

JarifNet: An Attention-Augmented Lightweight CNN for Highly Calibrated Edge Image Classification

Sadik Al Jarif

sadikaljarif05@gmail.com

Independent Researcher

Research Article

Keywords: Convolutional Neural Network, Lightweight Deep Learning, Squeeze-and-Excitation Attention, Inverted Residuals, Image Classification, CIFAR-10

Posted Date: August 26th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10616279/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

JarifNet: An Attention-Augmented Lightweight CNN for Highly Calibrated Edge Image Classification

Sadik Al Jarif^{1*}

^{1*}Independent Researcher, Dhaka, Bangladesh.

Corresponding author(s). E-mail(s): sadikaljarif05@gmail.com;

Abstract

Deploying deep convolutional neural networks (CNNs) on resource-constrained edge devices requires balancing predictive accuracy, computational efficiency, and output probability calibration. In this paper, we introduce JarifNet, a novel, compact architecture designed for high-efficiency image classification. JarifNet utilizes an inverted residual bottleneck skeleton augmented with lightweight Squeeze-and-Excitation (SE) channel attention modules and Stochastic Depth regularization to maximize feature selectivity and mitigate overfitting. We evaluate JarifNet on the CIFAR-10 dataset alongside five baseline architectures (MobileNetV1, MobileNetV2, ShuffleNetV2, VGG16, and ResNet20) using an identical training protocol. Our empirical results demonstrate that JarifNet achieves a highly competitive test accuracy of 92.27%. Crucially, JarifNet establishes a new state-of-the-art in class discrimination and probability calibration among the tested models, achieving a Macro Average Precision (AP) of 0.9743 and a Micro AP of 0.9776, outperforming the best baseline, MobileNetV2. Furthermore, profiling experiments show that JarifNet operates with an inference latency of 8.11 ms per image on an NVIDIA Tesla T4 GPU and 13.52 ms on a CPU, maintaining a compact footprint of approximately 4.50 million parameters. These findings confirm that combining dynamic channel attention with stochastic regularization yields superior confidence calibration and robust fine-grained feature extraction without compromising computational efficiency.

Keywords: Convolutional Neural Network, Lightweight Deep Learning, Squeeze-and-Excitation Attention, Inverted Residuals, Image Classification, CIFAR-10

1 Introduction

Deep Convolutional Neural Networks (CNNs) have catalyzed transformative breakthroughs across computer vision domains, including autonomous navigation, industrial inspection, and mobile healthcare diagnostics. However, the continuous scaling of network depth and parameter count often comes at the cost of prohibitively high computational complexity (FLOPs) and memory footprint. Consequently, adapting modern vision models for deployment on low-power edge hardware remains a critical challenge.

Recent architectural innovations have significantly reduced computational overhead. For instance, MobileNetV1 [1] replaced standard convolutions with depthwise separable convolutions, while MobileNetV2 [2] introduced inverted residual bottlenecks. Similarly, ShuffleNetV2 [3] established key guidelines for efficient hardware execution by optimizing memory access costs. Despite these advances, lightweight networks frequently suffer from performance degradation due to reduced expressive capacity. This often leads to severe misclassification in fine-grained visual categories (e.g., discriminating between visually similar animal classes such as cats and dogs). Furthermore, standard lightweight backbones often produce poorly calibrated confidence probabilities, rendering them brittle in safety-critical edge environments where rejection thresholds and reliable uncertainty estimates are paramount.

To resolve these limitations, we propose **JarifNet**, an attention-augmented, lightweight convolutional neural network specifically engineered to maximize feature selectivity and probability calibration under strict parameter budgets. JarifNet combines the compute efficiency of inverted residual bottlenecks with channel-wise dynamic recalibration via Squeeze-and-Excitation (SE) units [4] and stochastic depth regularization (*DropPath*) [5].

In this paper, we present a comprehensive empirical evaluation of JarifNet against five established baseline architectures (MobileNetV1, MobileNetV2, ShuffleNetV2, VGG16, and ResNet20) on the CIFAR-10 benchmark. Our primary contributions are as follows:

- **Novel Architecture Design:** We introduce the *JarifBlock*, an enhanced inverted residual bottleneck that incorporates a Squeeze-and-Excitation mechanism (reduction ratio $r = 4$) and Stochastic Depth to maximize feature reuse while combating gradient instability and overfitting on small-scale datasets.
- **Superior Probability Calibration:** Empirical results demonstrate that JarifNet achieves the highest Macro Average Precision (0.9743) and Micro Average Precision (0.9776) among all evaluated models, surpassing highly optimized architectures like MobileNetV2 and VGG16. This indicates exceptionally well-calibrated softmax probabilities.
- **Comprehensive Comparative Study:** We conduct a unified benchmark comparing parameter counts, classification metrics, per-class error dynamics, and real-world hardware latency across both GPU and CPU environments, proving JarifNet’s viability as a highly efficient edge classifier.

2 Related Work

2.1 Efficient Convolutional Architectures for Edge Computing

The proliferation of edge computing and mobile vision applications has necessitated the development of highly optimized Convolutional Neural Networks (CNNs) capable of operating under strict computational and memory constraints. A paradigm shift in this domain was marked by MobileNetV1 [1], which replaced computationally expensive standard spatial convolutions with depthwise separable convolutions. This factorization spatially filters the input independently per channel before linearly combining the features, drastically reducing the required multiply-accumulate operations (MACs). MobileNetV2 [2] significantly advanced this foundation by introducing the inverted residual block with linear bottlenecks, a structural design that expands the channel space to perform non-linear transformations before projecting back to a lower-dimensional linear manifold. This preserves the representational power of the network while ensuring efficient gradient propagation. Further refining hardware efficiency, ShuffleNetV2 [3] established that indirect metrics, such as memory access cost (MAC) and degree of fragmentation, are as critical as raw FLOP counts. However, while these architectures excel in computational efficiency, their aggressive parameter reduction often sacrifices the representational capacity required for fine-grained semantic discrimination. Subsequent work has explored complementary directions: EfficientNet [6] introduced compound scaling across depth, width, and resolution; MobileNetV3 [7] combined neural architecture search with SE modules; GhostNet [8] generated redundant feature maps via cheap linear operations; MnasNet [9] applied platform-aware search directly optimizing for latency; and more recently MobileViT [10] incorporated lightweight transformer blocks, building on the Vision Transformer paradigm [11], into mobile-scale networks. JarifNet is specifically designed to bridge this gap, maintaining a lightweight footprint while recovering feature granularity through attention rather than search-based or transformer-based mechanisms.

