Improved Compression of Artificial Neural Networks through Curvature-Aware Training

1st Reemt Hinrichs
Institut für Informationsverarbeitung
Leibniz University Hannover
Hannover, Germany
hinrichs@tnt.uni-hannover.de

2nd Kai Liang
Institut für Informationsverarbeitung
Leibniz University Hannover
Hannover, Germany

3rd Ze Lu
Institut für Informationsverarbeitung
Leibniz University Hannover
Hannover, Germany

4th Jörn Ostermann
Institut für Informationsverarbeitung
Leibniz University Hannover
Hannover, Germany

***Abstract*—Artificial neural networks achieve state-of-the-art performance in many branches of engineering. As such they are used for all kinds of tasks and nowadays are desired to be used on mobile devices like smartphones. Due to limited hardware resources or limited channel capacity on mobile devices, compression of neural network models to reduce storage or transmission costs is desired. Furthermore, reduced complexity is of interest. This work investigates introducing the curvature of the loss surface in the training of artificial neural networks and analyzes its benefit for the compression of neural networks through quantization and pruning of its weights. As proof-of-concept, three small LeNet-based neural networks were trained using a novel loss function consisting of a weighted average of the cross-entropy loss and the Frobenius norm of the hessian matrix. That way both, the loss as well as the local curvature, are minimized concurrently. Using the proposed method, mean test accuracies on the MNIST and FashionMNIST datasets after quantization were considerably improved by up to about $47.6\,\%$ for 1 bit quantization on MNIST and about $27.8\,\%$ on FashionMNIST compared to quantization after training without curvature information. Additionally, pruning was found to benefit from introducing curvature in the training as well with an increase of up to about $14.6\,\%$ mean test accuracy compared to pruning after training without curvature except for isolated cases. Training the artificial neural networks first without curvature information and subsequent training by only one epoch using curvature information allowed to increase the mean test accuracy after quantization at 1 bit by about $16\,\%$. The proposed method can potentially improve the accuracy after compression irrespective of the compression method applied.**

***Index Terms*—loss surface, curvature, compression, neural networks**

I. Introduction

Artificial neural networks (ANNs) nowadays are applied to many tasks achieving state-of-the-art performance in many branches of engineering. A common application is image classification where an artificial neural network maps a given image to one out of several classes [1]. Such an application among others is desireable to also be used on e.g. mobile devices [2]. Currently, a common approach is to send data to a remote computer that runs an ANN and returns the result to the mobile device [3]. This indirect application shifts the computational burden to other devices and avoids the need for greater hardware power. However, this approach requires an internet connection, introduces delay due to the required data transmission and gives rise to privacy issues. Therefore, compression methods are investigated to allow the deployment and application of artificial neural networks, which can have up to hundreds of millions of parameters and more, on mobile devices like smartphones or other devices with comparatively little hardware resources. To tackle this issue, the Movie Picture Expert Group (MPEG) recently standardized [4] a collection of methods to compress ANNs. One of the most common approaches to the compression of ANNs is certainly pruning [5]. Pruning methods generally set a certain subset of all weights of a given ANN to zero, usually those below some given threshold or the smallest $x\,\%$ of the weights with respect to their magnitude [6]. A large number of specialized pruning

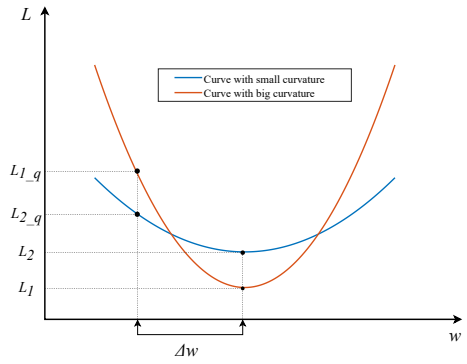


Fig. 1: Depicted are two example courses of a loss function with respect to a single weight for illustration. The red curve has higher curvature around its minimum but achieves a lower value at its minimum than the blue curve. The red curve, while prequantization allowing for the smallest loss, is more sensitive around its minimum to quantization, yielding the loss value L_{1_q}, and thus after quantization the blue curve achieves the lowest loss value L_{2_q} in this example.

methods exist like pruning using weight regularization, pruning via loss sensitivity or pruning using first order derivatives [5], [7]. Many other approaches to ANN compression have been proposed like weight sharing [8], tensor decompositions of weights [9], transform coding [10], knowledge distillation [11] or quantization [12]. Quantization in the context of ANN compression means the quantization of the weights of the network. A proxy for the performance of an ANN is the (task dependent) loss $\mathbf{L} \equiv \mathbf{L}(\mathbf{w}) \equiv \mathbf{L}(\mathbf{w}; \mathbf{x}, \mathbf{y})$ which depends on the input data $\mathbf{x}$, output data $\mathbf{y}$ and the weights of the network $\mathbf{w} = (w_1, \ldots, w_N) \in \mathbb{R}^N$. N denotes the total number of weights of the respective ANN.

Quantization of the weights is equivalent to adding a vector $\Delta\mathbf{w}$ yielding the new weights

$$\mathbf{w}_Q := \mathbf{w} + \Delta\mathbf{w} \tag{1}$$

usually leading to an increase of the loss, i.e.

$$\mathbf{L}_Q := \mathbf{L}(\mathbf{w}_Q; \mathbf{x}, \mathbf{y}) > \mathbf{L}(\mathbf{w}; \mathbf{x}, \mathbf{y}). \tag{2}$$

Because, after training, the weights are set such that $\mathbf{L}(\mathbf{w})$ is approximately (locally) minimal, quantization of the weights will generally increase the loss and therefore, given a meaningful loss function, decrease the performance of the ANN, e.g. the accuracy in classification tasks.

