Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Statistical Analysis of Weight Discretization in Deep Learning

Master's Final Degree Project

Kamilis Jonkus

Project author

Assoc. prof. dr. Tomas Iešmantas

Supervisor

Kaunas, 2026

Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Statistical Analysis of Weight Discretization in Deep Learning

Master's Final Degree Project

Applied Mathematics (6211AX006)

Kamilis Jonkus

Project author

Assoc. prof. dr. Tomas Iešmantas

Supervisor

Doc. dr. Mindaugas Kavaliauskas

Reviewer

Kaunas, 2026

Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Kamilis Jonkus

Statistical Analysis of Weight Discretization in Deep Learning

Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;

2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;

3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;

4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Kamilis Jonkus

Confirmed electronically

Jonkus Kamilis. Statistical Analysis of Weight Discretization in Deep Learning. Master's Final Degree Project / supervisor assoc. prof. dr. Tomas Iešmantas; Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Study field and area (study field group): Applied Mathematics (Mathematical Sciences).

Keywords: neural networks, weight quantization, integer quantization, clipping threshold, statistical distributions.
Kaunas, 2026. 54.

Summary

The practical application of deep neural networks requires substantial computational resources. These costs can be reduced by approximating model parameters and layer activations with lower-precision numerical formats. However, when the bit width is too low, model accuracy may decrease substantially. This degradation is related to clipping distortion at the edges of the quantization interval and rounding distortion within the discretization grid.

Neural network weight quantization was investigated using statistical methods. Four parametric distributions, Gaussian, Laplace, Student's t, and generalized Gaussian, were fitted to trained convolutional network weights at both layer and channel granularity. For each tensor or channel, the best-fitting distribution was selected. Based on the selected distribution, an MAE optimal symmetric clipping bound was derived for integer quantization.

The analytical clipping bound reduced mean absolute quantization error in most evaluated settings compared with the standard MinMax method. However, lower weight reconstruction error did not consistently lead to higher classification accuracy. On ImageNet, the analytical method achieved lower accuracy than MinMax for ResNet18, even when its MAE was smaller. This indicates that a clipping threshold that is optimal for weight approximation is not necessarily optimal for the final classification task.

To explain this discrepancy, the relationship between quantization-induced output mean shifts and accuracy degradation was investigated. Previous studies have shown that weight quantization can shift the means of layer-output distributions and thereby reduce model accuracy

[1]. The hypothesis was tested by training 500 small convolutional networks on MNIST. Under 4-bit quantization, a statistically significant positive correlation was found between total output mean shift and accuracy drop for both clipping methods: 0.761 for the analytical method and 0.707 for MinMax. Bias correction was then applied to compensate for these output mean shifts.

The floating-point ResNet18 model achieved 69.86% top-1 accuracy on ImageNet. After bias correction, the accuracy of the MinMax-quantized INT8 model increased from 69.71% to 69.76%, while the accuracy of the analytically quantized INT8 model increased from 67.56% to 68.57%. Thus, bias correction reduced part of the accuracy deficit caused by quantization, especially for the analytical method, but MinMax remained the more accurate clipping method in the final ImageNet evaluation.

Jonkus Kamilis. Giliųjų neuroninių tinklų svorių diskretizavimo statistinė analizė. Magistro baigiamasis projektas / vadovas assoc. prof. dr. Tomas Iešmantas; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Taikomoji matematika (Matematikos mokslai).

Reikšminiai žodžiai: neuroniniai tinklai, svorių diskretizavimas, sveikaskaitis diskretizavimas, nukirpimo riba, statistiniai skirstiniai.

Kaunas, 2026. 54 p.

Santrauka

Giliųjų neuroninių tinklų panaudojimas praktikoje reikalauja didelės apimties skaičiavimo išteklių. Šias sąnaudas galima mažinti parametrus ir tinklo sluoksnių aktyvacijas aproksimuojant mažesnio skaitinio tikslumo formatu. Vistik naudojant pernelyg mažą bitų skaičių tokiai aproksimacijai tikslumas reikšmingai mažėja. Tai sietina su dviem priežastimis: nukirpimo (angl. *clipping*) iškraipymu svorių intervalo kraštuose ir apvalinimo iškraipymu diskretizavimo tinklelyje.

Neuroninių tinklų svorių kvantavimas nagrinėtas sistemingai taikant statistinius metodus. Keturi parametriniai skirstiniai (Gauso, Laplaso, Stjudento t ir apibendrintasis Gauso) buvo taikyti apmokytų ResNet tipo tinklų svoriams sluoksnių ir kanalų lygmeniu: kiekvienu atveju parinktas geriausiai tinkantis skirstinys. Buvo išvesta optimali simetrinė svorių nukirpimo riba, priklausanti nuo parinkto skirstinio funkcijos, minimizuojant vidutinę absoliutinę kvantavimo paklaidą.

Gautos analitinės nukirpimo ribos panaudojimas leido sumažinti vidutinę absoliutinę paklaidą daugumoje sluoksnių, lyginant su klasikiniu praktikoje dažnai naudojamu MinMax metodu. Tačiau ImageNet duomenų klasifikavimo užduotyje analitinė riba lėmė mažesnį klasifikavimo tikslumą nei MinMax riba. Tai indikuoja, jog optimalus kvantavimo paklaidos prasme slenkstis nebūtinai yra optimalus užduoties, sprendžiamos tuo konkrečiu tinklu, atžvilgiu: klasifikavimo rezultatai nepranoko paprastesnės euristinės kvantavimo taisyklės.

Siekiant paaiškinti šį skirtumą, buvo tiriamas ryšys tarp sluoksnių išvesčių skirstinių poslinkio ir klasifikavimo tikslumo degradacijos. Ankstesni tyrimai parodė, kad pritaikius svorių kvantavimą atsiranda sluoksnių išvesčių skirstinių vidurkių poslinkiai, kurie lemia modelio tikslumo kritimą [1]. Hipotezė buvo tikrinama apmokant 500 mažų konvoliucinių tinklų su MNIST duomenų rinkiniu. Atliekant svorių diskretizavimą į 4 bitų sveikojo skaičiaus formatą, nustatyta statistiškai reikšminga koreliacija tarp sluoksnių išvesčių skirstinių vidurkių poslinkių ir tikslumo degradacijos abiem nukirpimo metodams – 0,761 analitiniam ir 0,707 MinMax metodui. Tuomet buvo pritaikyta sluoksnių poslinkių korekcija.

Slankiojo kablelio ResNet-18 modelio ImageNet duomenų rinkinio klasifikavimo tikslumas yra 69,86 proc. Pritaikius korekciją, MinMax metodu kvantuoto modelio tikslumas padidėjo nuo 69,71 proc. iki 69,76 proc, o analitiniu metodu kvantuoto modelio – nuo 67,56 proc iki 68,57 proc. Taigi, korekcija sumažino dalį didesnio analitinio metodo tikslumo trūkumo, išlaikant nedidelį MinMax metodo pranašumą.

Table of contents

List of figures **8**
List of tables **9**
List of abbreviations and terms **10**
Introduction **11**
1. Literature review **12**
1.1. Neural networks efficiency optimization 12
1.1.1. Efficient architectures 12
1.1.2. Knowledge distillation 12
1.1.3. Numerical precision reduction 12
1.1.4. Self compressing neural networks 12
1.1.5. Batch Normalization 13
1.1.6. Reparametrisation 13
1.1.7. Quantization aware REPVGG 14
1.2. Integer quantization 15
1.2.1. Layer quantization 15
1.2.2. Static and dynamic calibration 16
1.2.3. Quantization operator 16
1.2.4. Symmetric and asymmetric quantization 17
1.2.5. Per-layer and per-channel quantization 18
1.2.6. Non-uniform quantization 18
1.2.7. Clipping methods 19
1.2.8. Analytical clipping of integer quantization 19
1.2.9. Bias correction 20
1.3. Justifications of thesis aim and objectives 21
2. Methodology **22**
2.1. Integer quantization error 22
2.1.1. Decomposition of quantization error 22
2.1.2. Clipping error term expression simplification 23
2.1.3. Rounding error term simplification. 23
2.1.4. Analytical expressions for optimal clipping bounds 24
2.2. Distributions and distribution fitting 25
2.2.1. Likelihood 26
2.2.2. Gaussian distribution 26
2.2.3. Laplace distribution 26
2.2.4. Student`s t distribution 26
2.2.5. Generalized Gaussian distribution 27
2.3. Quantization specification 27
2.4. Performance evaluation 27
2.4.1. Data pre-processing 28
2.4.2. Mean absolute error 28
2.4.3. Classification accuracy 28
2.4.4. Class activation mapping 28
2.4.5. Custom CNN 29
2.5. Statistical significance evaluation 29

2.5.1. Training 30
2.5.2. Wilcoxon signed-rank test 30
2.5.3. Spearman correlation 31
2.6. Implementation resources 32
3. Results 33
3.1. Statistical properties of layer weights distributions 33
3.2. Batch normalization fusion effects 34
3.3. Weights distributions fitting 35
3.3.1. Per-layer ResNet18 weights distributions fitting 35
3.3.2. Per-channel ResNet18 weights distributions fitting 37
3.3.3. Custom CNN weights distributions fitting 39
3.3.4. Summary 41
3.4. Quantization accuracy evaluation 41
3.4.1. ResNet18 41
3.4.2. Custom CNN 41
3.4.3. Summary 43
3.5. Output distributions mean shift after quantization 43
3.5.1. Custom CNN outputs mean shift 43
3.5.2. Custom CNN bias correction 45
3.5.3. ResNet18 outputs mean shift and bias correction 45
3.5.4. ResNet18 accuracy evaluation 47
3.5.5. Class activations evaluation 48
3.5.6. Summary 50
3.6. Summary 50
Conclusions 51
List of references 52

List of figures

Fig. 1. RepVGG blocks fusion illustration (fused blocks on the right) [14] 14
Fig. 2. RepVGG (a) and QARepVGG (b) blocks fusion scheme [15] 15
Fig. 3. Schematic overview of a convolutional layer quantization [16] 16
Fig. 4. Per-layer and per-channel quantization comparison [17] 18
Fig. 5. Comparison between clipping ranges of different clipping methods for input activations to layer 3 of ResNet50 [13] 19
Fig. 6. Quantization error visualization. **A –** clipping error**, B** – rounding error 22
Fig. 7. Custom CNN architecture scheme with layer input and output shapes 29
Fig. 8. Per-layer weight histograms of pre-trained ResNet18, in forward order from top left to bottom right 33
Fig. 9. Per-layer empirical moments. The red line in the kurtosis panel marks the Gaussian baseline at 3 34
Fig. 10. Per-channel weight ranges of ResNet18 layer2.1.conv2 before and after batch normalization fusion 34
Fig. 11. Empirical histogram of ResNet18 layer3.0.conv1 weights with the four fitted distributions and the per-layer MinMax and ACIQ clip thresholds. GED – Generalized Gaussian distribution .. 36
Fig. 12. Empirical histogram of fully connected layer weights with the four fitted distributions and the per-layer MinMax and ACIQ clip thresholds. GED – Generalized Gaussian distribution 36
Fig. 13. First 16 output channels of layer3.0.conv1. Each subplot shows the channel weight histogram with the layer-level best-fit density drawn as a dashed line and the channel level best fit drawn as a solid line. Line colors identify the distribution family per the legend. GED – Generalized Gaussian distribution 38
Fig. 14. First 10 MNIST CNN training runs train and validation losses 40
Fig. 15. Mean and standard deviation of the per-block output mean shift across 500 trained models (INT4) 44
Fig. 16. Per-block and total mean shift against accuracy drop across 500 trained models (INT4)... 44
Fig. 17. Per-block and total mean shift against accuracy drop across 500 trained models (INT8)... 44
Fig. 18. Per-layer output mean shift under 8-bit per-channel MinMax weight quantization, without and with the bias correction step 46
Fig. 19. Per-layer output mean shift under 8-bit per-channel ACIQ weight quantization, without and with the bias correction step 46
Fig. 20. Per-image CAM cosine similarity between the FP32 ground truth class CAM and each quantized variant's ground-truth-class CAM. 48
Fig. 21. Class activation maps under quantization for four ImageNet validation images. Cell titles report the variant's predicted class and softmax probability, coloured green on a correct prediction and red on a wrong one 49

List of tables

Table 1. Per-layer best fit distribution and 8-bit per-layer symmetric quantization results. The Gain column is the ratio MAE_{mm}/MAE_{aciq}. 37
Table 2. Per-channel best-fit distribution counts across the 21 weight modules of ResNet18. The n_c column is the number of output channels in the module. 39
Table 3. Per-layer best fit distribution counts across the 5 weight modules of the custom CNN. Counts are given as cumulative counts over 500 runs. 40
Table 4. Per-channel best fit distribution counts across the 5 weight modules of the custom CNN. The n_c column is the number of output channels in the module. Counts are given as averages per-500 runs. 40
Table 5. ImageNet classification accuracy comparison of quantization methods under different quantization granularity and different integer precision. Analysed model is ResNet18. 41
Table 6. MAE comparison of MinMax and ACIQ clipping methods on 500 trained custom CNNs under different quantization granularities and integer precisions. The p-value is computed using the paired Wilcoxon signed-rank test. 42
Table 7. MNIST classification accuracy comparison of MinMax and ACIQ clipping methods on 500 trained custom CNNs under different quantization granularities and integer precisions. The p-value is computed using the paired Wilcoxon signed-rank test. 43
Table 8. Effect of bias correction on MNIST classification accuracy of quantized custom CNNs. Results compare MinMax and ACIQ clipping before and after bias correction for INT8 and INT4 quantization. The p-value is computed using the paired Wilcoxon signed-rank test. 45
Table 9. Accuracy comparison of quantization methods. 47

List of abbreviations and terms

Abbreviations:

ACIQ – analytical clipping of integer quantization;

CAM – class activation mapping;

CDF – cumulative distribution function;

CNN – convolutional neural network;

MAE – mean absolute error;

MSE – mean squared error;

PDF – probability density function;

PTQ – post-training quantization;

QAT – quantization aware training;

ReLU – rectified linear unit.

Introduction

Deep neural networks are increasingly used in business, scientific, medical, and embedded applications, where large amounts of data are often processed with limited computational resources. Although modern neural network architectures achieve high accuracy, their deployment may be constrained by memory usage, inference latency, and energy consumption. These limitations are especially relevant when models are used on devices with limited hardware capabilities or in systems that require real-time processing.

One practical approach for reducing computational cost is to reduce numerical precision. Instead of storing and processing weights and activations in a 32-bit floating-point format, quantization represents them using lower precision numerical formats, commonly integers. Integer quantization can reduce model size and accelerate inference, but it may also degrade neural network accuracy. This degradation is caused by the approximation error introduced when continuous floating-point values are mapped to a finite set of discrete levels.

The quality of quantization depends not only on the number of bits used but also on how the quantization interval is selected. A wider interval preserves extreme values but increases rounding error for values near the center of the distribution. A narrower interval improves resolution for frequently occurring values but clips values in the distribution tails. Therefore, the statistical properties of neural network weights are important when choosing quantization thresholds.

This work investigates weight discretization from a statistical perspective. Parametric distributions are fitted to neural network weight tensors, and the resulting distributions are used to derive quantization clipping thresholds by minimizing mean absolute error (MAE). The proposed methodology is compared with the classical MinMax approach. In addition, the relationship between weight discretization, layer-output distribution shifts, and model accuracy degradation is analyzed.

Aim of the thesis:

Develop a statistically optimal weight quantization method for convolutional neural networks and compare it to the standard MinMax approach in terms of quantization error and classification accuracy.

Objectives of the thesis:

1. Review the literature on integer quantization of convolutional neural networks, focusing on clipping methods and the effect of quantization on layer output distributions.
2. Derive a closed form expression for the MAE optimal symmetric clipping bound in terms of the weight distribution cumulative distribution function (CDF).
3. Fit four parametric distributions (Gaussian, Laplace, Student's t, generalized Gaussian) to the weight tensors of a trained ResNet18 at layer and channel granularity and assess which family describes weights the best.
4. Compare the analytical clipping method to the MinMax baseline on ImageNet classification under per-layer and per-channel granularity.
5. Investigate the relationship between quantization induced shifts in layer output means and the drop in classification accuracy, and test bias correction as a mitigation.

1. Literature review

Deployment of modern convolutional neural networks is limited by their inference cost in memory and compute [2]. Integer quantization reduces this cost by replacing 32-bit floating point weights and activations with low bit integers [3]. Introduction narrows from the broader efficiency optimization landscape to the integer quantization methodology.

1.1. Neural networks efficiency optimization

Modern convolutional networks consume substantial compute, memory, and energy at inference. This section covers some research areas in neural network optimization.

1.1.1. Efficient architectures

Inference can be made cheaper by designing more efficient architectures. MobileNet splits each convolution into a depthwise step and a 1×1 pointwise step. The split reduces computation by roughly nine times for 3×3 kernels at small accuracy loss on ImageNet [4].

1.1.2. Knowledge distillation

Inference cost can be reduced by training a small student network to imitate a large pre-trained teacher [5]. DistilBERT applies the technique to language pre-training. A BERT teacher is distilled during the pre-training stage into a student 40% smaller, 60% faster, and retaining 97% of the teacher's language understanding score [6]. Distillation can be combined with other efficiency optimization methods such as quantization or pruining.

1.1.3. Numerical precision reduction

Inference rarely needs the 32-bit floating point precision used during training, so deployment formats trade dynamic range against bit count. The 8-bit integer format maps each tensor through an affine relation, executed on integer multiply accumulate units widely available on commodity hardware [3]. The 8-bit floating point specification defines two formats: E4M3 (4 exponent, 3 mantissa bits) for the forward pass, and E5M2 (5 exponent, 2 mantissa bits) for gradients [7]. Later work pushes FP8 through gradients, optimizer states, and distributed communication for transformer language model training [8].

The extreme case constrains weights to $\{-1,+1\}$ so that multiplications collapse to sign flips, removing the multiplier from the inner loop entirely [9]. Recent work on large language models revives the idea with a ternary $\{-1,0,+1\}$ scheme that matches full precision baselines at billion parameter scale [10]. The result suggests that low precision discretization scales up favorably with model size.

1.1.4. Self compressing neural networks

One interesting piece of work is self compressing neural networks. A training scheme in which bit width is a learnable parameter optimized jointly with the network weights. A regularization term penalizes the total number of bits the model consumes, so the optimizer is incentivized to reduce precision wherever the task loss permits. The training objective itself governs precision allocation. Layers whose weights are more sensitive to rounding converge toward higher bit widths. Insensitive

layers converge toward lower precision. It is also allowed to prune weights entirely. No separate sensitivity analysis or calibration step is required [11].

1.1.5. Batch Normalization

Batch normalization is a layer operation that normalizes intermediate neural network values and then applies a learned affine transformation. During training, the mean and variance are computed from the mini batch. This stabilizes the distributions passed between layers and improves neural network training. For inference, batch normalization does not depend on the current mini batch. Instead, fixed mean and variance estimates collected during training are used [12].

For an input value x, batch normalization is defined by formula (1).

$$y(x) = \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}}\gamma + \beta \tag{1}$$

here x – input vector; μ – estimated mean; σ^2 – estimated variance; ϵ – small constant for numerical stability; γ – learned scale parameter; β – learned shift parameter.