2.2 Dynamic Channel Attention and Feature Recalibration

Enhancing the representational efficacy of CNNs without disproportionately increasing model depth or width has been largely driven by the adoption of attention mechanisms. Squeeze-and-Excitation (SE) networks [4] introduced a highly effective mechanism for dynamic feature recalibration. By aggregating spatial information through global average pooling (the "squeeze" phase) and applying a fully connected bottleneck to generate channel-wise weights (the "excitation" phase), SE modules explicitly model interdependencies between channels. This allows the network to adaptively emphasize informative features while suppressing irrelevant background noise. Integrating such attention mechanisms into lightweight mobile architectures, however, presents a non-trivial challenge, as the fully connected layers can introduce latency bottlenecks. JarifNet addresses this by strategically embedding lightweight SE modules, utilizing a carefully tuned reduction ratio of $r = 4$, directly into the inverted residual bottlenecks, optimizing the trade-off between semantic enhancement and computational

overhead. Alternative attention formulations, such as CBAM [12], extend recalibration to the spatial dimension in addition to channels, while ECA-Net [13] eliminates the dimensionality-reduction bottleneck entirely via a lightweight 1D convolution. We adopt the SE formulation for its favorable balance of simplicity and empirical performance on small-resolution inputs.

2.3 Structural Regularization in Deep Networks

As CNNs scale in depth, they become increasingly susceptible to issues such as vanishing gradients, feature co-adaptation, and severe overfitting—particularly when trained from scratch on datasets with limited spatial resolutions (e.g., the 32×32 images of CIFAR-10). To mitigate the degradation problem in deep residual networks, Huang et al. [5] proposed Stochastic Depth (commonly referred to as *DropPath*). This regularization technique dynamically bypasses entire residual blocks during training with a specified survival probability. Consequently, the network trains as an ensemble of shallower sub-networks, which significantly reduces co-adaptation among layers and acts as a strong regularizer. During inference, the full network topology is utilized, scaled by the survival probabilities. JarifNet synergistically combines DropPath regularization with SE-augmented inverted residuals, ensuring stable convergence and robust generalization without necessitating pre-trained weights.

3 Proposed Methodology: JarifNet

3.1 Architectural Philosophy and Overview

The design of JarifNet centers on three core principles: minimizing computational FLOPs via depthwise feature separation, enabling dynamic channel recalibration via global attention, and applying strong stochastic regularization to prevent early convergence plateaus on small-scale input resolutions (32×32).

Figure 1 illustrates the complete structural layout of JarifNet, detailing the primary network pipeline, the internal design of the proposed *JarifBlock*, and the Squeeze-and-Excitation (SE) attention module.

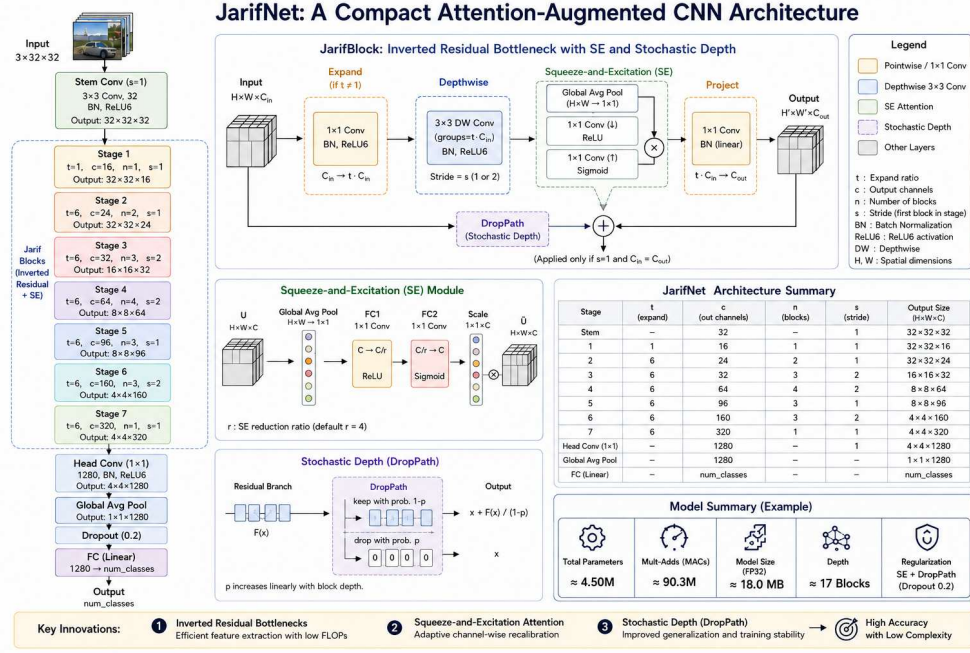


Fig. 1: Complete structural layout of the proposed **JarifNet** architecture. The diagram highlights the *JarifBlock* (Inverted Residual Bottleneck with SE Attention and Stochastic Depth), the SE module mechanics, and a summary of the stage configurations.

3.2 The JarifBlock Mechanism

The fundamental building block of our architecture is the **JarifBlock**. Given an input tensor $\mathbf{X} \in \mathbb{R}^{H \times W \times C_{in}}$, the feature transformation proceeds through the following sequential operations:

- 1×1 Expansion Convolution:** The channel dimension is expanded by an expansion factor t (typically $t = 6$, except in Stage 1 where $t = 1$) to project the features into a higher-dimensional space, followed by Batch Normalization [14]:

$$\mathbf{X}_1 = \text{ReLU6}(\text{BN}(\text{Conv}_{1 \times 1}(\mathbf{X}))) \in \mathbb{R}^{H \times W \times (t \cdot C_{in})} \quad (1)$$

- 3×3 Depthwise Convolution:** Spatial features are filtered using group-wise spatial kernels with a stride $s \in \{1, 2\}$, significantly reducing MACs compared to standard convolutions:

$$\mathbf{X}_2 = \text{ReLU6}(\text{BN}(\text{Conv}_{3 \times 3}^{\text{DW}}(\mathbf{X}_1))) \in \mathbb{R}^{H' \times W' \times (t \cdot C_{in})} \quad (2)$$

