Exploring Layer-specific Quantization in CNN-based Selective Sweep Detection

TIM VAN DE WETERING, University of Twente, The Netherlands

This study explores the complex field of population genetics known as "selective sweep detection," in which the fast spread of particular genetic variations within a population creates patterns in the genome. The ASDEC framework and the novel SweepNet architecture have made noteworthy advancements in genomic scan sensitivity, success rates, and detection accuracy. However running these CNNs requires a lot of resources, which presents a significant obstacle. Extensive experiments revealed that quantizing the layers separately produces encouraging outcomes. The majority of layers show a notable decrease in the required precision; some layers only require 2 bits, and in other circumstances, 1 bit is sufficient. By optimising resource utilisation and decreasing the memory requirement of these CNNs' operations, the layer-by-layer quantization technique makes these models run faster with fewer available resources.

Additional Key Words and Phrases: Convolutional Neural Network, Selective Sweep, Genome, DNA, Quantization, QKeras, Neural Network

1 INTRODUCTION

A selective sweep in population genetics refers to a specific phenomenon in population genetics where a particular genetic variant, often associated with a beneficial trait, rapidly increases in frequency within a population. Unique patterns are imprinted on the genetic terrain by selective sweeps. The area around the selected genetic variant usually shows reduced genetic diversity, as other variants in the neighbourhood get swept up with the selected variant's increasing frequency. Acknowledging this reduced diversity is an important signal for scientists searching for genomic areas affected by selection sweeps. Revealing these genomic areas is critical for comprehending population dynamics, identifying genes associated with adaptive characteristics, and gaining important knowledge about which factors are shaping genetic diversity. Selective sweep detection can be used to develop more potent pharmacological therapies as well as to explain how and why organisms survive in a given environment.

Originally, neural networks were used to tackle this problem, but with recent developments, a paradigm change has occurred where the problem is now framed as one of image classification. Convolutional Neural Networks (CNNs) have become essential tools in this context for efficiently handling the complexities of the classification problem. Significant improvements have been made over time to maximise these techniques' accuracy and speed.

ASDEC [10] is a notable framework that has been created to enable genome-wide scans to identify specific sweeps. This novel method highlights the continuous improvement and development in computational genomics. Furthermore, a thorough investigation of hyperparameters has led to significant advancements in CNN architecture, including the development of SweepNet [9]. SweepNet has a higher training efficiency than existing CNNs and requires considerably fewer epochs to achieve high validation accuracy. Furthermore, it performs consistently better in the presence of confounding factors [9].

When used for genomic scanning, the ASDEC framework performs better in terms of increased sensitivity, success rates, and detection accuracy. Additionally, the SweepNet design demonstrates a more consistent training behaviour in comparison to modern CNNs commonly used in population genetics. It demonstrates the capacity to distinguish between neutrality and a selective sweep even when confounding variables are present. Furthermore, SweepNet consistently achieves higher validation accuracy with fewer training epochs. Specifically, it uses raw genetic data from photos to classify them, saving time on data preparation by not depending on summary statistic distributions.

1.1 Motivation

Although the previously mentioned improvements improve selective sweep detection by tackling issues like speed and accuracy, a drawback of the proposed strategy is the amount of computing resources needed to run these CNNs. For a large bulk of data, running the model takes a very long time. It is essential to speed up this process, which can be done by quantization and building specialised hardware. To see if building specialised hardware makes sense, the quantization option should be tested first. Quantization aims to reduce the number of resources required to run the network in the field. If per-layer quantization is carried out without sacrificing a reasonable degree of accuracy, it could help speed up the image classification task. This can speed up research. In some cases, when the CNN is not very big, it could even make it possible to run on smaller and lighter devices. Selective sweep detection on more compact and lightweight devices could greatly expand research possibilities. This problem leads to the following questions:

1.2 Research Questions

The problem statement will lead to the following research question: *How does layer-specific quantization enhance the efficiency of CNNs in selective sweep detection for population genomics?*
And as a sub-question: *What insights can be gained by quantizing specific layers of CNNs in selective sweep detection for population genomics?*

1.3 Overview

This paper explores the existing literature on the quantization of CNNs for selective sweep detection. The first sections lay a solid basis by introducing CNNs and highlighting relevant related research. The methodology section explores the model's structure, the datasets that were employed, and the experimental setting to justify the choice to apply a layer-wise quantization strategy. The

TScIT 40, January 28, 2024, Enschede, The Netherlands

outcomes of these experiments are then shown in the results section. The results are summarised in the discussion that follows, which also points out the study's limitations and makes recommendations for other research directions. The paper concludes by answering the previously mentioned research questions.