During inference, μ, σ^2, ϵ, γ, and β are constants. Therefore, batch normalization is a fixed affine transformation. If it follows a fully connected layer, it can be fused into that layer by changing only the weights and biases, as shown in formula (2).

$$y_j = \frac{\sum_{k=1}^{p} x_k \cdot w_{kj} + b_j - \mu}{\sqrt{\sigma^2+\epsilon}}\gamma + \beta = \sum_{k=1}^{p} \frac{x_k \cdot w_{kj} \cdot \gamma}{\sqrt{\sigma^2+\epsilon}} + \frac{(b_j-\mu)\cdot\gamma}{\sqrt{\sigma^2+\epsilon}} + \beta = \sum_{k=1}^{p} x_k \cdot w'_{kj} + b'_j \tag{2}$$

here $w'_{kj} = \frac{w_{kj}\cdot\gamma}{\sqrt{\sigma^2+\epsilon}}$; $b'_j = \frac{(b_j-\mu)\cdot\gamma}{\sqrt{\sigma^2+\epsilon}} + \beta$.

The same idea applies to convolutional layers. Batch normalization parameters are estimated separately for each output feature map, so every output channel receives its own scale and shift. For the j-th convolutional output channel, fusion is given by formula (3) [13].

$$y_i = \frac{W_j * x + b_j - \mu}{\sqrt{\sigma^2+\epsilon}}\gamma + \beta = \frac{W_j * x}{\sqrt{\sigma^2+\epsilon}}\gamma + \frac{(b_j-\mu)\cdot\gamma}{\sqrt{\sigma^2+\epsilon}} + \beta = W'_j * x + b'_j \tag{3}$$

here $W'_j = \frac{W_j\cdot\gamma}{\sqrt{\sigma^2+\epsilon}}$; $b'_j = \frac{(b_j-\mu)\cdot\gamma}{\sqrt{\sigma^2+\epsilon}} + \beta$.

After this fusion transformation, the batch normalization layer no longer needs to be executed as a separate operation, therefore neural network efficiency is improved. This is also relevant for post-training quantization, because quantization is applied to the fused weights and biases rather than to the original unfused layer parameters [13].

1.1.6. Reparametrisation

Reparameterization decouples the topology used during training from the topology used at inference. Trained branches can be folded into a single equivalent layer before deployment. RepVGG trains with a 3×3 convolution, a 1×1 convolution, and an identity shortcut in parallel. The three branches are fused into one 3×3 convolution at inference. Fusion is illustrated in figure 1. The rewrite is algebraic, not approximate, so inference accuracy is preserved [14].

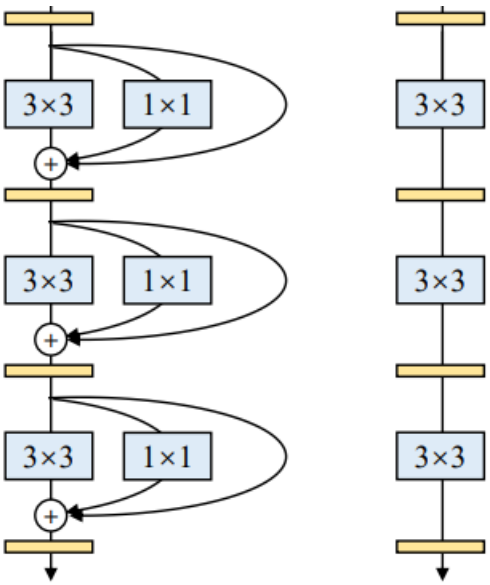

Fig. 1. RepVGG blocks fusion illustration (fused blocks on the right) [14]

1.1.7. Quantization aware REPVGG

However, reparametrization does not couple well with integer post-training quantization (PTQ). RepVGG-A0 drops from 72.2% to 50.3% top-1 on ImageNet under standard INT8 PTQ. Drops above 20 percentage points are observed across the RepVGG family [15].

PTQ accuracy is determined by the post-fusion weight and activation distributions. Two distribution level conditions determine PTQ success. The fused weight distribution must have a small variance and few outliers. The per-layer activation distribution must satisfy the same condition. Activation errors form part of the next layer's input. Errors, therefore, compound with depth [15].

The original RepVGG training objective applies a custom weight decay term. It was shown that this custom weight decay term inflates per-branch output variance during training. Replacing the custom term with standard L2 regularization raises INT8 accuracy from 50.3% to 61.6% on RepVGG-A0 [15].

The QARepVGG architecture removes batch normalization from the identity and 1×1 branches. It appends a single batch normalization layer after the three branch summation. The change preserves identical fusion to a 3×3 convolution at deploy time. RepVGG and QARepVGG block architectures and their fusion are illustrated in figure 2. On ImageNet, QARepVGG-A0 reaches 72.2% top-1 under FP32 and 70.4% under INT8 PTQ. The post-PTQ gap stays below 2 percentage points across the QARepVGG family. Each modification was justified by tracing its effect on a specific distributional condition rather than by post-hoc parameter tuning [15].

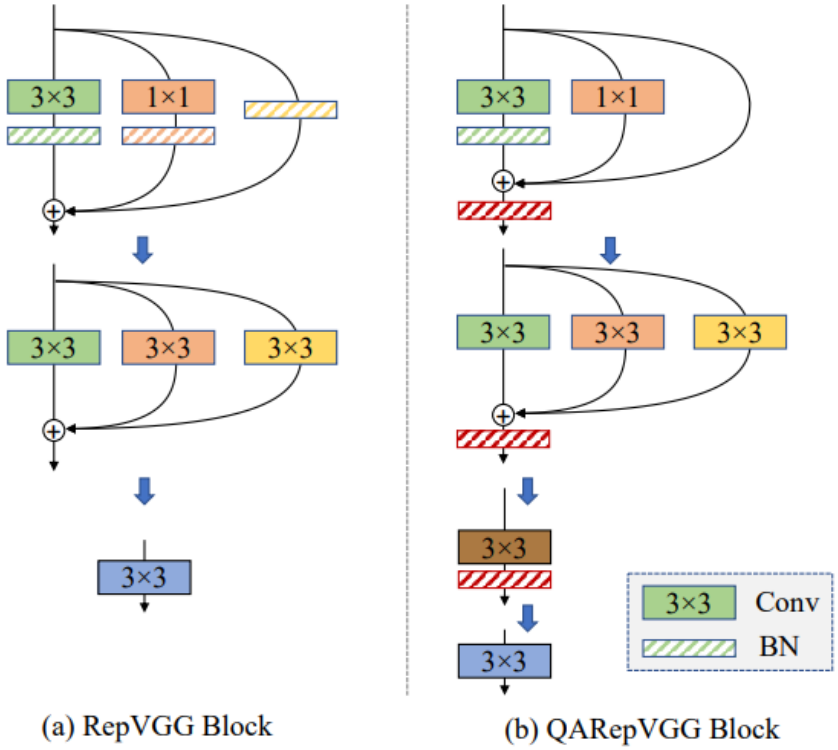

Fig. 2. RepVGG (a) and QARepVGG (b) blocks fusion scheme [15]

Results show that analisys of only quantization methodology is not enough. Integer quantization accuracy can be significantly recovered by introducing model architecture and training loss modifications. In the case of the QARepVGG architecture, such modifications were informed by statistical analyses of weight and activation distributions and by mathematical formulations of how these distributions are affected by the model architecture and training configuration.

1.2. Integer quantization

This section covers the PTQ scheme. Models trained with usual training schemes are eligible for PTQ quantization, however the model architecture and training configuration influence quantization results [13, 15]. Quantization aware training (QAT) often results in better quantized models' accuracy, but it requires significant training algorithm modifications [13, 16].

1.2.1. Layer quantization

Integer arithmetic inference fixes a numerical format for every tensor along the forward path of a layer [3]. Weights and layer inputs are stored in a low bit width integer format, typically INT8. The fully connected or convolutional layer intermediate low precision computation results are summed to an accumulator held at higher precision, typically INT32, to prevent overflow. Biases are added into the accumulator at INT32 precision. The INT32 result is then quantized back to the low precision format and used as the next layer’s input [16]. Figure 3 shows this flow for a convolutional layer.

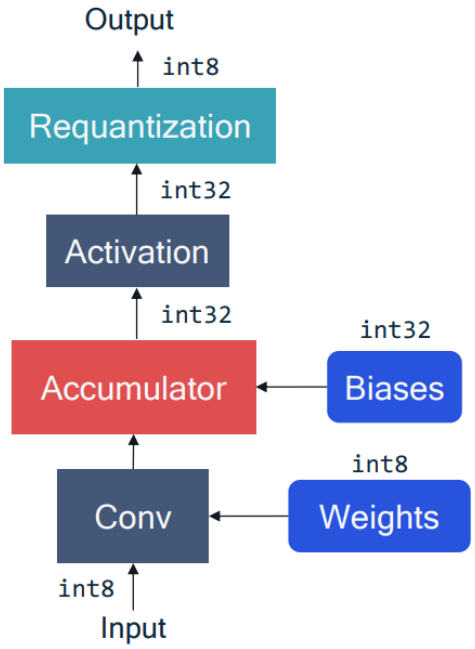

Fig. 3. Schematic overview of a convolutional layer quantization [16]

1.2.2. Static and dynamic calibration

Calibration determines the layer activations distribution. Trained weights and biases are fixed. Their distribution does not change across inference samples, so the clipping range can be derived directly from the trained tensor. Activations are functions of the input. Their range shifts from sample to sample. Determining activation quantization parameters, therefore, requires a calibration step, while weight and bias parameters do not [16].

Static calibration estimates the activation range once on a small sample of unlabeled inputs. The clipping thresholds are then fixed at deployment. The estimator runs offline, so it can be expensive. Dynamic calibration recomputes the clipping range from the live activation tensor on every forward pass. The recomputation runs inside the inference loop, so the estimator must be cheap. Static calibration is the default for fixed cost inference. Dynamic calibration is used when the input distribution drifts far enough from the calibration sample to make the offline estimate unreliable [17].

1.2.3. Quantization operator

The quantization operator maps a real valued tensor to a finite set of integer values. The asymmetric quantization operator is given by the formula (4). The clipping operator is given by the formula (5). Scale parameter s and zero point parameter z are calculated according to formulas (6) and (7) [13, 16].

$$x_q = Q(x) = clip(\lfloor s \cdot x + z \rceil, -2^{b-1}, 2^{b-1} - 1) \tag{4}$$

here x – input value; $\lfloor \cdot \rceil$ – round to nearest operation; b – integer bit width.

$$clip(x; a, c) = \begin{cases} a, & x < a, \\ x, & a \le x \le c, \\ c, & x > c. \end{cases} \tag{5}$$

$$s = \frac{2^b - 1}{q_{max} - q_{min}} \tag{6}$$

here q_{min} – original tensor lower clipping bound; q_{max} – original tensor upper clipping bound.

$$z = -\lfloor q_{min} \cdot s \rceil - 2^{b-1} \tag{7}$$

The dequantization function is given by formula (8). It does not return x exactly. The discrepancy $x - \hat{x}$ is the quantization error [13, 16].

$$x \approx \hat{x} = \frac{1}{s}\left(x_q - z\right) \tag{8}$$

For the symmetric case, the quantization operator is defined by formula (9), the scale parameter by formula (10), the dequantization operator by formula (11), the zero point parameter z does not exist, and $q_{min} = -q_{max}$ [3, 16]. For the symmetric case, the integer range is not [−128, 127], but [−127, 127]. Such a modification allows for further optimization of computation [3, 13].

$$x_q = Q(x) = clip(\lfloor s \cdot x \rceil, -(2^{b-1} - 1), 2^{b-1} - 1) \tag{9}$$

$$s = \frac{2^{b-1} - 1}{q_{max}} \tag{10}$$

$$x \approx \hat{x} = \frac{1}{s} x_q \tag{11}$$

1.2.4. Symmetric and asymmetric quantization

The asymmetric quantization scheme introduces a zero point parameter, allowing better alignment with the original data distribution, especially when it is not centered around zero. However, this added flexibility leads to more complex computations during inference. When both weights W and activations x are quantized asymmetrically, their multiplication produces several components given by formula (12). Here, weights and activations are multiplied in quantized representation, including the dequantization operation [13, 16].

$$\frac{1}{s_w}\left(W_q - z_w\right)\frac{1}{s_x}\left(x_q - z_x\right) = \frac{1}{s_w s_x} W_q x_q - \frac{1}{s_w s_x} z_w x_q - \frac{1}{s_w s_x} z_x W_q + \frac{1}{s_w s_x} z_w z_x \tag{12}$$

The first component is present only if both weights and activations are quantized using the symmetric scheme. The third term arises from asymmetric quantization of activations. All its components (s_w, s_x, z_x, W_q) are known before inference, so they can be computed during model optimization and even fused into the bias term. The fourth term arises from asymmetric quantization of weights and activations. Its computation overhead is negligible, the same as the third term. The second term arises from the asymmetric quantization of weights. This term depends on input x_q. Thus, it can not be fused into other parameters and results in significant computation overhead, which may even outweigh the efficiency gain of integer quantization [3, 13].

So, it is recommended to use a symmetric quantization scheme for neural network weights. An asymmetric quantization scheme can be used for activations quantization [13].

1.2.5. Per-layer and per-channel quantization

Quantization granularity determines how many scale parameters cover a single weight tensor. Per-layer quantization assigns a single scale to the entire tensor. The asymmetric variant adds a single zero point at the same granularity. Per-channel quantization assigns a separate scale per-output channel. A convolutional kernel with C_{out} filters then have C_{out} scales instead of one. Figure 4 contrasts the two schemes. The extra quantization metadata is proportional to C_{out} and stays small relative to the quantized weight tensor [17, 18].

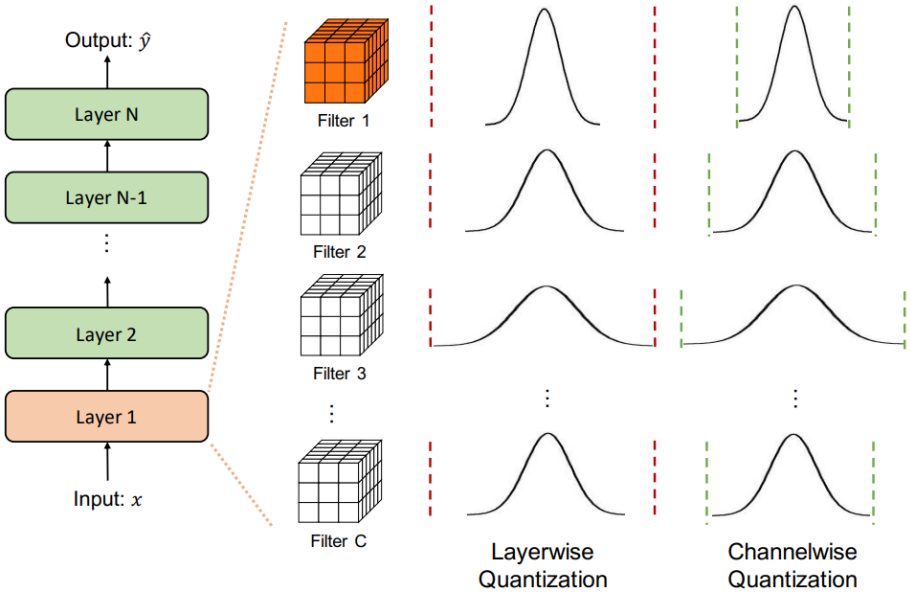

Fig. 4. Per-layer and per-channel quantization comparison [17]

The quantization accuracy is influenced by batch normalization. Batch normalization parameters γ and σ^2 are estimated separately per-output feature map. After batch normalization fusion, each filter is multiplied by the factor $\gamma/\sqrt{\sigma^2+\epsilon}$ independently. A per-layer scale must cover the largest filter. Smaller filters then use only a small subset of the available integer levels. Their granular error grows correspondingly. Batch normalization induced variation is a major contributor to per-layer weight quantization accuracy loss. Per-channel quantization addresses this by assigning each filter a quantization step size matched to its own range, thereby reducing quantization error and improving accuracy [18].

Per-channel weight quantization of weights is supported in mainstream inference frameworks and maps well to processor instructions. Activations follow the opposite convention. A per-channel activation scale would multiply each summand of the convolution by a different factor. The accumulation could no longer be expressed as one shared rescale of an integer sum. Activation quantization, therefore, stays per-layer even when weights are per-channel [13].

1.2.6. Non-uniform quantization

Uniform quantization places the levels on a regular grid with a constant step set by a single scale. Non-uniform quantization chooses the levels to fit the data distribution. Non-uniform spacing fits the distributions of trained weights and activations better than a uniform grid at a fixed bit width, enabling better accuracy. The drawback is that a non-uniform scheme is difficult to deploy efficiently on

hardware, and it is a more complex algorithm, so uniform quantization is mostly used in quantization applications [17].

1.2.7. Clipping methods

The clipping range [q_{min},q_{max}] trades clipping error on values outside the range against granular error from the rounding step. Three data driven methods estimate the range directly from the empirical tensor distribution. MinMax method sets the bounds to the observed extrema. The estimate has no clipping error on the calibration sample. A single outlier inflates the step and raises granular error on the bulk of the values. Percentile replaces the extrema with a chosen quantile, typically the 99.9th. The choice trades a small clipping error on the tails for a tighter step. Entropy calibration searches over candidate thresholds for the bound that minimizes the Kullback-Leibler divergence between the original tensor and its quantized counterpart. The three methods produce different ranges on the same tensor (figure 5). Entropy calibration is a default workflow in the Nvidia integer quantization library [13].

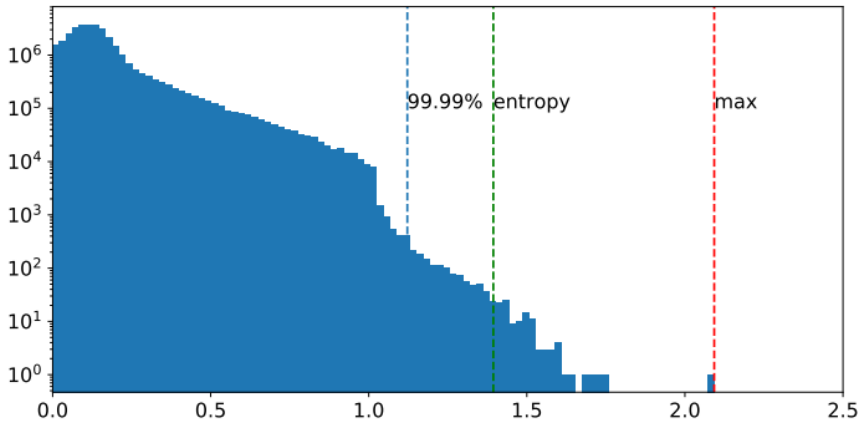

Fig. 5. Comparison between clipping ranges of different clipping methods for input activations to layer 3 of ResNet50 [13]

1.2.8. Analytical clipping of integer quantization

Analytical clipping of integer quantization (ACIQ) replaces the search for optimal clipping bounds with a closed form expression. Under a parametric assumption on the tensor, typically Laplace or Gaussian, the symmetric clipping bound that minimizes expected mean squared error (MSE) follows from the distribution parameters. Simple expressions for quantization clipping bounds are derived from formula (13). Under the assumption of the Laplace distribution, optimal symmetric clipping bounds for 2, 3, 4 bit quantization are 2.83b, 3.89b, 5.03b, where b is the Laplace distribution parameter [19].

$$\int_{-\infty}^{-\alpha} f(x)(x+\alpha)^2\,dx + \sum_{i=0}^{2^b-1} \int_{-\alpha+i\Delta}^{-\alpha+(i+1)\Delta} f(x)(x-q_i)^2 dx + \int_{\alpha}^{\infty} f(x)(x-\alpha)^2\,dx \qquad (13)$$

here f(x) – floating point precision tensor probability density function; Δ – integer bin width in floating point domain; q_i – i-th integer bin; α – symmetric clipping bound.