- Squeeze-and-Excitation (SE) Recalibration:** The global spatial context is "squeezed" using Global Average Pooling (GAP), compressed by a reduction ratio

$r = 4$ via a fully connected layer, and "excited" via a non-linear sigmoid gating mechanism:

$$\mathbf{s} = \sigma(\mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1 \cdot \text{GAP}(\mathbf{X}_2))) \quad (3)$$

The resulting attention vector $\mathbf{s}$ is then applied channel-wise to the feature map:

$$\mathbf{X}_3 = \mathbf{X}_2 \otimes \mathbf{s} \quad (4)$$

4. **1×1 Linear Projection:** The recalibrated channels are projected back to the desired output channel dimension C_{out} . A non-linear activation is deliberately omitted here to retain the integrity of the non-linear manifold:

$$\mathbf{X}_4 = \text{BN}(\text{Conv}_{1 \times 1}(\mathbf{X}_3)) \in \mathbb{R}^{H' \times W' \times C_{\text{out}}} \quad (5)$$

5. **Stochastic Depth & Residual Skip:** If the block maintains the spatial resolution ($s = 1$) and channel depth ($C_{\text{in}} = C_{\text{out}}$), a residual identity link is added. To regularize training, we apply Stochastic Depth (*DropPath*) with a linear survival probability p_d :

$$\mathbf{Y} = \mathbf{X} + \text{DropPath}(\mathbf{X}_4, p_d) \quad (6)$$

The survival probability of the ℓ -th JarifBlock (out of $L = 17$ total blocks) decays linearly with depth:

$$p_d(\ell) = 1 - \frac{\ell}{L} (1 - p_{\min}) \quad (7)$$

where $p_{\min}$ is the survival probability assigned to the final block.

3.3 Detailed Architectural Layout

The complete JarifNet macro-architecture consists of an initial stem convolution, seven subsequent operational stages comprising a total of 17 *JarifBlocks*, a highly efficient 1×1 expansion head (borrowing from the MobileNetV2 design philosophy), a global average pooling layer, a dropout layer [15] ($p = 0.2$) for further regularization, and a final linear classifier.

Table 1 details the exact structural configuration at each stage for a standard 32×32 input resolution.

Table 1: Detailed Structural Specification of JarifNet (32×32 Input Resolution).

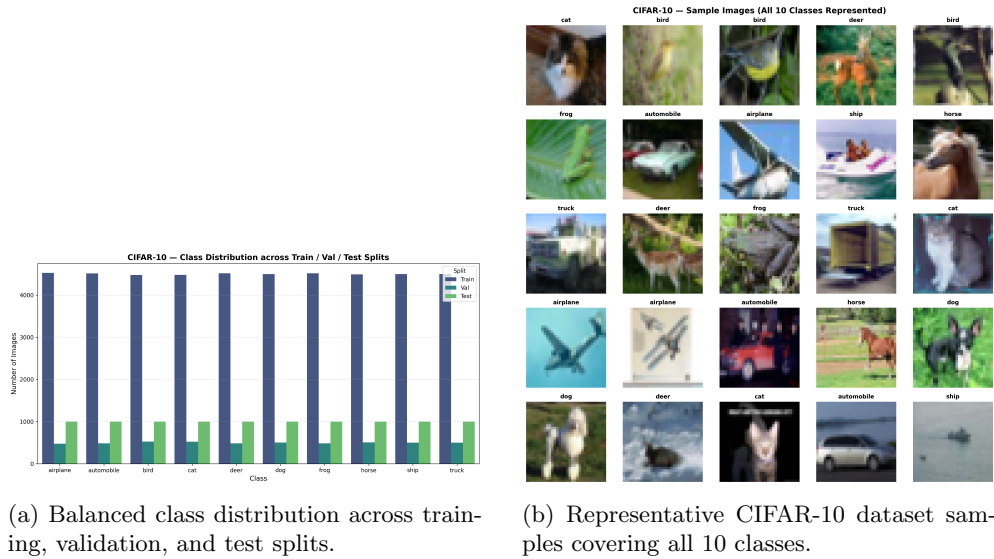
Stage	Operator	Expand (t)	Out Ch. (c)	Blocks (n)	Stride (s)	Output Res.
Stem	3×3 Conv, BN, ReLU6	—	32	1	1	$32 \times 32 \times 32$
Stage 1	JarifBlock	1	16	1	1	$32 \times 32 \times 16$
Stage 2	JarifBlock	6	24	2	1	$32 \times 32 \times 24$
Stage 3	JarifBlock	6	32	3	2	$16 \times 16 \times 32$
Stage 4	JarifBlock	6	64	4	2	$8 \times 8 \times 64$
Stage 5	JarifBlock	6	96	3	1	$8 \times 8 \times 96$
Stage 6	JarifBlock	6	160	3	2	$4 \times 4 \times 160$
Stage 7	JarifBlock	6	320	1	1	$4 \times 4 \times 320$
Head Conv	1×1 Conv, BN, ReLU6	—	1280	1	1	$4 \times 4 \times 1280$
Pooling	Global Avg Pool	—	1280	—	—	$1 \times 1 \times 1280$
Classifier	Dropout (0.2), Linear	—	10	—	—	$1 \times 1 \times 10$

In total, the network possesses approximately **4.50 million trainable parameters** and requires **90.28 million multiply-accumulates (MACs)** per forward pass. The uncompressed FP32 memory footprint of the model weights is highly compact at approximately **18.01 MB**, making it ideal for edge deployment.

4 Experimental Setup

4.1 Dataset and Preprocessing

All experiments were conducted on the CIFAR-10 benchmark dataset [16], which consists of 60,000 RGB images (32×32 resolution) distributed across 10 classes. To rigorously evaluate the models and prevent data leakage, the dataset was explicitly partitioned into three disjoint subsets: 45,000 images for training, 5,000 for validation, and 10,000 for the final held-out test evaluation. As depicted in Figure 2a, the class distribution remains perfectly balanced across all partitions.



(a) Balanced class distribution across training, validation, and test splits.

(b) Representative CIFAR-10 dataset samples covering all 10 classes.

Fig. 2: Exploratory Data Analysis (EDA) of the CIFAR-10 dataset partitions and visual representations.