Therefore methods are required to minimize

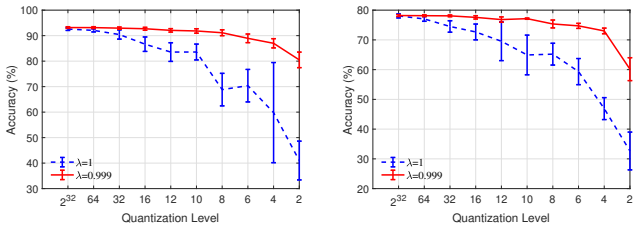
$$\Delta\mathbf{L} := \mathbf{L}_Q - \mathbf{L}. \tag{3}$$

Hessian-aware or loss-aware quantization has been proposed for this purpose [13]–[16]. These methods attempt to apply quantization such that the distortion of the weights due to quantization is not minimized with respect to the mean-squared error, the most common metric, but with respect to the respective loss function.

In contrast, in this work we investigate using curvature information in the loss function to directly yield weights $\mathbf{w_{opt}}$ such that the network is intrinsically more robust to the error $\Delta\mathbf{w}$ of the weights induced by quantization or pruning.

Curvature is a property of curves and, more prominently, surfaces [17], a special case being the loss surface described by $\mathbf{L}(\mathbf{w})$. Obviously, the impact of $\Delta\mathbf{w}$ decreases with decreasing curvature of $\mathbf{L}(\mathbf{w})$. Therefore we would like the training of the ANN to yield weights $\mathbf{w}_{opt}$ which minimize $\mathbf{L}(\mathbf{w})$, and therefore the performance of the ANN, but concurrently minimize the curvature of $\mathbf{L}$ (interpreted as a surface) at and around this very point $\mathbf{w}_{opt}$. This should yield ANNs intrinsically more robust to changes of theirs weights and thus intrinsically more robust to quantization or any other compression algorithm that affects the weights of an ANN after training. It turns out that this can be achieved by using a weighted average of the original loss function and a term related to the curvature of the loss function.

As a proof of concept, three small ANNs, all based on LeNet [18], were trained on the MNIST [19] and FashionMNIST [20] dataset and their performance before and after quantization investigated with respect to the training method. Two baseline methods were considered and compared to the proposed approach with respect to classification accuracy achieved across compression ratios. Additionally, the potential of combining our approach with pruning was investigated. The proposed idea is, when implemented with standard frameworks like pytorch, very expensive to compute due to the large number of second order derivatives required. This issue is discussed at the end of this manuscript.



(a) LeNet-496 on MNIST (b) LeNet-496 on FashionMNIST

Fig. 2: Mean accuracy and standard deviation of the LeNet-496 model on a) MNIST and b) FashionMNIST across codebook size of the quantizer when training with $\lambda = 1$ and $\lambda = 0.999$ with λ as in Eq. 5. The value 2^{32} denotes the uncompressed networks. A substantial decrease of the mean test accuracy occured only for codebook sizes of 16 or less suggesting to limit the investigations to these codebook sizes.

The structure of this manuscript is as follows: Section II explains the datasets used as well as in detail the general idea of this work. Furthermore the baseline methods are outlined. Section III shows the impact of quantization on the classification accuracy of the ANNs for all methods considered. The results as well as methods to decrease the computational complexity are discussed in Section IV and the paper is concluded in Section V.

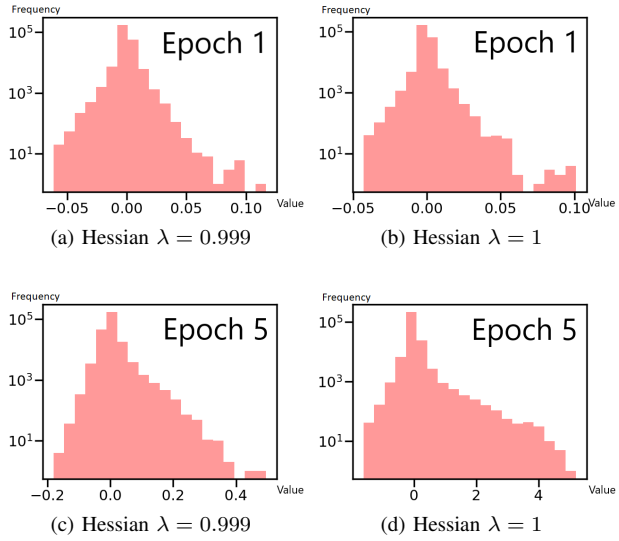
II. Methods and Materials

A. Proposed Method

The approach investigated in this work makes use of the fact that quantization acts effectively like an addition of a vector to the weights of a given ANN. Assuming for simplicity the existence of only one minimum, then, the smaller the curvature of the loss function especially around that minimum, the smaller the increase of the loss will be if we move from the

TABLE I: LeNet-model with 2026 parameters. For each convolution layer, ReLu is applied as activation function. Other models were structurally identical but with smaller number of weights per layer. The number of weights specified includes biases.