Other ACIQ method applications also utilize the MSE metric [20]. However, other work suggests the MSE metric is sensitive to outlier weights and proposes MAE as a better optimization target [21].

1.2.9. Bias correction

Post-training quantization often assumes that quantization error has zero mean. Under this assumption, positive and negative errors cancel when many quantized weights are multiplied by activations and summed. In practice, this cancellation is not always achieved. A finite set of weights can have a non zero average quantization residual. When these residuals are multiplied by layer inputs, they introduce a shift in the output mean. This shifted output becomes the input to the next layer, so the error can accumulate through the network depth [1].

Bias correction compensates for this effect without changing the trained weights. The expected output shift is estimated for every output channel and subtracted from the corresponding bias parameter. If W_q denotes the quantized weights and W_{fp} denotes the original floating point weights, the quantization residual is $\epsilon = W_q - W_{fp}$. The correction term is given by formula (14) [1].

$$\Delta b = \varepsilon \mathrm{E}[x] \tag{14}$$

here E[x] – expected input to the layer.

For convolutional and fully connected layers, the correction is computed separately for each output channel, because each output channel has its own bias parameter. The bias is then updated before inference using formula (15) [1].

$$b_{\text{new}} = b_{\text{orig}} - \Delta b \tag{15}$$

The expected input E[x] can be estimated in two ways. The first approach uses a small calibration dataset and directly measures the average activation entering each layer. The second approach avoids calibration data by using the parameters of preceding batch normalization layers. In this case, the pre-activation distribution is assumed to be Gaussian, with mean and variance determined by the batch normalization shift and scale parameters. If a rectified linear unit (ReLU) activation follows batch normalization, the negative part of this Gaussian distribution is clipped to zero. The resulting expected value is given by formula (16).

$$\mathrm{E}[x] = \gamma f(\beta/\gamma) + \beta F(\beta/\gamma) \tag{16}$$

here, f – Gaussian distribution probability density function (PDF); F – Gaussian distribution CDF; β – batch normalization learned shift parameter; γ – batch normalization learned scale parameter.

The expected inputs are propagated through the network in topological order. For the first layer, the input statistics are obtained from the input pre-processing procedure. For residual additions, the input mean is computed as the sum of the branch means. This allows the correction term to be computed for every convolutional and fully connected layer before inference [1].

Bias correction is especially important when quantization produces systematic output shifts rather than only random rounding noise. In MobileNetV2, bias correction alone increased INT8 ImageNet top-1 accuracy from 0.12% to 52.02%, showing that output mean shifts can be a major source of post-training quantization degradation [1].

1.3. Justifications of thesis aim and objectives

Analytical clipping methods for post-training quantization derive clipping thresholds from assumed parametric distributions of quantized tensors. Existing analytical approaches commonly rely on Gaussian or Laplace approximations, but the suitability of these distributions for describing trained neural network weight tensors is not always examined in detail [19, 20]. Since the selected distribution directly affects the resulting clipping threshold, this motivates a broader statistical analysis of weight distributions and the evaluation of alternative parametric families, including heavier tailed distributions.

The choice of optimization criterion is also important. Previous applications of analytical clipping use the MSE objective [19, 20], although this metric may be sensitive to outliers in the weights. Since neural network weight distributions often contain tail values, MAE may provide a more robust target for deriving clipping thresholds [21].

Furthermore, bias correction methods have shown that output distribution shifts can substantially affect quantized model accuracy, but the statistical relationship between output mean shift and accuracy degradation remains insufficiently evaluated [1].

These limitations motivate the development and experimental assessment of a statistically grounded weight discretization methodology.

2. Methodology

This chapter develops the analytical and experimental methodology. Sections 2.1 and 2.2 derive the integer quantization error decomposition, the optimal clipping bounds, and the maximum likelihood selection of the best parametric weight fit. Sections 2.3 and 2.4 fix the quantization specification and the evaluation metrics. Section 2.5 specifies the statistical significance evaluation protocol, including model training, the Wilcoxon signed-rank test, and Spearman correlation analysis. Section 2.6 lists the implementation resources.

2.1. Integer quantization error

Figure 6 splits the integer quantization error into two intuitive parts. Section A highlights values that fall outside the clipping bounds. Such outliers are snapped to the nearest bound. The lost distance from that bound is the clipping error. Section B highlights values that fall inside the clipping bounds. Such inliers are snapped to the nearest discrete bin centre. The gap between the original value and bin center is the rounding error.

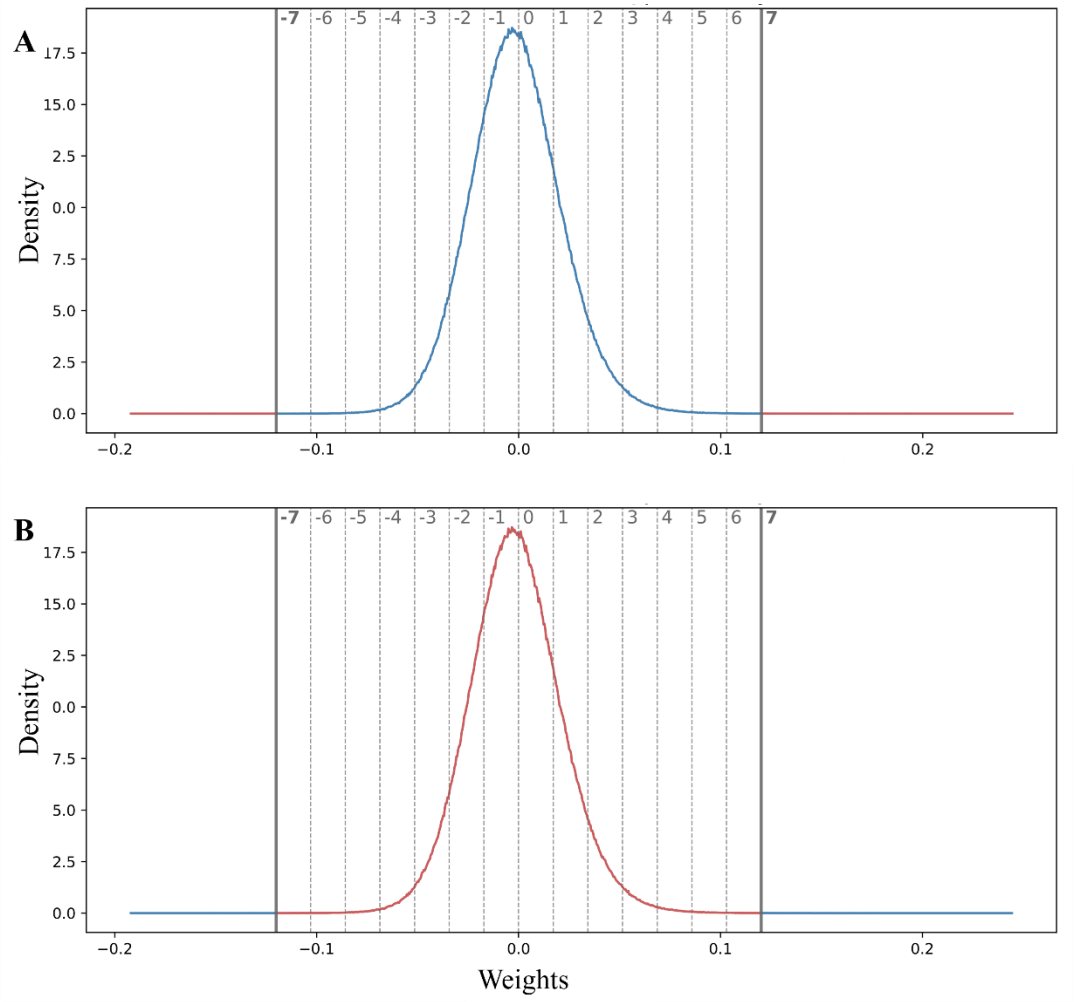

Fig. 6. Quantization error visualization. **A –** clipping error**, B** – rounding error

2.1.1. Decomposition of quantization error

Let f(x) a PDF and F(x) a CDF of neural network weights or activations. Considering an asymmetric quantization scheme with distribution clipping bounds [β,α] (denoted as [q_{min},q_{max}] in section 1.2.3.).

The quantization step size is given by formula (17). A continuous signal is discretized into discrete bins q∈R, where the set R size is 2^b.

$$\Delta = \frac{2\alpha}{2^b} \tag{17}$$

The total MAE D_{MAE} is the sum of errors from the outlier and inlier regions and is calculated using formula (20). Outlier regions contribute to the clipping error, which is separately calculated using formula (18). The inliers region contributes to granular rounding error and is calculated using formula (19).

$$D_{clip} = \int_{-\infty}^{\beta}(\beta - x)\, f(x)dx + \int_{\alpha}^{\infty}(x - \alpha)\, f(x)dx \tag{18}$$

$$D_{round} = \sum_{i=0}^{2^b-1} \int_{R_i}(x - q_i) f(x)dx \tag{19}$$

$$D_{MAE} = \sum_{i=0}^{2^b-1} \int_{R_i}(x - q_i) f(x)dx + \int_{-\infty}^{\beta}(\beta - x)\, f(x)dx + \int_{\alpha}^{\infty}(x - \alpha)\, f(x)dx \tag{20}$$

2.1.2. Clipping error term expression simplification

Expanding formula (18) terms results in formula (21).

$$D_{clip} = \beta \int_{-\infty}^{\beta} f(x)dx - \int_{-\infty}^{\beta} x\, f(x)dx + \int_{\alpha}^{\infty} x\, f(x)dx - \alpha \int_{\alpha}^{\infty} f(x)dx \tag{21}$$

Formulas (22) and (23) hold true, so formula (21) is further simplified to formula (24).

$$\int_{\alpha}^{\infty} f(x)dx = 1 - F(\alpha) \tag{22}$$

$$\int_{-\infty}^{\beta} f(x)dx = F(\beta) \tag{23}$$

$$D_{clip} = \beta F(\beta) - \int_{-\infty}^{\beta} x\, f(x)dx + \int_{\alpha}^{\infty} x\, f(x)dx - \alpha(1 - F(\alpha)) \tag{24}$$

2.1.3. Rounding error term simplification.

The rounding error integral in formula (19) cannot be evaluated in closed form. For a small bin width Δ, a simplification is available. Bennett's high resolution assumption replaces f(x) throughout each bin by its midpoint value $f(q_i)$. The bin local quantization error $\varepsilon = x - \hat{x}$ then follows a uniform distribution on [−Δ/2,Δ/2]. The same distribution applies to every bin [22].

Formula (25) gives the expected absolute error conditional on x falling in the bin i. The result does not depend on the bin index i.

$$E[\,|\varepsilon| \mid x \in R_i\,] = \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2} |\varepsilon|\, d\varepsilon = \frac{2}{\Delta}\int_{0}^{\Delta/2} \varepsilon\, d\varepsilon = \frac{2}{\Delta} \cdot \frac{(\Delta/2)^2}{2} = \frac{\Delta}{4} \tag{25}$$

The total rounding error decomposes into per-bin contributions weighted by bin probabilities given by formula (26).

$$D_{\text{round}} = \sum_{i=0}^{2^b-1} P\,(x \in R_i) \cdot E[\,|\varepsilon| \mid x \in R_i\,] = \frac{\Delta}{4}\sum_{i=0}^{2^b-1} P\,(x \in R_i) \approx \frac{\Delta}{4} \tag{26}$$

The conditional value $\frac{\Delta}{4}$ is constant in i and pulls out of the sum. The remaining $\sum_i P\ (x \in R_i)$ is the inlier probability mass over [β,α]. This mass is approximately unity once the clipping bounds capture the bulk of f(x). The per-bin and total rounding error expressions, therefore, coincide. Substituting Δ from formula (17) gives the simplified rounding error in terms of bit width and clipping bounds, formula (27).

$$D_{\text{round}} \approx \frac{\alpha-\beta}{2^{b+2}} \tag{27}$$

The approximation holds when f(x) is slowly varying on the scale of one bin. It degrades when f(x) has features narrower than Δ [22].

The framing is asymptotic. Under a fixed probability density function f(x), increasing the bit width strictly shrinks Δ. The approximation is therefore more accurate at higher bit widths than at lower ones, under fixed f(x). At 8 bits, the small step assumption is more strongly satisfied than at 4 bits or at 2 bits [22].

2.1.4. Analytical expressions for optimal clipping bounds

Combination of D_{clip} and D_{round} simplified forms (24) and (27) are denoted by formula (28).

$$D_{MAE} = \frac{\alpha-\beta}{2^{b+2}} + \beta F(\beta) - \int_{-\infty}^{\beta} x\, f(x)dx + \int_{\alpha}^{\infty} x\, f(x)dx - \alpha(1-F(\alpha)) \tag{28}$$

For optimal values of clipping bounds α and β, derivatives of D_{MAE} are calculated. Derivative by term α is given in formula (29) and derivative by term β is given in formula (30).

$$\begin{aligned}&\frac{\partial D_{MAE}}{\partial \alpha} = \frac{1}{2^{b+2}} + 0 - 0 + \frac{\partial}{\partial \alpha}\int_{\alpha}^{\infty} xf(x)dx - \frac{\partial}{\partial \alpha}\alpha\big(1-F(\alpha)\big) = \Big[\frac{\partial}{\partial \alpha}\int_{\alpha}^{\infty} xf(x)dx = \\ &-\alpha f(\alpha); \big(u(x)v(x)\big)' = u'(x)v(x) + u(x)v'(x)\Big] = \frac{1}{2^{b+2}} - \alpha f(\alpha) - \Big(1\cdot\big(1-F(\alpha)\big) + \\ &\alpha\cdot\big(-f(x)\big)\Big) = \frac{1}{2^{b+2}} - \alpha f(\alpha) - 1 + F(\alpha) + \alpha f(\alpha) = \frac{1}{2^{b+2}} - 1 + F(\alpha)\end{aligned} \tag{29}$$

$$\begin{aligned}&\frac{\partial D_{MAE}}{\partial \beta} = -\frac{1}{2^{b+2}} + \frac{\partial}{\partial \beta}\beta F(\beta) - \frac{\partial}{\partial \beta}\int_{-\infty}^{\beta} x\, f(x)dx + 0 - 0 = \Big[\frac{\partial}{\partial \beta}\int_{\alpha}^{\infty} xf(x)dx = \\ &\beta f(\beta); \big(u(x)v(x)\big)' = u'(x)v(x) + u(x)v'(x)\Big] == -\frac{1}{2^{b+2}} + 1\cdot F(\beta) + \beta\cdot f(\beta) - \\ &\beta f(\beta) = -\frac{1}{2^{b+2}} + F(\beta)\end{aligned} \tag{30}$$

To find the minimum quantization MAE by each clipping bound, derivatives are equated to zero in formulas (31) and (33). The optimal clipping bounds for the asymmetric quantization scheme are given by formulas (32) and (34).

$$\frac{\partial D_{MAE}}{\partial \alpha_*} = \frac{1}{2^{b+2}} - 1 + F(\alpha_*) = 0 \tag{31}$$

$$\alpha_* = F^{-1}\left(1 - \frac{1}{2^{b+2}}\right) \tag{32}$$

$$\frac{\partial D_{MAE}}{\partial \beta_*} = -\frac{1}{2^{b+2}} + F(\beta_*) = 0 \tag{33}$$

$$\beta_* = F^{-1}\left(\frac{1}{2^{b+2}}\right) \tag{34}$$

For the symmetric quantization case β is substituted with negative α in simplified D_{MAE} formula (28) and the result is denoted by formula (35). A similar process for finding optimal clipping bounds is applied as in the asymmetric quantization scheme. D_{MAE} The derivative by clipping bound term is calculated in formula (36), equated to zero in formula (37), and the optimal symmetric clipping bound is denoted by formula (38).

$$D_{MAE} = \frac{\alpha}{2^{b+1}} - \alpha F(-\alpha) - \int_{-\infty}^{-\alpha} x\, f(x)dx + \int_{\alpha}^{\infty} x\, f(x)dx - \alpha(1 - F(\alpha)) \tag{35}$$

$$\frac{\partial D_{MAE}}{\partial \alpha} = \frac{1}{2^{b+1}} - \left(1 \cdot F(-\alpha) + \alpha \cdot \left(-f(-\alpha)\right) - \alpha f(-\alpha) - \alpha f(\alpha) - \left(1 \cdot \left(1 - F(\alpha)\right) + \alpha \cdot \left(-f(x)\right)\right)\right) = \frac{1}{2^{b+1}} - F(-\alpha) + \alpha f(-\alpha) - \alpha f(-\alpha) - \alpha f(\alpha) - 1 + F(\alpha) + \alpha f(\alpha) = \frac{1}{2^{b+1}} - F(-\alpha) - 1 + F(\alpha) = F(\alpha) - F(-\alpha) - 1 + \frac{1}{2^{b+1}} \tag{36}$$

$$\frac{\partial D_{MAE}}{\partial \alpha_*} = F(\alpha_*) - F(-\alpha_*) - 1 + \frac{1}{2^{b+1}} = 0 \tag{37}$$

$$F(\alpha_*) - F(-\alpha_*) = 1 - \frac{1}{2^{b+1}} \tag{38}$$

For probability distributions that are symmetric around zero, the cumulative distribution function satisfies F(−x)=1−F(x). Using this property, the formula (38) simplifies to formula (42) with intermediary steps shown in formulas (39), (40), and (41).

$$F(\alpha_*) - (1 - F(\alpha_*)) = 1 - \frac{1}{2^{b+1}} \tag{39}$$

$$2F(\alpha_*) = 2 - \frac{1}{2^{b+1}} \tag{40}$$

$$F(\alpha_*) = 1 - \frac{1}{2^{b+2}} \tag{41}$$

$$\alpha_* = F^{-1}\left(1 - \frac{1}{2^{b+2}}\right) \tag{42}$$

However, to preserve generality, no symmetry assumption on the distribution is imposed. Therefore, in the general case, the symmetric optimal clipping value α_* must be obtained by numerically solving (38).

2.2. Distributions and distribution fitting

The optimal clipping condition derived in section 2.1.4 requires an explicit CDF function F(x). To obtain a tractable F(x) of empirical weight tensors, a parametric probability distribution is fitted to each quantization segment, and its closed form CDF is substituted into the optimality condition.

The choice of the parametric family has a direct effect on the clipping bound. A family that overestimates tail mass will place the clipping bound α_* too far from zero, underutilizing the available quantization levels. While a family that underestimates it will clip too aggressively, discarding the

weight values that carry the signal. Four candidate families are considered: Gaussian, Laplace, Student`s t, and generalized Gaussian.

Each family is fitted independently to the neural network weights array. Maximum likelihood estimation is used to determine distributions parameters. The family with the highest log-likelihood is selected for substitution into the optimal clipping bounds expression.