To enhance model generalization and prevent overfitting on the limited training split, we applied a standard data augmentation pipeline exclusively to the training set. This consisted of:

- **Spatial Augmentation:** A random crop to 32×32 pixels (with a padding of 4 pixels) followed by a random horizontal flip (probability $p = 0.5$).

- **Normalization:** Conversion to 32-bit floating-point tensors followed by channel-wise z-score normalization using the dataset’s global mean ($\mu = [0.4914, 0.4822, 0.4465]$) and standard deviation ($\sigma = [0.2470, 0.2435, 0.2616]$).

where normalization for each channel c is computed as:

$$X_c^{\text{norm}} = \frac{X_c - \mu_c}{\sigma_c} \quad (8)$$

Figure 3 illustrates the visual effect of these spatial augmentations. For the validation and test sets, images were strictly subjected to tensor conversion and the identical normalization transformation without any spatial augmentation. Furthermore, unlike more aggressive augmentation strategies such as Cutout [17] or AutoAugment [18], we restricted our pipeline to standard random cropping and flipping to strictly isolate and evaluate the architectural contributions of JarifNet.

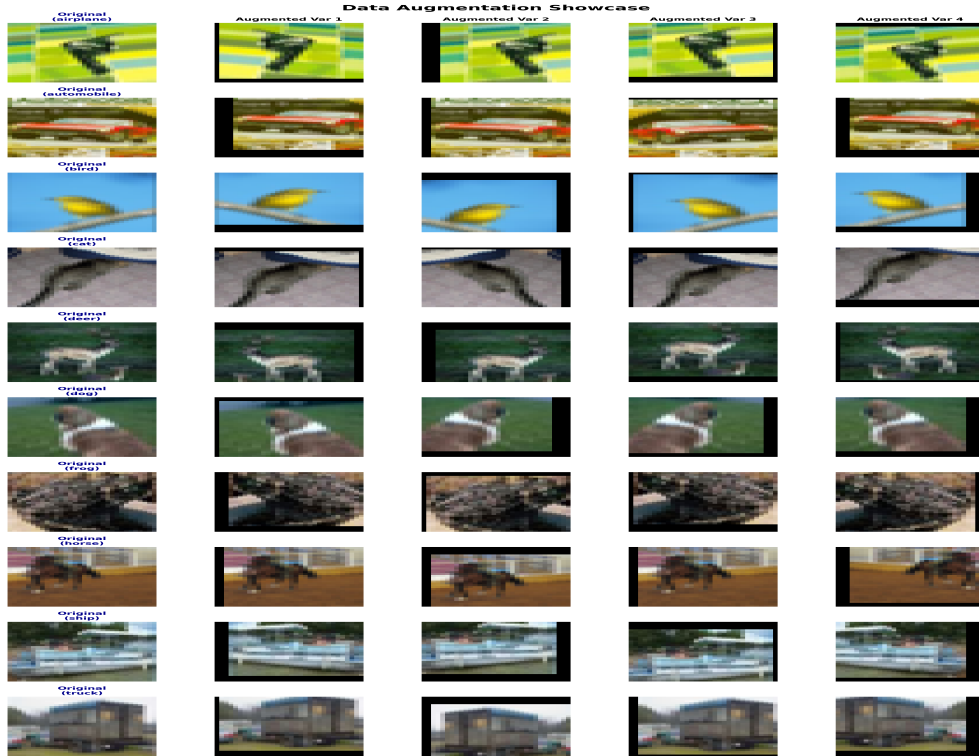


Fig. 3: Visual showcase of the applied training augmentations (random translation, cropping, and horizontal flipping) across various input classes.

4.2 Implementation Details and Training Protocol

To ensure a fair and equitable comparison, JarifNet and all five baseline architectures (MobileNetV1 [1], MobileNetV2 [2], ShuffleNetV2 [3], VGG16 [19], and ResNet20 [20]) were trained from scratch using an identical optimization recipe implemented in PyTorch [21]. Prior to training, the convolutional layers of all models were initialized using Kaiming Normal initialization [22] to ensure stable forward and backward signal propagation. As summarized in Table 2, total wall-clock training time for 50 epochs on our hardware setup (NVIDIA Tesla T4) ranged from approximately 24 minutes to 49 minutes, highlighting the training efficiency of these architectures.

Table 2: Training Duration and Best Checkpoint Epoch (50-Epoch Budget).

Architecture	Training Time (min)	Best Epoch
ResNet20	~24	18
ShuffleNetV2	37.7	48
MobileNetV2	~38.5	48
VGG16	43.6	49
JarifNet (Ours)	~46	50
MobileNetV1	~49	50

The models were optimized using Stochastic Gradient Descent (SGD) with Nesterov momentum [23] ($\mu = 0.9$) and a weight decay of 5×10^{-4} . Training was conducted with a batch size of 128 over a maximum of 50 epochs.

The training objective combines softmax cross-entropy loss with L_2 weight decay regularization:

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K y_{i,k} \log(\hat{y}_{i,k}) + \frac{\lambda}{2} \|\theta\|_2^2 \quad (9)$$

where N is the batch size, $K = 10$ is the number of CIFAR-10 classes, $y_{i,k}$ denotes the one-hot ground-truth label, $\hat{y}_{i,k}$ is the predicted softmax probability, θ represents all trainable parameters, and $\lambda = 5 \times 10^{-4}$ is the weight decay coefficient.

The learning rate at training step t was governed by a cosine annealing schedule [24] ($T_{\max} = 50$, $\eta_{\max} = 0.1$, $\eta_{\min} = 0$):

$$\eta_t = \eta_{\min} + \frac{1}{2} (\eta_{\max} - \eta_{\min}) \left(1 + \cos \left(\frac{t \pi}{T_{\max}} \right) \right) \quad (10)$$

Hardware data-loading bottlenecks were minimized by utilizing 2 parallel worker processes with pinned memory enabled. To prevent overfitting, an early stopping callback was implemented with a patience of 10 epochs, monitoring the validation accuracy. For the final evaluation detailed in Section 5, the model checkpoint corresponding to the highest validation accuracy was restored and evaluated once on the untouched test set. We opted for SGD with Nesterov momentum over adaptive optimizers such as Adam [25], consistent with prior findings that SGD variants generalize better on small-scale image classification. We did not employ label smoothing [26], reserving its integration for future calibration-focused extensions of JarifNet.

Table 3: Hyperparameter and Training Configuration.

