MASTER'S THESIS 2026

Quantization of CNNs & LLMs for integer-only hardware

David Stålmarck, Henning Tollin

ISSN 1650-2884
LU-CS-EX: 2023-79

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2023-79

Quantization of CNNs & LLMs for integer-only hardware

Kvantisering av CNN:er och LLM:er för hårdvara som endast stöder heltal

David Stålmarck, Henning Tollin

Quantization of CNNs & LLMs for integer-only hardware

David Stålmarck

Henning Tollin
henning.tollin@gmail.com

January 28, 2026

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Davide Grohman, davide.grohman@arm.com
Jonas Skepstedt, jonas.skeppstedt@cs.lth.se
Gareth Callanan, gareth.callanan@cs.lth.se

Examiner: Flavius Gruian, flavius.gruian@cs.lth.se

Abstract

This document describes the Master's Thesis format for the theses carried out at the Department of Computer Science, Lund University.

Your abstract should capture, in English, the whole thesis with focus on the problem and solution in 150 words. It should be placed on a separate right-hand page, with an additional *1cm* margin on both left and right. Avoid acronyms, footnotes, and references in the abstract if possible.

Leave a *2cm* vertical space after the abstract and provide a few keywords relevant for your report. Use five to six words, of which at most two should be from the title.

Keywords: MSc, BSc, template, report, style, structure

Acknowledgements

If you want to thank people, do it here, on a separate right-hand page. Both the U.S. *acknowledgments* and the British *acknowledgements* spellings are acceptable.

We would like to thank Lennart Andersson for his feedback on this template.

We would also like thank Camilla Lekebjer for her contribution on this template, as well as Magnus Hultin for his popular science summary class and example document.

Thanks also go to the following (former) students for helping with feedback and suggestions on this template: Mikael Persson, Christoffer Lundgren, Mahmoud Nasser.

Contents

Chapter 1

Evaluation

In this chapter, we present the evaluation of the models introduced in the previous chapter. The chapter is organized as follows. In Section 1.1, we report the accuracy results for the deep convolutional models, with the performance of the *4Conv*, *10Conv*, and *20Conv* architectures discussed in Section 1.1.1.

We then evaluate the *Cifar10* model in Section 1.1.2, followed by the results for the *MAGIC gamma* model in Section 1.1.3.

1.1 Performance in Accuracy

In this section, we present results comparing ONNX mixed-precision models with their corresponding fully integer TOSA models. For all experiments, the quantized ONNX model is used as the baseline for accuracy. A positive difference therefore indicates that the TOSA model achieves higher accuracy than the corresponding ONNX model, while a negative difference indicates a decrease in accuracy.

1.1.1 Deep model

For the deep convolutional models, three different network depths were evaluated. The results are summarized in Table 1.1.

Table 1.1: The results for mixed precision and integer models of different depths. The models were evaluated on a test set of 9,984 images

Model	ONNX accuracy	TOSA accuracy	Difference
4Conv	0.6523	0.6533	+0.0010
10Conv	0.6016	0.5987	−0.0029
20Conv	0.5233	0.5266	+0.0033

Table 1.1 reports the model name, the accuracy of the quantized ONNX QDQ-model, the accuracy of the corresponding TOSA full integer model, and the difference between the two. The results indicate that increasing model depth does not have a negative impact on the accuracy of the TOSA models relative to the ONNX models. Instead, the accuracies of the two representations closely track each other across all tested depths, even though overall performance decreases for deeper models. This suggests that the quantization error introduced when converting from mixed-precision to a fully integer representation does not accumulate in a way that significantly affects model accuracy.

Beyond comparing predicted class labels, it is also informative to examine the numerical outputs of the models directly. Such a comparison can provide additional insight into the differences between the two representations.

1.1.1.1 4Conv

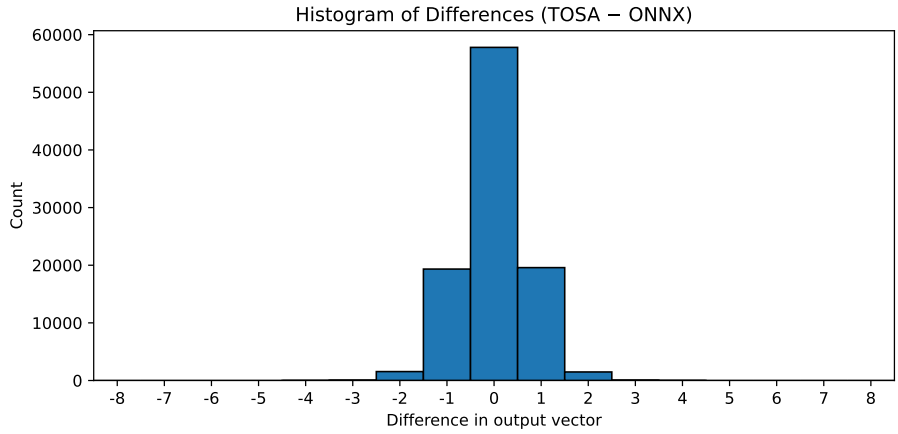


Figure 1.1: For the *4Conv* model. Shows the size of the numerical difference in the output vector between ONNX mixed precision and TOSA integer model.

Figure 1.1 illustrates the numerical differences between the output vectors produced by the ONNX and TOSA models. Although the classification accuracy of the two models is nearly identical, the figure shows that there are small numerical discrepancies in their output values. The x-axis represents the numerical difference between corresponding elements in the output vectors, while the y-axis indicates the count of each difference. Negative values indicate

cases where the ONNX QDQ-model produces a larger output value, whereas positive values indicate cases where the TOSA integer model produces a larger value.

From this figure, we conclude that the two models are not numerically identical at the output level. However, despite these differences in the output vectors, the predicted class often remains unchanged. This explains why the observed differences in accuracy are minimal, even though small numerical deviations are present.

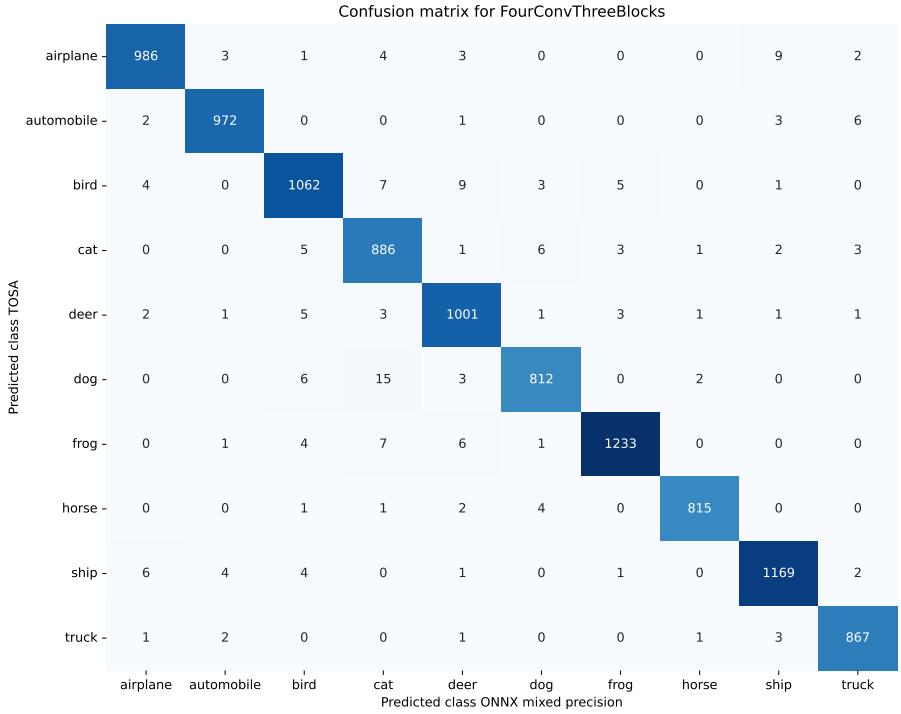


Figure 1.2: Confusion matrix between TOSA integer model and ONNX mixed precision model for the 4Conv model

Figure 1.2 shows the confusion matrix comparing the predictions of the *4Conv* ONNX QDQ-model and the corresponding TOSA model.