2 LITERATURE

To maximise model performance, speed, and resource consumption, researchers have investigated a variety of approaches in the field of quantizing CNNs. This section explores the relevant literature, beginning with a summary of basic CNN principles to help lay the groundwork for comprehension of the quantization approach exploration that follows. In light of this background, significant contributions to the literature are emphasised, presenting a range of methods, instruments, and conclusions that have influenced the understanding and utilisation of quantization in the context of CNNs.

2.1 Context

A basic understanding of CNNs becomes essential when delving into the complexities of quantization in the context of CNN-based selective sweep detection. This research goes into the complicated procedures of quantizing CNNs and clarifies its consequences for selective sweep detection. Understanding the foundations of CNNs is necessary to appreciate the significance of quantization:

2.1.1 Convolutional Neural Network (CNN). A CNN is a type of artificial neural network [6] designed for tasks involving images and visual data. CNNs are particularly effective in image recognition, object detection, and other computer vision tasks. The key feature of CNNs is the use of convolutional layers, which apply convolution operations to input data.

2.1.2 Quantization. Quantization is the division of a quantity into a discrete number of small parts [5]. In the context of CNNs, this refers to the process of reducing the precision or representation of a set of values. The range of values that can be represented effectively decreases when an operation's bit width is reduced. As a result, this decrease increases the likelihood of round-off errors and their size. Such a drop in precision causes loss and hurts operating accuracy. Yet, quantization to smaller bit widths has several advantages despite these difficulties. It reduces the memory needed and speeds up processes on hardware platforms that can effectively handle smaller bit widths.

2.1.3 Layers. To attain an understanding of the behaviour of the quantization on the different layers, it is required to understand what the layers do. The different tasks that the layers carry out affect how the model reacts to quantization, which means it is required for understanding the results attained in this study:

(i) **Convolution Layer:** Convolutional operations involve sliding a small filter (also known as a kernel) over the input data, performing element-wise multiplications and summing the results. This operation allows the network to capture local patterns and features in the input data [6].

(ii) **Max Pooling Layer:** This downsamples the input along its spatial dimensions (`width` and `height`). It does this by computing the maximum value of its input window for each input channel. The input window is defined at the time of creating the layer (using the `pool_size` argument for `Keras`) [2].

(iii) **Global Average Pooling Layer:** This downsamples the input along its spatial dimensions, similar to (ii), but computing the average instead of the maximum. This also does not use a local region size. The global version goes over the entire spatial dimension [2].

(iv) **Activation Layer:** Activation layers apply an activation function to an output. A neural network's (NN) activation functions are mathematical operations that are applied to each neuron's output to give the model non-linearity [7]. An activation function essentially maps the input to a certain output range. NNs require non-linearity for the network to understand intricate linkages and patterns in the input. Since the composition of linear processes is still linear, an NN without activation functions would be effectively a linear model `Sigmoid`, `ReLu` and `Softmax` are used in `SweepNet`, you can find the formulas of these activation functions in Appendix A.

(v) **Multiply Layer:** Performs element-wise multiplication [2].

(vi) **Dense Layer:** Also called the fully-connected layer. Mathematically, for a *Dense* layer with n neurons, given an input vector $\mathbf{x} = [x_1, x_2, \ldots, x_m]$ from the previous layer, the output y_i of the i^{th} neuron in the *Dense* layer is calculated as [2]:

$$y_i = \text{activation}\left(\sum_{j=1}^{m} w_{ij} \cdot x_j + b_i\right)$$

- w_{ij} is the weight of the connection between the j^{th} neuron in the previous layer and the i^{th} neuron in the *Dense* layer.
- b_i is the bias term associated with the i^{th} neuron.
- activation$(\cdot)$ is an activation function applied element-wise to the result.

2.2 Related Work

Coelho Jr. et al. [3] present the `QKeras` library, which is an extension of the `Keras` library [2]. `QKeras` allows for heterogeneously quantized versions of deep neural network models, through drop-in replacement of `Keras` layers. These models are trained quantization-aware, where the user can trade off model area or energy consumption by accuracy. The training and deployment of extremely low-resource, low-latency heterogeneously quantized deep neural networks on-chip using `QKeras` and `hls4ml` is made possible by the authors' fully automated design methodology. By taking advantage of `QKeras`'s ability to swap out `Keras` layers easily, this method makes it easier to create models with varying levels of precision for each layer, all the while being trained with quantization awareness.