2.2.1. Likelihood

The likelihood quantifies how plausible a set of distribution parameters is, given an observed sample. For a sample $\{x_1, \dots, x_N\}$ drawn independently from a distribution with a probability density function f(x;θ), the likelihood is given by formula (43) [23]. The parameter value that maximizes the likelihood is the maximum likelihood estimate of the distribution parameters θ [24].

$$L(\theta; x_1, \dots, x_N) = \prod_{i=1}^{N} f\,(x_i; \theta) \tag{43}$$

The natural logarithm of the likelihood is called the log-likelihood and is defined by formula (44). It is easier to work with, because it represents not the product of density multiplications, but the sum of densities' logarithms [24].

$$l(\theta; x_1, \dots, x_N) = \sum_{i=1}^{N} \ln\, f(x_i; \theta) \tag{44}$$

2.2.2. Gaussian distribution

The Gaussian distribution is parameterized by location μ and scale σ. The probability density function is given by formula (45) [25].

$$f(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{45}$$

For a sample $\{x_1, \dots, x_N\}$ drawn independently from the Gaussian distribution, the maximum likelihood estimators of μ and σ are the sample mean and the sample standard deviation [25].

2.2.3. Laplace distribution

The Laplace distribution is parameterized by location μ and scale b. The probability density function is given by formula (46) [25].

$$f(x; \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right) \tag{46}$$

For a sample $\{x_1, \dots, x_N\}$ drawn independently from the Laplace distribution, the maximum likelihood estimators of μ and b are the sample median and the mean absolute deviation from the median [25].

2.2.4. Student`s t distribution

The Student`s t distribution is parameterized by degrees of freedom ν, location μ, and scale σ. The probability density function is given by formula (47). The probability density function uses the gamma function defined by formula (48) [25].

$$f(x;\nu,\mu,\sigma) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\nu\pi}\,\sigma}\left(1+\frac{1}{\nu}\left(\frac{x-\mu}{\sigma}\right)^2\right)^{-\frac{\nu+1}{2}} \tag{47}$$

$$\Gamma(c) = \int_0^\infty u^{c-1}\exp(-u)\,du \tag{48}$$

For a sample $\{x_1,\ldots,x_N\}$ drawn independently from the Student`s t distribution, the parameters ν, μ, and σ are estimated by maximum likelihood. The estimates are obtained numerically by maximizing the log-likelihood [26].

2.2.5. Generalized Gaussian distribution

The generalized Gaussian distribution is parameterized by shape s, location μ, and scale σ. The probability density function is given by formula (49) [27]. The probability density function uses the gamma function defined by formula (48) [25, 27].

$$f(x;s,\mu,\sigma) = \frac{s}{2\sigma\,\Gamma(1/s)}\exp\left(-\left(\frac{|x-\mu|}{\sigma}\right)^s\right) \tag{49}$$

For a sample $\{x_1,\ldots,x_N\}$ drawn independently from the generalized Gaussian distribution, the parameters s, μ, and σ can be estimated by maximum likelihood. In general, the estimates are obtained by solving the likelihood equations for the location, scale, and shape parameters, typically using numerical methods [27].

2.3. Quantization specification

The integer quantizer is applied only to the weight tensors of convolutional and fully connected layers. Biases and activations are kept in floating point format. Each weight tensor is mapped to 4-bit or 8-bit integer levels through symmetric uniform quantization with a clipping range [−α,α] and scale given by formula (10). Symmetric quantization is the standard choice for weights quantization as covered in section 1.2.4.

Two per-channel and per-layer quantization granularities are evaluated, which were described in section 1.2.5. Both granularities operate on the batch normalized weights produced as described in section 1.1.5.

For each granularity, two strategies for choosing the clipping bound are compared. MinMax sets α, which is a maximum value across the weights slice. ACIQ first fits the four candidate distributions described in section 2.2 to the same slice, selects the candidate with the highest log-likelihood, and computes the MAE optimal symmetric clipping bound for the selected fitted distribution using the optimality condition derived in section 2.1.4, formula (38).

2.4. Performance evaluation

Methods were evaluated using two metrics: MAE between the original and dequantized weights, and top-1 accuracy on the ImageNet or MNIST dataset. Additionaly, for qualitative insights, several images were evaluated with class activation mapping (CAM) [28].

ResNet18 and custom convolutional neural network (CNN) were used as experiments subjects. ResNet18 was evaluated using the ImageNet dataset and custom CNN was evaluated using the MNIST dataset.

2.4.1. Data pre-processing

For any use case of ImageNet dataset, the image is resized so its shorter side is 256 pixels, center cropped to 224×224, and normalized per-channel with the standard ImageNet mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225).

MNIST dataset inputs are scaled to $[0,1]$ and standardized with a mean 0.1307 and standard deviation 0.3081. Such pre-processing was used not only for training but also for all other use cases requiring inference with this model.

2.4.2. Mean absolute error

Across all convolutional and linear layers, the total MAE between the floating point 32 weight tensor x and its dequantized version $\hat{x}$ is calculated according to the formula (50).

$$\mathrm{MAE}(W) = \frac{1}{N}\sum_{i=1}^{N}|\mathrm{x} - \hat{x}| \tag{50}$$

here, N – number of weights parameters in neural network.

2.4.3. Classification accuracy

Classification accuracy was evaluated as top-1 accuracy and calculated using the formula (51). This metric was used to evaluate classification accuracy on the ImageNet and MNIST validation sets.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}\,[y_i = \hat{y}_i\,] \tag{51}$$

here, N – the validation set size; y – image ground truth class; $\hat{y}$ – class prediction, which has the highest logit score; $\mathbb{1}[\cdot]$ – an operator that produces 1 if the condition is met, else 0.

2.4.4. Class activation mapping

CAM attaches a coarse spatial localisation map to a convolutional classifier with no architectural change beyond reading the last feature map and the head's class specific weights. The map highlights the spatial regions of the input that most strongly support the predicted class [28].

With X_{Latent} the last convolutional feature map before global average pooling and W_B the output neuron's weight vector, the CAM matrix is the dot product, defined as formula (52).

$$X_{\mathrm{CAM}} = X_{\mathrm{Latent}} \cdot W_B. \tag{52}$$

The same construction generalises to a multi class head by selecting the weight row of the predicted class [28].The resulting map carries the spatial resolution of the last feature tensor. It is bilinearly upsampled to the input resolution and rendered as a colour mapped layer over the original image.

2.4.5. Custom CNN

The custom CNN was designed for classification of MNIST digits. Neural network has four convolutional blocks followed by a linear classifier. Each block applies a 3×3 convolution, batch normalization, and a ReLU activation. Output channel widths are 32, 64, 64, and 128 across the four blocks. The second and fourth convolutions use a stride of 2. This halves the spatial resolution from 28×28 to 14×14 and then to 7×7. The other two convolutions preserve spatial size. A global average pooling layer reduces the final 128×7×7 feature map to a 128 dimensional vector. A single fully connected layer maps this vector to ten class logits. The architecture is shown in figure 7.

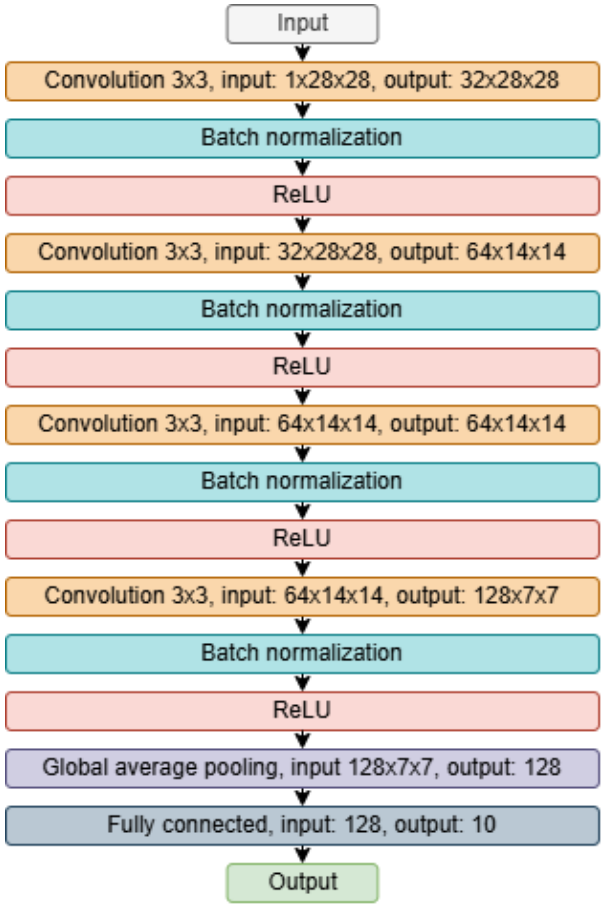

Fig. 7. Custom CNN architecture scheme with layer input and output shapes

2.5. Statistical significance evaluation

A set of 500 independently trained custom CNN variants was used for three statistical evaluations. First, the models were used to compare the classification accuracy obtained with the classical MinMax clipping method and the analytical clipping method. Second, they were used to assess the relationship between layer output distribution mean shifts and quantization induced accuracy degradation. Third, they were used to test whether bias correction produced a statistically significant improvement in quantized model accuracy.

The relationship between output mean shifts and quantization accuracy was evaluated using the Spearman correlation coefficient. For each trained model, layer output distribution statistics were collected before and after quantization. The total output mean shift was then compared with the corresponding decrease in classification accuracy. Since convolutional layer activations can have different distributions across output channels, the mean shift was first evaluated separately for each

output channel. This is important because a single layer level mean could hide channel level effects – positive shifts in some channels and negative shifts in other channels may cancel each other out.

2.5.1. Training

CNNs were trained from distinct random seeds. Each model was trained for fifteen epochs. The loss function is sparse categorical cross entropy, and the optimization algorithm is AdamW with hyperparameters β_1=0.9, β_2=0.999, ε=10^{-8} [29]. Other training parameters: learning rate 10^{-3}, weight decay 0.01, and mini batch size 512.

2.5.2. Wilcoxon signed-rank test

The paired Wilcoxon signed-rank test was used to evaluate whether two quantization variants produce statistically different results under the same experimental conditions. The test is appropriate because each comparison is paired by the same random seed and model. For each pair, the difference between the compared metric values is calculated by formula (53) [30].

$$d_i = x_i - y_i, i = 1, \dots, n \tag{53}$$

here x_i and y_i are the compared metric values for the same experimental run.

The null hypothesis states that the paired differences are centered at zero. The two-sided alternative is used because the test is intended to detect a systematic difference in either direction. The hypotheses are given by formulas (54) and (55) [30].

$$H_0: \text{median } (d_i) = 0 \tag{54}$$

$$H_1: \text{median } (d_i) \neq 0 \tag{55}$$

The Wilcoxon signed-rank test replaces the numerical differences with ranks. This was originally proposed by Wilcoxon for paired comparisons, where ranks are assigned to the magnitudes of the differences and signs are then attached according to the direction of the original differences. Zero differences are removed before ranking [30].

Let r_i denote the rank of $|d_i|$. If equal absolute differences occur, average ranks are assigned. The positive and negative signed-rank sums are calculated by formulas (56) and (57) [30].

$$W^+ = \sum_{d_i>0} r_i \tag{56}$$

$$W^- = \sum_{d_i<0} r_i \tag{57}$$

For a two-sided test, the test statistic is the smaller of the two signed-rank sums, as shown in formula (58) [30].

$$W = \min (W^+, W^-) \tag{58}$$

Under the null hypothesis, the signs of the nonzero paired differences are equally likely to be positive or negative. The exact p-value is obtained by considering all possible sign assignments to the nonzero ranks. If m is the number of nonzero differences and $\mathcal{S}$ is the set of all 2^m sign assignments to the array of absolute differences, then the two-sided exact p-value is calculated by formula (59) [30].

$$p = \frac{1}{2^m} \sum_{s \in \mathcal{S}} \mathbb{1}\{W(s) \leq W_{\text{obs}}\} \quad (59)$$

here W_{obs} − the observed value of the Wilcoxon statistic, W(s) − the statistic obtained under sign assignment s and $\mathbb{1}\{W(s) \leq W_{\text{obs}}\} = \begin{cases} 1, & W(s) \leq W_{\text{obs}}, \\ 0, & W(s) > W_{\text{obs}}. \end{cases}$

In this work, the number of paired observations is n=500. Exact enumeration of all sign assignments is therefore unnecessary and computationally impractical. For this reason, p-values were computed using the asymptotic large sample approximation of the Wilcoxon signed-rank statistic [31].

The p-value was interpreted at the significance level α=0.001. If p<α, the null hypothesis was rejected, indicating a statistically significant paired difference between the compared quantization variants. The direction and practical size of the effect were evaluated separately using the median paired difference.

2.5.3. Spearman correlation

The Spearman rank correlation coefficient was used to evaluate whether larger output mean shifts tend to occur together with larger drops in classification accuracy. The method compares the order of observations rather than their raw numerical values. Therefore, it is suitable when the relationship is expected to be monotonic but not necessarily linear. In this work, each observation corresponds to one trained CNN. For the i-th model, x_i denotes the total output mean shift after quantization, and y_i denotes the corresponding accuracy drop.

Let R_i be the rank of x_i, and let S_i be the rank of y_i, for i=1,…,n. In this work, the number of observations is n=500, because the relationship is evaluated across 500 independently trained CNNs. If there are no tied ranks, the same coefficient can also be computed from the rank differences, as shown in formulas (60) and (61) [32].

$$d_i = R_i - S_i \quad (60)$$

$$\rho_s = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n^3 - n} \quad (61)$$

here d_i– difference between the two ranks of the i-th model.

The value of ρ_s lies between −1 and 1. A value close to 1 means that models with larger output mean shifts also tend to have larger accuracy drops. A value close to −1 means that larger shifts tend to correspond to smaller accuracy drops. A value close to 0 indicates no clear monotonic relationship between the two variables.

The null hypothesis states that there is no rank correlation between total output mean shift and accuracy drop. The two-sided alternative is used because the test evaluates whether a monotonic relationship is present in either direction. The hypotheses are given by formulas (62) and (63).

$$H_0: \rho_s = 0 \quad (62)$$

$$H_1: \rho_s \neq 0 \quad (63)$$

For large samples, the statistical significance of the observed Spearman coefficient can be tested using Student's t approximation. The test statistic is calculated according to formula (64) [32].

$$t = \rho_s \sqrt{\frac{n-2}{1-\rho_s^2}} \tag{64}$$

Under the null hypothesis, this statistic is approximately distributed according to Student's t distribution with n-2 degrees of freedom. The two-sided p-value is calculated by formula (65) [32].

$$p = 2P(T_{n-2} \geq | \, t \, |) \tag{65}$$

here T_{n-2} – Student's t distributed random variable with n-2 degrees of freedom.

The p-value was interpreted at the significance level $\alpha=0.001$. If $p<\alpha$, the null hypothesis was rejected, indicating a statistically significant rank correlation between total output mean shift and accuracy drop. The sign and magnitude of ρ_s were then used to interpret the direction and strength of this relationship.

2.6. Implementation resources

The Python programming language[1] was used to implement the discussed methodology. The complete source code is released under an MIT license[2]. The SciPy Python library was used for statistical analysis functionalities, and some statistical analysis functionalities were re-implemented [33]. The NumPy Python library was used for array modifications, handling, and basic statistical analysis functionalities [34]. The Matplotlib Python library was used for figure plotting [35]. The Pillow Python library was used for loading images[3].

Methodology was evaluated using ResNet18 model, an image classification neural network pre-trained on ImageNet [36]. To gain a better understanding of the algorithms, the neural network was re-implemented using the tinygrad[4] deep learning library. Pre-trained weights from the PyTorch torchvision distribution were loaded into the tinygrad models without retraining [37].

[1] https://github.com/python/cpython
[2] https://github.com/kamilisjon/aciq
[3] https://github.com/python-pillow/Pillow
[4] https://github.com/tinygrad/tinygrad

3. Results

This section presents the empirical results of the proposed weight quantization methodology and its evaluation on ResNet18 and custom CNN models.

3.1. Statistical properties of layer weights distributions

Pre-trained ResNet18 weights were extracted from all convolutional and linear layers. Before extraction, batch normalization layers were fused into the preceding convolutional layers, so the analyzed tensors correspond to the weights used during quantized inference. For each layer, the empirical mean, variance, skewness, and kurtosis were computed. The resulting per-layer histograms are shown in figure 8, and the corresponding empirical moments are summarized in figure 9. The mean and variance describe the location and scale of the weight distribution, while skewness and kurtosis quantify asymmetry and tail heaviness. These statistics provide an initial assessment of how well simple parametric models, especially the Gaussian distribution, can represent trained network weights and support the subsequent choice of quantization clipping thresholds.

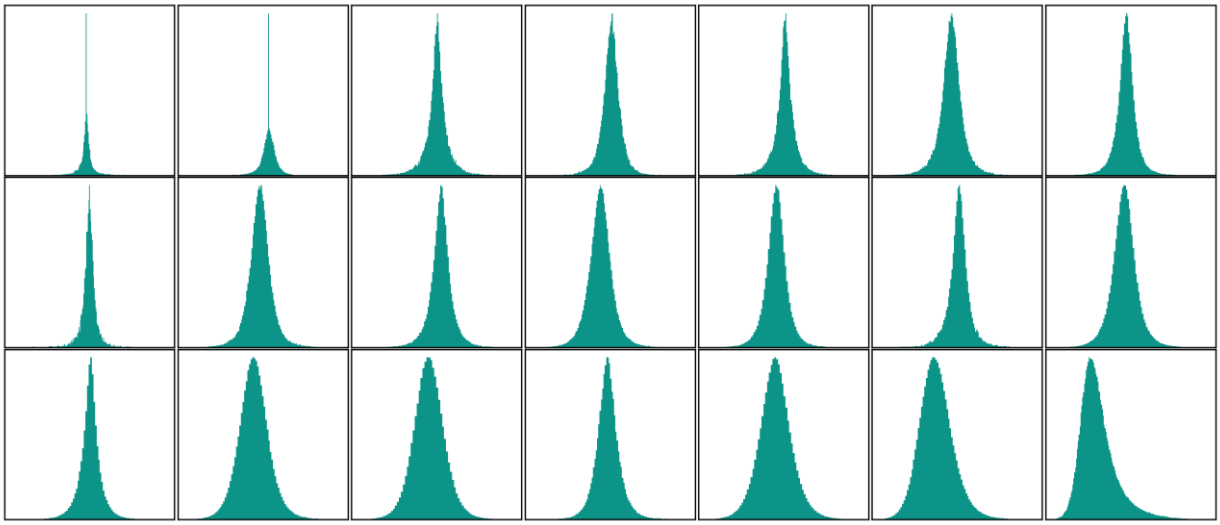

Fig. 8. Per-layer weight histograms of pre-trained ResNet18, in forward order from top left to bottom right

The empirical mean remains close to zero in all layers, with the largest absolute value equal to 3.80×10^{-3}. Twenty of the twenty-one layers have slightly negative means, while only the fully connected head has a positive mean. Since these deviations are small, a symmetric clipping interval $[-\alpha,\alpha]$ remains appropriate.

Skewness changes sign across network depth. Early convolutional layers are mostly negatively skewed, whereas later convolutional layers and the fully connected head are positively skewed. The absolute skewness exceeds 1 only at the first and last layers. In the remaining nineteen layers, it stays below 0.71. This indicates that most layer weight distributions are only moderately asymmetric. Since symmetric clipping matches zero skewness distributions most naturally, increasing skewness may cause the interval to overrepresent the lighter tail and underrepresent the heavier tail.

Kurtosis exceeds the Gaussian baseline of 3 in every layer, with the largest values observed in the early blocks. This indicates heavier tails than those of a Gaussian distribution. Higher kurtosis also implies a sharper central peak and more probability mass in the tails. Therefore, a Gaussian fit underestimates both the concentration near zero and the frequency of extreme weights.