A confusion matrix is a representation of how well two different classification models agree with each other. The confusion matrix is interpreted as follows: The diagonal means that the two models predicted the same class, they are in agreement. The offdiagonals means that the models did not predict the same class and their respective predictions can be read from the labels on the row/columns.

The figure indicates that the predictions of the two models differ for a small subset of the inputs. From this result, we can conclude that although the models sometimes produce different predictions, this does not imply that one model is correct while the other is incorrect. In some cases, both models may misclassify the same input but assign it to different incorrect classes. This observation further supports the conclusion that small numerical differences between the models do not necessarily translate into meaningful differences in overall classification performance.

1.1.1.2 10Conv

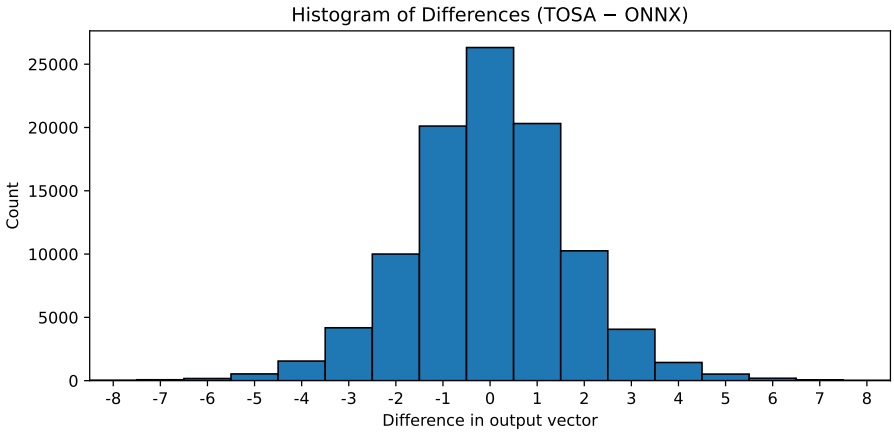


Figure 1.3: Shows the numerical difference in the output vector between ONNX mixed precision and TOSA model.

Figure 1.3 presents a histogram of the numerical differences between the outputs of the *10Conv* ONNX QDQ-model and the corresponding TOSA integer model. The figure illustrates the degree of similarity between the two models in terms of numerical behavior, but it does not directly reflect their classification accuracy. The x-axis shows the size of the numerical difference between the model outputs, while the y-axis indicates the count with which each difference occurs.

The results indicate that, in some cases, there are noticeable numerical differences between the two models, with differences occasionally reaching values as large as 7. While the distribution appears approximately symmetric, it is not perfectly so. Instead, the differences are centered around zero and approximately normally distributed, which explains the observed symmetry.

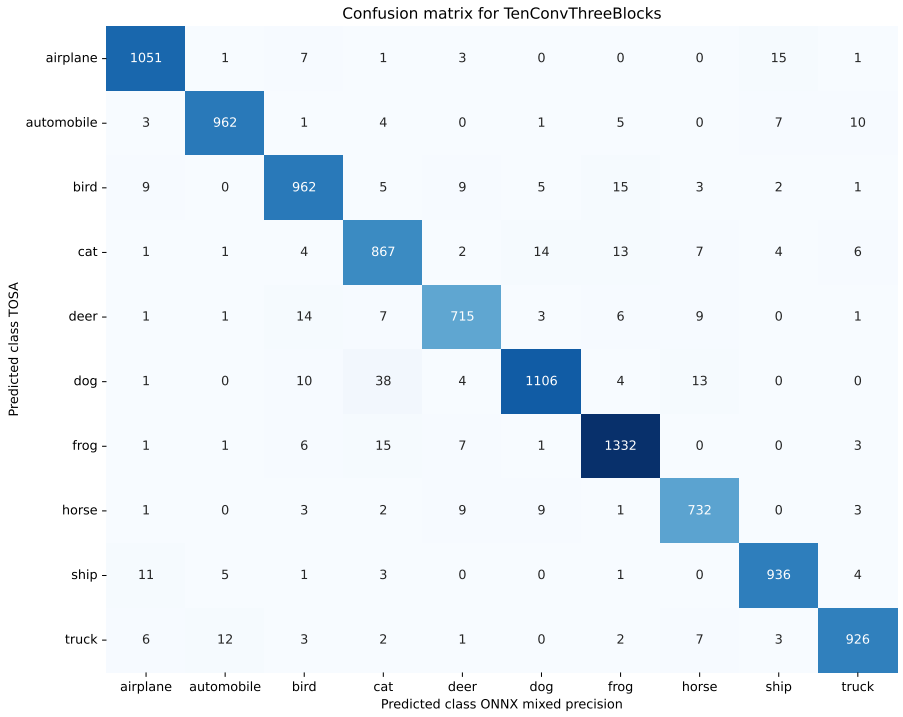


Figure 1.4: Confusion matrix between TOSA integer model and ONNX mixed precision model for the 10Conv model

Figure 1.4 shows the confusion matrix for the *10Conv* model. The purpose of this figure is to illustrate how the predictions of the ONNX QDQ-model and the TOSA integer model compare across the 9,984 test images. Each row corresponds to the class predicted by the TOSA model, while each column corresponds to the class predicted by the ONNX model. Each entry in the matrix therefore represents the number of samples for which the two models produced a specific combination of predictions.

The diagonal elements indicate the number of samples for which both models predicted the same class, which is the desired outcome. The relatively strong diagonal dominance observed in the matrix indicates a high level of agreement between the two models, despite the small numerical differences observed in their outputs.

1.1.1.3 20Conv

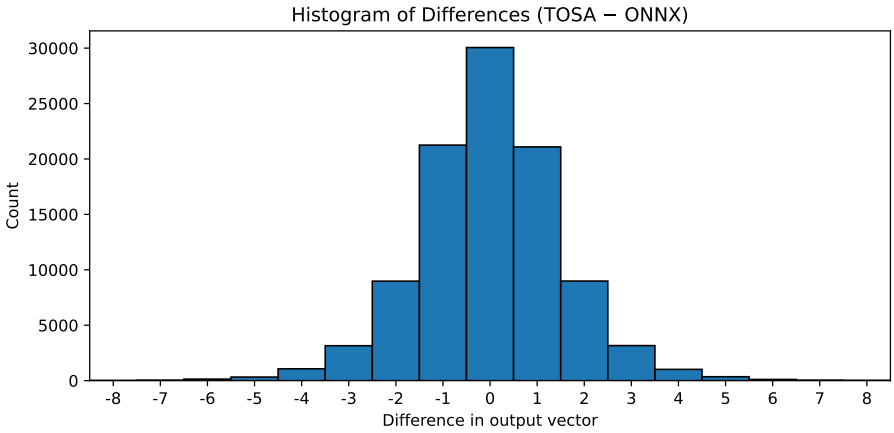


Figure 1.5: Shows the numerical difference in the output vector between ONNX mixed precision and TOSA model.

Figure 1.5 shows the numerical differences between the output vectors of the *20Conv* ONNX QDQ-model and the corresponding TOSA integer model. The x-axis represents the magnitude of the numerical difference, while the y-axis shows the frequency with which each difference occurs. Positive values indicate cases where the TOSA model produces a larger output value, whereas negative values indicate cases where the ONNX QDQ-model produces a larger value.

The figure indicates that, for a small subset of outputs, the numerical differences are relatively large. This suggests that some degree of error propagation may occur in deeper models. However, these numerical deviations do not imply that the TOSA model performs worse than the ONNX model in terms of classification accuracy. Rather, they reflect differences in the output vectors that do not necessarily affect the final predicted class.