Wu et al. [8] proposed an efficient framework to simultaneously accelerate and compress CNNs. Their Quantized CNN approach focuses on quantizing filter kernels in convolutional layers and weighting matrices in fully connected layers to minimize estimation errors, achieving remarkable speed-up and compression rates. The experiments conducted on the ILSVRC-12 benchmark demonstrated a 4~6× speed-up and 15~20× compression with merely a one percentage loss of classification accuracy. Additionally, the authors have

implemented their Q-CNN framework into an Android application designed for CNN-based image classification on mobile devices. The experiments were conducted on a single CPU core without GPU acceleration. Notably, the Q-CNN framework achieved a remarkable 3× speed-up for AlexNet and a 4× speed-up for CNN-S. Moreover, storage consumption was compressed by 20×, and the required run-time memory was reduced to only one-quarter of the original model. In light of these improvements, the loss in top-5 classification accuracy did not exceed 1%. Consequently, the proposed approach enhances run-time efficiency across multiple aspects, making the deployment of CNN models feasible on mobile platforms.

There has been a growing interest in recent years in exploring the impact of quantization techniques on deep learning models, particularly in the context of CNNs. Gluska et al. [4] have made a contribution to this field with their paper, where they research the layer-specific aspects of quantization and its application to the specific task of selective sweep detection in CNNs.

3 METHODOLOGY

The methodology of this research consists of four main steps: converting the existing CNN, generating test data, quantizing the model and evaluation.

The existing CNN, `SweepNet`, has been converted to a parameter-based quantized version using TensorFlow [1] and `QKeras`. Due to `SweepNet` being developed in the Python programming language, it was used for this research too.

In-house software was used to generate test data in the form of images. This software is hosted in the same repository as `SweepNet`. These images are specifically generated from simulated genomic areas, offering a regulated and repeatable dataset for in-depth examination. `SweepNet`'s repository also contains a link to the datasets used.

The CNN's classification performance for various degrees of quantization has been measured. This was done strategically to reach optimal configurations as fast as possible. For every configuration, 10 training epochs were used for each dataset. The choice of conducting 10 training epochs stems from various considerations. Firstly, a longer training duration was deemed impractical due to the significant time required for training various configurations, which would substantially impede the timely completion of the research. Additionally, the decision aligns with the example command from `SweepNet`, chosen for its adequacy in addressing this study's objectives. Furthermore, empirical evaluations indicated that the quantization outcomes settled after the 10^{th} epoch.

The research results were evaluated and an optimal configuration, with the least amount of bits used, was created. This optimal configuration can be found in Table 6.

3.1 Layer-wise Quantization

There are several reasons why it is important to implement quantization layer-by-layer. Because each layer in a CNN serves a distinct purpose and responds differently to quantization, quantizing layer-wise enables fine-grained optimisation. The layer-wise approach also makes it easier to carry out an iterative optimisation procedure in which the effects of quantization on each layer are examined separately. This analysis keeps going until the accuracy loss becomes appreciable at a certain point in the overall size of the model. Furthermore, the layer-wise method improves analysis and interpretability by independently looking at quantization's impacts on each layer. This contributes to increased interpretability and transparency in understanding how quantization affects the internal representations of the model by offering insightful information about the model's behaviour.

3.2 Model Structure

The model structure is an important aspect of defining the performance and functionality of the research model. Furthermore, it serves as a crucial determinant in illustrating how the model responds to the quantization process. `SweepNet`'s layer structure was kept the same, however, some layers were swapped out with their `QKeras`' equivalent, and given names to allow for quantization and easier visualization of the changes. The detailed information about the model can be observed in Table 1.

Not all layers were able to be quantized, due to the lack of support in `QKeras`. An overview of the layers that *could* be quantized using `QKeras` is shown in Table 1. All layers which got a different layer type (notably, with a `Q`-prefix) are layers from `QKeras` and could be quantized. The other layers which were kept the same between old and new did not have a `QKeras` equivalent.

3.3 Datasets

The quantization process was done using 6 different datasets available via `SweepNet`'s repository. The datasets were converted using the tool `win2img.py`. This tool converts the `.txt` files of the binary matrices into images, as illustrated in Figure 1. Of the 6 different datasets, 2, 4 and 5 were the most difficult. The examination of these datasets was particularly thorough to ensure that the observed lack of significant accuracy reduction during quantization was not limited to straightforward datasets. Rather, it was verified that even more challenging datasets maintained their performance under the quantization process.

The datasets contain training and test data. The training data is split between training and validation data. A validation split of 0.2 is used to assess the model's performance during training. The datasets contain 2 image classes, neutral and selection, containing neutrality and selective sweep data respectively.

Each dataset comprises 1000 images for training data, distributed between training and validation sets, and an additional 1000 images for testing.

3.3.1 Difficulty. The differences in difficulty stem from how the datasets were generated. Three different demographic models confound selective sweep detection simulated, i.e., population bottlenecks, migration, and recombination heterogeneity [9]. These demographic models introduce unique confounding factors, detailed in Table 2. Notably, due to these factors, datasets 2, 4, and 5 present the most challenges.

