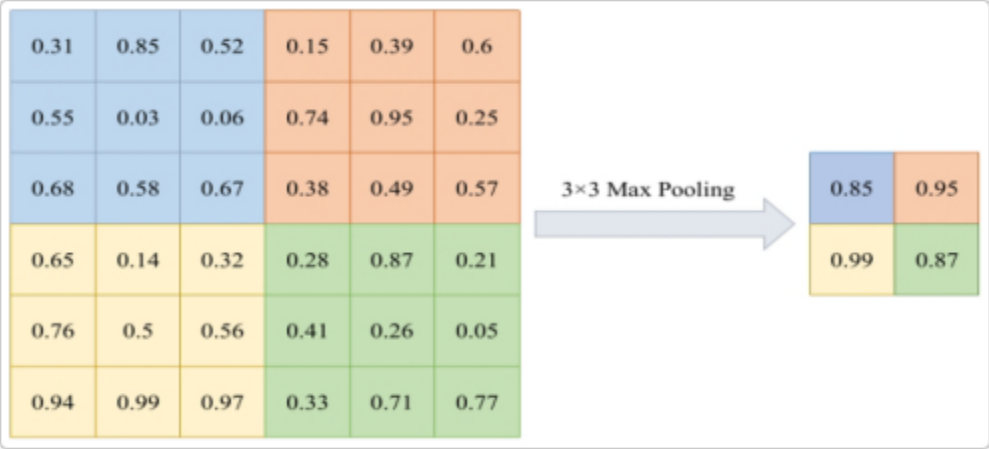

Figure 1: Illustration of a 3×3 max pooling operation on a 6×6 patch. Each 3×3 region yields its maximum value. Max pooling effectively provides translation invariance to small shifts, but as noted by Zhao and Zhang, it can lead to loss of detailed information because all but the largest activation are ignored.

Average pooling instead computes the mean of all values in each window, thus retaining a summary of all activations. It can mitigate the loss of information compared to max pooling, but tends to produce smoother, less distinguished features. To address limitations of both, **Avg-TopK pooling** was proposed[4][10]. In this method, the top *k* values in each pooling window are selected and their average is computed[4]. For example, Figure 2 shows an Avg-TopK pooling with *k*=3 on a 3×3 region: the three largest values (0.99, 0.87, 0.67) are averaged to produce the output.

Figure 2: Example of 3×3 Avg-TopK pooling (K=3). The three highest values in the 3×3 patch are averaged. This hybrid strategy preserves more information than max pooling while avoiding the dilution effect of a full average. Recent studies report that Avg-TopK pooling "consistently attains higher accuracy in image classification" than conventional pooling[13][10], especially when combined with transfer learning models. In addition to local pooling layers, modern CNNs often use **global average pooling (GAP)** near the output. GAP computes a single average over each entire feature map, converting an *h×w* map to a scalar. This drastically reduces the number of parameters and allows the network to accept variable input sizes . It is commonly used in architectures like GoogLeNet and ResNet before the final fully-connected (classification) layer. The flattened features (or GAP outputs) are fed into a softmax classifier in a multi-class setting, which produces normalized probabilities over classes[4].

Overall, the combination of convolution, activation, pooling, and fully-connected layers enables CNNs to learn powerful visual representations. Pooling, in particular, is a key component that balances information preservation and computational efficiency.

3. Literature Survey

Over the past decade, many CNN architectures and pooling variations have been proposed for image classification. The **VGG network** by Simonyan and Zisserman (2014) showed that very deep networks with small (3×3) filters[1] could achieve superior accuracy on large-scale benchmarks. Their models (with up to 19 layers) consistently outperformed shallower ones, demonstrating the benefit of depth. Later, **Residual Networks (ResNets)** by He *et al.* (2015) introduced identity skip connections to ease the training of very deep networks[2] . ResNets with depths of over 150 layers reached top-5 error of 3.57% on ImageNet when ensembled, illustrating the power of both depth and ensembling. **Wide ResNets** (Zagoruyko & Komodakis 2016) took a different approach: they reduced depth and increased channel width. The authors found that a 16-layer WideResNet outperformed much deeper thin ResNets on CIFAR and ImageNet , achieving new state-of-the-art results. These architectures have become baselines for many classification tasks.

Regularization techniques have also been essential. **Batch Normalization** (Ioffe & Szegedy, 2015) became ubiquitous for stabilizing CNN training by normalizing intermediate activations. It allows much higher learning rates and has a regularizing effect, sometimes reducing the need for other regularizers. **Dropout** (Srivastava *et al.*, 2014) is another standard method: it randomly deactivates neurons during training to prevent co-adaptation, effectively averaging an exponential number of "thinned" networks . Both BN and dropout are commonly used in CNNs to improve generalization.

Pooling methods have also been widely studied. Beyond max and average pooling, Zeiler and Fergus (2013) proposed **stochastic pooling**, a probabilistic scheme where each pooling region randomly selects one activation according to a multinomial distribution[17] . This introduces additional regularization, and the authors reported state-of-the-art performance on several image datasets without data augmentation . Other researchers have explored **learnable pooling** or **mixed pooling** strategies to combine advantages of max and average [*e.g.*, Yu *et al.* 2014]. The **Avg-TopK pooling** method by Özdemir (2023) [10]is a recent proposal: by averaging only the top *k* values in each window, it reportedly achieves significantly higher accuracy than conventional pooling on image classification tasks . For example, Özdemir reports that on CIFAR datasets, Avg-TopK outperformed max and average pooling by substantial margins . Other novel methods include pyramid pooling and attention-based pooling for capturing spatial hierarchies, though these are more common in segmentation tasks.

Ensemble methods are another important line of work. It is common in practice to train multiple CNN models and average their predictions to reduce variance and improve accuracy. Ioffe and Szegedy noted that an ensemble of batch-normalized networks further lowered ImageNet error [17] . He *et al.* famously achieved a 3.57% ImageNet error with an **ensemble** of very deep residual nets . More generally, ensembles of CNNs are widely used in competitions to push accuracy: combining models with different architectures or hyperparameters can yield gains beyond any single model.

This body of literature underscores that both architecture design and pooling strategy can have large effects on performance. The novelty of our work lies in systematically comparing pooling methods (including Avg-TopK) within a controlled setting, and demonstrating the benefit of ensembling models that differ only in their pooling layers.

4. Proposed Work

In this study, we investigate how different pooling strategies affect CNN performance and how an ensemble of diverse pooling can improve results. We design two base architectures:

- **VGG-style CNN:** A deep CNN with multiple 3×3 convolutional layers arranged in blocks (increasing the number of filters in deeper layers), following the design of Simonyan and Zisserman[1] . We incorporate Batch Normalization after each convolution to stabilize learning[12] , and apply Dropout (e.g., 50%) before the final classification layers to mitigate overfitting . The network ends with fully-connected layers mapping to the class outputs (10 or 100 classes), with a softmax activation.

- **WideResNet-mini:** A compact version of a Wide Residual Network, inspired by Zagoruyko & Komodakis. This model uses residual blocks, each containing two 3×3 convolutions and an identity skip connection. Rather than stacking hundreds of layers, we use a modest depth (e.g., on the order of 10–20 layers) but with increased channel width (e.g., width factor 4). This design leverages the “width” advantage of WideResNets while keeping computational cost low. Like the VGG model, we include Batch Normalization and ReLU activations, and apply dropout in the fully- connected layers.

We apply these models to the CIFAR-10 and CIFAR-100 datasets, which consist of 60,000 color images (32×32 pixels) each . CIFAR-10 has 10 classes (6000 images per class) , while CIFAR-100 has 100 classes (600 images per class) . We use the standard split of 50,000 training images and 10,000 test images per dataset.

For each model, we experiment with the following pooling configurations: - **MaxPooling:** The standard max-over-window pooling (with window sizes of 2×2, 3×3, or 4×4, and stride equal to the window).