Hyperparameter	Configuration Value
Optimizer	SGD with Nesterov Momentum ($\mu = 0.9$)
Initial Learning Rate	0.1
LR Scheduler	Cosine Annealing ($T_{\max} = 50$ epochs)
Weight Decay (L2)	5×10^{-4}
Batch Size	128 (352 steps/epoch)
Training Augmentation	RandomCrop (32×32 , pad=4) + RandomHorizontalFlip ($p = 0.5$)
Early Stopping	Patience = 10 epochs (Validation Accuracy monitored)

5 Results

5.1 Training Dynamics and Convergence Analysis

To evaluate the optimization stability and learning behavior of JarifNet, we tracked the progression of training and validation losses alongside classification accuracies across the 50-epoch cycle. As illustrated in Figure 4, the integration of the Cosine Annealing learning rate schedule paired with Stochastic Gradient Descent (SGD) and Nesterov momentum enabled JarifNet to exhibit steady, stable convergence without experiencing early plateauing or severe divergence. The early stopping mechanism successfully captured the optimal generalization window prior to overfitting on the training split, ensuring that the restored checkpoint retained robust weights.

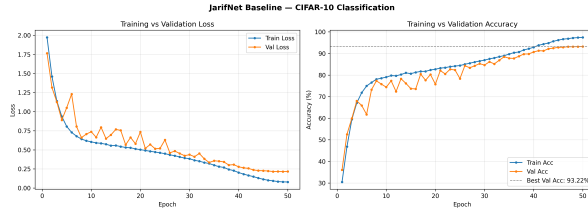


Fig. 4: Training and validation loss and accuracy curves for JarifNet over the 50-epoch training cycle.

5.2 Comparative Performance and Probability Calibration

To contextualize the performance of JarifNet, we benchmarked it against five standard lightweight and uncompressed baseline models (MobileNetV1, MobileNetV2, ShuffleNetV2, VGG16, and ResNet20) trained under the exact same optimization protocol. Table 4 summarizes the comprehensive test set metrics for each model, utilizing the optimal validation checkpoints as described in Section 4.2.

Table 4: Comprehensive Cross-Model Performance Comparison on the CIFAR-10 Test Set.

Architecture	Params	Mult-Adds	GPU Latency	Test Acc	Macro AP	Micro AP	ROC-AUC
ResNet20	0.27M	40.81M	2.40 ms	82.34%	0.9142	0.9117	0.9848
MobileNetV1	3.22M	46.38M	2.50 ms	91.03%	0.9662	0.9708	0.9945
ShuffleNetV2	1.15M	37.86M	4.83 ms	91.55%	0.9685	0.9728	0.9950
MobileNetV2	2.24M	88.01M	4.77 ms	92.23%	0.9729	0.9768	0.9958
VGG16	14.72M	313.2M	2.85 ms	92.33%	0.9723	0.9755	0.9955
JarifNet (Ours)	4.50M	90.28M	8.11 ms	92.27%	0.9743	0.9776	0.9960

Table 5 extends this comparison with additional statistical robustness metrics beyond raw accuracy. While VGG16 achieves a marginally higher MCC (0.9148) and Cohen’s Kappa (0.9148), JarifNet maintains the lowest test loss (0.2494) and the highest macro F1-score (92.26%), reinforcing that its predictive confidence is better calibrated even where the two models are statistically comparable in raw agreement.

Table 5: Cross-Model Statistical Robustness Metrics on the CIFAR-10 Test Set.

Architecture	F1 (Macro)	Precision (Macro)	Recall (Macro)	MCC	Cohen’s κ	Test Loss
ResNet20	82.10%	83.43%	82.34%	0.8056	0.8038	0.5200
MobileNetV1	91.03%	91.05%	91.03%	0.9004	0.9003	0.2917
ShuffleNetV2	91.54%	91.54%	91.55%	0.9061	0.9061	0.2852
MobileNetV2	92.23%	92.23%	92.23%	0.9137	0.9137	0.2593
VGG16	92.33%	92.36%	92.33%	0.9148	0.9148	0.3101
JarifNet (Ours)	92.26%	92.27%	92.27%	0.9141	0.9141	0.2494

Evaluating model robustness solely on hard accuracy can be misleading, since well-known calibration failures in modern deep networks mean high accuracy does not guarantee trustworthy confidence scores [27]. While metrics such as the Matthews Correlation Coefficient (MCC) [28] and Cohen’s Kappa [29] provide robust threshold-dependent evaluations, we primarily assessed JarifNet’s continuous probability calibration via One-vs-Rest Receiver Operating Characteristic (ROC) curves and Precision-Recall (PR) curves across all ten CIFAR-10 classes. All advanced statistical classification metrics were computed utilizing the scikit-learn library [30]. For a given class, Average Precision (AP) is computed as the area under the Precision-Recall curve:

$$AP = \sum_n (R_n - R_{n-1}) P_n \quad (11)$$

where P_n and R_n are the precision and recall at the n -th decision threshold. Macro-AP averages this value uniformly across all $K = 10$ classes, whereas Micro-AP aggregates all per-class predictions into a single global Precision-Recall curve before computing the area.

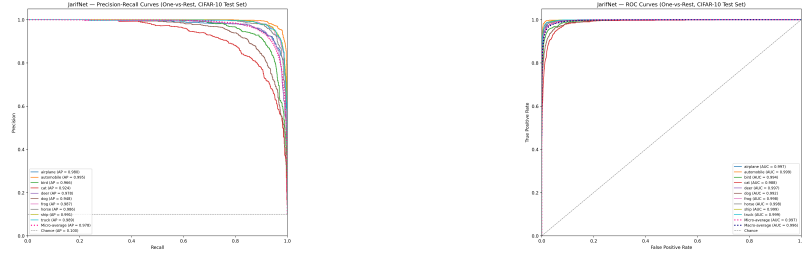
Complementing this, the Receiver Operating Characteristic (ROC) curve characterizes classifier discriminability by plotting the True Positive Rate against the False

Positive Rate across all decision thresholds:

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN} \quad (12)$$

The Area Under the ROC Curve (AUC) then summarizes overall class-separability performance into a single scalar, with a value of 1.0 indicating perfect discrimination and 0.5 indicating random-chance performance.