Layer	Weights	Kernel Size	Stride	Channels In	Channels Out
Convolutional Layer	208	5×5	2	1	8
Max Pooling	-	2×2	1	-	-
Convolutional Layer	1168	3×3	1	8	16
Max Pooling	-	2×2	1	-	-
Fully Connected	650	-	-	-	-



(a) Hessian $\lambda = 0.999$ (b) Hessian $\lambda = 1$

(c) Hessian $\lambda = 0.999$ (d) Hessian $\lambda = 1$

Fig. 3: Histograms showing the distribution of the values of the hessian matrix for the LeNet-496 model with (a,c) $\lambda = 0.999$ and (b,d) $\lambda = 1$ after one and after five epochs of one example training. For $\lambda = 0.999$ the values of the hessian matrix remain tightly distributed around zero.

location of the minimum A to some other location B. Due to this, it is natural to consider the curvature of the loss surface. It is evident that the value of the loss function, and thus the entire ANN, should be (locally) more robust towards quantization of the ANN's weights if the loss surface is of smaller curvature, i.e. more flat with respect to the weights. The general idea is conveyed in Figure 1. While possibly achieving greater performance, i.e. a smaller loss, before quantization, after quantization it is possible that the lesser curved loss yields better performance.

There are several measures of curvature known and investigated in mathematics like gaussian curvature, mean curvature and so on. Gaussian curvature seems to be the curvature of interest in this regard, albeit, due to its locality, total curvature, i.e. the integral across some surface area A of the gaussian curvature, could be even more suitable.

The gaussian curvature $GC(\omega)$ of a (loss) surface $\mathcal{L}$ can be calculated according to [21]

$$GC(\omega) = \frac{\det \mathbf{H}_{\mathcal{L}}(\omega)}{(||\nabla \mathcal{L}(\omega)||_2^2 + 1)^2} \tag{4}$$

which becomes $\det \mathbf{H}_{\mathcal{L}}(\omega)$ at local optima. $\mathbf{H}_{\mathcal{L}}(\omega)$ is the hessian at point ω corresponding to the loss surface $\mathcal{L}$. While this is in principle implementable in standard frameworks, the determinant is difficult to tract. Sufficient for the vanishing of $GC(\omega)$ is the vanishing of $||\mathbf{H}||_{2,2}$, the Frobenius norm of the hessian matrix, which can be implemented in a straightforward manner in the training of artificial neural networks.

In this work we considered standard classification problems to test our idea and thus used the cross-entropy loss function $\mathbf{L}_{CE}$ to measure the performance of the ANNs. Therefore, to consider the curvature of the cross-entropy loss surface during training, the loss function was changed to

$$\mathbf{L}_{new} = \lambda \cdot \mathbf{L}_{CE}(\omega) + (1-\lambda) \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\frac{\partial^2 \mathbf{L}_{CE}(\omega)}{\partial \omega_i \partial \omega_j}\right)^2 \tag{5}$$

$$\equiv \lambda \cdot \mathbf{L}_{CE}(\omega) + (1-\lambda)||\mathbf{H}||_{2,2}^2 \tag{6}$$

with $\lambda \in [0, 1]$ and the number of weights N of the ANN. $||\mathbf{H}||_{2,2}$ is the Frobenius norm of the hessian matrix of the cross-entropy loss $\mathbf{L}_{CE}(\omega)$. $\lambda = 1$ yields the vanilla cross-entropy loss, while $\lambda < 1$ adds curvature information to the loss function.

B. Datasets and Neural Networks

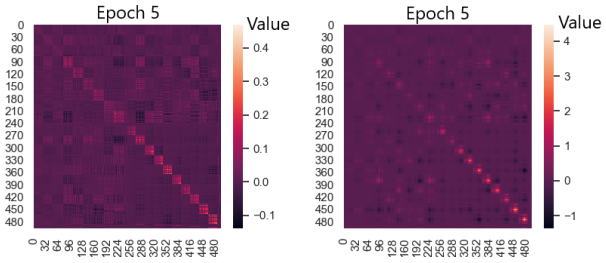
The approach was tested using two different image datasets, the well known MNIST dataset [19] consisting of images of handwritten digits and the FashionMNIST [20] consisting of images of common cloths. While MNIST is certainly the most used dataset to test classifiers, FashionMNIST was selected due to consisting of ten classes like MNIST but being considered somewhat more difficult to classify. This allowed investigating the impact of the dataset on the proposed compression method.

Both datasets have a dedicated training set consisting of 60,000 images and a dedicated test set consisting of 10,000 images.

The neural networks used were based on LeNet-5 [18] but used three layers to reduce the computational complexity and had 496, 1306 and 2026 parameters/weights, respectively. These models were respectively labeled LeNet-496, LeNet-1306 and LeNet-2026. The general structure, identical for all three models, is given in Table I with the number of weights corresponding to LeNet-2026.

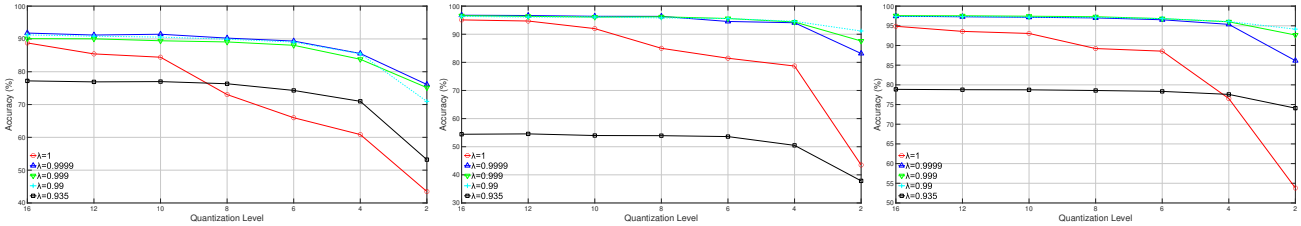
C. Baselines

Two baseline methods were considered: direct quantization of the weights of the ANNs trained using the regular cross-



(a) Hessian $\lambda = 0.999$ (b) Hessian $\lambda = 1$

Fig. 4: Heatmaps showing the elements of the hessian matrix of the LeNet-496 model after training for 5 epochs on MNIST using (a) $\lambda = 0.999$ and thus introducing curvature in the training and (b) $\lambda = 1$. No clear change in the pattern was observed and both hessians see an approximately linear increase of values of the diagonal elements.