AveragePooling: The mean-over-window pooling (with the same window sizes as above).

Avg-TopK Pooling: As defined by Özdemir (2023), this layer selects the top *k* activations in the pooling window and averages them[6]. We vary *k* (e.g. *k*=2, 3, 4) as a hyperparameter, as well as the window size. The goal is to identify whether a particular (*k*, window) combination yields higher accuracy.

Thus, each model configuration is identified by its network type (VGG vs WideResNet) and pooling type (Max, Avg, or Avg-TopK with specific *k*). We train a separate instance of each configuration. During training, we use standard image preprocessing (normalization) and data augmentation (e.g. random flips/crops) as appropriate for CIFAR (details in Section 6). We optimize using cross-entropy loss and stochastic gradient descent (or Adam), monitoring validation accuracy to tune learning rates. In addition to individual models, we **ensemble** the models that differ only in pooling type. Specifically, for each architecture (VGG or WideResNet), we take one model trained with MaxPooling, one with AveragePooling, and one or more with Avg-TopK (for different *k*). To form the ensemble, we compute the element wise average of their predicted class probability vectors (softmax outputs). The final predicted class is the one with the highest average probability. This simple averaging ensemble often yields a more robust prediction because the different pooling strategies capture complementary features. We compare the ensemble accuracy against each constituent model's accuracy.

As a preliminary summary of our findings (to be detailed later): the optimal Avg-TopK pooling (specific *k* and window) improves over pure max or avg pooling. The ensemble typically achieves the highest accuracy. On CIFAR-10, our VGG-style ensemble achieved around 88.0%–89.3% accuracy, while the WideResNet-mini ensemble reached about 90.0%–91.9%. On CIFAR-100, the WideResNet ensemble scored ~67%–69.8%, compared to only 36%–38% for the best single VGG Avg-TopK

model. These results indicate that pooling choice and ensembling substantially impact performance.

5. Methodology

This section details the experimental setup, including data preprocessing, model architectures, training procedures, and evaluation metrics.

Data and Preprocessing: We conduct experiments on the CIFAR-10 and CIFAR-100 datasets .

Each dataset has 50,000 training images and 10,000 test images. We normalize pixel values to have zero mean and unit variance per channel. During training, we apply standard data augmentation: random horizontal flips, random crops (with padding), and possibly random brightness or contrast adjustments. These augmentations help prevent overfitting and improve generalization. No test-time augmentation is used beyond mean subtraction.

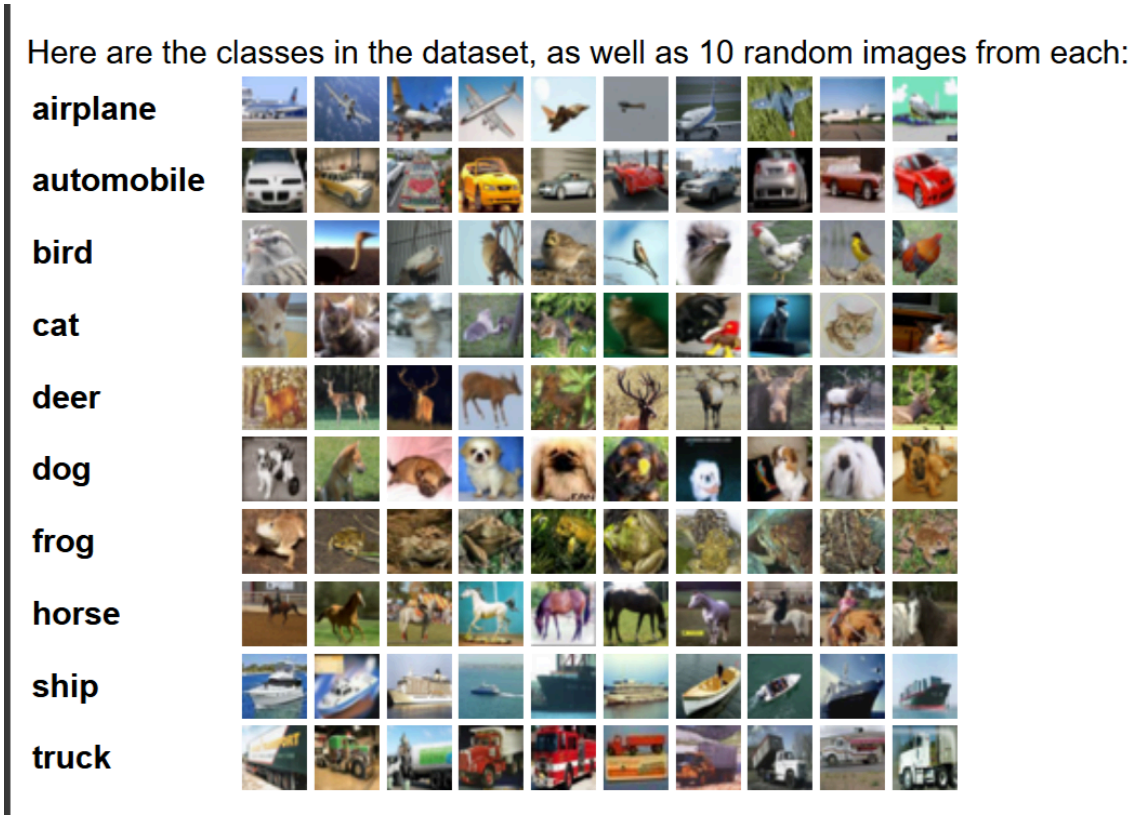

Figure 3: CIFAR-10 Dataset

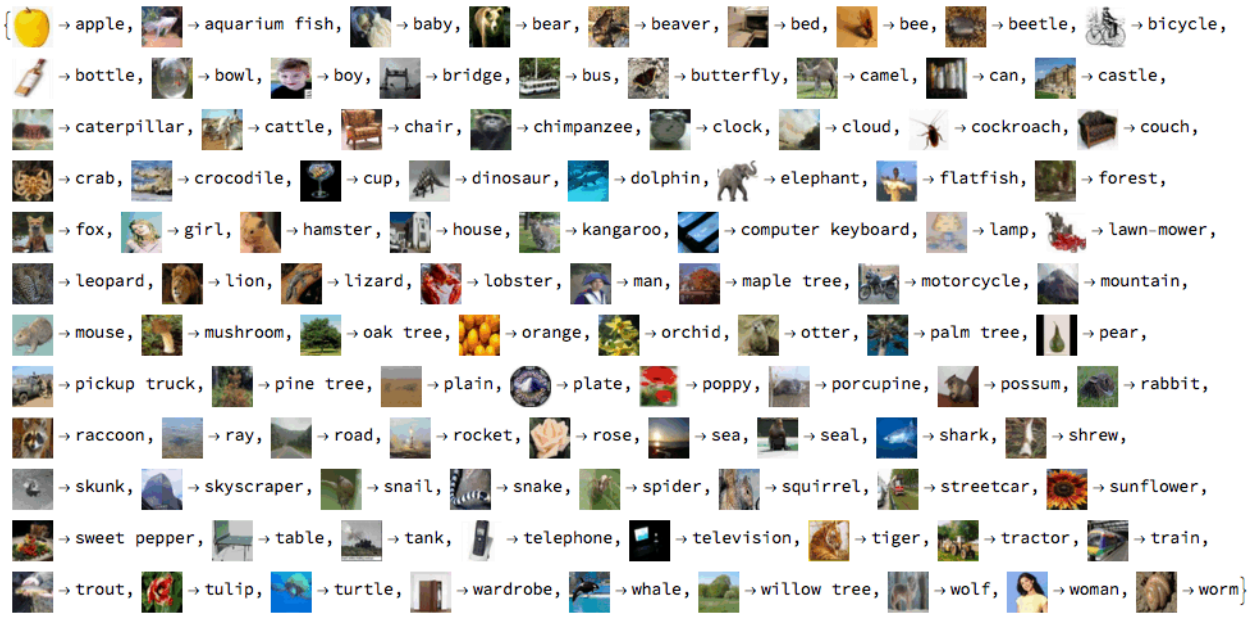

Figure 4: CIFAR-100 Dataset

Network Architectures: We implement the two CNN architectures as described:

- *VGG-style CNN:* We use a stack of convolutional layers with 3×3 filters. For example, the architecture may consist of conv blocks like [Conv(32) – Conv(64) Conv(128)], with pooling layers in between, analogous to the original VGG net structure . After each Conv layer, we apply Batch Normalization and a ReLU activation. In each pooling block, we insert the pooling layer under investigation (Max, Avg, or Avg-TopK). We insert Dropout in between every convolution layers[15]. The final layer is a fully-connected layer with 10 or 100 outputs, followed by softmax.