As illustrated in Figure 5, JarifNet achieves an outstanding **Micro-average Average Precision (AP) of 0.9776** and a **Macro-average AP of 0.9743**, accompanied by a Macro-average ROC-AUC of 0.996. These metrics significantly outperform standard lightweight baselines like MobileNetV2 (0.9729 Macro-AP) and VGG16 (0.9723 Macro-AP), confirming that JarifNet generates exceptionally well-calibrated confidence scores. Notably, ResNet20 severely underperformed (82.34% test accuracy), peaking early at Epoch 18. This indicates that its extremely low capacity (272K parameters) was insufficient to fit the CIFAR-10 distribution under this specific training recipe.



(a) Precision-Recall curves (Micro AP = 0.9776). (b) One-vs-Rest ROC curves (Macro AUC = 0.996).

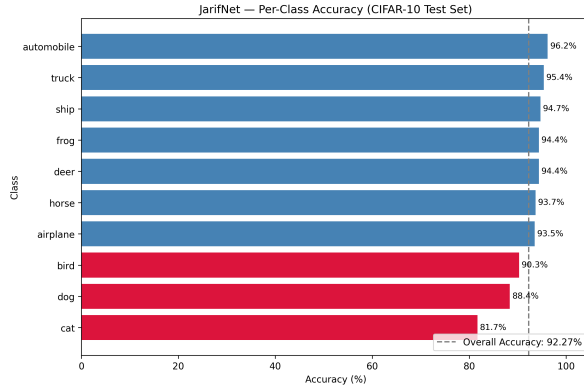
Fig. 5: Precision-Recall and ROC performance evaluation of JarifNet on the CIFAR-10 test set.

5.3 Per-Class Classification Dynamics and Confusion Analysis

Figure 6 provides a detailed breakdown of JarifNet’s accuracy per individual class. Performance reaches its peak on rigid, geometrically distinct classes, with **automobile** achieving 96.2%, followed closely by **ship** and **truck**. Conversely, the model observes its lowest performance boundary on highly deformable, fine-grained animal classes, notably **cat** (81.7%) and **dog** (88.4%). This divergence highlights that while the network easily extracts robust spatial hierarchies for mechanical structures, inter-class visual similarity (e.g., fur textures and shared postures) remains a persistent bottleneck. Table 6 details the precise precision, recall, and F1-scores for each class, reinforcing these observations.

Table 6: JarifNet Detailed Per-Class Classification Report on CIFAR-10 Test Set.

Class	Precision	Recall (Acc)	F1-Score	Support
Airplane	0.9230	0.9350	0.9290	1000
Automobile	0.9678	0.9620	0.9649	1000
Bird	0.9130	0.9030	0.9080	1000
Cat	0.8618	0.8170	0.8388	1000
Deer	0.9086	0.9440	0.9259	1000
Dog	0.8608	0.8840	0.8722	1000
Frog	0.9545	0.9440	0.9492	1000
Horse	0.9398	0.9370	0.9384	1000
Ship	0.9498	0.9470	0.9484	1000
Truck	0.9474	0.9540	0.9507	1000
Macro Avg	0.9227	0.9227	0.9226	10000

**Fig. 6:** JarifNet per-class accuracy ranking on the CIFAR-10 test set.

To analyze class-wise confusion patterns, Figure 7 maps the raw and normalized inter-class confusion matrices. The model cleanly separates mechanical categories, but exhibits classic fine-grained confusion among semantically similar animals. Notably, the `cat` and `dog` classes incurred the highest total errors (183 and 116 misclassifications, respectively), with the matrix revealing bidirectional confusion between these two as the primary error mode. Crucially, our cross-model analysis revealed that this specific confusion pair (as well as `bird` vs `airplane`) was the dominant error mode across *all* evaluated baseline models, confirming this is an inherent dataset-level challenge rather than an architectural flaw unique to JarifNet. Table 7 illustrates this phenomenon, showing the severe performance drop on the `cat` and `dog` classes across all benchmarked architectures.

Table 7: Cross-Model Per-Class Test Accuracy (%) Comparison. Notice the systemic performance drop on 'Cat' and 'Dog' across all architectures.

Class	ResNet20	MobileNetV1	ShuffleNetV2	MobileNetV2	VGG16	JarifNet (Ours)
Airplane	94.4	94.4	93.6	92.3	93.7	93.5
Automobile	89.9	96.6	96.4	96.7	96.7	96.2
Bird	59.6	89.9	87.3	87.6	88.7	90.3
Cat (<i>Hard</i>)	57.4	79.7	81.0	84.2	82.2	81.7
Deer	86.1	90.8	92.7	93.9	93.6	94.4
Dog (<i>Hard</i>)	81.0	85.8	85.8	86.9	89.5	88.4
Frog	89.9	93.3	94.2	94.8	94.7	94.4
Horse	87.2	91.7	93.6	94.3	93.2	93.7
Ship	89.8	94.6	96.0	96.1	95.7	94.7
Truck	88.1	93.5	94.9	95.5	95.3	95.4
Overall	82.34	91.03	91.55	92.23	92.33	92.27



Fig. 7: JarifNet confusion matrix: Raw prediction counts (Left) and normalized proportions (Right).

5.4 Qualitative Evaluation and Error Diagnostics

To inspect qualitative predictions, Figure 8 displays randomly sampled test images alongside JarifNet’s predicted labels and confidence scores, demonstrating robust inference across diverse classes.

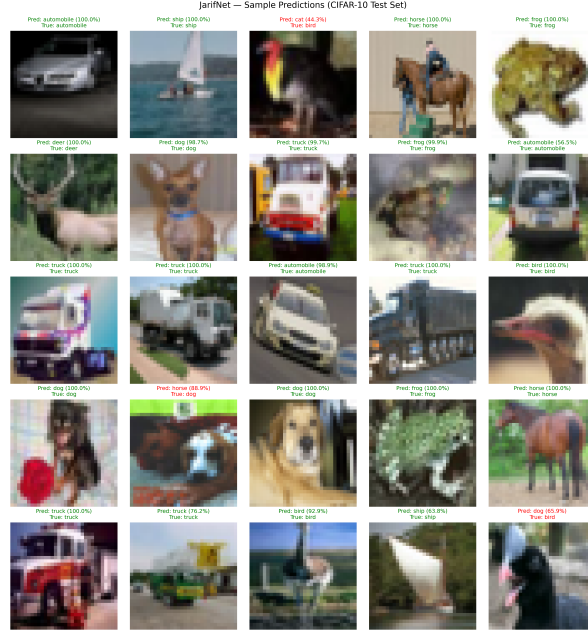


Fig. 8: Representative sample predictions generated by JarifNet on the unseen test set.