(a) LeNet-496 on MNIST (b) LeNet-1306 on MNIST (c) LeNet-2026 on MNIST

Fig. 5: Mean test accuracies of the three models investigated across five repetitions for several values of λ on MNIST. Little difference between λ values was observed for eight and more quantization levels. However, a significant difference was observed at two quantization levels, where the mean test accuracy could vary by about 5 – $7\,\%$. $\lambda = 0.935$ was included to show how the proposed method breaks down if the curvature is emphasized too much in the loss function.

entropy loss, i.e. using Eq. 5 with $\lambda = 1$, also called naive training method, and pruning with fine-tuning of the weights. For pruning, using Pytorch's prune function a percentage p was specified and $p\,\%$ of the weights with the smallest magnitude were then set to zero after training using Eq. 5 with $\lambda = 1$. Then, to optimize the performance, the ANN was fine-tuned, i.e. was trained again for a few epochs using the smaller network obtained through pruning. Fine-tuning was always peformed using $\lambda = 1$.

D. Quantization

Initially, minimum mean-square error quantization using the well known Max-Lloyd algorithm and uniform quantization were investigated. However, uniform quanization was found to perform considerably better and thus was the only quantization type investigated further. The stepsize of the mid-rise uniform quantizer was optimized for a given number of quantization levels using all the weights of the ANNs after training, i.e. weights were not distinguished by e.g. the layers they belonged to.

E. Training and Evaluation

Each ANN was trained five times, i.e. five random initializations with subsequent training were performed, using the adam solver with a learning rate of 0.001 and a weight decay of 0.0001 unless otherwise specified as well as a batchsize of 1024. A $80\,\%/20\,\%$ training/validation split of the dedicated training data was used, while all results reported were achieved on the dedicated test sets. All ANN were trained for 30 epochs.

The compression ratio in this work was defined as the ratio of the model size before and after compression.

A network with n weights, each weight encoded with b bits before quantization, requires a total storage of nb bits. After quantization with k clusters/quantization levels, only $log_2(k)$ bits are needed to encode the index, the total storage is thus $nlog_2(k)$. In addition, the codebook of the quantizer has to be transmitted. For uniform quantization, the codebook is determined once the stepsize and the center of the quantizer is known. Thus, $2b$ additional bits have to be transmitted. The compression ratio r_Q was therefore computed according to

$$r_Q = \frac{nb}{nlog_2(k) + 2b}. \tag{7}$$

$b = 32$ was used in this work which is the default precision in Python. For pruning, the compression ratio r_p was calculated according to

$$r_P = \frac{N}{(N - B)(1 - pr) + B} \tag{8}$$

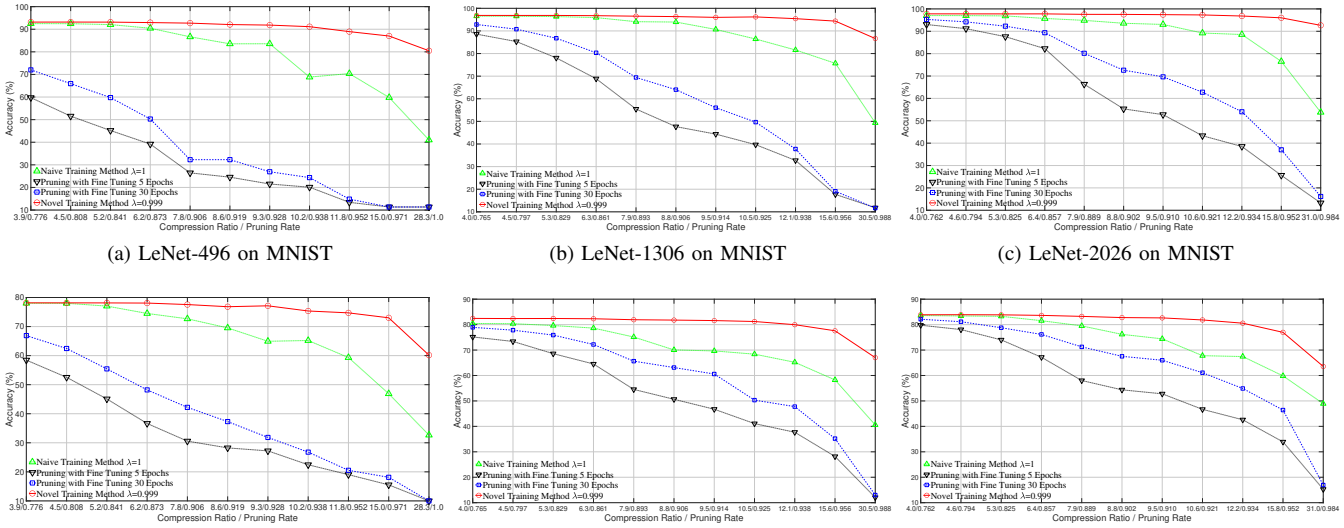
with the total number of weights N, the total number of biases B and the pruning rate $pr \in (0, 1]$. The biases were excluded from pruning as it is common practice due to their usually large impact on the ANNs performance, and thus $pr = 1$ corresponds to the pruning of all network weights except for the bias. $pr = 1$ was used when the desired compression ratio r_{target} would require to prune more than $N - B$ weights. This occured only for LeNet–496.