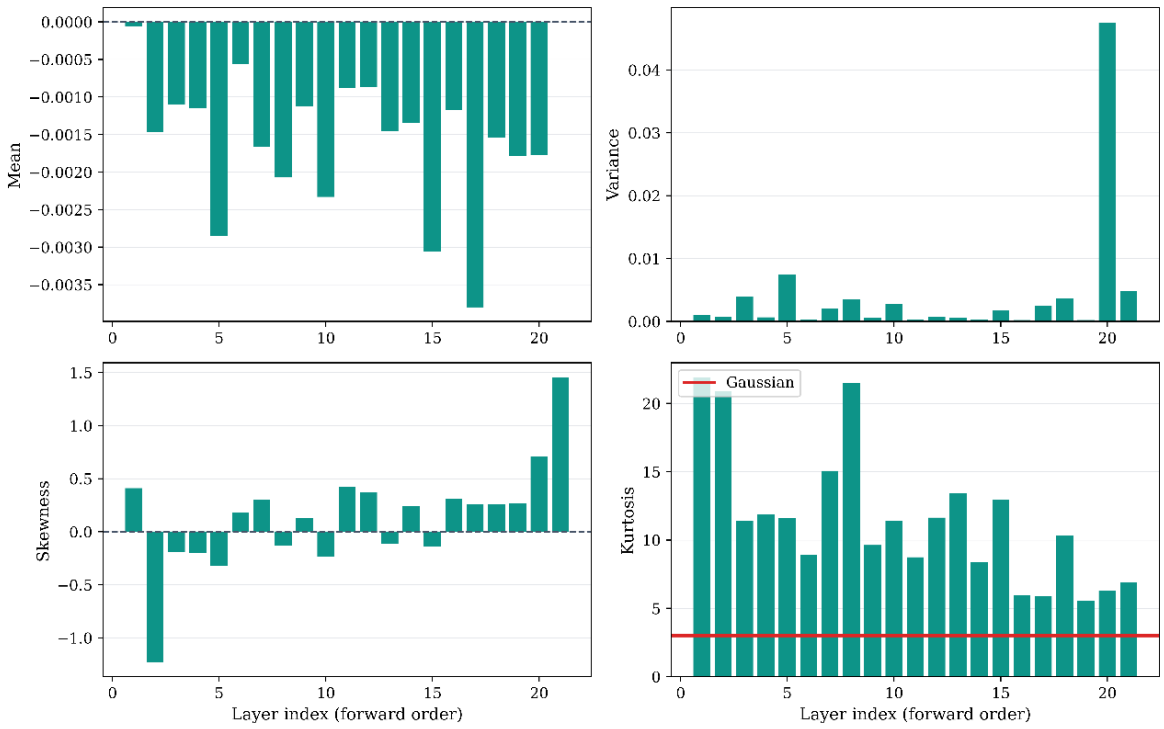

Fig. 9. Per-layer empirical moments. The red line in the kurtosis panel marks the Gaussian baseline at 3

Overall, the ResNet18 weight distributions are approximately centered but consistently heavier tailed than Gaussian distributions. A Gaussian model can represent the near zero location and approximate symmetry of many layers, but it does not capture the tail behavior. Consequently, clipping thresholds derived from a Gaussian assumption may truncate extreme weights too aggressively. Heavier tailed parametric families are therefore evaluated in Section 3.3.

3.2. Batch normalization fusion effects

During inference, batch normalization becomes a fixed affine transformation applied independently to each output channel. It can therefore be fused into the weights and biases of the preceding convolutional or linear layer. This fusion is a standard pre-processing step before post-training quantization [13].

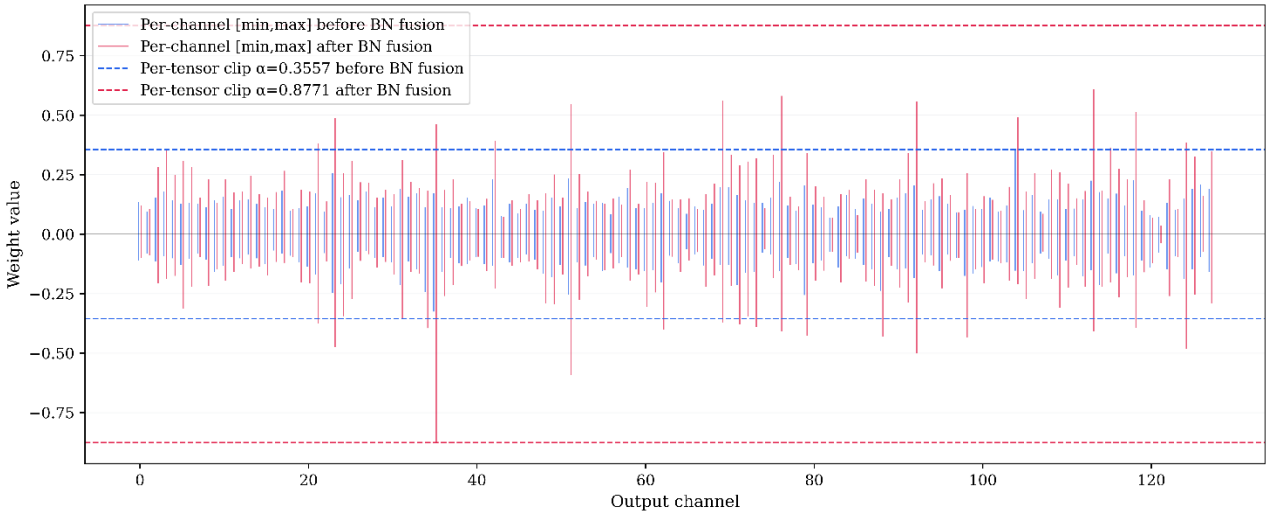

Fig. 10. Per-channel weight ranges of ResNet18 layer2.1.conv2 before and after batch normalization fusion

Each output channel is rescaled by its own batch normalization factor. After fusion, weight magnitudes therefore change independently across channels. Figure 10 illustrates this effect for one ResNet18 layer by showing the spread of per-channel weight ranges before and after fusion. Under per-layer quantization, the symmetric clipping bound is determined by the channel with the largest post-fusion range. As a result, channels with smaller ranges use only a limited part of the available quantization interval.

3.3. Weights distributions fitting

Each fused weight tensor was treated as a one dimensional sample. Four parametric families were fitted by maximum likelihood: Gaussian, Laplace, Student's t, and generalized Gaussian distributions. For each layer, the best fitting family was selected by the highest log-likelihood value. The analytical clipping method then selected the symmetric clipping threshold that minimizes the expected mean absolute quantization error under the fitted distribution.

The same fitting procedure was also applied at the channel level. Per-channel quantization assigns a separate clipping threshold to each output channel instead of using one threshold for the full weight tensor [18]. Its benefit depends on the extent to which channels within the same layer differ in scale and distributional shape.

3.3.1. Per-layer ResNet18 weights distributions fitting

Figure 11 shows the fitted distributions for the ResNet18 layer3.0.conv1 weights, while figure 12 presents the same comparison for the fully connected layer. In both cases, the empirical histogram is shown together with the fitted Gaussian, Laplace, Student's t, and generalized Gaussian densities. The figures also mark the MinMax and ACIQ clipping thresholds.

Despite their different roles in the network, both layers are best represented by the Student's t distribution. The Gaussian curve underestimates the sharp central peak in both cases. Since most weights are concentrated near zero, this mismatch affects the part of the distribution that contributes most to quantization error.

These two layers differ in symmetry. The layer3.0.conv1 weights are approximately balanced around zero, while the fully connected layer has a longer right tail. In both cases, the ACIQ clipping threshold lies inside the empirical weight range. The MAE optimal threshold therefore prioritizes resolution in the high density central region over preserving the most extreme tail values.

Table 1 summarizes the best fitting distribution, clipping thresholds, and 8-bit per-layer MAE results for all layers. The Student's t distribution is selected in 16 of the 21 layers, while the generalized Gaussian distribution is selected in the remaining five. Neither the Gaussian nor the Laplace distribution provides the best fit for any layer.

ACIQ reduces per-layer MAE by a factor of 2.31 on average. The largest gain is observed in layer4.0.conv2, where the MAE ratio reaches 4.33. The gain is smaller when the empirical maximum is already close to the fitted optimum. The weakest improvement is observed in layer2.0.downsample, where the gain is 1.09. Thus, the benefit of ACIQ increases when extreme weights lie far from the main mass of the distribution.

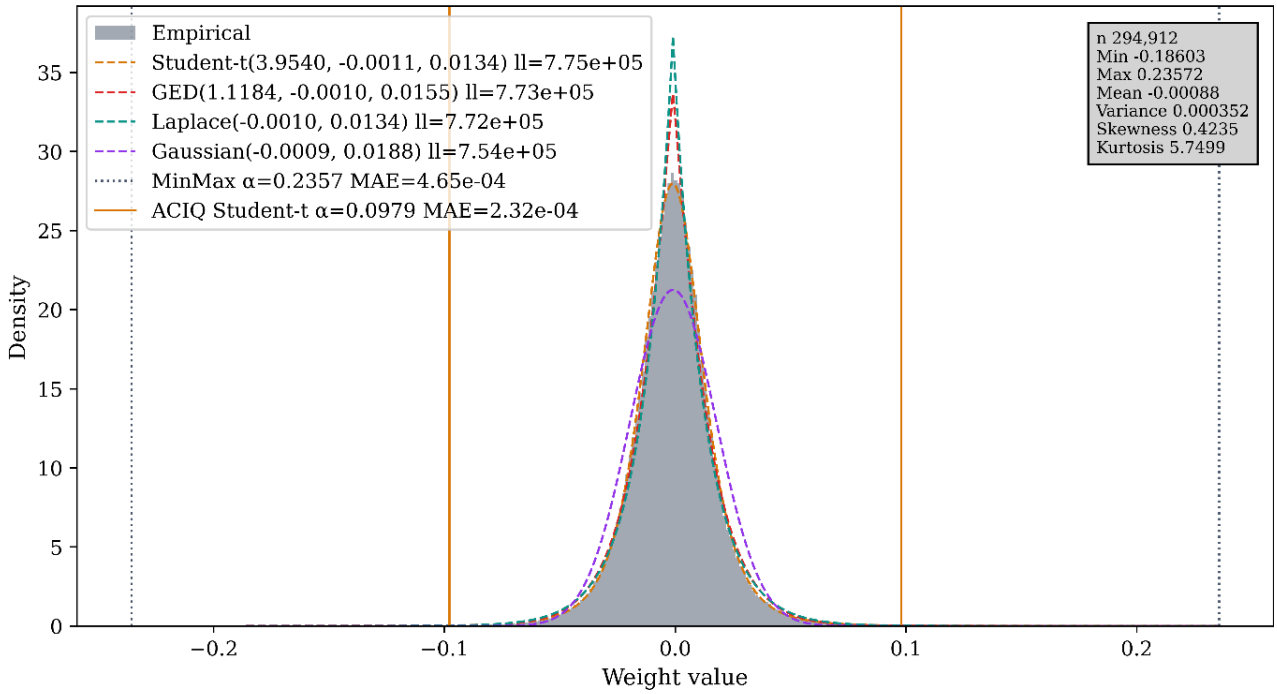

Fig. 11. Empirical histogram of ResNet18 layer3.0.conv1 weights with the four fitted distributions and the per-layer MinMax and ACIQ clip thresholds. GED – Generalized Gaussian distribution

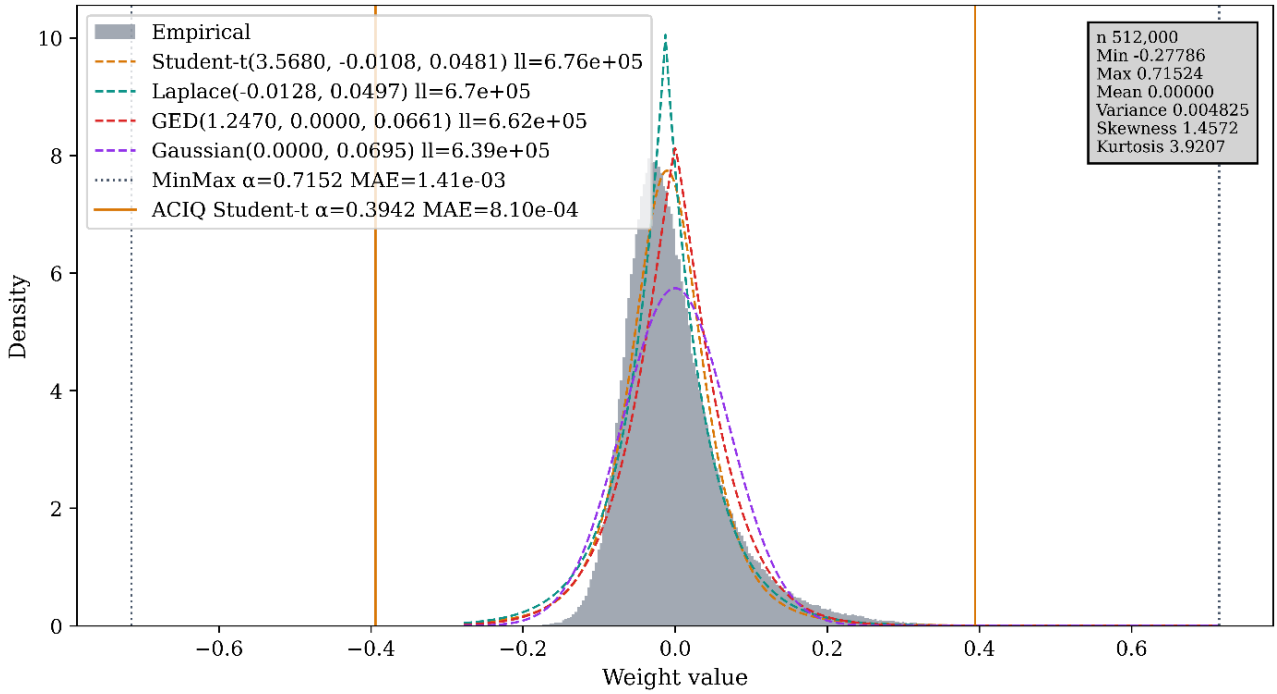

Fig. 12. Empirical histogram of fully connected layer weights with the four fitted distributions and the per-layer MinMax and ACIQ clip thresholds. GED – Generalized Gaussian distribution

Table 1. Per-layer best fit distribution and 8-bit per-layer symmetric quantization results. The Gain column is the ratio MAE_{mm}/MAE_{aciq}.

#	Layer	Fit	α_{mm}	α_{aciq}	MAE_{mm} x10^{-3}	MAE_{aciq} x10^{-3}	Gain
1	stem	Generalized Gaussian	0.394	0.394	0.674	0.674	1.00×
2	layer1.0.conv1	Generalized Gaussian	0.375	0.375	0.650	0.650	1.00×
3	layer1.0.conv2	Generalized Gaussian	0.771	0.308	1.509	0.849	1.78×
4	layer1.1.conv1	Student-t	0.280	0.159	0.550	0.368	1.49×
5	layer1.1.conv2	Generalized Gaussian	1.047	0.413	2.063	1.243	1.66×
6	layer2.0.conv1	Student-t	0.213	0.105	0.418	0.221	1.89×
7	layer2.0.conv2	Student-t	0.724	0.310	1.429	0.692	2.07×
8	layer2.0.downsample	Student-t	0.692	0.634	1.361	1.254	1.09×
9	layer2.1.conv1	Student-t	0.311	0.140	0.613	0.311	1.97×
10	layer2.1.conv2	Student-t	0.877	0.348	1.727	0.747	2.31×
11	layer3.0.conv1	Student-t	0.236	0.0979	0.465	0.232	2.00×
12	layer3.0.conv2	Student-t	0.564	0.153	1.111	0.370	3.00×
13	layer3.0.downsample	Student-t	0.408	0.190	0.801	0.397	2.02×
14	layer3.1.conv1	Student-t	0.272	0.0885	0.536	0.208	2.58×
15	layer3.1.conv2	Generalized Gaussian	0.970	0.191	1.911	0.619	3.09×
16	layer4.0.conv1	Student-t	0.302	0.0627	0.594	0.155	3.83×
17	layer4.0.conv2	Student-t	1.144	0.203	2.252	0.520	4.33×
18	layer4.0.downsample	Student-t	0.998	0.328	1.963	0.788	2.49×
19	layer4.1.conv1	Student-t	0.293	0.0597	0.577	0.142	4.06×
20	layer4.1.conv2	Student-t	3.648	0.865	7.179	2.315	3.10×
21	fully connected	Student-t	0.715	0.394	1.407	0.810	1.74×

3.3.2. Per-channel ResNet18 weights distributions fitting

For layer3.0.conv1, the best per-layer fit is the Student's t distribution. Therefore, the dashed curve in each subplot of figure 13 represents the same Student's t density fitted to the full layer. Most individual channels are also best fitted by the Student's t distribution, so their channel level densities closely overlap with the per-layer reference. A smaller number of channels are better fitted by the generalized Gaussian distribution, and their densities differ more visibly from the dashed curve. Thus, even within a single layer, individual channels may follow different distributional shapes.

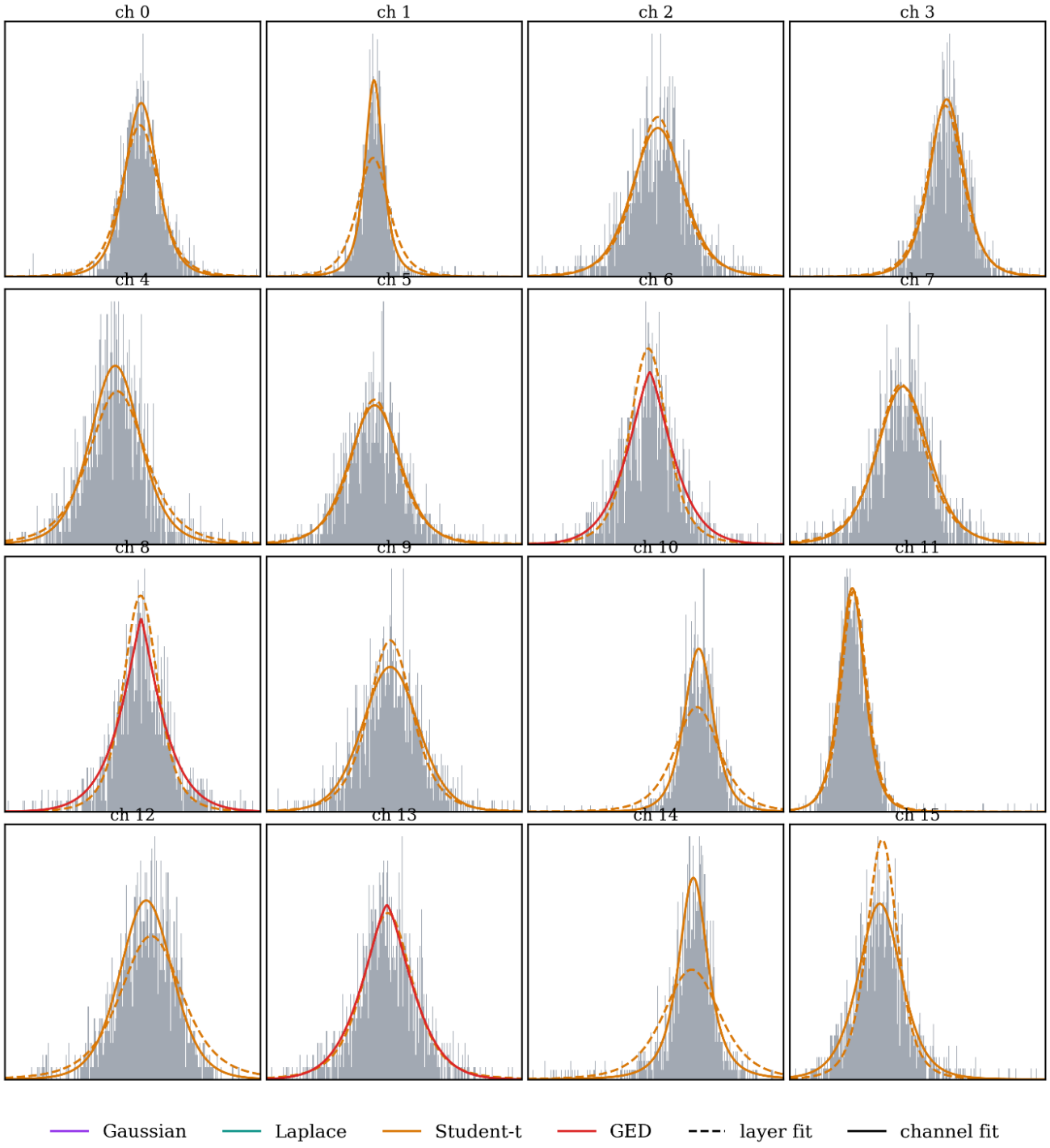

Fig. 13. First 16 output channels of layer3.0.conv1. Each subplot shows the channel weight histogram with the layer-level best-fit density drawn as a dashed line and the channel level best fit drawn as a solid line. Line colors identify the distribution family per the legend. GED – Generalized Gaussian distribution

Table 2 summarizes the per-layer best fits and the corresponding per-channel best fit counts across the 21 weight modules of ResNet18. Across all 5800 output channels, the selected per-channel distribution is always either Student's t or generalized Gaussian. Gaussian and Laplace distributions are not selected for any channel.