Table 1. The structure of SweepNet, the model used during research.

Layer Index	Layer Name	Old Layer Type	New Layer Type	Architecture Component
0	sweepnet_first_conv	Conv2D	QConv2D	SweepNet
1	sweepnet_first_max_pool	MaxPooling2D	MaxPooling2D	SweepNet
2	sweepnet_second_conv	Conv2D	QConv2D	SweepNet
3	sweepnet_second_max_pool	MaxPooling2D	MaxPooling2D	SweepNet
4	sweepnet_third_conv	Conv2D	QConv2D	SweepNet
5	sweepnet_third_max_pool	MaxPooling2D	MaxPooling2D	SweepNet
6	se_block_first_global_avg	GlobalAvgPool2D	GlobalAvgPool2D	SE_block
7	se_block_first_conv	Conv2D	QConv2D	SE_block
8	se_block_first_activation	Activation	QActivation	SE_block
9	se_block_second_conv	Conv2D	QConv2D	SE_block
10	se_block_second_activation	Activation	QActivation	SE_block
11	se_block_first_multiply	Multiply	Multiply	SE_block
12	sweepnet_first_dense	Dense	QDense	SweepNet
13	sweepnet_first_global_avg	GlobalAvgPool2D	GlobalAvgPool2D	SweepNet
14	sweepnet_second_dense	Dense	QDense	SweepNet

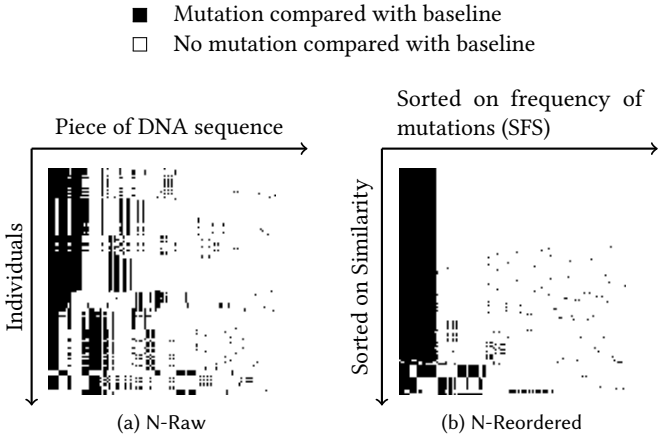

Fig. 1. The images converted from the .txt data from the datasets.

Table 2. The confounding factors for each dataset [9].

Dataset	Confounding Factor
1	Mild bottleneck
2	Severe bottleneck
3	Recent migration
4	Old migration
5	Low intensity recombination
6	High intensity recombination

3.4 Experiments

The quantization process was executed (almost) exclusively using QKeras' quantized_bits quantizer. Table 5 provides the average accuracy of the non-quantized model for each dataset, serving as a baseline against which the results of other configurations can be compared.

3.4.1 Bit widths. This study conducted an extensive look into quantization by examining several bit widths. The bit widths chosen were 1, 2, 4, 6, 8, and 32 bits. By including these various bit widths, a more nuanced understanding of how quantization affects model correctness across a range of precision levels was provided. Additionally, the selection of 32 bits was crucial due to it being the default precision of a non-quantized model. This meant the accuracy could be compared to the baseline accuracy, offering a full examination of the influence of quantization across the whole range of bit widths evaluated.

3.4.2 Initial Quantization. First, each layer of the neural network was quantized separately. Every layer was quantized using all the chosen bit widths mentioned previously. This provided a comprehensive insight into the impact of each bit width on both the accuracy and loss of the model. Additionally, it shed light on how the choice of a specific bit width for a single layer influenced the overall accuracy and loss of the entire model.

3.4.3 Uniform Quantization. After the initial quantization was done, all layers were quantized uniformly. They were quantized uniformly to speed up the process of finding the best configuration. Due to the knowledge gained during the initial quantization, it was expected that most layers do not need a high amount of bits.

3.4.4 Quantization Using 1 Bit. Building upon the earlier findings, additional rounds of quantization were executed. It was observed that reducing the bit width to 2 bits marked the threshold before a notable decline in accuracy was evident. The next step following the previous findings was quantizing all layers to have 2 bits, but every time only 1 layer would have 1 bit. The results gained prompted further testing with layers 7 and 9 using only 1 bit at the same time.

3.4.5 Activation Layers. The activation layers (layers 8 and 10) were then quantized. All the quantization experiments involved treating the two layers separately, using the same base model configuration as depicted in Table 3. Subsequently, the activation layers were both quantized at the same time.

Table 3. The accuracy and loss of the model with all layers quantized to use 2 bits, aside from layers 7 and 9, which used only 1 bit.