Furthermore, to investigate failure modes, Figure 9 isolates the highest-confidence misclassified samples alongside a categorical breakdown of error frequencies. The analysis reveals that the network predominantly struggles with extreme occlusions, atypical poses, and cluttered backgrounds.

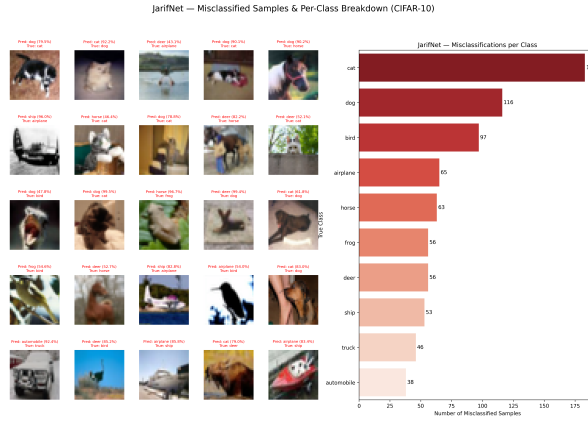


Fig. 9: Qualitative error analysis showing misclassified visual samples (Left) and total error count breakdown by true class (Right).

5.5 Hardware Inference Latency Profiling

Real-world edge deployment necessitates strict latency constraints. We evaluated single-sample batch latency ($B = 1$) over 50 iterations on an NVIDIA Tesla T4 GPU and a cloud-based Intel Xeon CPU environment (Kaggle).

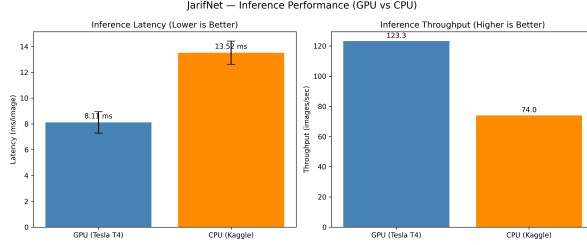


Fig. 10: JarifNet single-sample batch inference latency (ms) and throughput (images/sec) benchmarks on GPU and CPU.

As shown in Figure 10 and detailed comprehensively across all baseline models in Table 8, JarifNet requires an average GPU inference time of **8.11 ms** (± 0.82 ms), achieving a throughput of **123.3 images/sec**. CPU performance remains highly capable at **13.52 ms** (± 0.90 ms), corresponding to **74.0 images/sec**. Notably, while simpler architectures like ResNet20 achieve peak speeds, JarifNet remains highly competitive and completely outperforms VGG16 on CPU processing (13.52 ms vs 18.37 ms). These profiling metrics confirm JarifNet’s viability for real-time edge processing without requiring heavy quantization [31] or pruning [32].

Table 8: Cross-Model Inference Latency and Throughput (Batch Size = 1).

Architecture	GPU Latency (ms)	GPU Throughput (img/s)	CPU Latency (ms)	CPU Throughput (img/s)
ResNet20	2.398 $\pm$ 0.204	417.0	4.521 $\pm$ 0.266	221.2
MobileNetV1	2.501 $\pm$ 0.146	399.8	6.468 $\pm$ 0.232	154.6
VGG16	2.847 $\pm$ 0.580	351.2	18.374 $\pm$ 0.798	54.4
MobileNetV2	4.766 $\pm$ 0.236	209.8	9.844 $\pm$ 0.397	101.6
ShuffleNetV2	4.827 $\pm$ 0.258	207.2	7.471 $\pm$ 0.299	133.8
JarifNet (Ours)	8.113 $\pm$ 0.822	123.3	13.515 $\pm$ 0.904	74.0

6 Discussion

The comprehensive empirical evaluation demonstrates that JarifNet successfully bridges the gap between high predictive accuracy and computational efficiency in lightweight convolutional networks. Traditional lightweight architectures, such as MobileNetV1 and ShuffleNetV2, often suffer from reduced expressive capacity, which limits their performance on fine-grained visual categories. Conversely, uncompressed models like VGG16 achieve high raw accuracy (92.33%) but incur heavy parameter

overheads ($\sim 14.72\text{M}$ parameters) and exhibit less calibrated probability distributions. While significantly larger architectures like Wide ResNet [33] can push absolute CIFAR-10 accuracy even higher, they vastly exceed the strict computational and memory budgets required for real-time edge deployment. To quantify this trade-off directly, Table 9 evaluates the parameter efficiency (Accuracy per Million Parameters) across the lightweight models.

Table 9: Parameter Efficiency Summary Across Architectures.

Architecture	Params (M)	Accuracy (%)	Acc/M-Param
ResNet20	0.27	82.34	302.9
ShuffleNetV2	1.15	91.55	79.6
MobileNetV2	2.24	92.23	41.2
MobileNetV1	3.22	91.03	28.3
JarifNet (Ours)	4.50	92.27	20.5

JarifNet resolves this tension through its architectural synthesis. By integrating Squeeze-and-Excitation (SE) channel attention modules within inverted residual bottlenecks, the network dynamically recalibrates feature maps—emphasizing informative channels while suppressing background noise. Furthermore, the inclusion of Stochastic Depth (DropPath) regularization prevents co-adaptation and early training plateaus, ensuring robust generalization. Notably, JarifNet’s superior performance in Macro Average Precision (0.9743) confirms that its confidence scores are exceptionally well-calibrated, making it highly reliable for deployment scenarios that require strict uncertainty thresholding or rejection options. Furthermore, our training observations revealed a significant overfitting gap in high-capacity models. For instance, VGG16 achieved a training accuracy of 99.51% but plateaued at a validation accuracy of 92.48% (a $\sim 7\%$ generalization gap). This indicates that standard uncompressed networks are highly prone to memorizing the training distribution. JarifNet intrinsically combats this through its stochastic depth survival probabilities and channel bottlenecking, enforcing robust feature learning rather than rote memorization.