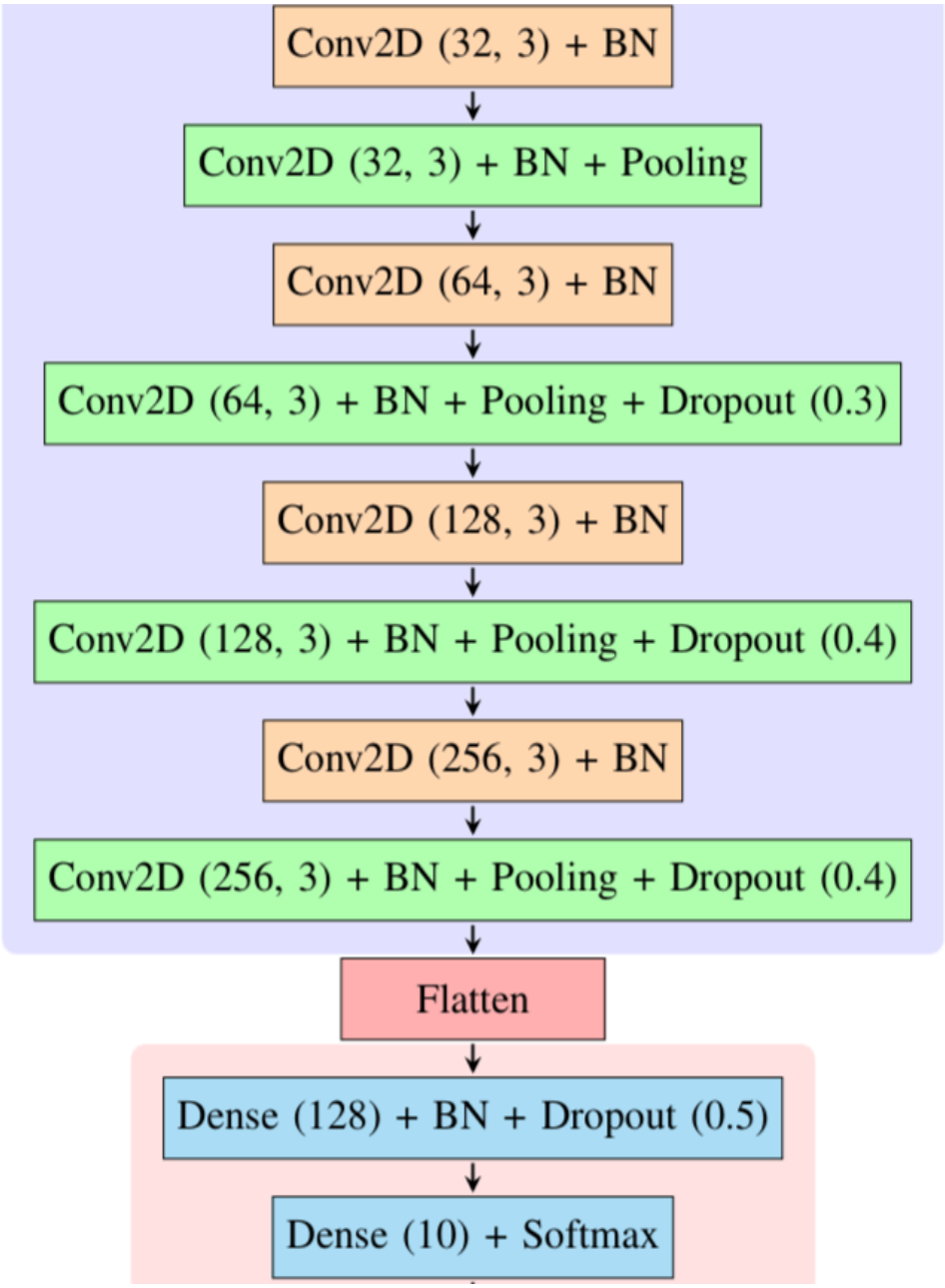

Figure 5: VGG Style Architecture

- *WideResNet-mini:* The network is based on wide residual blocks . Concretely, we use three groups of residual blocks, each group doubling the number of channels. Within each block, we have two 3×3 convolutions (with BN+ReLU). The "width" (number of channels) in the first group is set (e.g., 16 filters) and increased in subsequent groups. The total depth is relatively shallow (e.g., around 16 layers total) since we are not building a full-scale WideResNet. A 1×1 convolution is used in the skip connection when changing channel dimensions.

A pooling layer (Max/Avg/Avg-TopK) is applied after each group to downsample (we experiment with various window sizes). A global average pooling is applied at the end of the feature extractor to flatten for the classifier. Finally, a 10- or 100-way softmax layer produces class probabilities .

- Pooling Variations: For each model, we run separate training with three pooling configurations:

- *MaxPooling:* After each convolutional block, we apply max pooling. We experiment with pooling window sizes of 2×2, 3×3, and 4×4. The stride is set equal to the window size (non-overlapping pooling).

- *AveragePooling:* We similarly experiment with average pooling layers of the same window sizes (2×2, 3×3, 4×4).

Avg-TopK Pooling: For this, we insert a pooling layer that takes the top *k* values in each window and averages them . We test different values of *k* (e.g., 2, 3, 4) for each window size. For example, one configuration could be AvgTopK with window 3×3 and k=3. When *k* equals 1, this effectively becomes max pooling, and when *k* equals the total number of elements, it is average pooling. Values in between provide the hybrid effect. This pooling is implemented efficiently by sorting or partial selection of the local values.

During training, all other hyperparameters are kept the same across pooling types to ensure fair comparison. We train each network for a fixed number of epochs (e.g., 30, 75, 50) or until convergence, using a categorical cross-entropy loss and an optimizer such as Adam. The learning rate is scheduled (e.g., decayed by a factor at milestones). We monitor validation accuracy to select the best model if early stopping is used.

Ensemble Method: To build an ensemble, we take the trained models with

different pooling layers but the same base architecture. For example, we train a VGG-style network with MaxPooling, another with AveragePooling, and one with Avg-TopK (for a chosen *k*). After training, each model produces a probability vector over classes for a given input. The ensemble prediction is obtained by averaging these vectors element wise (i.e., arithmetic mean of probabilities) and then selecting the class with maximum average probability. This approach effectively combines the learned features of each model. We evaluate ensemble accuracy on the test set and compare it with individual model accuracies. As noted in prior work, such ensembling often yields higher accuracy than any single model.

Evaluation Metrics: We report the classification accuracy (percentage of correctly classified test images) for each model and ensemble. We also analyze training curves to ensure convergence and compute confusion matrices for deeper analysis (to be presented later). Statistical significance of differences can be assessed if needed.

By systematically varying the pooling type and forming ensembles, our methodology thoroughly explores the effect of pooling on CNN performance.