When comparing pruning to quantization the networks were first quantized with a given number of quantization levels and the corresponding compression ratio r_Q was computed. Then, the required pruning rate pr was determined such that the compression ratio r_P was as close to r_Q as possible.

III. Results

First, the rough range of interest of the quantization levels were investigated. For several quantizer levels ranging from 2 to 64, including the uncompressed reference denoted by 2^{32}, the respective mean test accuracy across five repetitions of the LeNet-496 model on MNIST is given in Fig. 2 together with the standard deviation. A major drop in accuracy started to occur for all networks and datasets at about 4 bit or 16 quantization levels. The standard deviation was found to be greatly reduced when training with $\lambda < 1$ and the results were found to very repeatable. Additionally, standard deviation decreased with increasing number of parameters and tended to increase from MNIST to FashionMNIST.

To investigate the impact of the loss function on the networks in more detail, the distribution of the values of the hessian matrix when training with $\lambda = 0.999$ and $\lambda = 1$ was



(a) LeNet-496 on MNIST (b) LeNet-1306 on MNIST (c) LeNet-2026 on MNIST

(d) LeNet-496 on FashionMNIST (e) LeNet-1306 on FashionMNIST (f) LeNet-2026 on FashionMNIST

Fig. 6: Mean test accuracy of the three models investigated across five repetitions and both datasets after quantization and training with $\lambda = 1$ and $\lambda = 0.999$. Additionally, results for pruning after five and 30 epochs are depicted, too. For pruning, $\lambda = 1$ was used. Quantization proved to considerably superior with respect to the achieved mean test accuracy after quantization/pruning of the networks. The mean test accuracy on MNIST for the LeNet-2026 model was about $80\,\%$ larger after quantization than after pruning at a compression ratio of 31.0.

assessed for the LeNet-496 model expecting a tight distribution of the values around zero. This was confirmed and an example comparing the distributions of the values of the hessian after one epoch and after five epochs is depicted in Fig. 3. A heatmap of the hessian matrix is shown in Fig. 4. Interestingly, the diagonal elements appeared to linearly increase for either method. No clear difference in the heatmap patterns were apparent, only the magnitude of the weights was one order less when training with $\lambda = 0.999$.

Initially, values of λ ranging from 0 to 1 in steps of 0.1 were investigated. But values below 0.9 turned out to generally emphasize the curvature too much resulting in very poor accuracies prior to quantization. Thus Fig. 5 depicts for all three models the mean test accuracies across five repetitions on MNIST for values of λ very close to 1 as well as $\lambda = 0.935$. The latter value was chosen as an example where the benefit of reduced curvature turned into affecting the overall accuracy negatively. The sweet spot at which the proposed method broke down appeared to be around $\lambda = 0.95$. $\lambda = 0.99$ and $\lambda = 0.999$ were found to perform the best across all models and datasets with a minor advantage for $\lambda = 0.999$ on the FashionMNIST for the LeNet-496 model.

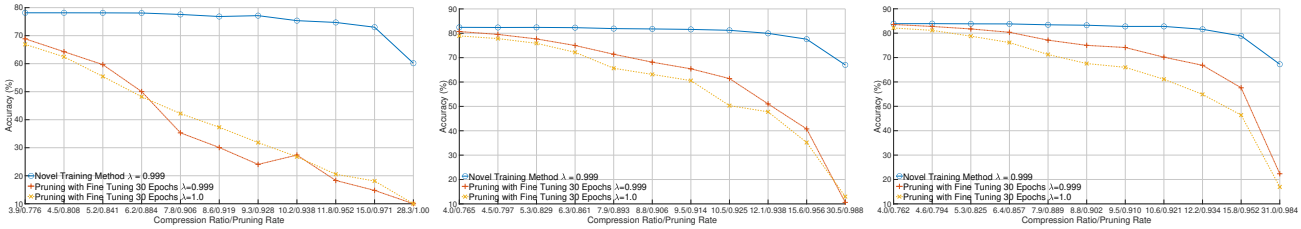
For reasonable values of $\lambda < 1$ the mean accuracy after quantization was found to be up to $40\,\%$ better than when training with $\lambda = 1$, i.e. without considering curvature, on MNIST and up to $25\,\%$ better on FashionMNIST irrespective of the network model.

The decrease in mean test accuracy with decreasing number of quantization levels was observed to decrease with increasing number of parameters of the networks. On MNIST and $\lambda = 0.99$, for the LeNet-496 the mean test accuracy decreased from about $93\,\%$ for the uncompressed network to about $75\,\%$ at two quantization levels or 1 bit. For the LeNet-1306, the mean test accuracy decreased from about $96\,\%$ to $91\,\%$ at two quantization levels and for the Lenet-2026 from about $97\,\%$ to $95\,\%$ at two quantization levels.

On FashionMNIST and $\lambda = 0.999$, for the LeNet-496 the mean test accuracy decreased from about $78.2\,\%$ for the uncompressed network to about $60.2\,\%$ at two quantization levels. For the LeNet-1306, the mean test accuracy decreased from about $81.7\,\%$ to $67.0\,\%$ and for the Lenet-2026 from about $83.9\,\%$ to $63.6\,\%$.

Fig. 6 depicts mean test accuracies for all models and both datasets and all compression methods investigated. For pruning, $\lambda = 1$ was used before and after pruning the networks and accuracies for fine-tuning with five and 30 epochs are given. Generally, pruning was found to perform the worst out of all methods investigated but improved considerably with increasing model size.

