Proseminar Wide Reduced-Precision Networks

Jan Langbecker
Supervisor: Haibin Zhao

Karlsruher Institut für Technologie, 76131 Karlsruhe, Germany
jan.langbecker@student.kit.edu

Abstract. One major problem of neural networks that often hinders them impractical outside of an experimental and scientific context are their resource requirements both during training and when applying the model after Training. There are many papers about techniques that aim to lessen some of the resource requirements of neural networks, many do so by quantising them thereby reducing the precision or limiting the range of input data and weights to the neural network when running the model.
One such quatisation technique is called "Wide Reduced Precision Networks"[1]. This paper compares WRPN to other similar techniques [7][3][8] in terms of their effectiveness in achieving their respective goals of reducing resource consumption. And explains key concepts and terms required to understand these techniques.

1 Introduction

Motivation In recent years there has been a trend of drastically increasing the size of the input layer of neural networks as well as their depth - especially with convolutional neural networks used in tasks like computer vision. This causes a massive increase in memory requirements and computational requirements to run and train such neural networks as well as storage requirements to store such a neural network. This causes the issues of such neural networks being difficult to deploy in the real world[10] as well as them being expensive to train.

One proposed solution to this problem is network quantisation: it aims to reduce those requirements while having a limited impact on the accuracy of neural networks. In recent years a lot of different quantisation techniques, each with their own set of advantages and disadvantages have been introduced. One such technique are Wide Reduced Precision Networks [1] which tries to overcome the main issues of previous quantisation techniques: Mainly them either having a major reduction in resource requirements or minimal loss in accuracy but not both.

In this paper we will explain terminology and concepts required to understand WRPN and quantisation in general, the quantisation technique used in WRPN and the quantisations schemes used in other similar techniques. We will also

compare WRPN to other previously used quantisation techniques on which WRPN are based.

2 Basic Terminology and Concepts

2.1 Neural networks

A neural network is a mathematical model that maps input(s) to a set of output values. It consists of 3 types of layers: An input layer containing the input value(s) to the neural network, an output layer containing the output values and at least one hidden layer, each consisting of nodes (values), and connections between nodes of the current layer and the nodes of the next layer, each connection having an associated weight. Those weights are also referred to as model parameters.

The neural network calculates the values of one layer based on the values of the previous layer and the weights of their connections. Most of the time this is done by simple multiplication and accumulation or in other words matrix multiplication, however other types of functions such as convolution can be applied as well. This is done for each layer until the output layer is reached, the values that are calculated as values of the output layer are the outputs of the entire network.[4]

2.2 Convolutional Neural Networks

Convolutional Neural Networks or CNN's are a type of neural network in which some layers calculations are not performed by the usual matrix multiplication used in general neural networks, but instead by performing convolution. This proved beneficial in a lot of classification tasks - especially in image recognition and natural language processing.[9]

2.3 Convolution

The discrete convolution of two complex-valued functions is defined as [2]:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

For our intents and purposes the two functions f and g map complex coordinates to one component of the inputs of the first function and to the weights of the layer for the second function.

2.4 Backpropagation

Backpropagation is the process used to improve the neural network during "training". It aims to adapt the weights of a layer to reduce the mean error of the network. This reduction of the mean error of the network is usually achieved using gradient descent.

2.5 Quantisation/Quantized Neural Networks

Quantisation is the process of mapping a larger input set to a smaller output set. The output often has a finite number of elements. In the context of neural networks quantisation can be applied to either input values, output values, network weights or any combination of these for any of the layers of the neural network.

"While it is possible to train a quantized [neural network] from scratch, the majority of research has shown that starting with a pre-trained model to be more effective at minimising accuracy loss when quantizing a [neural network]"[10]. This means that quantisation is almost always applied to pre-trained models (Meaning that the Network already has been trained using training data). Quantisation is usually either applied as so called quantisation-aware training or as post-training quantisation. While post-training quantisation involves just the quantisation of the network itself, quantisation-aware training also involves additional training after the quantisation "so that [the neural network can] adjust [..] to the newly quantized values."[10]

The goal of quantisation is to reduce the storage and computational requirements of neural networks at the cost of a loss in precision of the output layer. [10] Good quantisation techniques have a very limited precision loss impact on the output, while achieving a major reduction in either storage requirements or computational requirements or both.

A neural network with quantisation is also referred to as a quantized neural network. Quantisation can be achieved using a variety of techniques, each with their own output set and specific advantages and disadvantages.

2.6 The Clipping Function

One very important and often used function for the quantisation of a neural network is the clip-function [7]:

$$clip(x, u, l) = max(l, min(u, x)) \tag{1}$$

The clip function limits the set of a value of x to a range between an upper bound u and a lower bound l by "clipping" values above u to u and values below l to l or in other words: Replacing values larger than u with u and values smaller than l with l.