	Reference Accuracy[1]	Accuracy	Loss
Dataset 1	0.9925	0.9900	0.0269
Dataset 2	0.8999	0.9024	0.2985
Dataset 3	0.9850	0.9875	0.0359
Dataset 4	0.8324	0.8600	0.3776
Dataset 5	0.8025	0.8650	0.3295
Dataset 6	0.9925	0.9875	0.0330

[1] The reference or non-quantized model accuracy, which is also found in Table 5.

Table 4. The accuracy and loss of the model with all layers quantized to use 2 bits, aside from layers 7-10, which used only 1 bit.

	Reference Accuracy[1]	Accuracy	Loss
Dataset 1	0.9925	0.9900	0.0228
Dataset 2	0.8999	0.9024	0.2931
Dataset 3	0.9850	0.9825	0.0334
Dataset 4	0.8324	0.7425	0.5466
Dataset 5	0.8025	0.7875	0.4920
Dataset 6	0.9925	0.9674	0.1156

[1] The reference or non-quantized model accuracy, which is also found in Table 5.

4 RESULTS

A range of results was revealed by the in-depth examination of various quantization configurations described in 3.4. The results of some configurations are presented in Table 5, offering a detailed overview across all datasets.

4.1 Initial Quantization

In the experimental phase of quantizing each layer separately, detailed examinations were carried out on the impact of various bit widths on both accuracy and loss for the entire model. To visualize the findings, Figure 2 presents the accuracy of every layer separately quantized on dataset 2.

In this phase, it was discovered that most precisions did not cause any significant accuracy drop, however when using 1 bit the model started to exhibit a substantial decrease in accuracy. This observation prompted further investigation into the optimal bit width for maintaining accuracy while achieving quantization benefits.

4.2 Uniform Quantization

While the majority of configurations showcased minimal accuracy loss, a significant reduction in accuracy was observed at 1 bit, indicating that this bit width proved insufficient for the majority of layers. This substantial decrease in accuracy is visually depicted in Figure 3. The significant accuracy drop is illustrated alongside the accuracies corresponding to all other uniform bit widths across each dataset.

4.3 Quantization Using 1 Bit

In this phase, all layers were quantized to use 2 bits, but every time only 1 layer would have 1 bit. These configurations resulted in mostly significant loss, aside from 2 layers. Layers 7 and 9 did not drop the accuracy significantly when quantized to only use 1 bit. Both layers using 1 bit at the same time also maintained approximately the same level of accuracy without experiencing any significant decline, as shown in Table 3.

4.4 Activation Layers

No noticeable degradation in precision was observed for the lower bit widths as outlined in Table 4. The activation layers required only 1 bit each, which did not result in a reduction in accuracy.

4.5 Test

The outcomes of the experiments, as detailed in the preceding subsections, reveal an optimal configuration. Table 6 presents this configuration, specifically emphasizing the bits utilized per layer in the model.

The test accuracy is a crucial metric that validates the generalization capability of the trained model on unseen data. In addition to the validation/train accuracy, the corresponding test accuracy results can be found in Table 7.

5 DISCUSSION

This study has demonstrated that implementing per-layer quantization in a CNN can significantly reduce the model size without a substantial loss in accuracy. Detailed configurations and their accuracy are presented in Table 5. The optimal configuration (Table 6), a result of the entire quantization process described in 3.4, leverages previous findings to maintain high accuracy while achieving a noteworthy reduction in model size. Remarkably, the optimal configuration is only approximately 6.2% of the size of the non-quantized model, while maintaining a comparable level of accuracy. Upon examining the results, several interesting observations emerge, shedding light on the effects of quantizing `SweepNet` layer-wise.

The results shown in Table 7 indicate that the model had difficulties with the selection dataset, suggesting that the dataset is particularly challenging. Interestingly, the selection datasets show a more noticeable deterioration in accuracy, suggesting a possible higher need for precision. These findings point to a potential higher bit width requirement for the datasets used in the selection process. Further research is needed to confirm these speculations.

A noteworthy trend appears in Figure 3, where datasets 4 and 5 *decline* accuracy when using more than 4 bits. This behaviour could perhaps result from the model's overfitting tendencies. Higher precision may make overfitting more effective, while lower precision makes overfitting less effective. More research is needed to support this hypothesis.