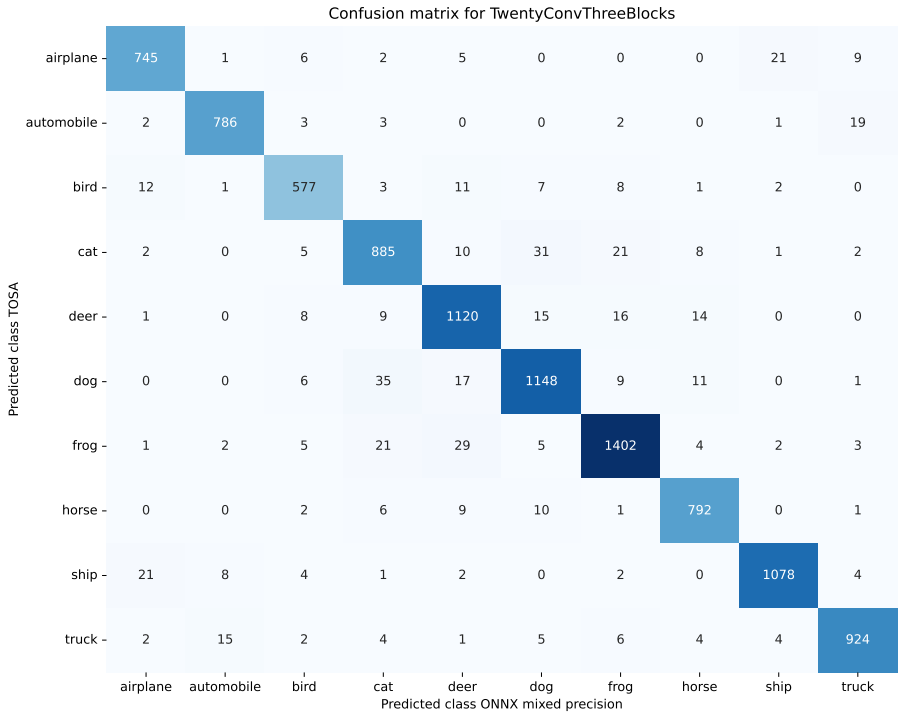


Figure 1.6: Confusion matrix between TOSA integer model and ONNX mixed precision model for the 20Conv model

Figure 1.6 displays the confusion matrix for the *20Conv* model. The figure illustrates the class predictions produced by the ONNX QDQ-model and the corresponding integer model. It can be observed that the predictions differ for a non-negligible number of samples, indicating that the two models occasionally assign different classes to the same input. This does however not indicate that one model is better then the other, both models might predict different incorrect classes.

1.1.2 Cifar model

The accuracy of the CIFAR-10 model is presented in Table 1.2.

Table 1.2: Results of the model intended for the Cifar data set. It was tested on a imageset of 9,984 images

Model	ONNX accuracy	TOSA accuracy	Difference
Cifar model	0.8056	0.8058	+0.002

Table 1.2 presents the accuracy comparison between the ONNX QDQ-model and the corresponding TOSA model for the *Cifar10* model. The results show that the proposed translation approach also performs well for models that are specifically designed to achieve high accuracy on the target task, with only negligible differences between the two representations.

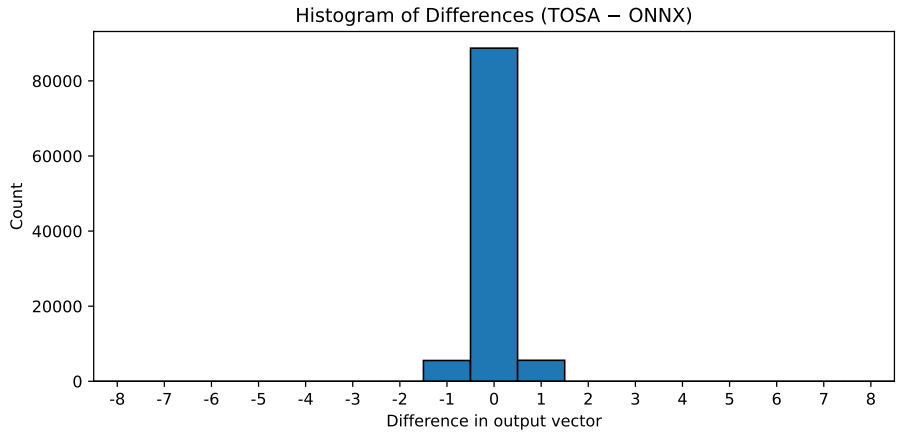


Figure 1.7: Shows the numerical difference in the output vector between ONNX mixed precision and TOSA model.

Figure 1.7 shows a numerical comparison between the outputs of the ONNX QDQ-model and the corresponding TOSA model. The figure illustrates the magnitude of the numerical differences in the output vectors produced by the two models. The x-axis represents the size of the difference, while the y-axis indicates the frequency with which each difference occurs. Negative values correspond to cases where the TOSA model produces a larger output value, whereas positive values indicate cases where the ONNX model produces a larger value.

A notable observation from this figure is that the majority of elements in the output vectors are identical between the two models. When differences do occur, their magnitude is at most 1. This indicates that the two models exhibit very similar numerical behavior.

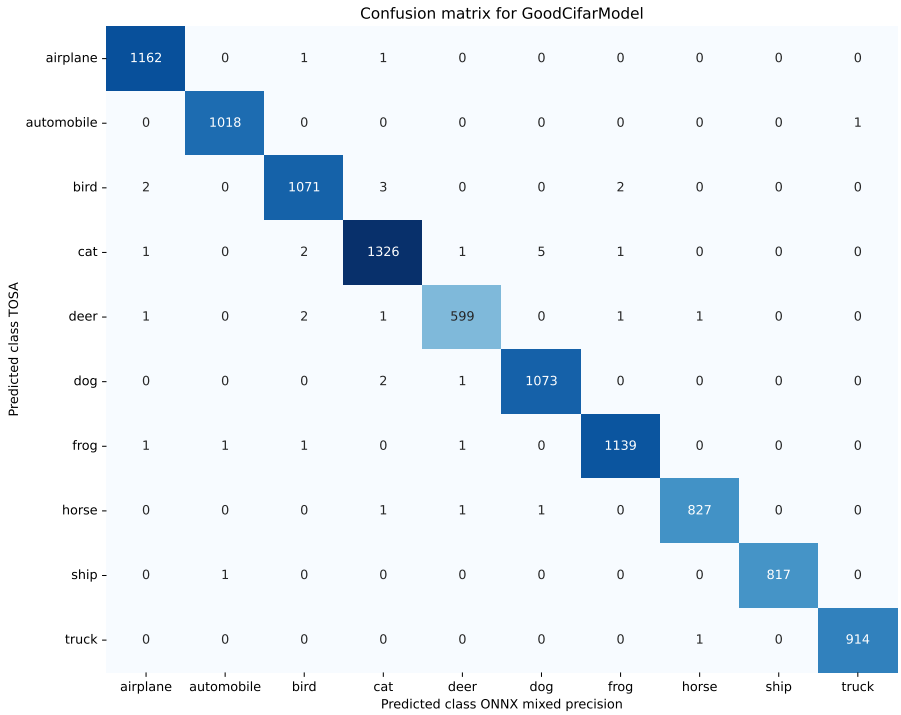


Figure 1.8: Confusion matrix between TOSA integer model and ONNX mixed presicion model for *Cifar10* model

Figure 1.8 presents the confusion matrix comparing the predicted classes of the *Cifar10* TOSA integer model and the corresponding ONNX QDQ-model. The figure shows that the two models produce different predictions for only a small number of samples, indicating a high level of agreement between their classification outputs.

1.1.3 MAGIC gamma telescope

The accuracy for the model trained on the MAGIC gamma telescope dataset presented in Table 1.3.