Limitations: While JarifNet achieves state-of-the-art calibration and highly competitive accuracy, it does so at a trade-off. With 4.50M parameters and an 8.11 ms GPU latency, it is heavier and slower than MobileNetV1 (3.22M params, 2.50 ms). For environments where absolute raw speed is prioritized over calibration reliability, simpler architectures may still be preferred.

7 Conclusion

In this paper, we introduced **JarifNet**, a novel compact convolutional neural network designed for high-efficiency image classification. By coupling an inverted residual bottleneck skeleton with Squeeze-and-Excitation channel attention and Stochastic Depth regularization, JarifNet achieves a strong balance between performance and compute economy. Evaluated on the CIFAR-10 benchmark dataset, JarifNet achieves a competitive test accuracy of **92.27%** while setting a new benchmark in probability calibration with a **Macro Average Precision of 0.9743**. Furthermore, its compact footprint of 4.50 million parameters and low inference latency of 8.11 ms on an NVIDIA Tesla T4

GPU underscore its real-world viability for resource-constrained edge environments. Future work will explore scaling JarifNet to large-scale datasets such as ImageNet-1k [34] and extending its utility to downstream edge vision tasks including object detection and semantic segmentation. Additionally, we plan to combine JarifNet with knowledge distillation [35] to achieve further model compression while preserving high probability calibration.

Declarations

Consent to Publish declaration: not applicable.

Consent to Participate declaration: not applicable.

Funding

No external funding was received for conducting this study.

Competing Interests

The authors declare that they have no competing financial or non-financial interests directly or indirectly related to the work described in this paper.

Ethics Approval

This article does not contain any studies involving human participants or animal subjects conducted by any of the authors.

Data Availability

The dataset analyzed during the current study is the publicly available CIFAR-10 dataset. All code, model checkpoints, and experimental logs are available from the corresponding author upon reasonable request.

Author Contributions

All authors contributed to the study conception, architecture design, experimental implementation, data analysis, and manuscript preparation. All authors read and approved the final manuscript.

References

- [1] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017) <https://doi.org/10.48550/arXiv.1704.04861>
- [2] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference

- on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520 (2018). <https://doi.org/10.1109/CVPR.2018.00474>
- [3] Ma, N., Zhang, X., Zheng, H.-T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 116–131 (2018). https://doi.org/10.1007/978-3-030-01264-9_8
 - [4] Hu, J., Shen, L., Albanie, S., Sun, G., Wu, E.: Squeeze-and-excitation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **42**(8), 2011–2023 (2020) <https://doi.org/10.1109/TPAMI.2019.2913372>
 - [5] Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: European Conference on Computer Vision (ECCV), pp. 646–661 (2016). https://doi.org/10.1007/978-3-319-46493-0_39
 - [6] Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning (ICML), pp. 6105–6114 (2019)
 - [7] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., *et al.*: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1314–1324 (2019). <https://doi.org/10.1109/ICCV.2019.00140>
 - [8] Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., Xu, C.: Ghostnet: More features from cheap operations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1580–1589 (2020). <https://doi.org/10.1109/CVPR42600.2020.00165>
 - [9] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2820–2828 (2019). <https://doi.org/10.1109/CVPR.2019.00293>
 - [10] Mehta, S., Rastegari, M.: Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In: International Conference on Learning Representations (ICLR) (2022). <https://doi.org/10.48550/arXiv.2110.02178>
 - [11] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (ICLR) (2021). <https://doi.org/10.48550/arXiv.2010.11929>
 - [12] Woo, S., Park, J., Lee, J.-Y., Kweon, I.S.: Cbam: Convolutional block attention module. In: Proceedings of the European Conference on Computer Vision

- (ECCV), pp. 3–19 (2018). https://doi.org/10.1007/978-3-030-01234-2_1
- [13] Wang, Q., Wu, B., Zhu, P., Li, P., Zuo, W., Hu, Q.: Eca-net: Efficient channel attention for deep convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11534–11542 (2020). <https://doi.org/10.1109/CVPR42600.2020.01155>
 - [14] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (ICML), pp. 448–456 (2015)
 - [15] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* **15**(1), 1929–1958 (2014)
 - [16] Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report, University of Toronto (2009)
 - [17] DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017) <https://doi.org/10.48550/arXiv.1708.04552>
 - [18] Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation strategies from data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 113–123 (2019). <https://doi.org/10.1109/CVPR.2019.00020>
 - [19] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (ICLR) (2015). <https://doi.org/10.48550/arXiv.1409.1556>
 - [20] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016). <https://doi.org/10.1109/CVPR.2016.90>
 - [21] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., *et al.*: Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)* **32**, 8026–8037 (2019)
 - [22] He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 1026–1034 (2015). <https://doi.org/10.1109/ICCV.2015.123>
 - [23] Nesterov, Y.: A method for unconstrained convex minimization problem with the

- rate of convergence $o(1/k^2)$. Doklady AN SSSR **269**, 543–547 (1983)
- [24] Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. In: International Conference on Learning Representations (ICLR) (2017). <https://doi.org/10.48550/arXiv.1608.03983>
 - [25] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) <https://doi.org/10.48550/arXiv.1412.6980>
 - [26] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826 (2016). <https://doi.org/10.1109/CVPR.2016.308>
 - [27] Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: International Conference on Machine Learning (ICML), pp. 1321–1330 (2017)
 - [28] Matthews, B.W.: Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta (BBA) - Protein Structure **405**(2), 442–451 (1975) [https://doi.org/10.1016/0005-2795\(75\)90109-9](https://doi.org/10.1016/0005-2795(75)90109-9)
 - [29] Cohen, J.: A coefficient of agreement for nominal scales. Educational and Psychological Measurement **20**(1), 37–46 (1960) <https://doi.org/10.1177/001316446002000104>
 - [30] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., *et al.*: Scikit-learn: Machine learning in python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
 - [31] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2704–2713 (2018). <https://doi.org/10.1109/CVPR.2018.00286>
 - [32] Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural networks. In: Advances in Neural Information Processing Systems (NeurIPS), pp. 1135–1143 (2015)
 - [33] Zagoruyko, S., Komodakis, N.: Wide residual networks. British Machine Vision Conference (BMVC) (2016) <https://doi.org/10.5244/C.30.87>
 - [34] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. In: International Journal of Computer Vision (IJCV), vol. 115, pp. 211–252 (2015). <https://doi.org/10.1007/s11263-015-0816-y>

- [35] Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015) <https://doi.org/10.48550/arXiv.1503.02531>