Tables 7 and 5 show that dataset 4 has a more noteworthy drop in precision compared to the other datasets. Table 7 shows the lost accuracy for neutral and selection separately (0.010 for neutral and 0.082 for selection). Even separately, dataset 4 has still lost the most accuracy out of all of them. This seems to be suggesting that the confounding factor used to create dataset 4, makes the model require more precision to operate. The previously mentioned overfitting

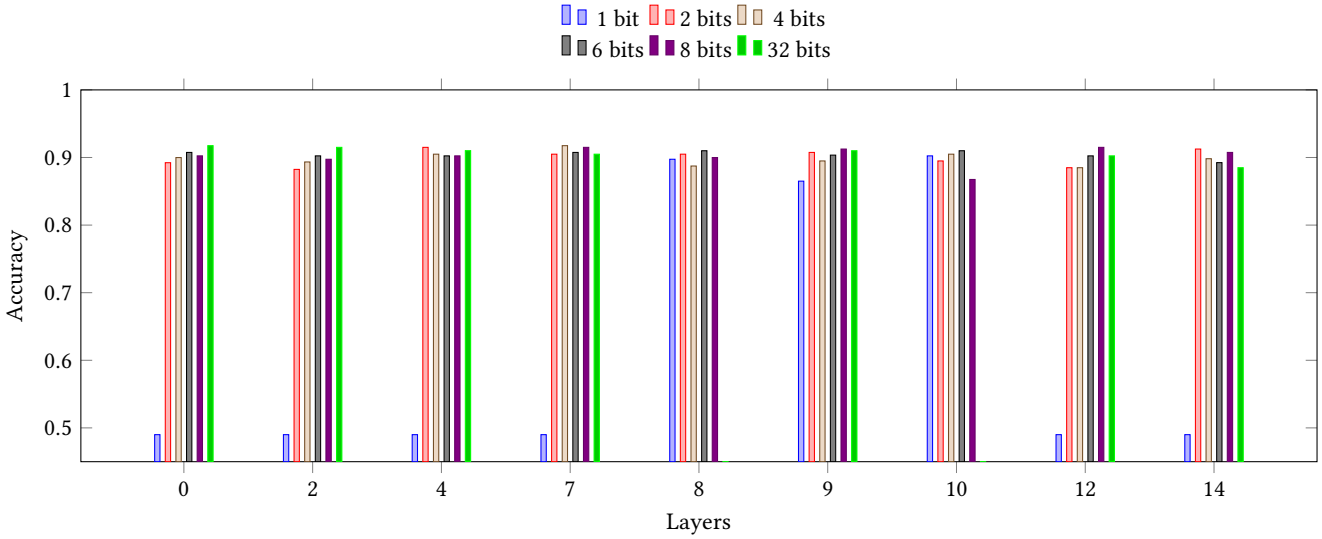

Fig. 2. The accuracy achieved by quantizing every layer separately on dataset 2.

Table 5. Details of varying configurations where the accuracy after the last epoch is provided for every dataset.

Configuration	Model Size (Bits)	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
Non-quantized	310400	0.9925	0.8999	0.9850	0.8324	0.8025	0.9925
All layers 2 bits[1]	19400	0.9925	0.9100	0.9900	0.7450	0.8700	0.9725
Optimal[2]	19238	0.9900	0.9024	0.9825	0.7425	0.7875	0.9674

[1] Aside from the activation layers as mentioned in 5.1.1.
[2] The optimal configuration is displayed in Table 6.

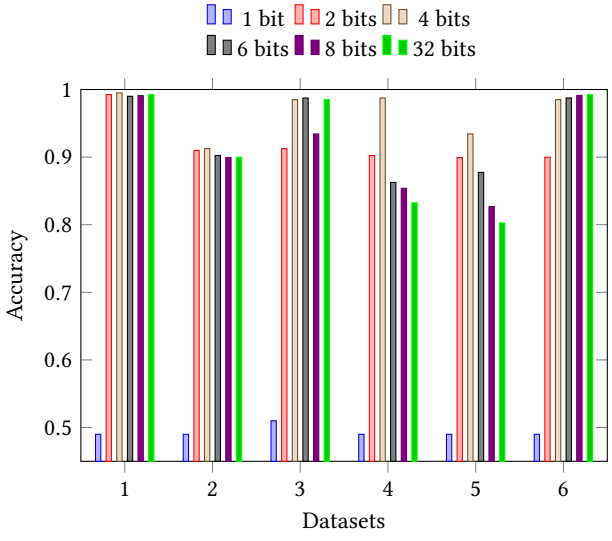

Fig. 3. The accuracy achieved by uniformly quantizing the model across all 6 datasets.

Table 6. The bits used per layer for the optimal configuration.

Layer Index	Layer Name	Bits Used
0	sweepnet_first_conv	2
1	sweepnet_first_max_pool	2
2	sweepnet_second_conv	2
3	sweepnet_second_max_pool	2
4	sweepnet_third_conv	2
5	sweepnet_third_max_pool	2
6	se_block_first_global_avg	2
7	se_block_first_conv	1
8	se_block_first_activation	1
9	se_block_second_conv	1
10	se_block_second_activation	1
11	se_block_first_multiply	2
12	sweepnet_first_dense	2
13	sweepnet_first_global_avg	2
14	sweepnet_second_dense	2

problem could also be playing a role here. More research is needed to ascertain the validity of these claims.