On MNIST, the difference between quantization for $\lambda = 0.999$ and pruning was up to $80\,\%$ for LeNet-2026 and about $47\,\%$ on FashionMNIST for two quantizer levels corresponding to a compression ratio $r_Q = 31.0$. However, pruning can be considered a special kind of quantization. Thus it was expected to improve when λ was set to values smaller than one. This assumption was confirmed by our investigations and results are given for all models on FashionMNIST in Fig. 7. Except for the LeNet–2026 model for two and four quantizer levels corresponding to compression ratios r_Q of 31.0 and 15.8, respectively, pruning after training with $\lambda = 0.999$



(a) LeNet-496 on FashionMNIST (b) LeNet-1306 on FashionMNIST (c) LeNet-2026 on FashionMNIST

Fig. 7: Mean test accuracy of the three models investigated across five repetitions on FashionMNIST after pruning and training with $\lambda = 1$ and after pruning and training with $\lambda = 0.999$. The subsequent fine-tuning was always performed using $\lambda = 1$. Additionally, mean test accuracy after quantization and training with $\lambda = 0.999$ is shown as well. Except for few cases pruning after training with $\lambda = 0.999$ achieved considerably increased mean test accuracy with up to $11\,\%$ increased mean test accuracy for the LeNet-2026 model at a pruning rate of 0.934. The improvements in mean test accuracy were observed to be on average greater for the more complex FashionMNIST.

yielded superior accuracies of between about $1\,\%$ and $11\,\%$ compared to training with $\lambda = 1$, the precise value depending on model and compression ratio. Similar results were achieved on MNIST but were left out to shorten the presentation of the results somewhat. Some results for pruning and quantization are summarized in Table II. For simplicity, compression ratios in the table are given which correspond to the LeNet-1306 model. The corresponding true compression ratios for the LeNet-496 and LeNet-2026 model were slightly different as can be seen in e.g. Fig. 7.

Finally it was investigated, using the LeNet-496 model and the MNIST dataset, if the training could be sped up by first training with $\lambda = 1$, avoiding the costly computation of the hessian and subsequent training for a few epochs with $\lambda < 1$ potentially considerably improving the robustness towards quantization. Thus the LeNet-496 model was trained for 30 epochs using $\lambda = 1$ and then was trained further for one to ten epochs using $\lambda = 0.9999$ with a learning rate of 0.008. These particular values were found to yield superior performance for this specific task after a few initital tests and sweeps across possible parameter values. A larger learning rate should be able to speed up training at the cost of potentially worse final accuracy. The result is depicted in Fig. 8.

As references, the mean test accuracies achieved after 30 epochs of training with $\lambda = 1$ as well as 30 epochs of training with $\lambda = 0.999$ were included, too. After only one epoch, the accuracy improved by about $16\,\%$ for two quantization levels. However, the accuracy did not increase considerably for more epochs, actually going down slightly for some. Furthermore, the accuracy of this speed up was considerably lower than when training right from the start with $\lambda = 0.999$ and a learning rate of 0.001, which achieved a mean test accuracy of $74.5\,\%$ compared to $59.5\,\%$ for the speed up after one epoch. Nonetheless, for six or more quantizer levels the mean test accuracy of the speed up almost reached the mean test accuracy when training with $\lambda = 0.999$ from the start.

IV. Discussion

This work investigated the feasibility of training artificial neural networks such that the curvature is minimized yielding artificial neural networks automatically more robust to quantization. It was found that considering the Frobenius norm of the hessian of the cross-entropy loss function can make artificial neural networks considerably more robust to quantization, especially at small numbers of quantization levels and also improves performance after pruning.

The reduced impact of quantization on the mean test accuracy on MNIST and FashionMNIST with increasing modelsize which can be seen in Fig. 6 can be explained by increasing overparametrization of the networks. As expected, this impact is greater for the more complex FashionMNIST compared to the classic MNIST dataset. A smaller neural network is expected to lose more of its expressive power than a large neural network. One would expect this impact to decrease further for larger networks which is in line with results obtained by [14], who considered loss-aware quantization, a method closely related to ours, who saw a decrease of accuracy at 1 bit quantization of only $-0.1\,\%$ for considerably larger networks.

The lower mean test accuracy of the LeNet-2026 model compared to the LeNet-1306 model on FashionMNIST can be explained by the number of epochs we trained. By loss curve inspection we noticed that for FashionMNIST the LeNet-2026, and less so the LeNet-1306, network had not completely converged. Because training with $\lambda < 1$ was very time consuming, and the focus did not lie on maximizing accuracy but rather investigating the impact of network compression, for comparability we decided to not increase the number of epochs further.

While the proposed method improved the mean test accuracy after pruning as well, its benefit was not as great as for quantization. In a few cases a decrease in performance was even observed. One reason for its reduced effectiveness could be the reduced network size after training. In contrast to quantization this can limit greatly the network's expressive

power irrespective of the retained resolution of the weights of 32 bits. The few cases of reduced network performance after pruning could be due to the locality of our method. Our training method minimizes the local curvature only and cannot guarantee that the curvature remains small in wider areas around the weights $\mathbf{w}_{opt}$ found after training. To do this, total curvature or similar quantaties would have to be considered. However, for quantization, a decrease in performance when training with the proposed method was never observed.