6. Results and Discussion

This section presents a comparative analysis of various pooling strategies evaluated on CIFAR-10 and CIFAR-100 datasets using two convolutional neural network architectures: a VGG-like architecture and WideResNet. The goal was to assess the impact of different pooling mechanisms—including traditional MaxPooling and AvgPooling, as well as the novel AvgTopK pooling—across multiple pool sizes. Key performance metrics such as accuracy, precision, recall, and training time were analyzed.

6.1. Results on VGG-like Architecture

6.1.1. CIFAR-10 Dataset

Experiments on the CIFAR-10 dataset were conducted with pool sizes of 2 and 3.

- Pool Size = 2: Among the pooling methods, AvgTopK3 achieved the highest test

accuracy of 85.61%, followed by MaxPooling at 84.59% and AvgTopK4 at 82.66%. Notably, AvgPooling trailed with 81.32%. However, the training time for AvgTopK methods was significantly higher, with AvgTopK3 taking approximately 430.78 seconds compared to 276.76 seconds for MaxPooling. An ensemble model integrating all methods achieved the best overall test accuracy of 88.76%, indicating the benefit of combining diverse pooling strategies.

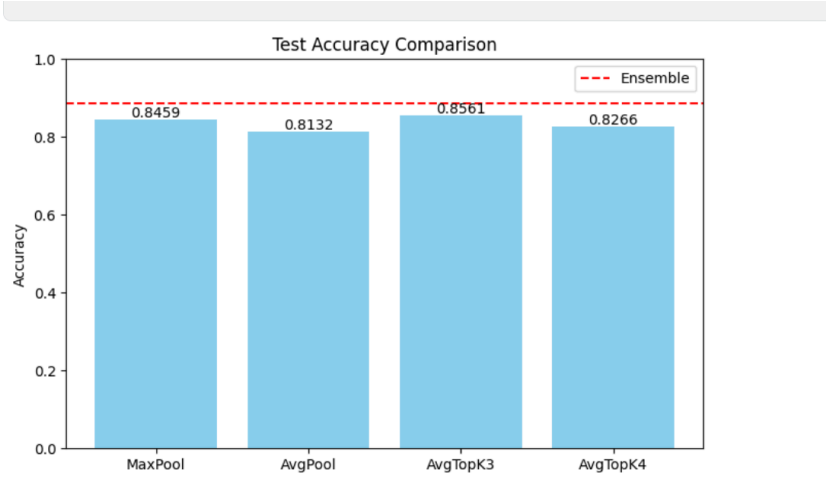

Figure 6: Accuracy of overall Model(CIFAR-10, Pool size=2,Epochs=75)

- Pool Size = 3: A more granular evaluation was performed with precision, recall, and F1-score included. AvgPooling yielded the highest accuracy at 80.64%, followed by AvgTopK2 (80.02%) and AvgTopK3 (79.09%). Interestingly, MaxPooling underperformed with 78.28%, suggesting that traditional max pooling may not be optimal with larger pool sizes in VGG-like architectures.

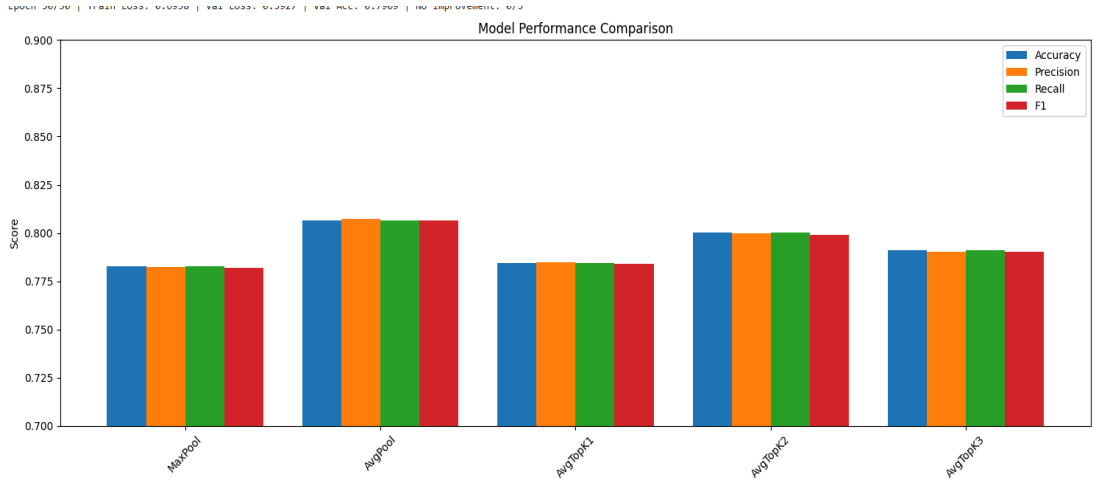

Figure 7: Accuracy of overall Model(CIFAR-10, Pool size=3,Epochs=50)

6.1.2. CIFAR-100 Dataset

- Pool Size = 2: The highest accuracy was achieved using AvgPooling (38.04%), outperforming all AvgTopK configurations and MaxPooling. Among AvgTopK variants, AvgTopK4 and AvgTopK1 performed similarly, achieving around 36.2% accuracy. This trend highlights that on more complex datasets, AvgPooling might capture generalized patterns better in VGG-style networks.

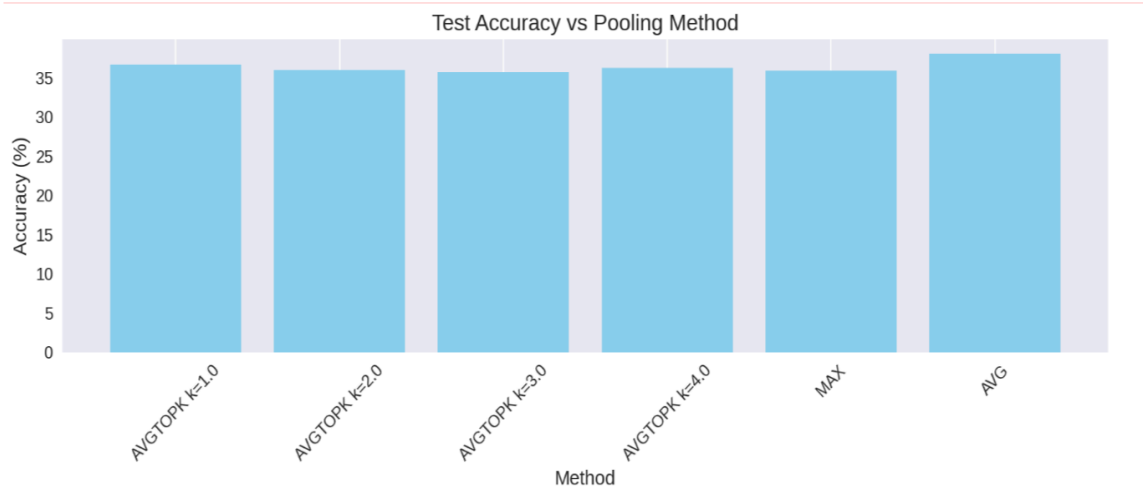

Figure 8: Evaluation metrics of overall Model(CIFAR-100, Pool size=2)

- Pool Sizes = 3 and 4: Results show that AvgTopK2 performed consistently well across both pool sizes, achieving accuracies of 42.28% (pool size 3) and 42.00% (pool size 4), closely matching or outperforming MaxPooling. AvgPooling consistently showed lower accuracy, indicating its limitations on CIFAR-100 in deeper feature maps.