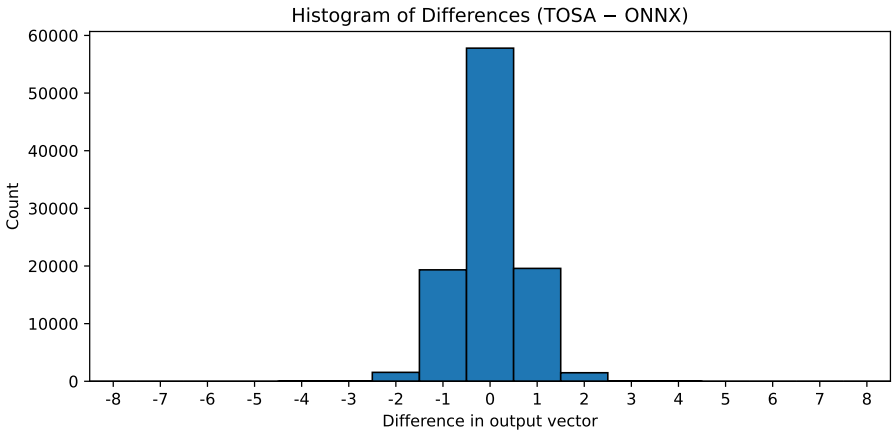
Table 1.3: Results of the model intended for the MAGIC gamma data set. It was tested on a imageset of 9,984 images

Model	ONNX accuracy	TOSA accuracy	Difference
MAGIC Gamma	0.8616	0.8616	0.000

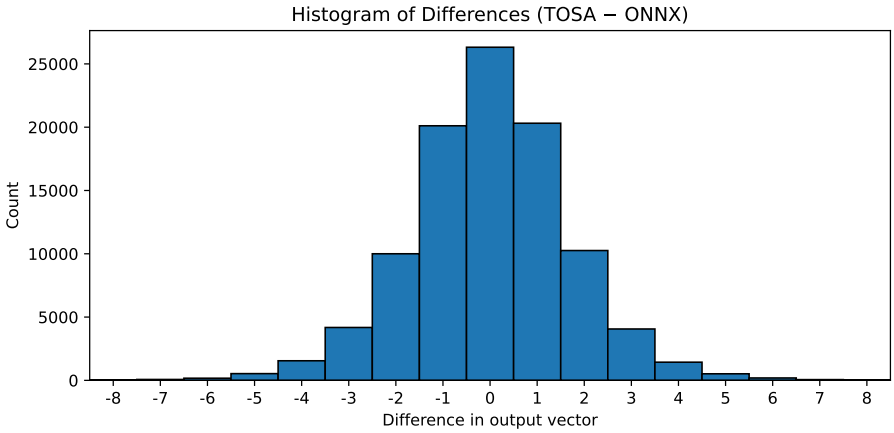
Table 1.3 presents the accuracy comparison for the *MAGIC gamma* model. The results show that the translation approach performs well for continuous-valued data, with both the ONNX QDQ-model and the TOSA model achieving identical accuracy.

1.2 Evaluation of Integer Softmax Methods

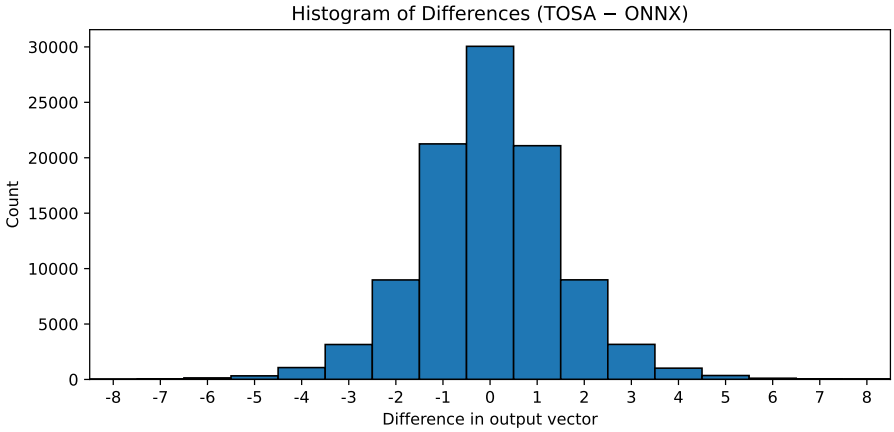
This section presents our experimental evaluation of integer softmax approximations. We evaluate six implementations: two single-table baselines (LUT-i8 and LUT-i16), polynomial



(a) Histogram for *4Conv*



(b) Histogram for *10Conv*



(c) Histogram for *20Conv*

Figure 1.9: Shows the size of the numerical difference in the output vector between ONNX mixed precision and TOSA integer model

approximations of varying degrees, and four DIGmax variants (DIGmax-i8-Log, DIGmax-i8-Linear, DIGmax-i16-Log, DIGmax-i16-Linear).

1.2.1 Implementation Overview

Representing softmax as a set of integer-only operations is not straightforward. We are aware of three general approaches: Taylor series approximations, lookup tables with interpolation (linear or polynomial), and power-of-two decomposition methods. These approaches have also been combined in various ways, as discussed in Section **??**. The two most common approaches are Taylor series and lookup tables with linear interpolation.

1.2.1.1 Single Lookup Table with Linear Interpolation (LUT-i16)

The LUT-i16 method uses a 513-entry lookup table with 7-bit linear interpolation, corresponding to the TOSA `TABLE` operator. While this approach works well in many cases, it has several limitations:

1. **Varying precision:** The approximation error depends on the distance from table entries. Figure 1.10 illustrates how the error exhibits a wave-like pattern. This occurs because the exponential function is approximated using linear segments between table entries. The relative error follows a repeating pattern regardless of the absolute value, since softmax normalization makes only relative differences matter.

2. **Fixed range coverage:** The table only covers a predetermined range $[0, x_{\max}]$. There is a trade-off: covering a wider range reduces precision, while covering a narrower range causes clipping for out-of-range inputs. In our experiments, setting $x_{\max} = 20$ caused the model to produce garbage output because a small number of logits fell outside this range and were handled incorrectly.

3. **Computational overhead:** While linear interpolation itself is inexpensive, it must be performed for every input element, adding to the total computation.

1.2.1.2 Polynomial Approximation

We also evaluated polynomial approximations using truncated Taylor series of degrees 2, 3, and 4. Figure 1.11 shows the approximation error for these methods.

The polynomial approach requires no lookup tables, but our experiments showed it is only accurate over a narrow range. Additionally, the computational cost scales with both the number of input elements and the polynomial degree. We concluded that standalone polynomial approximation is not practical for softmax. However, polynomials could potentially replace linear interpolation within a LUT-i16 scheme if the additional computation is acceptable.

1.2.1.3 DIGmax Methods

Our DIGmax approach addresses the range-precision trade-off by maintaining multiple lookup tables and selecting the appropriate one at runtime. We implemented four variants combining two table distributions (linear and logarithmic) with two output precisions (int8 and int16).

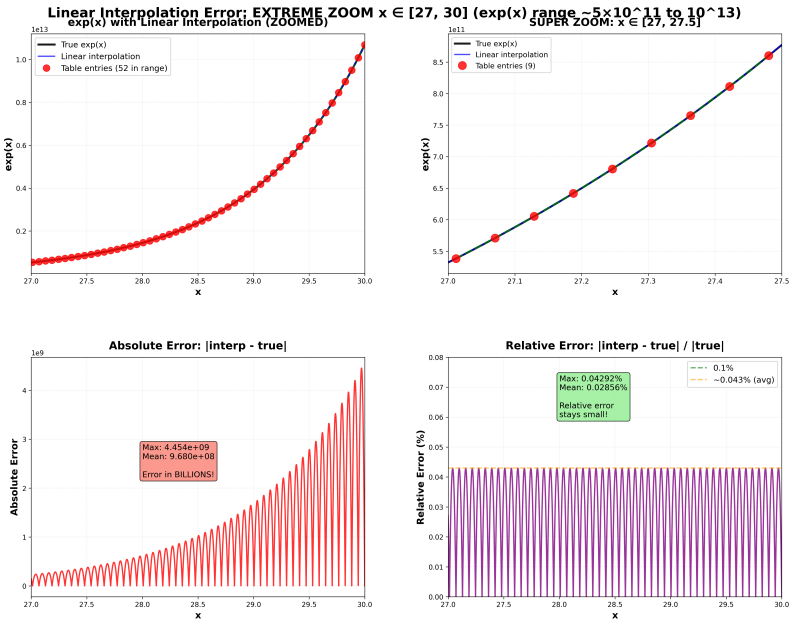


Figure 1.10: Linear interpolation error pattern for LUT-i16. The wave-like structure arises from approximating the exponential curve with linear segments between table entries. The error is smallest at table entries and largest at midpoints between entries.