While values of λ larger than about 0.95 generally, for all models and datasets, yielded at least somewhat improved robustness towards quantization, the question regarding generalization of reasonable choices of λ depends on the generalization of the elements of the hessian $\mathbf{H}$. This itself depends on the network structure and the data. While datasets or batches of data can be normalized and thus are potentially having little impact on the choice of λ, the structure of the network could largely affect the values of the hessian. Here a normalization of the hessian could be beneficial in the future.

The most important feature of the proposed approach is that it can improve any post-training compression method that affects the weights of the artificial neural network. Due to the intrinsic robustness to weight-changes due to the smaller curvature of the loss surface the proposed method could have very broad application. This makes it different to all other methods known to the authors.

A. Comparison to Other Work

As far as we know nobody attempted to minimize the curvature of the loss function during training. Also, as we applied our method to rather small networks, comparison is difficult. The publication most similiar to ours is [16] as well as [14], [22], [23]. All of these attempt to make quantization loss-aware and not in a sense the other way around, i.e. the loss quantization-aware, which is another way of looking at curvature-aware training. While quantization-aware training was mentioned in [24], it means introducing quantization during training which is entirely different from our approach. [16] finds that the curvature of the loss surface becomes very steep at two and four bits and thus quantization had a very large impact on the network performance. This is exactly where we see the greatest benefit of our approach in line with their findings. Hou et al. [14], in their best result, saw a decrease of only $-0.1\,\%$ at 1 bit quantization when using loss-aware quantization. This minor decrease corresponds well to the general trend seen in our work, where the decrease of the mean test accuracy decreased with increasing model complexity and was only $2\,\%$ on MNIST for the LeNet-2026 model. While the drop in accuracy was larger on the more complex FashionMNIST, considerably larger networks are expected to see a considerably lower impact of their performance due to quantization and their increasing overparametrization.

Most importantly, the proposed method allows to be combined with the other compression techniques discussed here. Due to the proposed novel loss function, the ANN is brought automatically into a state more robust to quantization (or any other method affecting its weights) but no quantization method is specified and thus any could be applied.

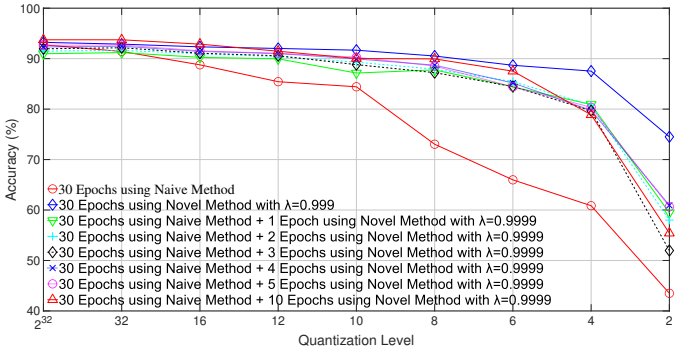


Fig. 8: Depicted are the mean test accuracies across five reptitions using the LeNet-496 model and the MNIST dataset when training first with $\lambda = 1$ and subsequently training for one to five and ten additional epochs using $\lambda = 0.9999$, i.e. curvature is introduced in a secondary training step. The learning rate for the secondary training step was set to 0.008. As reference, mean test accuracies when training for 30 epochs using $\lambda = 1$ only as well as 30 epochs using $\lambda = 0.999$ are depicted, too. After only one epoch of training with $\lambda = 0.9999$ the accuracy after quantization with 1 bit or two quantizer levels increased by about $17\,\%$ and slightly more when using six and eight quantizer levels.

B. Dealing with the Computational Burden

As explained, the computation of $||\mathbf{H}||_{2,2}^2$ prevented going beyond small toy networks in this work due to the large number of trainings performed. While networks with at least 10^5 weights are trainable in a reasonable time frame with a direct implementation and regular computers, larger networks pose a bigger problem. The complexity of computing the hessian is $\mathcal{O}(N^2)$, with the number of weights N of the ANN. As the number of weights in state-of-the-art networks can be up to one billion, this would yield a complexity of the order of 10^{18}. Additionally, the operations per element of the hessian could be rather large depending on the architecture, adding considerably to the computational burden.

Nonetheless this approach appears feasible. Super computers achieve above 10^{15} floating point operations per second (FLOP) [25] approaching one exaflop. Furthermore is the rectified linear unit commonly used in deep neural networks which yields a very simple form of the gradient and thus of the hessian. Additionally, the elements of the hessian usually are greatly connected with each other due to the layer structure of artificial neural networks. Finally, alot of computations could be parallelized. If additionally it was recognized that there were certain second order derivates of greater importance than others, the computational costs would decrease considerably. As it is not necessary to store all elements of the hessian, as the weight update according to Eq. 5 can be done componentwise, memory requirements are not a large issue.

V. Conclusion

This work proposed a curvature-aware cross-entropy loss function for the training of quantization-robust artificial neural networks (ANNs). For three small LeNet-based toy networks and on two datasets, namely MNIST and FashionMNIST, the ANNs after training with the proposed loss functions were compared with respect to the decrease of the mean test accuracy to two baseline methods, pruning as well as quantization after training with the regular cross-entropy loss.

It was found that adding curvature information in the training ("curvature-aware training") considerably increased the robustness of the ANN to quantization of their weights, increasing the mean test accuracy at two quantization levels by up to about $47\,\%$ compared to training without curvature information. The proposed method considerably outperformed the baseline pruning method by about $40\,\%$ to about $80\,\%$ at two quantization levels. Pruning after curvature-aware training was also found to perform better than curvature-unaware training, increasing in mean test accuracy by $5 - 11\,\%$. Additionally, training for only one additional epoch with the proposed loss function after training with the regular cross-entropy loss was found to improve the mean test accuracy after quantization by about $17\,\%$ at 1 bit.