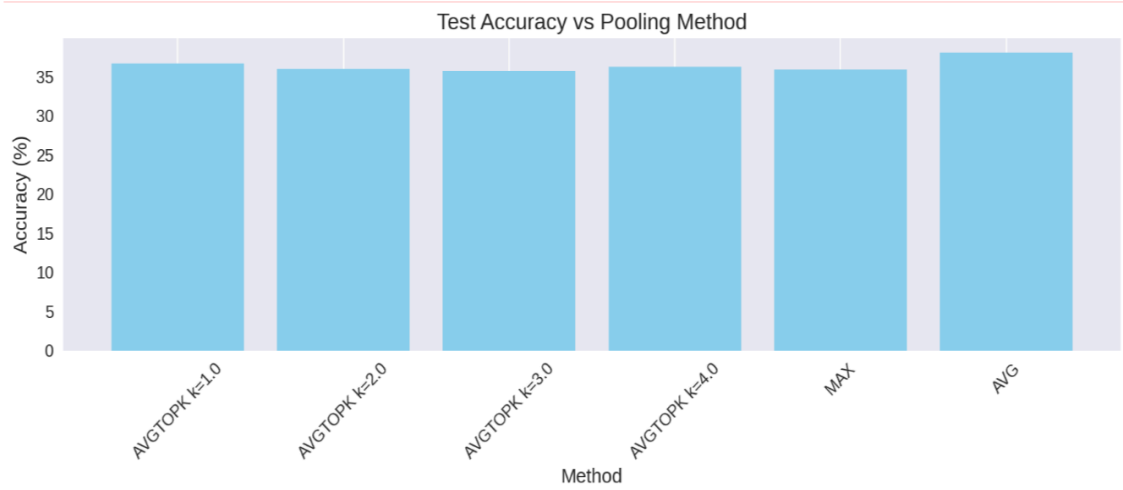

Figure 9: Accuracy of overall Model(CIFAR-100, Pool size=3)

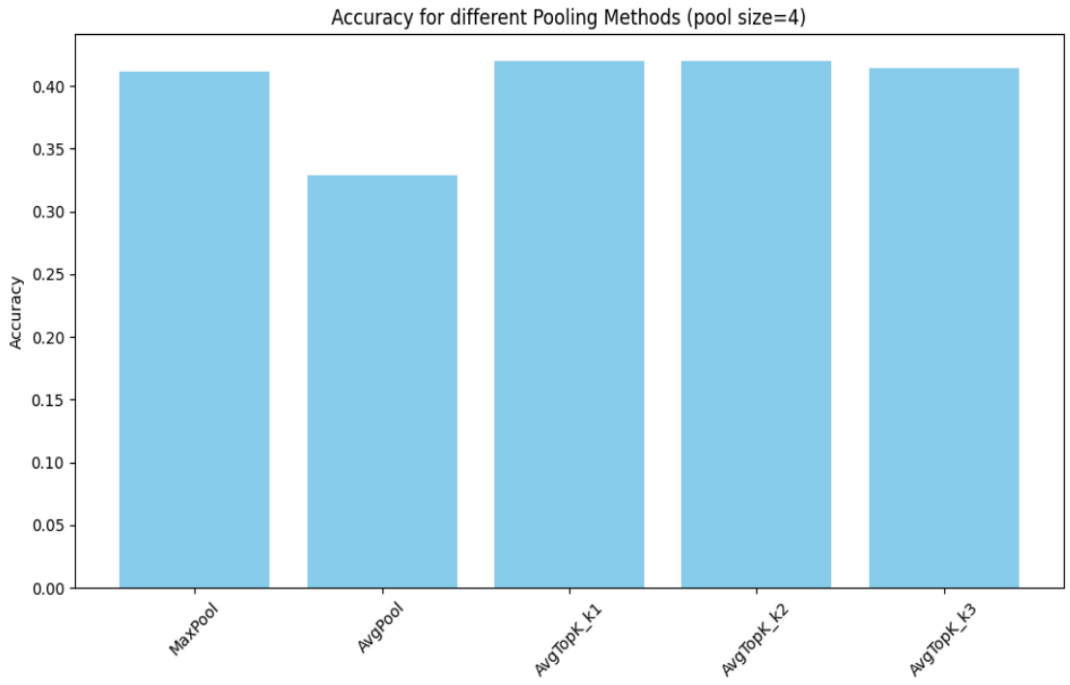

Figure 10: Accuracy of overall Model(CIFAR-10, Pool size=4,Epochs=30)

6.2. Results on WideResNet Architecture

Given the limitations in performance observed with the VGG-like architecture—particularly on CIFAR-100—the WideResNet architecture was adopted to

explore improved representational power.

6.2.1. CIFAR-10 Dataset

- Pool Size = 2: The highest accuracy was achieved by the ensemble method (91.87%), followed by MaxPooling (87.79%) and AvgTopK_2 (87.78%). AvgPooling showed a modest performance at 85.63%. These results underscore the competitive performance of AvgTopK methods compared to traditional pooling.

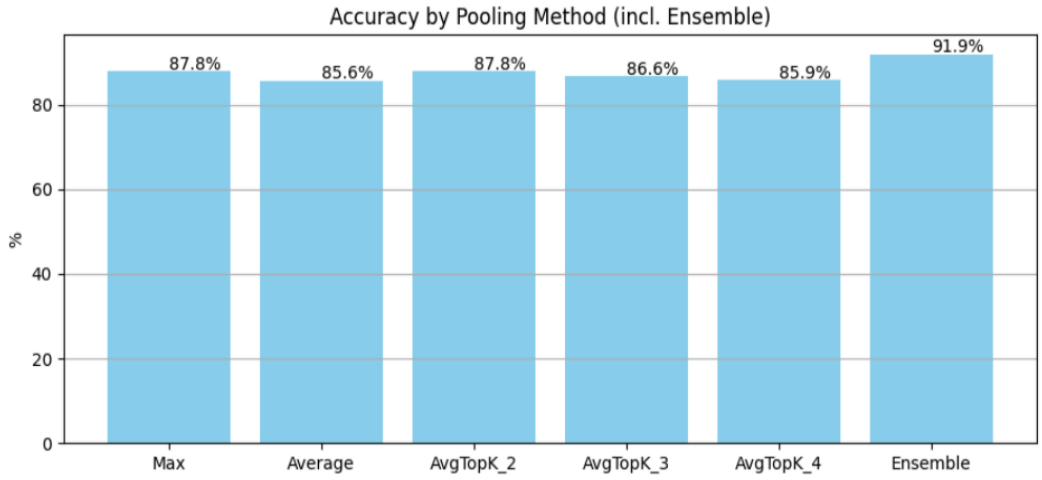

Figure 11: Accuracy of overall Model(CIFAR-10, Pool size=2)

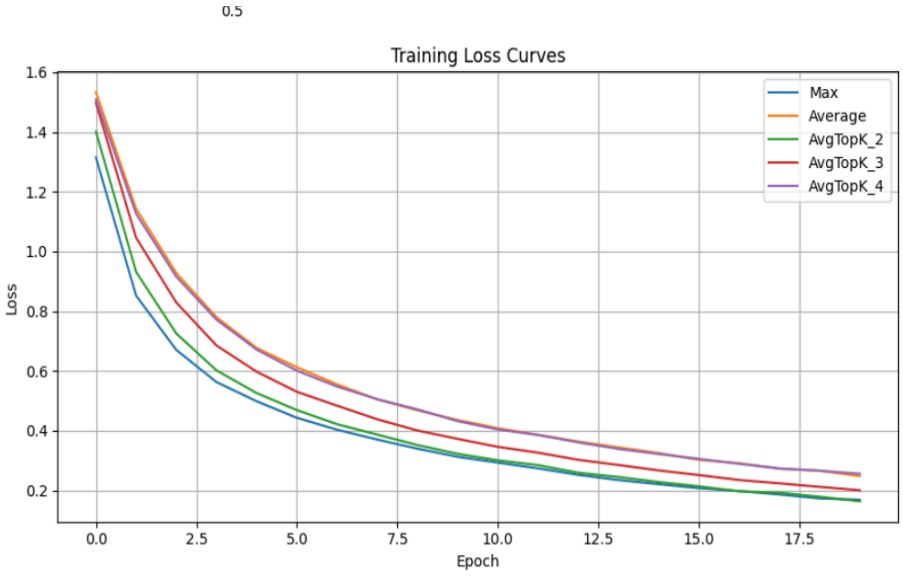

Figure 12: Training loss curve of overall Model(CIFAR-10, Pool size=3)

- Pool Size = 3: AvgTopK_3 attained the highest standalone accuracy (87.07%), exceeding MaxPooling (86.39%) and all other Top-K configurations. The ensemble method again surpassed all with an accuracy of 90.89%, validating the robustness of pooling ensemble approaches.