1.2.2 Theoretical Evaluation: Uniform Test Points

We evaluate each method on synthetic test data to characterize their approximation behavior. For each configuration, we sample $N = 10{,}000$ uniformly spaced points across the input range and compute both the function approximation and the relative error compared to the true exponential.

Table 1.4: Theoretical experiment configurations

$x_{\max}$	N	Purpose	Formulation
8	10000	Small range, high precision expected	$\exp(x)$ on $[0, 8]$
8	10000	Small range, high precision expected	$\exp(x)$ on $[-8, 0]$
32	10000	Medium range, typical attention patterns	$\exp(x)$ on $[0, 32]$
32	10000	Medium range, typical attention patterns	$\exp(x)$ on $[-32, 0]$
128	10000	Large range, covering almost all logits	$\exp(x)$ on $[0, 128]$
128	10000	Large range, covering almost all logits	$\exp(x)$ on $[-128, 0]$

Each figure shows two panels: the upper panel displays the function approximation $\hat{f}(x)$ compared to the true $\exp(x)$, while the lower panel shows the relative error on a logarithmic scale. We present results for both positive ranges $[0, x_{\max}]$ and negative ranges $[-x_{\max}, 0]$. Although mathematically equivalent, these formulations produce different quantization behavior: positive ranges yield large output values that stress the upper limits of integer representation, while negative ranges yield small output values that underflow to zero. Since softmax uses the shifted formulation where inputs lie in $[-r, 0]$, the negative range results are more directly applicable to attention computations.

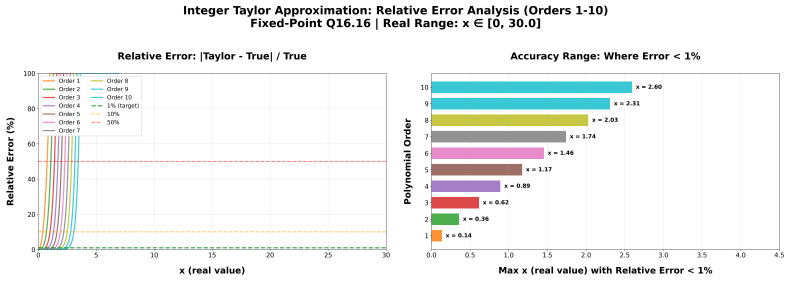


Figure 1.11: Approximation error for polynomial methods of varying degrees. Polynomials provide good accuracy only within a narrow range around the expansion point, with error growing rapidly outside this region.

The key phenomena visible in these results are:

1. **Zero-value regions:** Single-table methods (LUT-i8, LUT-i16) produce zero outputs beyond their effective range, visible as flat regions in the upper panel and very high relative error in the lower panel.

2. **Sawtooth error patterns:** DIGmax methods exhibit periodic error spikes at table boundaries where the selected table changes, followed by decreasing error within each table's range.

3. **Precision hierarchy:** Methods using int16 tables with interpolation consistently achieve lower baseline error than their int8 counterparts. By definition, the minimum resolution for int8 tables is $x_{\max}/2^8$, while for int16 with 7-bit interpolation it is $x_{\max}/2^{23}$.

4. **Range-precision trade-off:** As $x_{\max}$ increases, single-table methods must spread their limited precision across a wider range, degrading accuracy everywhere.

1.2.2.1 Small Range: $x_{\max} = 8$

For the small range $x_{\max} = 8$, all methods perform reasonably well. In the positive formulation (Figure 1.12), all methods track the exponential curve. LUT-i8 exhibits visible stepping due to its 8-bit output resolution, while LUT-i16 and DIGmax-i16 variants achieve smooth approximations.

In the negative formulation (Figure 1.13), LUT-i8 saturates to a constant value for $x < -2$ because $\exp(-2) \approx 0.135$ already approaches the resolution limit of 8-bit output. The characteristic wave pattern of LUT-i16's linear interpolation error is clearly visible. DIGmax methods show their sawtooth pattern at table boundaries but maintain bounded error throughout.

1.2.2.2 Medium Range: $x_{\max} = 32$

At $x_{\max} = 32$, the limitations of single-table methods become apparent. In the positive formulation (Figure 1.14), both LUT-i8 and LUT-i16 saturate and cannot represent the large values of $\exp(x)$ for $x > 5$. Their relative error reaches 100% (shown as 10^2 on the log scale) for most of the range.

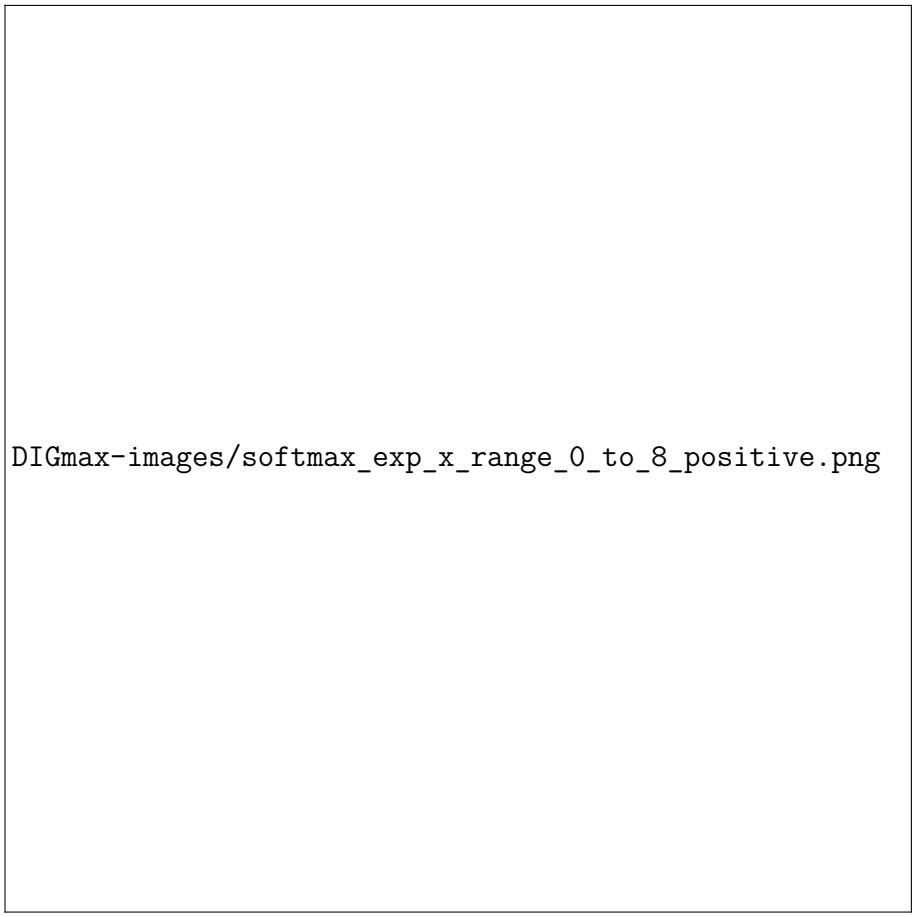


Figure 1.12: Function approximation and relative error for $\exp(x)$ on $[0, 8]$. All methods track the exponential curve well. LUT-i8 shows step-like behavior due to limited output resolution. DIGmax-i16 variants achieve the lowest error.

In the negative formulation (Figure 1.15), the zero-value regions are clearly visible: LUT-i8 outputs zero for approximately $x < -5$, and LUT-i16 for $x < -16$. DIGmax-i8-Log exhibits an interesting artifact: periodic spikes where the output jumps to 1.0 at table boundaries, causing relative error spikes up to 10^{12}. This occurs because the coarse logarithmic table selection occasionally maps inputs to an inappropriate table. DIGmax-i16 variants maintain error below 10% across nearly the entire range.