Table 2. Per-channel best-fit distribution counts across the 21 weight modules of ResNet18. The n_c column is the number of output channels in the module.

#	Layer	n_c	Layer fit	Gaussian	Laplace	Student-t	Generalized Gaussian
1	stem	64	Generalized Gaussian	0	0	7	57
2	layer1.0.conv1	64	Generalized Gaussian	0	0	0	64
3	layer1.0.conv2	64	Generalized Gaussian	0	0	41	23
4	layer1.1.conv1	64	Student-t	0	0	41	23
5	layer1.1.conv2	64	Generalized Gaussian	0	0	41	23
6	layer2.0.conv1	128	Student-t	0	0	79	49
7	layer2.0.conv2	128	Student-t	0	0	118	10
8	layer2.0.downsample	128	Student-t	0	0	54	74
9	layer2.1.conv1	128	Student-t	0	0	119	9
10	layer2.1.conv2	128	Student-t	0	0	102	26
11	layer3.0.conv1	256	Student-t	0	0	217	39
12	layer3.0.conv2	256	Student-t	0	0	247	9
13	layer3.0.downsample	256	Student-t	0	0	145	111
14	layer3.1.conv1	256	Student-t	0	0	242	14
15	layer3.1.conv2	256	Generalized Gaussian	0	0	213	43
16	layer4.0.conv1	512	Student-t	0	0	457	55
17	layer4.0.conv2	512	Student-t	0	0	496	16
18	layer4.0.downsample	512	Student-t	0	0	292	220
19	layer4.1.conv1	512	Student-t	0	0	457	55
20	layer4.1.conv2	512	Student-t	0	0	484	28
21	fully connected	1000	Student-t	0	0	967	33

3.3.3. Custom CNN weights distributions fitting

A custom CNN was trained on MNIST using 500 different random seeds. Figure 14 shows that the training runs converged to a similar final regime. A short increase in validation loss appears during the first 30 optimization steps, after which all curves decrease and remain below 0.2.

For the 500 trained custom CNN models, the per-layer fitting results indicate that Gaussian and Laplace distributions are rarely selected as the best fitting families. Most convolutional blocks are instead described by Student's t and generalized Gaussian distributions, as summarized in table 3. The first two blocks are dominated by the generalized Gaussian distribution, while the third block is consistently assigned to Student's t. In the fourth block, both heavy tailed families occur with similar frequency. These results confirm that the custom CNN weight tensors are not well described by simple Gaussian or Laplace models. Heavier tailed parametric families are more suitable for trained convolutional weights.

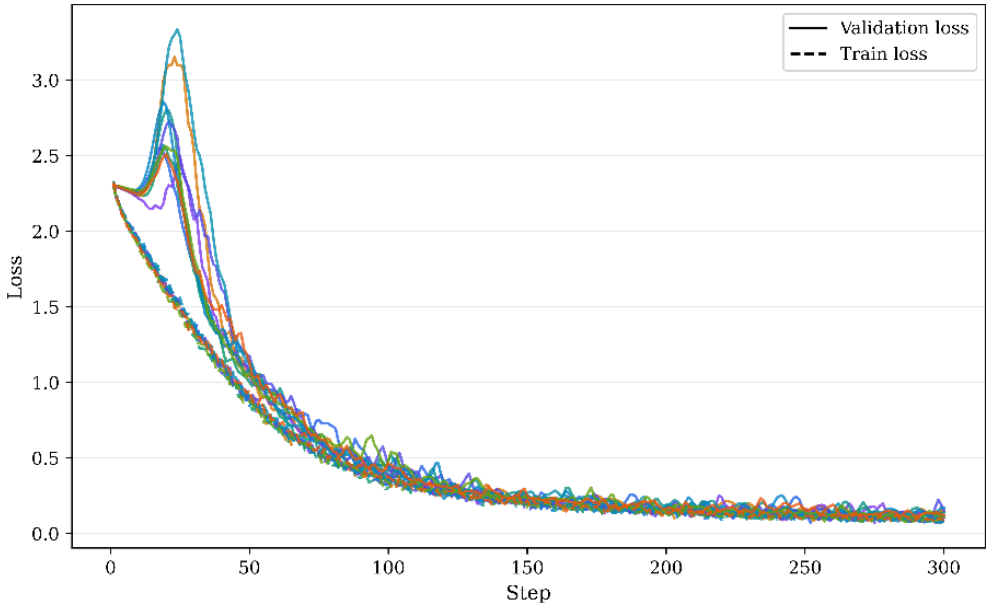

Fig. 14. First 10 MNIST CNN training runs train and validation losses

Table 3. Per-layer best fit distribution counts across the 5 weight modules of the custom CNN. Counts are given as cumulative counts over 500 runs.

#	Layer	Number of parameters	Gaussian	Laplace	Student-t	Generalized Gaussian
1	block1	288	0	0	74	426
2	block2	18432	0	0	124	376
3	block3	36864	0	0	500	0
4	block4	73728	0	0	248	252
5	fully connected	1280	0	0	0	500

At the channel level, the same pattern remains visible. When individual output channels are considered separately, the generalized Gaussian distribution becomes more dominant, as summarized in table 4. In all convolutional blocks, most channels are best described by the generalized Gaussian family, while Student's t appears as the second most common alternative. Gaussian and Laplace fits occur only rarely and mainly in the first block. The per-channel analysis therefore supports the per-layer result. Trained weights are generally heavy tailed and are better represented by flexible distribution families than by Gaussian or Laplace approximations.

Table 4. Per-channel best fit distribution counts across the 5 weight modules of the custom CNN. The n_c column is the number of output channels in the module. Counts are given as averages per-500 runs.

#	Layer	nc	Parameters per-channel	Gaussian	Laplace	Student-t	Generalized Gaussian
1	block1	32	9	0.186	0.122	0.6	31.1
2	block2	64	288	0.002	0	1.4	62.6
3	block3	64	576	0.004	0	16.2	47.8
4	block4	128	576	0.006	0	27.7	100.3
5	fully connected	10	128	0	0	0	10

3.3.4. Summary

Section 3.3 shows that trained weight tensors are consistently better described by heavy tailed distributions than by Gaussian or Laplace models. For ResNet18, the best per-layer fit is either Student's t or generalized Gaussian, and the same two families also cover all 5800 per-channel fits. The fitted distributions produce ACIQ clipping thresholds that usually lie inside the empirical MinMax range, improving resolution near the dense central part of the weight distribution. This reduces 8-bit per-layer MAE by a factor of 2.31 on average. The same tendency is observed across 500 custom CNN models trained on MNIST, where Gaussian and Laplace fits are rare and most convolutional weights are best described by Student's t or generalized Gaussian distributions.

3.4. Quantization accuracy evaluation

3.4.1. ResNet18

The practical effect of quantization on ResNet18 is evaluated on the ImageNet validation set in table 5. In the 8-bit setting, per-channel quantization preserves accuracy better than per-layer quantization for both clipping methods. The best result is obtained with per-channel MinMax quantization, where top-1 accuracy decreases only from 69.862% to 69.710%. Although ACIQ gives lower MAE than MinMax in both 8-bit settings, it causes a larger accuracy decrease. This indicates that minimizing weight reconstruction error alone does not guarantee better classification performance.

The difference is larger in the 4-bit setting. Per-layer quantization causes an almost complete accuracy collapse for both clipping methods. Per-channel quantization reduces this degradation, but only MinMax preserves a meaningful level of accuracy, reaching 44.526% top-1. ACIQ again gives lower MAE but much lower classification accuracy. Thus, the analytical clipping method improves weight approximation in terms of MAE, while MinMax is more robust with respect to the final classification task. This supports the broader conclusion that quantization error and task accuracy are related but not equivalent evaluation criteria.

Table 5. ImageNet classification accuracy comparison of quantization methods under different quantization granularity and different integer precision. Analysed model is ResNet18.

Method	Granularity	Precision	MAE x10^{-3}	Top-1	ΔTop-1
—	—	FP32	—	69.862	—
MinMax	Per-layer	INT8	2.43	**69.616**	−0.25
ACIQ	Per-layer	INT8	**0.764**	65.314	−4.55
MinMax	Per-channel	INT8	0.795	**69.710**	−0.15
ACIQ	Per-channel	INT8	**0.693**	67.562	−2.30
MinMax	Per-layer	INT4	39.4	0.124	−69.74
ACIQ	Per-layer	INT4	**7.01**	**0.454**	−69.41
MinMax	Per-channel	INT4	14.4	**44.526**	−25.37
ACIQ	Per-channel	INT4	**6.76**	2.066	−67.80

3.4.2. Custom CNN

A total of 500 custom CNNs were trained with different random seeds. For each model, distribution fitting was performed at both per-layer and per-channel granularity. The weights were then quantized

to INT8 and INT4 using the MinMax and ACIQ clipping methods. For each bit width and granularity, the MAE values of the MinMax and ACIQ variants were compared as paired observations using the Wilcoxon signed-rank test.

MAE results across the 500 trained models are summarized in table 6. ACIQ reduces MAE in most settings. The reduction is largest under 4-bit quantization, where per-layer MAE decreases from 15.344×10^{-3} with MinMax to 7.952×10^{-3} with ACIQ, and per-channel MAE decreases from 10.315×10^{-3} to 7.660×10^{-3}. The only exception is 8-bit per-channel quantization, where MinMax gives a slightly lower MAE than ACIQ. The Wilcoxon signed-rank test indicates that the MinMax and ACIQ MAE differences are statistically significant in all four settings. Thus, ACIQ generally improves weight reconstruction error, but the improvement depends on quantization granularity and bit width.

Table 6. MAE comparison of MinMax and ACIQ clipping methods on 500 trained custom CNNs under different quantization granularities and integer precisions. The p-value is computed using the paired Wilcoxon signed-rank test.

Method	Granularity	Precision	MAE $x10^{-3}$	p-value
MinMax	Per-layer	INT8	0.846	$1.26\ x10^{-83}$
ACIQ	Per-layer	INT8	0.633	
MinMax	Per-channel	INT8	0.568	$1.26\ x10^{-83}$
ACIQ	Per-channel	INT8	0.574	
MinMax	Per-layer	INT4	15.344	$1.26\ x10^{-83}$
ACIQ	Per-layer	INT4	7.952	
MinMax	Per-channel	INT4	10.315	$3.38\ x10^{-27}$
ACIQ	Per-channel	INT4	7.660	

The corresponding MNIST classification accuracies are compared in table 7. In contrast to the MAE results, lower quantization error does not consistently lead to higher classification accuracy. Under 8-bit per-layer quantization, MinMax achieves higher accuracy than ACIQ, and the paired Wilcoxon signed-rank test indicates that this difference is statistically significant. Under 8-bit per-channel quantization, both methods preserve the floating point baseline almost exactly, and the difference between them is not statistically significant. Under 4-bit quantization, per-channel quantization preserves accuracy better than per-layer quantization for both clipping methods. MinMax gives higher mean accuracy than ACIQ in both 4-bit settings, but the per-layer difference is not statistically significant. These results confirm that the clipping bound minimizing weight approximation error is not necessarily optimal for the final classification task.

Table 7. MNIST classification accuracy comparison of MinMax and ACIQ clipping methods on 500 trained custom CNNs under different quantization granularities and integer precisions. The p-value is computed using the paired Wilcoxon signed-rank test.

Method	Granularity	Precision	Top-1	p-value
—	—	FP32	97.216	—
MinMax	Per-layer	INT8	97.203	3.38×10^{-8}
ACIQ	Per-layer	INT8	97.055	
MinMax	Per-channel	INT8	97.216	0.41
ACIQ	Per-channel	INT8	97.215	
MinMax	Per-layer	INT4	84.622	0.88
ACIQ	Per-layer	INT4	83.558	
MinMax	Per-channel	INT4	94.542	1.72×10^{-5}
ACIQ	Per-channel	INT4	93.610	

3.4.3. Summary

Section 3.4 shows that ACIQ generally reduces weight reconstruction error, but this does not consistently improve classification accuracy. On ResNet18, per-channel quantization preserves ImageNet accuracy better than per-layer quantization, especially in the 4-bit setting. However, MinMax remains more robust for the final task, despite higher MAE. The same pattern appears across 500 custom CNNs trained on MNIST. ACIQ significantly reduces MAE in most bit width and granularity settings, but MinMax often gives equal or higher classification accuracy. These results confirm that quantization error and task accuracy are related but not equivalent evaluation criteria.

3.5. Output distributions mean shift after quantization

3.5.1. Custom CNN outputs mean shift

After training, each model was quantized per-layer to 4 bits using symmetric MinMax and ACIQ clipping. Output mean shift was measured for each block on the full test set. Figure 15 shows the across seed mean and standard deviation of the per-block mean shift. The shift increases with depth. Under MinMax, the block4 mean shift is approximately three times larger than the block3 shift and more than six times larger than the block1 shift. ACIQ reduces the mean shift in every block. The across seed variability also increases in deeper blocks.

Figure 16 relates the per-block mean shift to the accuracy drop across 500 random seeds. The accuracy drop is defined as the difference between floating point and quantized accuracy. Spearman's rank correlation was used to evaluate this relationship. For 4-bit quantization, the total mean shift has a statistically significant positive correlation with the accuracy drop. The correlation is $\rho=0.707$ for MinMax and $\rho=0.761$ for ACIQ.

The same correlation analysis was repeated for 8-bit quantization to determine whether the relationship also appears in the near lossless quantization regime. In contrast, no statistically significant correlation is observed for 8-bit quantization, as shown in figure 17.

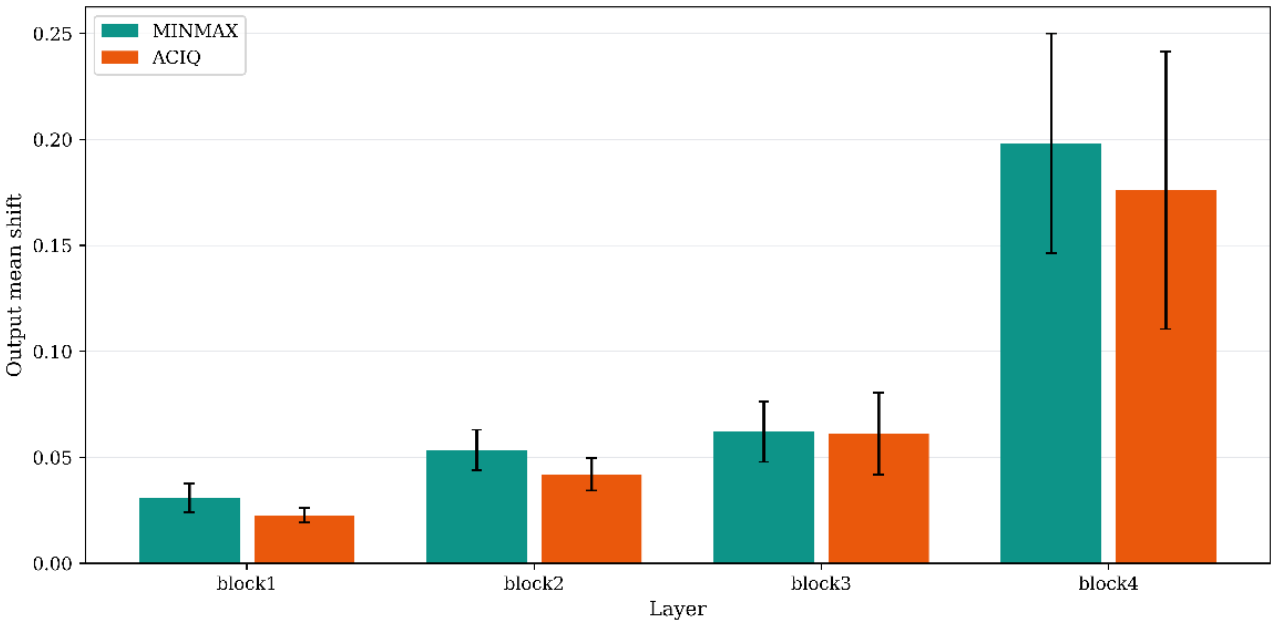

Fig. 15. Mean and standard deviation of the per-block output mean shift across 500 trained models (INT4)

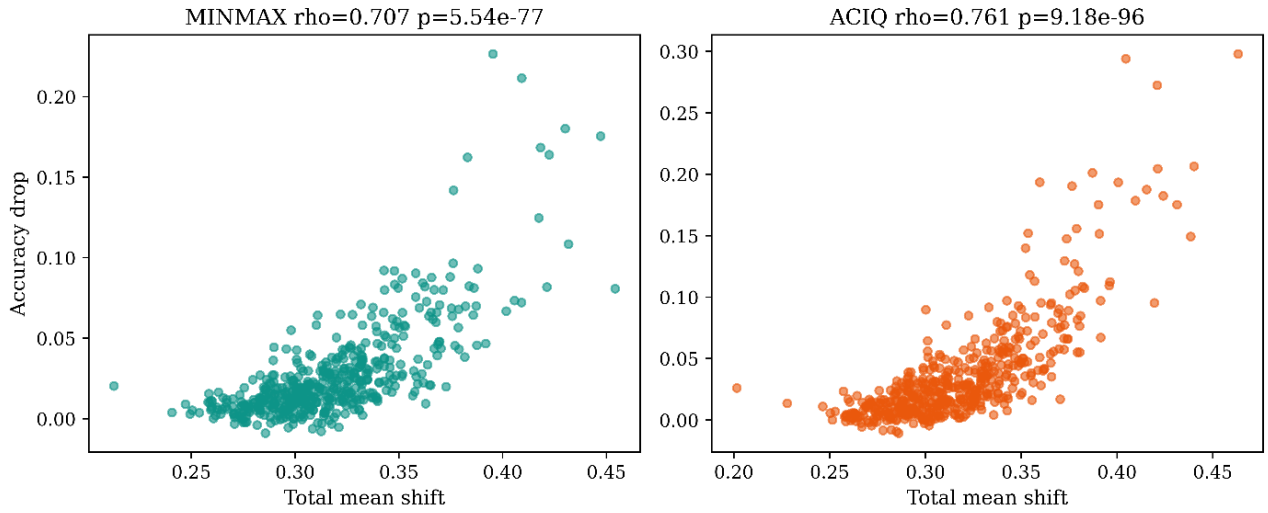

Fig. 16. Per-block and total mean shift against accuracy drop across 500 trained models (INT4)