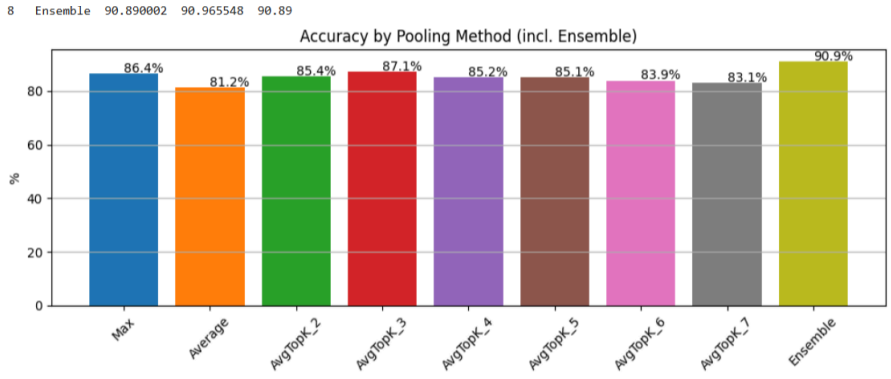

Figure 13: Accuracy of overall Model(CIFAR-10, Pool size=3,Epochs=30)

6.2.2. CIFAR-100 Dataset

- Pool Size = 2: A notable accuracy improvement was observed with the ensemble model reaching 69.80%, outperforming all individual pooling strategies. Among the standalone methods, MaxPooling achieved 60.79%, closely followed by AvgTopK_2 (59.98%), demonstrating the effectiveness of Top-K pooling in capturing critical features.

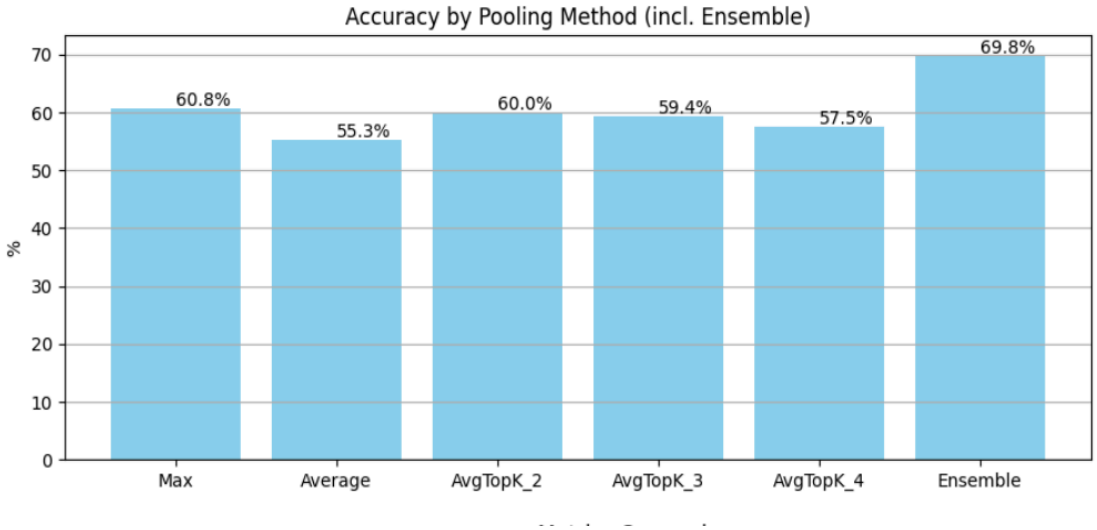

Figure 14: Accuracy of overall Model(CIFAR-100, Pool size=2)

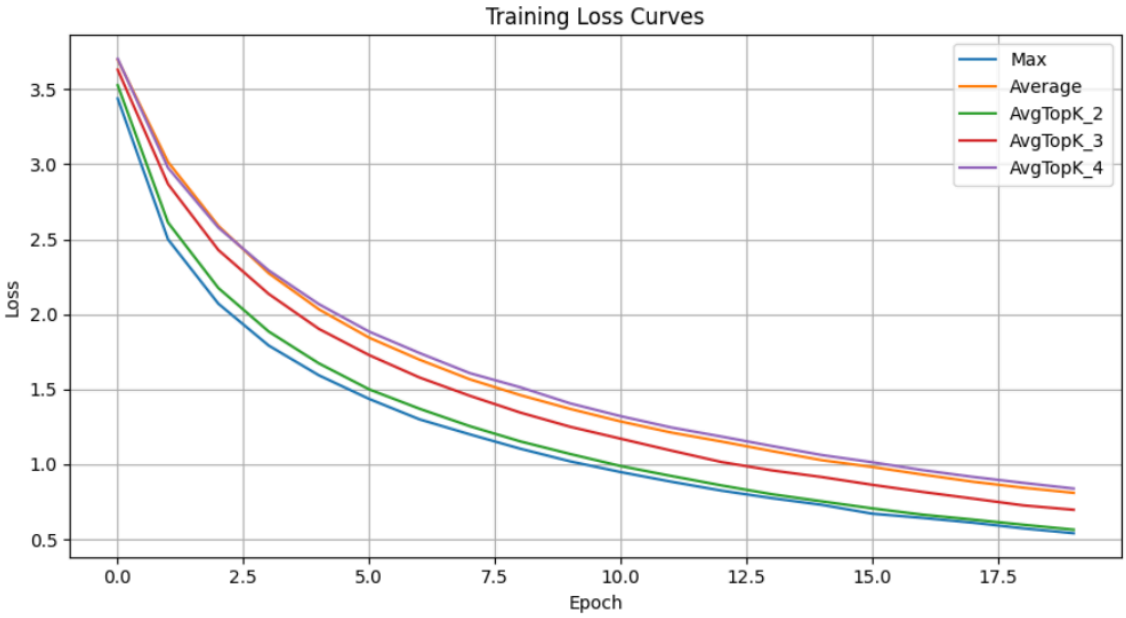

Figure 15: Training loss curve of overall Model(CIFAR-100, Pool size=2)

- Pool Size = 3: AvgTopK_2 again led among individual models with 59.30% accuracy, followed by AvgTopK_4 (58.9%). MaxPooling reached 58.5%, while AvgPooling lagged significantly at 45.15%. The ensemble yielded an impressive 68.1%, indicating enhanced feature diversity through model fusion.