1.2.2.3 Large Range: $x_{\max} = 128$

The large range $x_{\max} = 128$ represents an extreme stress test. In the positive formulation (Figure 1.16), single-table methods saturate almost immediately. DIGmax methods continue to approximate the function, though DIGmax-i8 variants show visible stepping and periodic error spikes.

In the negative formulation (Figure 1.17), the limitations become severe. LUT-i8 and LUT-i16 output zero for the vast majority of the range. DIGmax-i8-Linear shows relative error growing exponentially (linearly on the log scale), reaching 10^{57} at $x = -128$. This occurs

Figure 1.13: Function approximation and relative error for $\exp(x)$ on $[-8, 0]$. LUT-i8 shows early saturation to a constant value. The wave-like error pattern of LUT-i16 from linear interpolation is visible. DIGmax methods maintain consistent error across the range.

because the quantization error compounds: small absolute errors become enormous relative errors when the true value is $\exp(-128) \approx 10^{-56}$.

DIGmax-i16 variants perform best, maintaining relative error around 1–10% across most of the range. The logarithmic variant (DIGmax-i16-Log) shows slightly higher but more consistent error, while the linear variant (DIGmax-i16-Linear) achieves lower error in some regions but with more variation.

1.2.3 Key Observations

Several conclusions emerge from this theoretical evaluation:

1. **Range coverage matters more than precision:** A method that covers the full input range with moderate precision outperforms one with high precision over a limited range. Missing even a few outlier logits can corrupt the entire attention computation.
2. **Int16 with interpolation provides substantial benefits:** The 23-bit effective precision (2^{16+7} distinct output levels) of int16 tables extends the effective range from approxi-

Figure 1.14: Function approximation and relative error for $\exp(x)$ on $[0, 32]$. LUT-i8 and LUT-i16 saturate early. DIGmax methods continue to approximate the exponential across the full range, though with periodic error spikes at table boundaries.

mately 5.5 units (int8) to 16 units, reducing zero-value regions significantly.

3. **DIGmax adapts to input distributions:** By selecting tables matched to the actual input range, DIGmax avoids the precision loss that single-table methods incur when designed for worst-case ranges.

4. **Logarithmic distribution trades precision for robustness:** DIGmax-Log variants use only 6 tables compared to 256 for linear distribution, reducing memory by 42×. The coarser granularity causes occasional larger errors at table boundaries but provides consistent behavior across all ranges.

1.3 22222 Evaluation of Integer-only Softmax methods, including DIGmax

When it comes to representing Softmax as a set of integer-only operations its not trivial what is the best way.

Figure 1.15: Function approximation and relative error for $\exp(x)$ on $[-32, 0]$. Single-table methods show large zero-value regions. DIGmax-i8-Log shows periodic spikes to $\exp(x) = 1$ at table boundaries. DIGmax-i16 variants maintain the best accuracy.

There are three ways which we are aware of, trying to approximate it using a taylor series, a big lookup table with interpolation, both linear and using polynomials, to not become to memory intensive, and a power of 2 solution where you try to match things to a power of two. Later people have combined these as well into different matters, see background section about this.

The two most common ones are taylor series based approach and big lookup with linear interpolation.

Explained more in detail in background **??**.

We implemented one sanity check that would be a randomizer for the softmax logits, that should represent and help find what randomized numerical error we can get and still have a good result. Here we had a normalized

We implemented one straight forward LUTi16, the big option of a lookup table in tosa using linear interpolation. The lookup table with linear interpolation is good. However it has some drawbacks.

Figure 1.16: Function approximation and relative error for $\exp(x)$ on $[0, 128]$. Single-table methods fail entirely beyond small x. Only DIGmax methods can approximate the exponential across this extreme range.

1. The precision varies depending on how close to an entry it is. As we try to illustrate in this image is how the error behaves in a pattern of waves. This comes from that we inside these bubbles are trying to represent an exponential function using linear interpolation. Which is a bit naive. Works for small values, worse for large, however since we in softmax later normalize, we can just look at the relative error and we see then this beautiful pattern of how the error no matter size has a standardized wave shape it repeats. THere are a few potential solutins for this, but the all come at a cose. You could have a taylor approximation between each point. In our case we are mostly interested in values close to 0, with normal distribution around 0. so You could also try to do optimizations where it matters the most, to decrease relative error around 0.
2. All values must be moved from a quantized state to a fixed-point integer state. Which is computational intensive.
3. The linear interpolation is not that expensive, however it adds to it, especially since you have to do that with every single entry.
4. It only covers a certain scope on the real axis. It only covers what you tell it to cover,

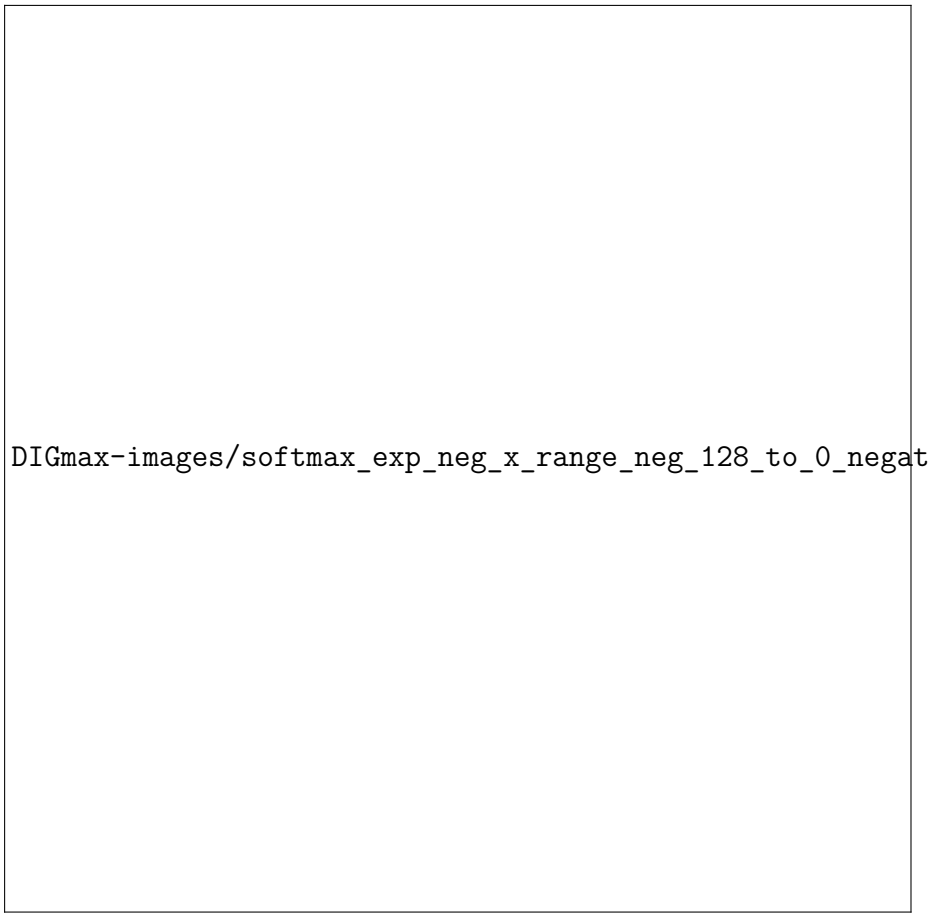


Figure 1.17: Function approximation and relative error for $\exp(x)$ on $[-128, 0]$. This extreme range causes all methods except DIGmax-i16 variants to exhibit large error regions. DIGmax-i8-Linear shows error growing linearly on the log scale (exponentially in absolute terms).

and it's a tradeof between the more you want to cover, the more precision you lose. Interesting enough when we tested this, it caused the LUTi16 (lookup table linear interpolation 16) to completely die and fail because, even though only a few logits where outside, it destroyed the full model and we only got garbage results. Mentioning this with an image a bit lower.

We implemented one polynomial approximation with different degrees.