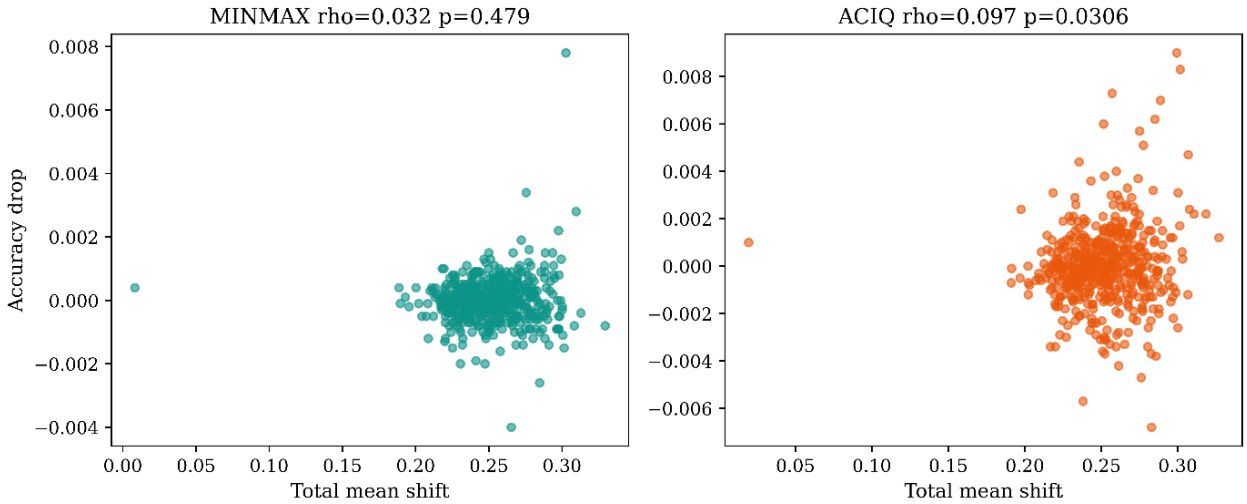

Fig. 17. Per-block and total mean shift against accuracy drop across 500 trained models (INT8)

3.5.2. Custom CNN bias correction

Bias correction on the custom CNN models is evaluated in table 8. For 8-bit MinMax quantization, the correction has no visible effect on mean top-1 accuracy, and the paired Wilcoxon signed-rank test does not indicate a statistically significant difference. For 8-bit ACIQ quantization, the correction slightly decreases accuracy, although the absolute change is very small. The effect is different in the 4-bit setting. Bias correction increases MinMax accuracy from 94.542% to 95.703% and ACIQ accuracy from 93.610% to 94.731%. Both 4-bit improvements are statistically significant. Thus, bias correction is most useful when quantization is aggressive enough to cause measurable accuracy degradation. In the near lossless 8-bit setting, it does not provide a practical benefit.

Table 8. Effect of bias correction on MNIST classification accuracy of quantized custom CNNs. Results compare MinMax and ACIQ clipping before and after bias correction for INT8 and INT4 quantization. The p-value is computed using the paired Wilcoxon signed-rank test.

Method	Bias correction	Precision	Top-1	p-value
Original		FP32	97.216	—
MinMax	No	INT8	97.216	0.75
MinMax	Yes	INT8	97.216	
ACIQ	No	INT8	**97.215**	0.15 x10^{-3}
ACIQ	Yes	INT8	97.184	
MinMax	No	INT4	94.542	3.38 x10^{-27}
MinMax	Yes	INT4	**95.703**	
ACIQ	No	INT4	93.610	1.57 x10^{-7}
ACIQ	Yes	INT4	**94.731**	

3.5.3. ResNet18 outputs mean shift and bias correction

ResNet18 with batch normalization fused into the preceding layers was quantized symmetrically per-output channel to 8 bits using MinMax and ACIQ clipping. For each method, the model was evaluated on the ImageNet validation set in two variants. The first used the quantized weights without further modification. The second applied bias correction to every weight layer.

Figure 18 shows the MinMax case. The largest mean shift occurs at the final activation, layer4.1.activation_2, where it decreases from 8.5×10^{-3} before correction to 2.9×10^{-3} after correction. Bias correction reduces the shift at every layer except the stem, where both values remain below 2×10^{-4} and are visually indistinguishable. At the final activation, the reduction is approximately threefold. A similar reduction is observed in the other deeper activations.

Figure 19 repeats the comparison for the ACIQ method. The y axis range is approximately ten times larger than in the MinMax case, which indicates substantially larger output mean shifts. ACIQ produces larger shifts than MinMax at every layer. The final activation reaches 7.9×10^{-2} before correction, and most of the total shift is concentrated in the deeper activations.

Bias correction reduces the shift in most layers. The same threefold or larger reduction pattern is visible at layer1.1.activation_2 and several middle block activations. The final activation is the main exception. There, the shift increases slightly from 7.9×10^{-2} to 8.8×10^{-2} after correction.

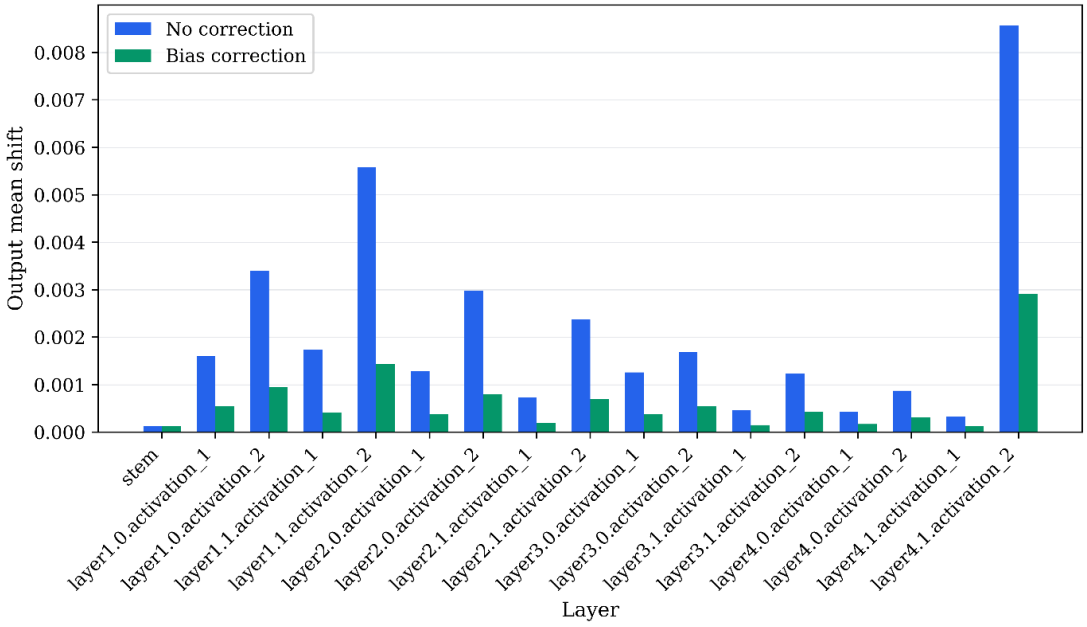

Fig. 18. Per-layer output mean shift under 8-bit per-channel MinMax weight quantization, without and with the bias correction step

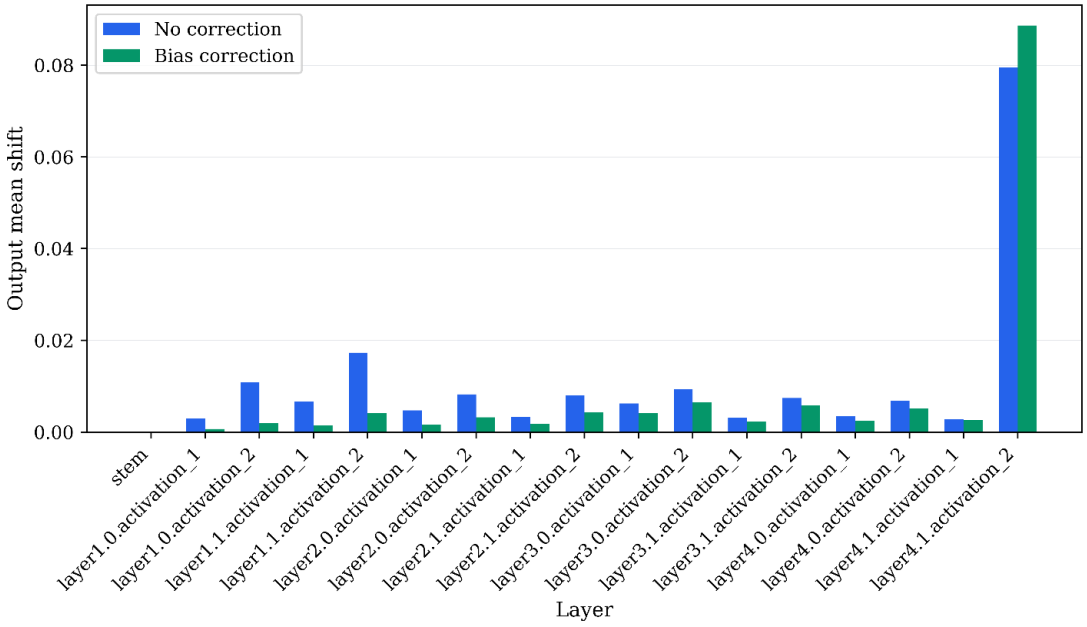

Fig. 19. Per-layer output mean shift under 8-bit per-channel ACIQ weight quantization, without and with the bias correction step

ACIQ uses a tighter clipping threshold to reduce weight MAE, but this also produces a larger output mean shift. The tighter threshold clips more of the per-channel weight tails. If the clipped tail is asymmetric, the resulting quantization error can shift the channel mean and propagate through later layers. Bias correction directly targets this effect and reduces the shift under both clipping methods. The remaining shift at the final ACIQ activation suggests that the closed form correction underestimates the effect when the clipped tail is heavy on one side. This indicates that data based calibration may be needed in such cases.

3.5.4. ResNet18 accuracy evaluation

The final ImageNet evaluation measures the effect of bias correction on the ResNet18 quantization variants analysed in the preceding section. The batch normalization layers are fused before quantization, and the full ImageNet validation set is used for evaluation. The comparison is restricted to per-channel weight quantization, because the earlier accuracy results showed that per-channel quantization preserves classification accuracy better than per-layer quantization. The floating point ResNet18 model is used as the reference. The evaluated variants are MinMax and ACIQ clipping with and without bias correction under INT8 and INT4 weight quantization. The resulting top-1 accuracies and accuracy gaps relative to the floating point model are given in table 9.

Table 9. Accuracy comparison of quantization methods.

Method	Bias correction	Precision	Top-1	ΔTop-1
Original		FP32	69.862	—
MinMax	No	INT8	69.710	−0.15
MinMax	Yes	INT8	**69.760**	−0.10
ACIQ	No	INT8	67.562	−2.30
ACIQ	Yes	INT8	**68.574**	−1.29
MinMax	No	INT4	44.526	−25.37
MinMax	Yes	INT4	**57.444**	−12.42
ACIQ	No	INT4	2.066	−67.80
ACIQ	Yes	INT4	**14.554**	−55.31

In the INT8 setting, MinMax quantization preserves the floating point accuracy almost completely. Without correction, the top-1 accuracy decreases from 69.862% to 69.710%, corresponding to a gap of 0.15 percentage points. After bias correction, the accuracy increases to 69.760%, reducing the gap to 0.10 percentage points. The improvement is small in absolute value, which indicates that the original MinMax INT8 model already has only a limited output distribution shift.

The effect is larger for ACIQ. Without correction, INT8 ACIQ reaches 67.562% top-1 accuracy, which is 2.30 percentage points below the floating point reference. After applying bias correction, the accuracy increases to 68.574%, reducing the gap to 1.29 percentage points. Thus, the correction recovers 1.01 percentage points of top-1 accuracy. This agrees with the preceding mean shift analysis: the correction has a stronger effect when the quantization method introduces a larger shift in layer output distributions.

The INT4 results show that bias correction becomes more important when the quantization error is stronger. For MinMax, the top-1 accuracy increases from 44.526% to 57.444%, reducing the gap from 25.37 to 12.42 percentage points. For ACIQ, the accuracy increases from 2.066% to 14.554%. Although this is a substantial relative improvement, the corrected ACIQ INT4 model still remains far below the floating point baseline and below the corrected MinMax INT4 model. Therefore, bias correction mitigates part of the degradation caused by aggressive quantization, but it does not fully compensate for the loss introduced by suboptimal clipping thresholds.

Overall, bias correction improves all quantized ResNet18 variants except where the uncorrected model is already close to the floating point reference. The strongest configuration remains INT8 MinMax with bias correction, reaching 69.760% top-1 accuracy and remaining within 0.10 percentage points of the original model. ACIQ benefits more from the correction, especially in the INT8 case, but it does not surpass MinMax in classification accuracy.

3.5.5. Class activations evaluation

Cosine similarity between the ResNet18 FP32 ground truth class CAM and each quantized variant's ground truth class CAM is computed at the native 7×7 resolution over all 50 000 ImageNet validation images for the four per-channel variants at 4 and 8 bits. The qualitative grid (figure 21) shows four images selected by FP32 variant prediction confidence: most confident correct (A), most confident incorrect (B), least confident correct (C), least confident incorrect (D).
Figure 20 shows that 8-bit per-channel quantization preserves CAM geometry across the full validation dataset. INT4 MinMax drops to mean 0.941 cosine similarity between CAMs and recovers to 0.969 under bias correction. INT4 ACIQ collapses to mean 0.482 and only partially recovers to 0.756 with bias correction, while individual images span the full [-1,1] range.

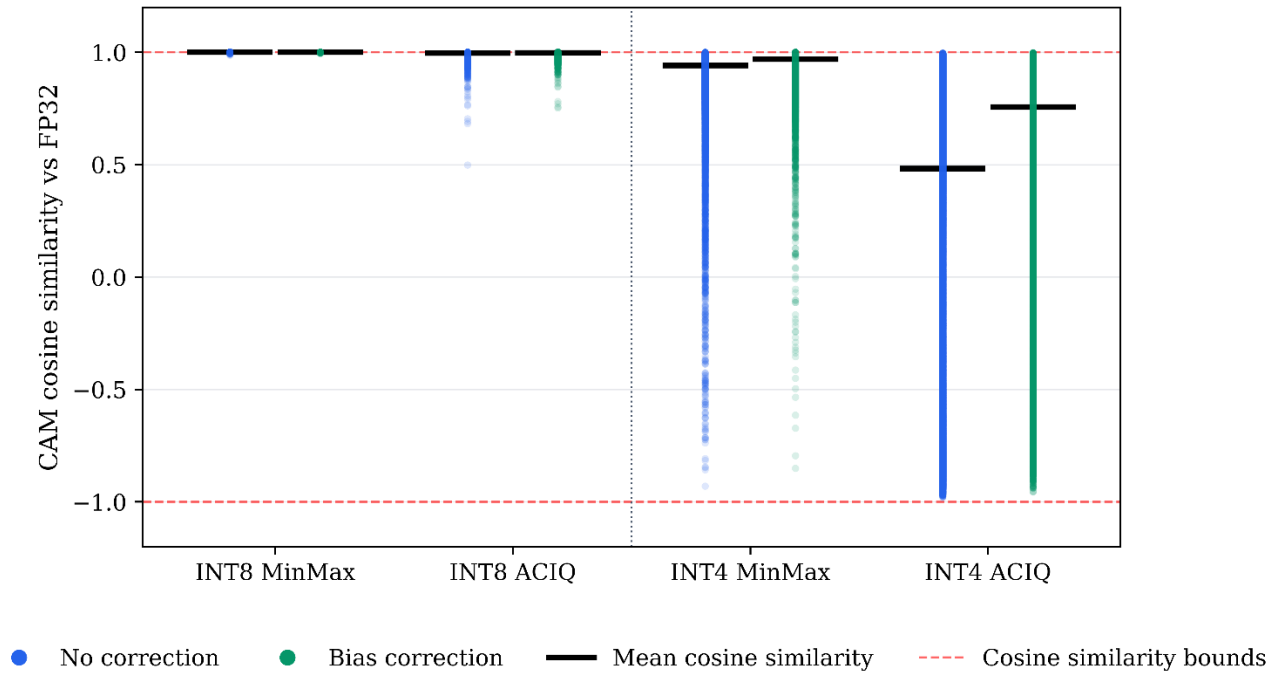

Fig. 20. Per-image CAM cosine similarity between the FP32 ground truth class CAM and each quantized variant's ground-truth-class CAM.

The qualitative grid confirms the same regime split (figure 21). Most INT8 quantized variants reproduce the FP32 decision, correct or incorrect, with CAMs visually close to FP32. Most INT4 quantized variants show noticeable heatmap displacement.
Section D's ground-truth panel is blank because the standard 224×224 centre crop excludes the labelled object. The model is evaluated on a region that does not contain the knee pad. This observations points out a limitation of standard ImageNet images prepocessing.