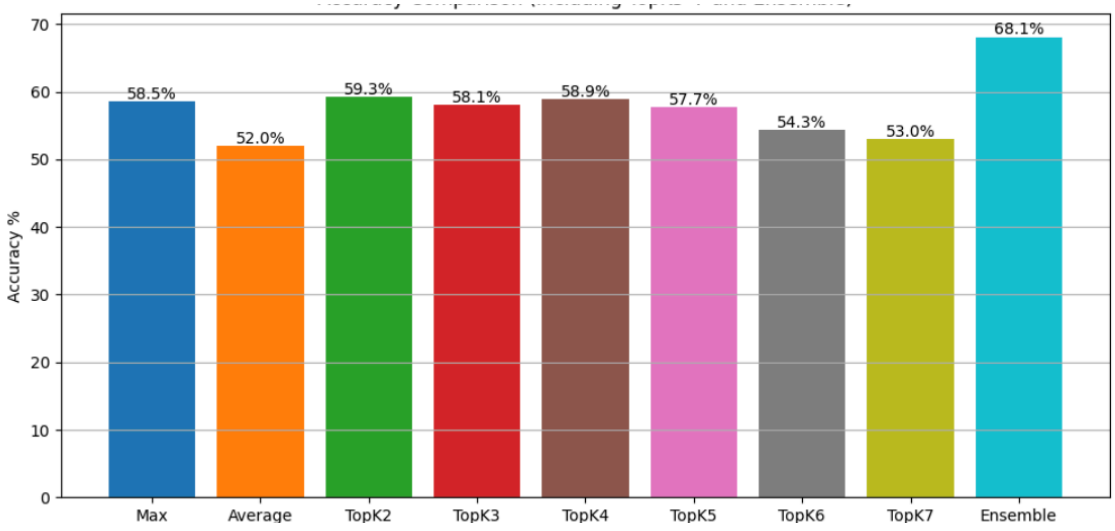

Figure 16: Accuracy of overall Model(CIFAR-100, Pool size=3,Epochs=30)

- Pool Size = 4: Results continued to favor Top-K pooling, with AvgTopK_2 achieving the best standalone accuracy of 58.1%. MaxPooling and AvgPooling posted 56.4% and 49.3%, respectively. The ensemble model again performed

best, with an accuracy of 67.5%, confirming the generalizability of pooling ensembles across wider and deeper architectures.

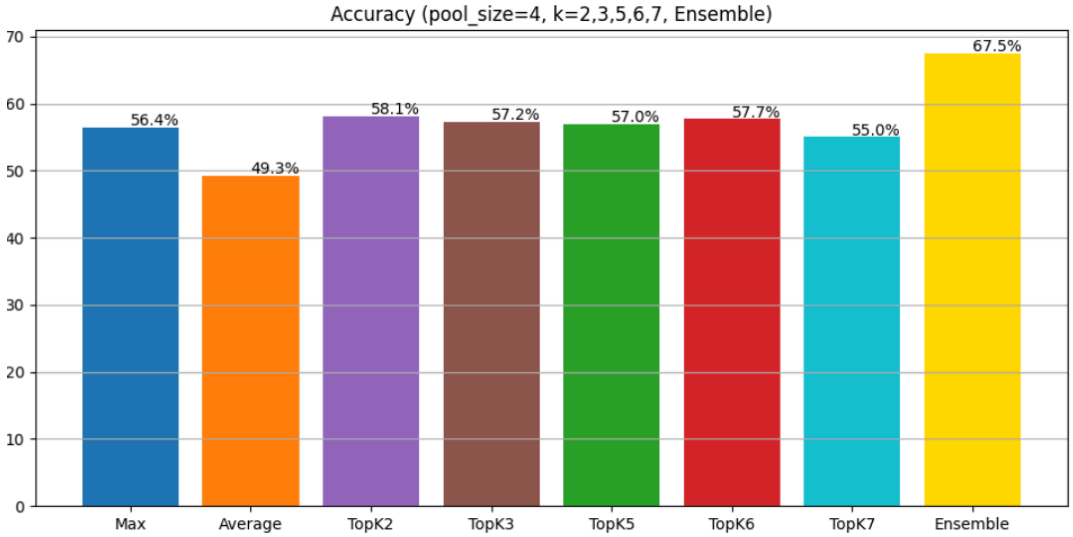

Figure 16: Accuracy of overall Model(CIFAR-100, Pool size=4,Epochs=30)

Best Results of the two models

Dataset	Architecture	Pool Size	Best Individual Accuracy	Best Ensemble Accuracy
CIFAR-10	VGG-like	2	85.61% (AvgTopK3)	88.76%
CIFAR-10	WideResNet	2	87.79% (MaxPool)	91.87%
CIFAR-100	VGG-like	2	38.04% (AvgPool)	41.08%
CIFAR-100	WideResNet	2	60.79% (MaxPool)	69.80%
CIFAR-10	WideResNet	3	87.07% (AvgTopK3)	90.89%
CIFAR-100	WideResNet	3	59.30% (AvgTopK2)	68.12%
CIFAR-100	WideResNet	4	58.46% (AvgTopK2)	69.53%

Result of Wide ResNet Architecture on CIFAR-100

Pool Size	Pooling Method	Accuracy (%)	Precision (%)	Recall (%)
2	MaxPooling	60.79	62.90	60.79
	AvgPooling	55.29	59.14	55.29
	AvgTopK_2	59.98	63.30	59.98

	AvgTopK_3	59.38	61.02	59.38
	AvgTopK_4	57.53	59.95	57.53
	Ensemble	**69.80**	**70.87**	**69.80**
3	MaxPooling	58.53	60.002	58.54
	AvgPooling	51.98	55.60	51.98
	AvgTopK_2	59.28	60.93	59.28
	AvgTopK_3	58.06	61.18	58.07
	AvgTopK_4	58.93	60.73	58.94
	AvgTopK_5	57.67	60.46	57.68
	AvgTopK_6	54.33	57.62	54.34
	AvgTopK_7	53.04	56.67	53.05
	Ensemble	**68.12**	**69.85**	**68.13**
4	MaxPooling	56.85	57.32	56.85
	AvgPooling	53.36	54.88	53.36
	AvgTopK_2	58.46	60.07	58.46
	AvgTopK_3	58.32	59.45	58.32
	AvgTopK_4	57.60	59.36	57.60
	AvgTopK_5	56.39	57.71	56.39
	AvgTopK_6	57.38	59.18	57.38
	AvgTopK_7	56.44	58.68	56.44
	Ensemble	**69.53**	**69.84**	**69.53**

Result of VGG style Architecture on CIFAR-100

Pool Size	Pooling Method	Accuracy (%)	Precision (%)	Recall (%)
2	AvgTopK_1	36.68	37.54	36.68
	AvgTopK_2	36.01	37.28	36.01
	AvgTopK_3	35.73	37.28	35.73
	AvgTopK_4	36.22	37.71	36.22
	MaxPooling	35.88	36.97	35.88
	AvgPooling	**38.04**	**38.96**	**38.04**
3	MaxPooling	42.06	43.57	42.06
	AvgPooling	36.75	36.81	36.75
	AvgTopK_1	41.67	43.03	41.67
	AvgTopK_2	**42.28**	**42.81**	**42.28**
	AvgTopK_3	42.04	43.00	42.04
4	MaxPooling	41.16	42.34	41.16
	AvgPooling	32.88	33.52	32.88
	AvgTopK_1	**42.02**	**43.25**	**42.02**
	AvgTopK_2	42.00	42.71	42.00
	AvgTopK_3	41.40	43.23	41.40