Here we saw that these where good for a very short range. They are also very computational expensive. Therefore we concluded this approach not useful at all. However potentially as a replacement in the linear interpolation for LUTi16, however, only if you can afford the extra compute.

And finally, our DIGmax approach, which we conducted some off-track from TOSA spec to test different granularitites.

So we can began with studying

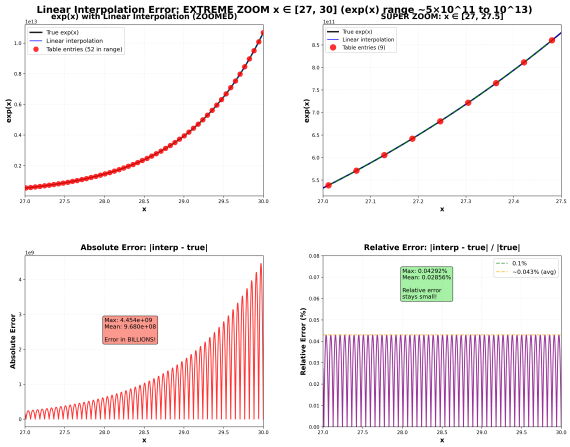


Figure 1.18: Caption

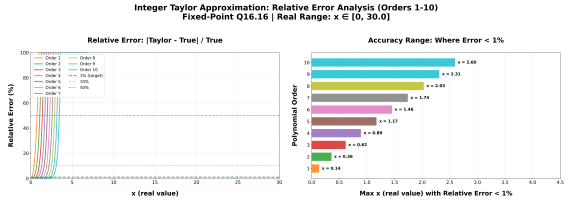


Figure 1.19: Caption

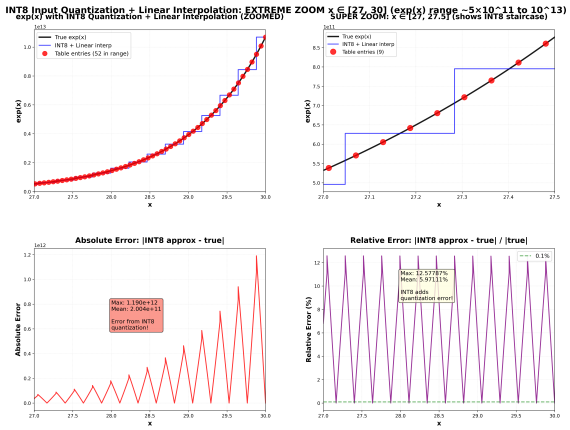


Figure 1.20: Caption

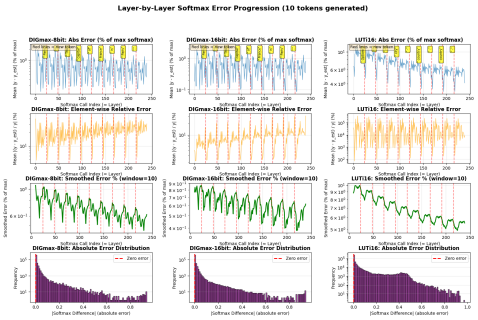


Figure 1.21: Caption

We saw many interesting things here regarding what seems to take effect.

Some interesting things we saw on the way to get there was for instance, for real worlds logits. Missing a few attention-logits destroys the full model. Therefore, making sure you can handle the full range is more important than precision sometimes. For instance we saw that with a xmax set to 20, so only handling range 0-20 with a LUTi16. For instance here is a table showing what happens here. We see that even using only 1 table of DIGmax where you quantize it down, aka, quantizing down to same scale becomes better than having a fixed range and trying to put it there. you see how the performance of a standard DIGmax perform better. This is because of xmax 20 however.

Just comparing the different implemntations on a straight line there are some differences in accuracy. Like LUTi16 has output [0, 2**31-1] while normal LUTi8 has [0, 2**7-1]

So here we have a comparison between some different configurations.

We see that as expected all version that uses the i16-tables perform better. Per definition is the i8 tables lowest accuaracy xmax/2**8, and for i16 xmax/2**23 (16+7 interpolated), meaning a much higher resolution for the output

1.3.1 Evaluation of full integer translation of transformer

When testing the full integer pipeline there are multiple things to have in mind.

The transformer have evolved a lot during the last few years, and it's a quite broad term in the sense on how to implement it. As menitoned in the sections above, we have multiple blocks inside a transformer, each block consist of a feed forward network and an attention layer, this attention layer consist of multiple heads of attention-heads (this is for parallellization and compute speed up reasons we want to have multiple heads). We have two things here, depth, how many blocks, and width, how many attention-heads, even though this may be less interesting.

1.3.1.1 Single-layer contribution of attention

So first we test for one single layer, what the contribution of error is.

Here we can see the contribution for each step in the process we described, we see that the big jump is when we perform the softmax that we get the degragation.

1.3.1.2 Error propagation through multiple layers of attention

After knowing the expected effect of one single layer we of course want to know how this propagates. It is not always obvious how. However, we can try to measure it by studying mean error and standard deviation of it.

From running tests we also see that, this is very noisy and hard to tell.

Chapter 2

Discussion

This chapter examines the significance of the experimental results, addresses the research questions posed in the introduction, and explores the broader implications of this work for edge deployment of neural networks. We structure the discussion around two main themes: the CNN translation in Section 2.1 and the integer softmax approach for transformers in Section 2.2.

2.1 CNN Translation Pipeline

The results presented in Section 1.1 demonstrate that translating quantized ONNX mixed-precision models to fully integer TOSA representations preserves classification performance to a high degree across a variety of model architectures and data modalities. Overall, the observed differences in accuracy are small and do not exhibit a systematic degradation in the TOSA models, supporting the viability of the proposed translation approach.

2.1.1 Impact of Model Depth

For the deep convolutional models, the comparison across different network depths shows that increased depth does not amplify accuracy discrepancies between the ONNX and TOSA representations. Although the absolute accuracy decreases as the models become deeper, this trend is consistent across both representations and is therefore attributable to model capacity and optimization rather than quantization effects.

The close tracking of accuracy between ONNX and TOSA models suggests that quantization errors introduced during the translation process do not accumulate in a way that significantly affects final predictions, even in deeper networks. This is a notable result, as deeper archi-

tectures are potentially more susceptible to numerical error propagation. While the output histograms for the deepest model indicate the presence of larger numerical deviations for a small subset of outputs as seen in Figures 1.3, 1.5, these deviations rarely influence the `argmax` operation that determines the predicted class.

2.1.2 Numerical Differences Versus Classification Outcomes

Across all evaluated models, a consistent pattern emerges: numerical differences between the ONNX and TOSA output vectors do not directly translate into meaningful differences in classification accuracy. The histogram analyses reveal that while output values may differ sometimes substantially, the predicted class often remains unchanged.

This observation highlights an important distinction between numerical equivalence and functional equivalence. Although the two representations are not numerically identical, they remain functionally equivalent for the classification task. The confusion matrices further support this interpretation by showing strong diagonal dominance, indicating a high level of agreement between predicted classes. When disagreements occur, they frequently involve both models making incorrect predictions but assigning different incorrect classes, rather than one model being systematically more accurate than its counterpart

2.1.3 Precision of input data

The experiments show that the translation to a fully integer *int8* representation behaves consistently across models operating on different types of input data. Both the *Cifar10* image classification model and the *MAGIC gamma* model were converted using the same integer-only quantization approach, despite their fundamentally different data characteristics.

For the *Cifar10* model, the conversion preserves classification accuracy and yields highly similar output values, with most outputs being identical and remaining differences limited to very small magnitudes. This indicates that the integer scaling parameters accurately capture the dynamic range of image-based activations.

For the MAGIC gamma telescope model, which operates on continuous-valued input features, the conversion results in identical accuracy between the two representations. This demonstrates that the integer-only execution is not limited to convolutional or image-based models, but generalizes to models with different input distributions and computational structures.

Overall, these results suggest that the proposed integer conversion method is robust across data modalities when using *int8* weights and activations, supporting its applicability as a general deployment strategy for integer-only inference.