References

[1] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[2] M. Zeman, E. Osipov, and Z. Bosnić, "Compressed superposition of neural networks for deep learning in edge computing," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[3] T. Tan and G. Cao, "Efficient execution of deep neural networks on mobile devices with npu," in *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (Co-Located with CPS-IoT Week 2021)*, ser. IPSN '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 283–298. [Online]. Available: https://doi.org/10.1145/3412382.3458272

[4] Motion Picture Expert Group. (MPEG), "Neural Network Compression for Multimedia Applications," International Organization for Standardization, Geneva, CH, Standard ISO/IEC 15938-17:2021, 2021.

[5] J. O. Neill, "An overview of neural network compression," *arXiv preprint arXiv:2006.03669*, 2020.

[6] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2017.

[7] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221010894

[8] K. Ullrich, E. Meeds, and M. Welling, "Soft weight-sharing for neural network compression," *arXiv preprint arXiv:1702.04008*, 2017.

[9] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, "Tensorizing neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 442–450.

[10] T. Laude and J. Ostermann, "Neural network compression using transform coding and clustering," in *NIPS 2018: Compact Deep Neural Network Representation with Industrial Applications Workshop*, 2018. [Online]. Available: https://openreview.net/pdf?id=HklTRIB957

[11] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531

[12] M. Loni, H. Mousavi, M. Riazati, M. Daneshtalab, and M. Sjödin, "Tas: Ternarized neural architecture search for resource-constrained edge devices," in *Proceedings of Design, Automation Test in Europe Conference Exhibition DATE'22*, vol. 1, 2022.

[13] Z. Dong, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer, "Hawq: Hessian aware quantization of neural networks with mixed-precision," *arXiv preprint arXiv:1905.03696*, 2019.

[14] L. Hou, Q. Yao, and J. T. Kwok, "Loss-aware binarization of deep networks," *arXiv preprint arXiv:1611.01600*, 2018.

[15] A. Zhou, A. Yao, K. Wang, and Y. Chen, "Explicit loss-error-aware quantization for low-bit deep neural networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9426–9435.

[16] Y. Nahshan, B. Chmiel, C. Baskin, E. Zheltonozhskii, R. Banner, A. M. Bronstein, and A. Mendelson, "Loss aware post-training quantization," *Machine Learning*, vol. 110, no. 11, pp. 3245–3262, 2021.

[17] S. Kobayashi, *Differential Geometry of Curves and Surfaces*. Springer, Singapore, 2019. [Online]. Available: https://doi.org/10.1007/978-981-15-1739-6

[18] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 12 1989. [Online]. Available: https://doi.org/10.1162/neco.1989.1.4.541

[19] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[20] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[21] R. Goldman, "Curvature formulas for implicit curves and surfaces," *Computer Aided Geometric Design*, vol. 22, no. 7, pp. 632–658, 2005, geometric Modelling and Differential Geometry. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167839605000737

[22] Z. Qu, Z. Zhou, Y. Cheng, and L. Thiele, "Adaptive loss-aware quantization for multi-bit networks," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7985–7994, 2020.

[23] Y. Li, R. Gong, X. Tan, Y. Yang, P. Hu, Q. Zhang, F. Yu, W. Wang, and S. Gu, "Brecq: Pushing the limit of post-training quantization by block reconstruction," *arXiv preprint arXiv:2102.05426*, 2021.

[24] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.

[25] P. Kogge, "The tops in flops," *IEEE Spectrum*, vol. 48, no. 2, pp. 48–54, 2011.

TABLE II: Accuracy of all models when using 1 bit to 3 bit quantization for compression on the (a) MNIST and (b) FashionMNIST dataset as well as corresponding results for pruning for (c) MNIST and (d) FashionMNIST. The corresponding compression ratios are 30.5, 15.6 and 10.5 (for the LeNet-1306 model; slightly different for the other models) which are specified in the table headings for all models for simplicity. For pruning, because no reasonable results were achieved at the highest compression ratio, results are specified for the compression ratios of 15.6, 12.1 and 10.5. Results are given for $\lambda = 1$ and $\lambda = 0.999$ in the form $Accuracy_{\lambda=1}/Accuracy_{\lambda=0.999}$ in percent.

(a) MNIST (Quantization)

Model	30.5	15.6	10.5
LeNet-496	41.01/80.49	59.81/87.0	68.83/91.14
LeNet-1306	49.35/86.63	75.67/94.34	86.42/96.2
LeNet-2026	53.8/92.66	76.52/96.02	89.22/97.3

(b) FashionMNIST (Quantization)

Model	30.5	15.6	10.5
LeNet-496	32.65/60.16	46.9/73.01	65.2/75.36
LeNet-1306	40.57/67.02	58.26/77.58	68.39/81.22
LeNet-2026	48.92/63.57	59.84/76.96	67.79/81.84

(c) MNIST (Pruning)

Model	15.6	12.1	10.5
LeNet-496	11.35/12.35	14.83/15.04	24.35/24.78
LeNet-1306	19.09/28.98	37.78/42.83	49.67/54.89
LeNet-2026	30.56/37.04	47.58/54.03	60.18/62.77

(d) FashionMNIST (Pruning)

Model	15.6	12.1	10.5
LeNet-496	18.12/14.77	20.53/18.32	26.78/27.38
LeNet-1306	35.19/40.75	47.77/51.03	50.32/61.42
LeNet-2026	46.42/57.6	54.9/66.83	61.1/70.15