5.1 Challenges

This study was not conducted without any obstacles:

Table 7. The test accuracy for neutral and selection classification for each dataset across some configurations.

	Non-quantized	Optimal[1]
Dataset 1 Neutral	1.0	1.0
Dataset 1 Selection	0.995	0.983
Dataset 2 Neutral	0.879	0.878
Dataset 2 Selection	0.915	0.909
Dataset 3 Neutral	0.988	0.988
Dataset 3 Selection	0.993	0.992
Dataset 4 Neutral	0.939	0.929
Dataset 4 Selection	0.781	0.699
Dataset 5 Neutral	0.856	0.887
Dataset 5 Selection	0.863	0.803
Dataset 6 Neutral	0.969	0.984
Dataset 6 Selection	0.998	0.986

[1] The optimal configuration is displayed in Table 6.

5.1.1 Missing Quantizers. The activation layers (layers 8 and 10) were not initially quantized due to the difference in how their quantizers were defined. All other layers have a separate `kernel_quantizer` and `bias_quantizer`. The `bias_quantizer` does not do anything significant in this case. Only the `kernel_quantizer` was of importance. The activation layers are different. They do not have a `kernel_quantizer` and `bias_quantizer`, they only have an `activation`. The `activation` itself decides what activation method to use and also if it is quantized. Since using the same `quantized_bits` quantizer would change the activation method, a different one was used. `quantized_relu` for the first activation layer (layer 8) since it uses the `relu` method. The second activation layer (layer 10) used the `sigmoid` function that was not documented to be implemented. Later `quantized_sigmoid` was found to be implemented but not documented, prompting its incorporation into the methodology for subsequent use.

5.1.2 Activation Layers. Although no noticeable degradation in precision was observed for any bit width as mentioned in 4.4, this was not the case for 32 bits. The `quantized_relu` and `quantized_sigmoid` quantizers both dropped to a flat zero accuracy across all datasets when utilizing 32 bits and `NaN`[1] for the loss. The outcomes observed with 32 bits suggest a potential bug; however, its impact was relatively small, as the performance with other bit widths remained consistent.

5.2 Limitations

In acknowledging the scope and constraints of this study, certain limitations deserve consideration.

5.2.1 Quantizers. In this study, the employed quantization methods include `quantized_bits` for the majority of layers, while the two activation layers utilize `quantized_relu` and `quantized_sigmoid` respectively. `QKeras`, however, encompasses numerous quantizers that were not subjected to testing, potentially yielding diverse results.

5.2.2 Model Structure. In the scope of this study, it is noteworthy that the structural elements of `SweepNet`'s model, including layer types, layer order, layer count, and input sizes, remained unaltered. The focus solely centred on exploring the effects of quantization on the existing architecture. The structure itself plays an important role in the effects of quantization.

5.2.3 Data. This study utilized the 6 datasets the developers of `SweepNet` provided. The analysis was conducted using the 'quick example' command provided by the developers. It's important to note that variations in datasets and inputs to the command may yield different results

5.2.4 Hardware Acceleration. Quantization diminishes the memory requirements for parameter storage, enabling the operation of CNNs on systems with fewer resources available. However, it brings forth the challenge of lacking hardware acceleration. While most CPUs natively support 32-bit precision, lower precisions such as 2-bits are not native, resulting in a slowdown in model execution that necessitates specialized hardware for acceleration.

However, this issue primarily pertains to general-purpose CPUs; specialized hardware is often built to natively support such low-precision operations, facilitating hardware acceleration.

5.3 Future work

The study primarily focused on layer-specific quantization using varying bit widths. A potential area for further exploration involves experimenting with different quantizers and exploring additional parameters within the `quantized_bits` quantizer, particularly `integer` and `symmetric`. The `integer` parameter dictates the number of bits to the left of the decimal point, while the `symmetric` parameter determines whether the bit representation represents an equal amount of positive and negative numbers.

Future research could also focus on different model structures and test data.

The absence of hardware acceleration on non-specialised hardware may incentivize further investigation into the trade-offs between achieving low memory usage and circumventing the requirement for specialized hardware to accelerate quantized models when specialised hardware is not desired.

6 CONCLUSION

In the context of resource-intensive CNNs, such as `SweepNet`, this study highlights the potential for significant resource reduction through layer-specific quantization, thereby addressing the resource demands associated with training and running complex models. In light of the potential resource reduction demonstrated through layer-specific quantization, the research questions stated can be answered:

[1] `NaN` (**N**ot **a** **N**umber) denotes an invalid or undefined value for floating-point numbers, such as the outcome of 0/0 or $\sqrt{X}, X < 0$.