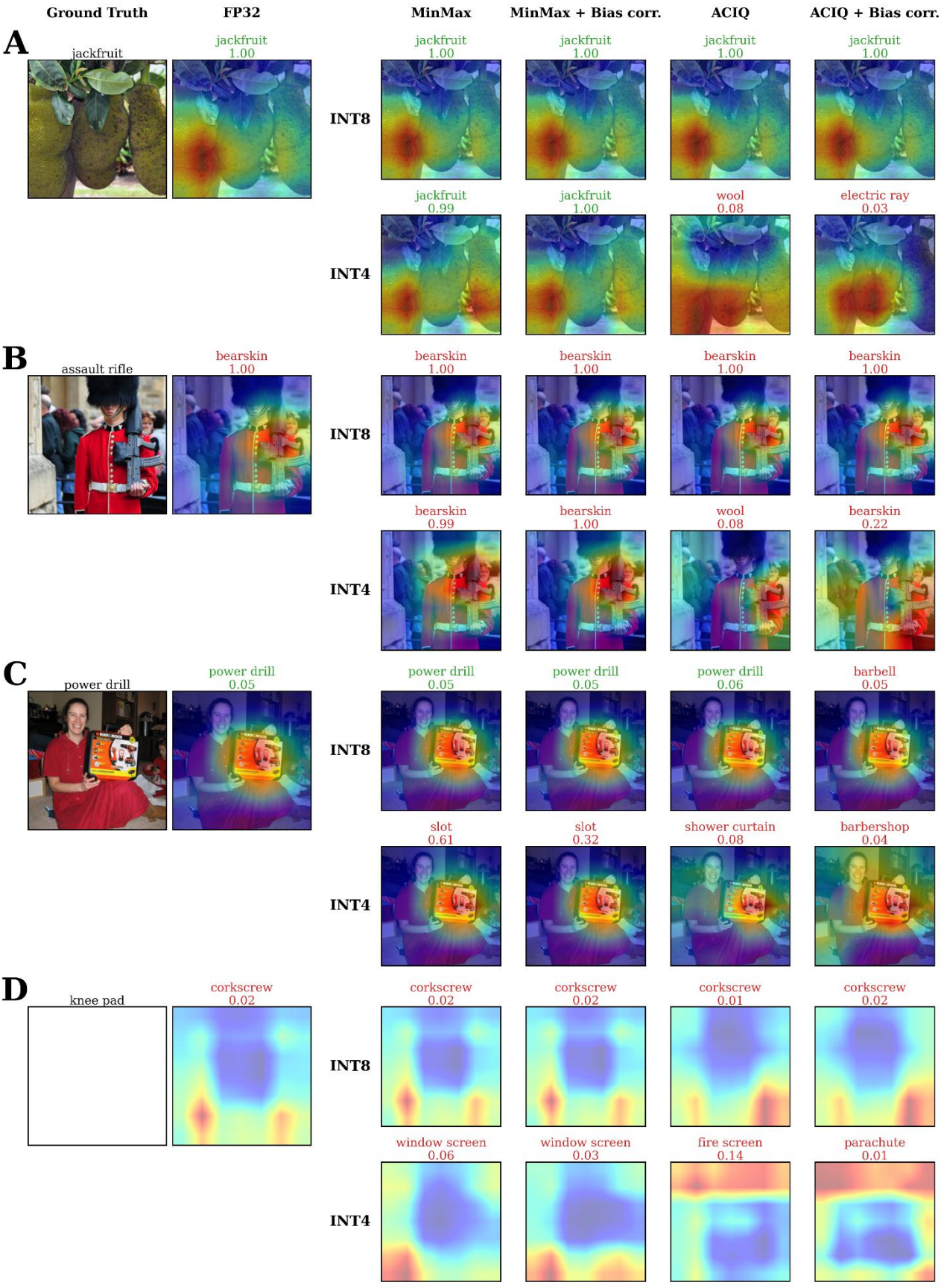

Fig. 21. Class activation maps under quantization for four ImageNet validation images. Cell titles report the variant's predicted class and softmax probability, coloured green on a correct prediction and red on a wrong one.

3.5.6. Summary

Section 3.5 shows that quantization introduces measurable shifts in layer output means and that these shifts are related to accuracy degradation. In the custom CNN experiment, 4-bit quantization produces mean shifts that increase with network depth and correlate significantly with accuracy drop across 500 random seeds.

Bias correction improves accuracy in the 4-bit setting for both MinMax and ACIQ, while it has little practical effect in the near lossless 8-bit setting. The same pattern is observed in ResNet18: bias correction reduces most output mean shifts and partially recovers ImageNet accuracy, especially for ACIQ and INT4 quantization. However, the remaining shift in the final ACIQ activation shows that closed form correction does not fully remove the effect of aggressive asymmetric tail clipping.

INT8 quantization preserves CAM geometry almost unchanged, while INT4, especially ACIQ, strongly distorts spatial feature maps. Bias correction reduces this distortion but does not fully remove it.

3.6. Summary

Section 3 shows that trained convolutional weights are approximately centered but consistently heavy tailed, which makes Student's t and generalized Gaussian distributions more suitable than Gaussian or Laplace models. The analytical clipping method reduces weight reconstruction error in most settings, but this improvement does not consistently translate into higher classification accuracy. Per-channel quantization is more robust than per-layer quantization, especially under aggressive 4-bit discretization. Across both ResNet18 and the custom CNN experiments, MinMax often preserves task accuracy better than ACIQ, despite higher MAE. The output mean shift analysis shows that quantization induced shifts increase with network depth and correlate with accuracy degradation in the 4-bit setting. Bias correction reduces these shifts and partially restores accuracy, especially when quantization error is large.

Conclusions

1. The literature review showed that post-training integer quantization is affected by clipping threshold selection, quantization granularity, batch normalization fusion, and quantization induced shifts in layer output distributions. These factors are important when comparing analytical clipping with the standard MinMax baseline.
2. A closed form condition for the mean absolute error optimal symmetric clipping threshold was derived in terms of the weight distribution cumulative distributions function. This allowed the quantization interval to be selected from a fitted parametric distribution rather than from the empirical maximum weight value.
3. Trained convolutional weights were consistently better described by heavy tailed distributions than by Gaussian or Laplace distributions. For ResNet18, the best layer level fit was either Student's t or generalized Gaussian in all 21 weight modules. At channel level, all 5800 output channels were also best fitted by one of these two families. This confirms that heavy tailed parametric models are more suitable for describing trained network weights.
4. Quantization granularity has a larger impact on 8-bit classification accuracy than the choice of clipping method. Switching from per-layer to per-channel granularity reduced the top-1 accuracy gap to the floating point baseline from 4.55% to 2.30% for the analytical method and from 0.25% to 0.15% for the MinMax method. At per-layer granularity, the analytical method achieves substantially lower mean absolute error than MinMax, 0.764×10^{-3} compared with 2.43×10^{-3}, but this advantage is largely reduced at per-channel granularity, 0.693×10^{-3} compared with 0.795×10^{-3}, where each channel obtains its own clipping scale and the heuristic estimator is no longer pulled wide by a few extreme channels. This indicates that the practical benefit of analytical clipping diminishes once per-channel scaling absorbs the cross channel variation introduced by batch normalization fusion.
5. The analytical clipping method reduced weight reconstruction error in most evaluated settings, but it did not consistently improve classification accuracy. On ResNet18, per-channel 8-bit integer quantization mean absolute error decreased from 0.795×10^{-3} with MinMax to 0.693×10^{-3} with analytical clipping, but top-1 accuracy decreased from 69.710% to 67.562%. Thus, mean absolute error optimal clipping is not necessarily accuracy optimal.
6. Quantization induced output mean shift was related to accuracy degradation under aggressive quantization. Across 500 custom CNNs, 4-bit quantization produced a statistically significant positive correlation between total output mean shift and accuracy drop, with $\rho=0.707$, $p<0.001$, for MinMax and $\rho=0.761$, $p<0.001$, for the analytical method. Bias correction also produced statistically significant ($p<0.001$) 4-bit accuracy improvements on MNIST, increasing MinMax accuracy from 94.542% to 95.703% and analytical clipping accuracy from 93.610% to 94.731%. This confirms that output mean correction is useful mainly when quantization introduces measurable accuracy degradation.

List of references

1. NAGEL, Markus, BAALEN, Mart van, BLANKEVOORT, Tijmen and WELLING, Max. *Data-Free Quantization Through Weight Equalization and Bias Correction*. Online. 25 November 2019. arXiv. arXiv:1906.04721. [Accessed 1 May 2026]. arXiv:1906.04721 [cs]

2. HAN, Song, MAO, Huizi and DALLY, William J. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. Online. 15 February 2016. arXiv. arXiv:1510.00149. [Accessed 6 May 2026]. arXiv:1510.00149 [cs]

3. JACOB, Benoit, KLIGYS, Skirmantas, CHEN, Bo, ZHU, Menglong, TANG, Matthew, HOWARD, Andrew, ADAM, Hartwig and KALENICHENKO, Dmitry. *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*. Online. 15 December 2017. arXiv. arXiv:1712.05877. [Accessed 3 May 2026]. arXiv:1712.05877 [cs]

4. HOWARD, Andrew G., ZHU, Menglong, CHEN, Bo, KALENICHENKO, Dmitry, WANG, Weijun, WEYAND, Tobias, ANDREETTO, Marco and ADAM, Hartwig. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. Online. 17 April 2017. arXiv. arXiv:1704.04861. [Accessed 3 May 2026]. arXiv:1704.04861 [cs]

5. HINTON, Geoffrey, VINYALS, Oriol and DEAN, Jeff. *Distilling the Knowledge in a Neural Network*. Online. 9 March 2015. arXiv. arXiv:1503.02531. [Accessed 3 May 2026]. arXiv:1503.02531 [stat]

6. SANH, Victor, DEBUT, Lysandre, CHAUMOND, Julien and WOLF, Thomas. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. Online. 1 March 2020. arXiv. arXiv:1910.01108. [Accessed 3 May 2026]. arXiv:1910.01108 [cs]

7. MICIKEVICIUS, Paulius, STOSIC, Dusan, BURGESS, Neil, CORNEA, Marius, DUBEY, Pradeep, GRISENTHWAITE, Richard, HA, Sangwon, HEINECKE, Alexander, JUDD, Patrick, KAMALU, John, MELLEMPUDI, Naveen, OBERMAN, Stuart, SHOEYBI, Mohammad, SIU, Michael and WU, Hao. *FP8 Formats for Deep Learning*. Online. 29 September 2022. arXiv. arXiv:2209.05433. [Accessed 3 May 2026]. arXiv:2209.05433 [cs]

8. PENG, Houwen, WU, Kan, WEI, Yixuan, ZHAO, Guoshuai, YANG, Yuxiang, LIU, Ze, XIONG, Yifan, YANG, Ziyue, NI, Bolin, HU, Jingcheng, LI, Ruihang, ZHANG, Miaosen, LI, Chen, NING, Jia, WANG, Ruizhe, ZHANG, Zheng, LIU, Shuguang, CHAU, Joe, HU, Han and CHENG, Peng. *FP8-LM: Training FP8 Large Language Models*. Online. 19 December 2023. arXiv. arXiv:2310.18313. [Accessed 3 May 2026]. arXiv:2310.18313 [cs]

9. COURBARIAUX, Matthieu, BENGIO, Yoshua and DAVID, Jean-Pierre. *BinaryConnect: Training Deep Neural Networks with binary weights during propagations*. Online. 18 April 2016. arXiv. arXiv:1511.00363. [Accessed 3 May 2026]. arXiv:1511.00363 [cs]

10. MA, Shuming, WANG, Hongyu, MA, Lingxiao, WANG, Lei, WANG, Wenhui, HUANG, Shaohan, DONG, Li, WANG, Ruiping, XUE, Jilong and WEI, Furu. *The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits*. Online. 27 February 2024. arXiv. arXiv:2402.17764. [Accessed 3 May 2026]. arXiv:2402.17764 [cs]

11. CSÉFALVAY, Szabolcs and IMBER, James. *Self-Compressing Neural Networks*. Online. 31 January 2023. arXiv. arXiv:2301.13142. [Accessed 2 May 2026]. arXiv:2301.13142 [cs]

12. IOFFE, Sergey and SZEGEDY, Christian. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. Online. 2 March 2015. arXiv. arXiv:1502.03167. [Accessed 21 March 2026]. arXiv:1502.03167 [cs]

13. WU, Hao, JUDD, Patrick, ZHANG, Xiaojie, ISAEV, Mikhail and MICIKEVICIUS, Paulius. *Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation*. Online. 20 April 2020. arXiv. arXiv:2004.09602. [Accessed 20 March 2026]. arXiv:2004.09602 [cs]

14. DING, Xiaohan, ZHANG, Xiangyu, MA, Ningning, HAN, Jungong, DING, Guiguang and SUN, Jian. *RepVGG: Making VGG-style ConvNets Great Again*. Online. 29 March 2021. arXiv. arXiv:2101.03697. [Accessed 3 May 2026]. arXiv:2101.03697 [cs]

15. CHU, Xiangxiang, LI, Liang and ZHANG, Bo. *Make RepVGG Greater Again: A Quantization-aware Approach*. Online. 11 December 2023. arXiv. arXiv:2212.01593. [Accessed 4 May 2026]. arXiv:2212.01593 [cs]

16. NAGEL, Markus, FOURNARAKIS, Marios, AMJAD, Rana Ali, BONDARENKO, Yelysei, BAALEN, Mart van and BLANKEVOORT, Tijmen. *A White Paper on Neural Network Quantization*. Online. 15 June 2021. arXiv. arXiv:2106.08295. [Accessed 3 May 2026]. arXiv:2106.08295 [cs]

17. GHOLAMI, Amir, KIM, Sehoon, DONG, Zhen, YAO, Zhewei, MAHONEY, Michael W. and KEUTZER, Kurt. *A Survey of Quantization Methods for Efficient Neural Network Inference*. Online. 21 June 2021. arXiv. arXiv:2103.13630. [Accessed 20 March 2026]. arXiv:2103.13630 [cs]

18. KRISHNAMOORTHI, Raghuraman. *Quantizing deep convolutional networks for efficient inference: A whitepaper*. Online. 21 June 2018. arXiv. arXiv:1806.08342. [Accessed 28 March 2026]. arXiv:1806.08342 [cs]

19. BANNER, Ron, NAHSHAN, Yury, HOFFER, Elad and SOUDRY, Daniel. *Post-training 4-bit quantization of convolution networks for rapid-deployment*. Online. 29 May 2019. arXiv. arXiv:1810.05723. [Accessed 20 March 2026]. arXiv:1810.05723 [cs]

20. HU, Ting, MEINEL, Christoph and YANG, Haojin. *Empirical Evaluation of Post-Training Quantization Methods for Language Tasks*. Online. 29 October 2022. arXiv. arXiv:2210.16621. [Accessed 17 May 2026]. arXiv:2210.16621 [cs.CL]

21. ZHANG, Luoming, HE, Yefei, FEI, Wen, LOU, Zhenyu, WU, Weijia, YING, YangWei and ZHOU, Hong. *Towards Accurate Post-training Quantization for Reparameterized Models*. Online. 25 February 2024. arXiv. arXiv:2402.16121. [Accessed 20 March 2026]. arXiv:2402.16121 [cs]

22. BENNETT, W. R. Spectra of Quantized Signals. *Bell System Technical Journal*. July 1948. Vol. 27, no. 3, p. 446–472. DOI 10.1002/j.1538-7305.1948.tb01340.x.

23. FISHER, R. A. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London, Series A: Containing Papers of a Mathematical or Physical Character*. 1 January 1922. Vol. 222, no. 594–604, p. 309–368. DOI 10.1098/rsta.1922.0009.

24. *Mathematical Statistics with Applications in R*. Online. Elsevier, 2015. [Accessed 10 May 2026]. ISBN 978-0-12-417113-8. DOI: 10.1016/C2012-0-07341-3

25. FORBES, Catherine, EVANS, Merran, HASTINGS, Nicholas and PEACOCK, Brian. *Statistical Distributions*. Online. 1. Wiley, 2010. [Accessed 10 May 2026]. ISBN 978-0-470-39063-4. DOI: 10.1002/9780470627242

26. LIU, Chuanhai and RUBIN, Donald B. ML ESTIMATION OF THE t DISTRIBUTION USING EM AND ITS EXTENSIONS, ECM AND ECME. *Statistica Sinica*. 1995. Vol. 5, no. 1, p. 19–39.

27. NADARAJAH, Saralees. A generalized normal distribution. *Journal of Applied Statistics*. 1 September 2005. Vol. 32, no. 7, p. 685–694. DOI 10.1080/02664760500079464.

28. AUGUSTAUSKAS, Rytis, ZABULIS, Lukas, LIPNICKAS, Arunas and JOKUBAUSKAS, Simas. Defects Localization in Images Using Deep Learning-Based Classification with CAM Output. In : . 7 September 2023. p. 487–492. DOI 10.1109/IDAACS58523.2023.10348813.

29. LOSHCHILOV, Ilya and HUTTER, Frank. *Decoupled Weight Decay Regularization*. Online. 4 January 2019. arXiv. arXiv:1711.05101. [Accessed 10 May 2026]. arXiv:1711.05101 [cs]

30. WILCOXON, Frank. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*. 1945. Vol. 1, no. 6, p. 80–83. DOI 10.2307/3001968.

31. CURETON, Edward E. The Normal Approximation to the Signed-Rank Sampling Distribution when Zero Differences are Present. *Journal of the American Statistical Association*. 1967. Vol. 62, no. 319, p. 1068–1069. DOI 10.2307/2283694.

32. ZAR, Jerrold H. Significance Testing of the Spearman Rank Correlation Coefficient. *Journal of the American Statistical Association*. 1972. Vol. 67, no. 339, p. 578–580. DOI 10.2307/2284441.

33. VIRTANEN, Pauli, GOMMERS, Ralf, OLIPHANT, Travis E., HABERLAND, Matt, REDDY, Tyler, COURNAPEAU, David, BUROVSKI, Evgeni, PETERSON, Pearu, WECKESSER, Warren, BRIGHT, Jonathan, WALT, Stéfan J. van der, BRETT, Matthew, WILSON, Joshua, MILLMAN, K. Jarrod, MAYOROV, Nikolay, NELSON, Andrew R. J., JONES, Eric, KERN, Robert, LARSON, Eric, CAREY, C. J., POLAT, İlhan, FENG, Yu, MOORE, Eric W., VANDERPLAS, Jake, LAXALDE, Denis, PERKTOLD, Josef, CIMRMAN, Robert, HENRIKSEN, Ian, QUINTERO, E. A., HARRIS, Charles R., ARCHIBALD, Anne M., RIBEIRO, Antônio H., PEDREGOSA, Fabian, MULBREGT, Paul van and CONTRIBUTORS, SciPy 1 0. SciPy 1.0--Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2 March 2020. Vol. 17, no. 3, p. 261–272. DOI 10.1038/s41592-019-0686-2. arXiv:1907.10121 [cs]

34. HARRIS, Charles R., MILLMAN, K. Jarrod, VAN DER WALT, Stéfan J., GOMMERS, Ralf, VIRTANEN, Pauli, COURNAPEAU, David, WIESER, Eric, TAYLOR, Julian, BERG, Sebastian, SMITH, Nathaniel J., KERN, Robert, PICUS, Matti, HOYER, Stephan, VAN KERKWIJK, Marten H., BRETT, Matthew, HALDANE, Allan, DEL RÍO, Jaime Fernández, WIEBE, Mark, PETERSON, Pearu, GÉRARD-MARCHANT, Pierre, SHEPPARD, Kevin, REDDY, Tyler, WECKESSER, Warren, ABBASI, Hameer, GOHLKE, Christoph and OLIPHANT, Travis E. Array programming with NumPy. *Nature*. 2020. Vol. 585, no. 7825, p. 357–362. DOI 10.1038/s41586-020-2649-2.

35. BARRETT, P., HUNTER, J., MILLER, J. T., HSU, J.-C. and GREENFIELD, P. matplotlib -- A Portable Python Plotting Package. In : *Astronomical Data Analysis Software and Systems XIV*. Online. 1 December 2005. p. 91. [Accessed 10 May 2026]. Available from: https://ui.adsabs.harvard.edu/abs/2005ASPC..347...91BADS Bibcode: 2005ASPC..347...91B

36. HE, Kaiming, ZHANG, Xiangyu, REN, Shaoqing and SUN, Jian. *Deep Residual Learning for Image Recognition*. Online. 10 December 2015. arXiv. arXiv:1512.03385. [Accessed 20 March 2026]. arXiv:1512.03385 [cs]

37. PASZKE, Adam, GROSS, Sam, MASSA, Francisco, LERER, Adam, BRADBURY, James, CHANAN, Gregory, KILLEEN, Trevor, LIN, Zeming, GIMELSHEIN, Natalia, ANTIGA, Luca, DESMAISON, Alban, KÖPF, Andreas, YANG, Edward, DEVITO, Zach, RAISON, Martin, TEJANI, Alykhan, CHILAMKURTHY, Sasank, STEINER, Benoit, FANG, Lu, BAI, Junjie and CHINTALA, Soumith. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Online. 3 December 2019. arXiv. arXiv:1912.01703. [Accessed 10 May 2026]. arXiv:1912.01703 [cs]