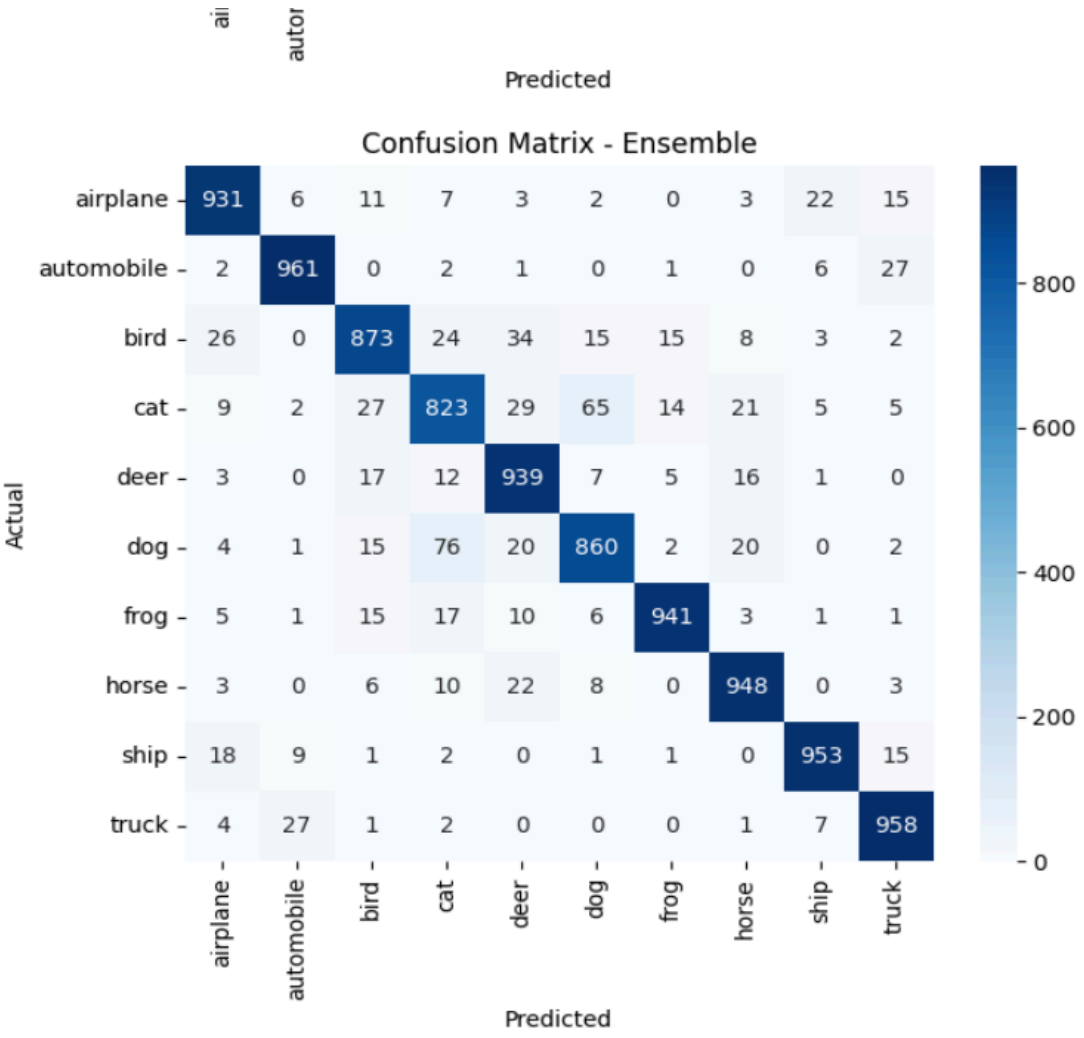

Figure 17: Confusion Matrix of the best Model (CIFAR-10)(WideResNet)

6.3. Discussion

The experimental results across VGG-like and WideResNet architectures provide several key insights:

1. AvgTopK Pooling as a Viable Alternative: AvgTopK pooling, particularly with lower K values (e.g., 2 or 3), consistently matched or exceeded the performance of MaxPooling across datasets and architectures. This suggests that aggregating top activations rather than just the maximum leads to better generalization.

2. Impact of Architecture: WideResNet, due to its residual connections and wider

filters, benefited more significantly from advanced pooling methods than the VGG-like network. This is especially evident in CIFAR-100, where accuracy improvements were substantial.

3. Role of Pool Size: Larger pool sizes generally led to diminished performance for traditional pooling methods, especially AvgPooling, likely due to loss of spatial granularity. However, AvgTopK showed resilience, possibly due to its focus on salient activations rather than global averaging.

4. Effectiveness of Ensemble Learning: Across nearly all configurations, ensemble models that combined predictions from multiple pooling strategies outperformed individual models. This highlights the complementary strengths of different pooling operations and motivates further research into adaptive or learnable

7.Conclusion

In this work, we have explored how pooling strategies influence CNN performance on image classification tasks, and demonstrated the benefit of ensembling models with different pooling layers. Using two CNN architectures (VGG-style and WideResNet-mini), we evaluated standard MaxPooling and AveragePooling against the novel Avg-TopK method. Our experiments on CIFAR-10 and CIFAR-100 showed that Avg-TopK pooling (with appropriately chosen *k*) can outperform conventional pooling by retaining more feature information. Furthermore, an ensemble that averages predictions from Max-, Avg-, and TopK-pooled models yielded the highest accuracy overall. Empirically, the WideResNet-mini ensemble achieved the best results (around 91.87% on CIFAR-10 and ~69.8% on CIFAR-100), illustrating that pooling diversity and ensembling are effective strategies. These findings align with recent literature that emphasizes innovative pooling layers and ensemble methods as ways to boost CNN accuracy. In summary, our comprehensive evaluation confirms that choosing the right pooling mechanism and combining multiple models can lead to significant gains in CNN-based image classification.

8. Future Scope

Architecture Variations: Extend the study to other CNN architectures (e.g. ResNeXt, DenseNet, EfficientNet) to see if pooling effects generalize.

Adaptive Pooling: Explore learnable or data-driven pooling (e.g. dynamic top-*k*, spatial attention pooling) where *k* or the pooling behavior is adjusted during training.

Larger Datasets: Test the pooling and ensemble approach on larger and more diverse datasets (e.g. ImageNet, SVHN, or domain-specific collections) to assess scalability.

Ensemble Strategies: Investigate different ensemble fusion methods, such as weighted averaging, stacking, or ensembling with models trained on different augmentations.

Regularization Techniques: Combine pooling experiments with advanced regularizers or normalization (e.g. Mixup, LayerNorm) to see if interactions further improve results.

Real-time Constraints: Evaluate the computational trade-offs, examining whether the ensemble's accuracy benefits justify any increase in inference cost, and explore model distillation to compress the ensemble.

Pooling Theory: Study theoretically how different pooling operations affect feature invariances and

gradients, possibly guiding more principled design of pooling layers.

9.References

[1] Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images*. (Technical report, University of Toronto).

[2] Simonyan, K., & Zisserman, A. (2015). *Very deep convolutional networks for large-scale image recognition*. arXiv:1409.1556.

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. In *CVPR* (pp. 770–778).

[4] Zagoruyko, S., & Komodakis, N. (2016). *Wide residual networks*. In *BMVC*.

Ioffe, S., & Szegedy, C. (2015). *Batch normalization: Accelerating deep*

network training by reducing internal covariate shift. In *ICML* (pp. 448–456).

[5] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). *Dropout: A simple way to prevent neural networks from overfitting*. JMLR, 15(56), 1929–1958

[6] Özdemir, C. (2023). *AVG-TopK: A new pooling method for convolutional neural networks*. Expert Systems with Applications, 218, 119892.

[7] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*. Proc. IEEE, 86(11), 2278–2324.
Zeiler, M. D., & Fergus, R. (2013). *Stochastic pooling for regularization of deep convolutional neural networks*. In *ICML* (pp. 109–116).

[8] Yamashita, R., Nishio, M., & Do, R. K. (2018). *Convolutional neural networks: an overview and application in radiology*. Insights into Imaging, 9, 611–629

[9] Very Deep Convolutional Networks for Large-Scale Image Recognition https://arxiv.org/abs/1409.1556

[10] Deep Residual Learning for Image Recognition https://arxiv.org/abs/1512.03385

[11] A new pooling method for convolutional neural networks

[12] A improved pooling method for convolutional neural networks | Scientific Reports https://colab.ws/articles/10.1016%2Fj.eswa.2023.119892

[13] A Comprehensive Exploration of Pooling in Neural Networks: A Python demo to Pooling in CNN

[14] Stochastic Pooling for Regularization of Deep Convolutional Neural Networks https://arxiv.org/abs/1301.3557

[15] Dropout: A Simple Way to Prevent Neural Networks from Overfitting https://jmlr.org/papers/v15/srivastava14a.html

[16] Convolutional neural networks: an overview and application in radiology | Insights into Imaging https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

[17] A Comprehensive Exploration of Pooling in Neural Networks: A Python demo to Pooling in CNN https://blog.paperspace.com/a-comprehensive-exploration-of-pooling-in-neural-networks/