6.1 Sub-question: What insights can be gained by quantizing specific layers of CNNs in selective sweep detection for population genomics?

Research on quantizing individual CNN layers for population genomics selective sweep detection shows that per-layer quantization leads to a significant reduction in model size with high accuracy. Problems arise, especially when it comes to the selection datasets, suggesting that increased precision may be required because of the apparent decline in accuracy. More than 4 bits appear to be associated with decreased accuracy for some datasets, which may be due to overfitting behaviours driven by precision levels. The precision of dataset 4 significantly decreases, suggesting a higher precision need that may be caused by confounding factors. These findings highlight the importance of carefully selecting precision based on dataset characteristics and conducting further investigation to validate claims and improve model effectiveness.

6.2 Research Question: How does layer-specific quantization enhance the efficiency of CNNs in selective sweep detection for population genomics?

The study demonstrates that layer-specific quantization in a CNN dramatically lowers the model size without sacrificing a significant amount of accuracy. The quantization method resulted in an ideal configuration that retains similar accuracy but achieves a notable reduction in model size, about `6.2%` of the non-quantized model's size. These results demonstrate how layer-specific quantization in resource-intensive CNNs such as `SweepNet` can lead to significant resource reduction.

Layer-specific quantization has efficiency benefits, but some drawbacks must also be considered. Specialised hardware is required because quantized bit lengths are not supported natively on most general-purpose CPUs, which might cause significant slowdowns. Nonetheless, because of the reduced memory requirements, the specialised hardware need not be as large as that required for comparable non-quantized models.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] François Chollet et al. 2015. Keras. https://keras.io.

[3] Claudionor N. Coelho Jr., Aki Kuusela, Shan Li, Hao Zhuang, Thea Aarrestad, Vladimir Loncar, Jennifer Ngadiuba, Maurizio Pierini, Adrian Alan Pol, and Sioni Summers. 2021. Automatic Heterogeneous Quantization of Deep Neural Networks for Low-Latency Inference on the Edge for Particle Detectors. *Nature Machine Intelligence* 3, 8 (June 2021), 675–686. https://doi.org/10.1038/s42256-021-00356-5 arXiv:2006.10159 [hep-ex, physics:physics]

[4] Shachar Gluska and Mark Grobman. 2020. Exploring Neural Networks Quantization via Layer-Wise Quantization Analysis. https://doi.org/10.48550/arXiv.2012.08420 arXiv:2012.08420 [cs, stat]

[5] R.M. Gray and D.L. Neuhoff. 1998. Quantization. *IEEE Transactions on Information Theory* 44, 6 (Oct. 1998), 2325–2383. https://doi.org/10.1109/18.720541

[6] Keiron O'Shea and Ryan Nash. 2015. An Introduction to Convolutional Neural Networks. https://doi.org/10.48550/arXiv.1511.08458 arXiv:1511.08458 [cs]

[7] Siddharth Sharma, Simone Sharma, and Anidhya Athaiya. 2020. ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology* 04, 12 (May 2020), 310–316. https://doi.org/10.33564/IJEAST.2020.v04i12.054

[8] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. 2016. Quantized Convolutional Neural Networks for Mobile Devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4820–4828. https://openaccess.thecvf.com/content_cvpr_2016/html/Wu_Quantized_Convolutional_Neural_CVPR_2016_paper.html

[9] Hanqing Zhao, Pavlos Pavlidis, and Nikolaos Alachiotis. 2023. SweepNet: A Lightweight CNN Architecture for the Classification of Adaptive Genomic Regions. In *Proceedings of the Platform for Advanced Scientific Computing Conference (PASC '23)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3592979.3593411

[10] Hanqing Zhao, Matthijs Souilljee, Pavlos Pavlidis, and Nikolaos Alachiotis. 2023. Genome-Wide Scans for Selective Sweeps Using Convolutional Neural Networks. *Bioinformatics* 39, Supplement_1 (June 2023), i194–i203. https://doi.org/10.1093/bioinformatics/btad265

A ACTIVATION FUNCTIONS

The activation functions used in `SweepNet` are [7]:

- **Sigmoid:**

$$\sigma(x) = \frac{1}{1+e^{-x}},$$
$$\sigma(x) \in (0,1)$$

- **ReLu:**

$$\mathrm{ReLU}(x) = \max(0,x),$$
$$\mathrm{ReLU}(x) \in [0,+\infty)$$

- **Softmax:**

$$\mathrm{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}},$$
$$\mathrm{Softmax}(x_i) \in (0,1)$$