2.1.4 Key Takeaways

The main conclusions that can be drawn from the experimental results are summarized as follows:

- The translation from quantized ONNX mixed-precision models to fully integer TOSA models preserves classification accuracy across all evaluated architectures and datasets, with observed differences consistently close to zero.
- Increasing network depth does not lead to systematic accuracy degradation in the TOSA models, indicating that quantization-related numerical errors do not accumulate in a way that meaningfully affects final predictions.
- Although numerical differences between ONNX and TOSA output vectors are present, these differences rarely alter the predicted class, demonstrating functional equivalence despite the lack of strict numerical equivalence.
- High-accuracy models, such as the *Cifar10* model, exhibit particularly strong robustness to integer-only execution, with most output values being identical across representations.
- The identical performance observed on the MAGIC gamma telescope dataset indicates that the proposed approach generalizes beyond image-based convolutional models to continuous-valued input data.
- Overall, the results support the feasibility of deploying fully integer TOSA models as a drop-in replacement for quantized ONNX models in integer-only inference environments.

2.2 Integer Softmax for Transformers

2.2.1 The DIGmax Approach

The DIGmax (Dynamical Integer-based Global-set-of-LUTs Softmax) approach represents a significant contribution toward enabling full-integer transformer inference without calibration. Unlike existing methods that rely on polynomial approximations (computationally expensive) or single global LUTs (limited accuracy), DIGmax uses a precomputed set of lookup tables that span the expected range of input scales. At runtime, the appropriate table is selected based on the dynamic range of the input tensor.

The translation invariance property of softmax is central to this approach. Since $\text{softmax}(x) = \text{softmax}(x + c)$ for any constant c, the output depends only on the relative spacing of input values, not their absolute magnitude. This spacing is captured by the quantization scale factor, which motivates organizing the LUT set by scale rather than by absolute value range.

2.2.2 Calibration-Free Operation

A crucial advantage of DIGmax is that it requires no calibration dataset. Traditional static quantization approaches must run representative data through the model to determine appropriate scale factors for each tensor. For large language models with diverse input distributions, obtaining a truly representative calibration set is challenging and can lead to accuracy degradation on out-of-distribution inputs.

DIGmax sidesteps this problem by computing the input scale dynamically at runtime. The statistical analysis presented in Section **??** provides a principled method for choosing the maximum expected value G based on the weight distributions initialized during training. By setting $G = 3\sqrt{\text{Var}(Z)}$, we capture 99.7% of expected values under the assumption that the attention logits follow a distribution derived from the weight initialization scheme.

2.2.3 Accuracy Analysis

The benchmark results demonstrate that DIGmax achieves accuracy comparable to the QDQ baseline in both the real domain $[0, 1]$ and the quantized domain $[0, 128]$. The MSE distributions show substantial overlap between DIGmax Heuristic and QDQ, with DIGmax Best (oracle table selection) achieving slightly better accuracy than both.

The table selection accuracy analysis reveals that the heuristic selection is off by an average of 1.81 tables from the optimal choice, with a median difference of 2 tables. Given that the global table contains $N = 256$ rows in the test configuration, this represents less than 1% deviation from optimal selection. The robustness of DIGmax to suboptimal table selection stems from the gradual variation between adjacent tables, neighboring entries in the global LUT approximate similar scale ranges.

2.2.4 Memory and Computational Trade-offs

The DIGmax approach introduces a trade-off between memory footprint and approximation accuracy. With N tables of 256 entries each (assuming 8-bit storage), the total LUT memory is $N \times 256$ bytes. For $N = 256$, this amounts to 64 KB, a modest overhead for modern hardware but potentially significant for extremely resource-constrained devices.

The computational overhead consists of:

1. Finding the row maximum for each softmax row (O(sequence_length) comparisons)
2. Computing the table index via a single fixed-point multiply-shift
3. Table lookup for each element
4. Integer division for normalization (using binary long division)

The binary long division algorithm (Code Snippet **??**) deserves particular attention. While division is typically expensive in integer-only hardware, the algorithm presented computes $2^K/d$ in $O(K)$ bit operations. For $K = 16$ fractional bits, this is 16 iterations of shift-compare-subtract, which is feasible for inference latency requirements.

2.2.5 Addressing Research Question 2

The second research question asked: *Which conversion methods yield the best results when transitioning from ONNX quantization schemes to TOSA in terms of accuracy and performance?*

Our experiments suggest that different approaches are optimal for different network components:

For linear operators (Conv, GEMM, MatMul): Static quantization with offline-computed rescale parameters yields excellent accuracy with minimal runtime overhead. The scale factors can be folded into multiplier-shift pairs at compile time.

For softmax: We still yet have to confirm this officially, but apple-to-apple comparison is being conducted atm. However we believe the results will be DIGmax since one Global LUT yielded extremely bad results: Dynamic quantization with DIGmax provides the best balance between accuracy and calibration requirements. The per-row dynamic approach adapts to varying input distributions without requiring offline calibration.

2.3 Limitations and Threats to Validity

2.3.1 Operator Coverage

The current implementation supports only a subset of ONNX operators: Conv, GEMM, ReLU, Flatten, and MaxPool. Many production models use additional operators such as batch normalization (not folded), various activation functions (GELU, SiLU), normalization layers (LayerNorm, GroupNorm), and attention mechanisms. Extending the pipeline to support these operators requires additional lowering passes and, in some cases, integer approximations for transcendental functions.

2.3.2 Model Scale

The evaluated models are relatively small compared to state-of-the-art architectures. The deepest model (20Conv) contains 60 convolutional layers, while modern networks like ResNet-152 or transformers with hundreds of attention layers present additional challenges. Error accumulation effects that were not observed in our experiments might become significant at larger scales.

2.3.3 Softmax Evaluation Scope

The DIGmax evaluation focuses on the softmax operation in isolation rather than end-to-end transformer inference. While the benchmark results are promising, the interaction between softmax quantization errors and subsequent operations (attention-weighted value aggregation, residual connections, layer normalization) requires further investigation.

2.3.4 Hardware Validation

Some of the experiments were conducted using the TOSA reference model, which provides bit-exact execution but does not reflect the performance characteristics of actual hardware implementations. Deployment on physical integer-only accelerators may reveal additional considerations related to memory bandwidth, cache behavior, and operator fusion opportunities.

Some of the experiments were conducted using simulations in python of integer-only hardware.

2.4 Practical Implications

2.4.1 Edge Deployment Viability

The results confirm that full-integer deployment pipelines built on MLIR and TOSA are viable for resource-constrained environments where floating-point execution is undesirable or infeasible. The negligible accuracy degradation observed across diverse model architectures and data types suggests that edge deployment can be achieved without sacrificing model quality.

2.4.2 Development Workflow

The pipeline established in this thesis, PyTorch → ONNX → ONNX-MLIR → TOSA, provides a clear path for practitioners seeking to deploy quantized models. By leveraging existing tools at each stage, the approach minimizes the implementation burden while maintaining flexibility for customization.

2.4.3 Transformer Quantization

The DIGmax approach opens new possibilities for transformer quantization on edge devices. By eliminating the calibration requirement, the approach simplifies deployment and enables adaptation to diverse input distributions. The memory overhead of the global LUT set is modest compared to the model weights themselves, making the trade-off favorable for most deployment scenarios.

2.5 Summary

The experimental results validate both contributions of this thesis. The CNN translation pipeline demonstrates that ONNX QDQ models can be converted to full-integer TOSA models with minimal accuracy loss, answering the first research question affirmatively. The DIGmax approach provides a practical solution for integer softmax computation without calibration, contributing to the broader goal of full-integer transformer inference.

The design of the lowering process, especially the factoring of scale parameters and the careful handling of tensor layouts, is essential to these outcomes. While the current implementation supports only a subset of operators, the findings indicate that the approach is fundamentally sound and can likely be extended to more complex architectures and operator sets. The consistency between ONNX and TOSA inference further suggests that full-integer deployment pipelines are viable for resource-constrained environments where floating-point execution is undesirable or infeasible.