2.7 Reduced-Precision Networks

Reduced-precision networks are a special kind of quantized neural network, where the output set of the quantisation is finite and set up in a way causing the individual output values to be less precise than the input values. This is for example the case when mapping floating point values to fixed point values or integers. Or when limiting the range of possible integers, e.g. when applying the clipping function. [10]

2.8 Filter Maps

Filter maps are convolutional layers of a neural network that act as filters. They are often applied multiple times in succession and often there are multiple different filters applied after each other. They differ from general convolutional layers because they are not fully connected meaning not every input influences every output and some outputs may not even be connected at all. Filter maps enable the input data to conventional layers in convolutional neural networks to be more usable since after applying a good filter the parts of the input data relevant to the classification will be more highlighted, while those data parts less important will be less dominant in the overall result.

2.9 Straight-Through Estimator

If the weights of a neural network are quantized in a way that limits the amount of possible values to a set of finite values, we got a problem when trying to train the network e.g when applying quantisation-aware training [10]:
There is no meaningful gradient function that we can apply during backpropagation to "train" the network. The straight-through estimator as explained in [11] is a solution to this problem. It is an estimated continuous gradient function which can be used to apply gradient descent and train the network.

2.10 AlexNet

AlexNet is a specific eight layers deep convolutional neural network originally trained for the ImageNet competition on image recognition which it subsequently won. It has since them been trained on different datasets and used in research papers as one example the demonstrate the effect of techniques introduced in the papers.[6]

2.11 ResNet

A ResNet or a Residual Neural Network is a specific kind of neural network which has some skip connections that "bypass" some of the layers of the neural networks, this means the for some values the original value is added to the output of one or more subsequent layers before passing the result of the addition

as an input to the next layer. They were introduced after the winner of the next ImageNet competition after AlexNet won by including more layers - which causes issues because errors in one layer can compound with each following layer. ResNet's won the following ImageNet competition.[5]

3 Wide Reduced-Precision Networks

3.1 Explanation

Wide reduced-precision networks or WRPN are one proposed solution to the major resource requirements of modern neural networks, and more specific convolutional neural networks, during execution as well as to storing the model. The main issue is that modern neural networks use a large input layer and high precision values in all layers, causing the storage requirements to store the model parameters to be very large.

WPRN's aim is to reduce these requirements like other reduced-precision networks by limiting the range and precision of weights of the neural network. Additionally different from other reduced-precision networks, WRPN also quantize the activations which aims to further reduce the computational requirements of the network. Since this causes a lot of precision loss in the output layer compared to a conventional network WRPN include additional filter maps to compensate for this loss.

"Although the number of raw compute operations increases as [...] the number of filter maps in a layer [is increased], the compute bits required per operation is now a fraction of what is required when using full-precision operations"[1] This enables the use of cheaper and more scalable hardware, which means that WRPN can still be less computationally expensive then conventional full precision networks. WRPN requires some training for the additional filter maps, this can be done before quantizing or by applying quantisation aware training.

WRPN main contribution is the fact that it enables the usage of neural network quantisation with all its benefits while being able to mitigate most of, or in some cases even all the downsides that usually arise when applying quantisation like a large loss in accuracy. This replaces the optimization for minimal loss in accuracy after applying 'traditional' quantisation techniques with the optimisation of the reduction of the computational cost of the neural network. Also it introduces the downsides of having to optimise for each data set individually when applying the procedure, and the possibility of higher computational costs than the non-quantized Network when improperly optimising for the data-set.

3.2 Quantisation Scheme

WRPN quantisation scheme works differently for weights and activations. It's quantisation scheme for weights it is given by:

$$w_q = \frac{1}{2^{k-1}-1} round((2^{k-1}-1) * clip(w, 1, -1))$$

Where w_q is the quantized weight, w the original weight, *round* is rounding half down and *clip* is the clipping function (1).

The quantisation scheme applied to input (activation) values is:

$$a_q = \frac{1}{2^k-1} round((2^k-1) * clip(a, 1, 0))$$

Where a_q is the quantized input value, a the original input value, *round* is rounding half down and *clip* is the clipping function (1).

Because this quantisation scheme is very general which might result in a compounding error due to the increased number of filter maps, which results in worse accuracy of the network than just applying quantisation. [1] circumvents this issue by trying several different combinations of the 3 variables increase in filter maps, bit-length of weights and bit-length of activations.

The upper bounds for these values are indirectly set by the computational cost of the original network (if you include too many filters while not decreasing the bit-lengths enough the performance of the WRPN can be worse than the original network - in which case the whole process was useless). And the lower bounds are directly set by the original amount of filter maps and 1 bit weights and 1 bit activations. This is the lower bound because with less filter maps then the original network the accuracy of the network would suffer even more due to the quantisation then just quantizing the network and when using 1 bit weights/activations we effectively perform binary connect and we can't reduce the precision of weights anymore then binary connect does.

This means that the optimal bit-lengths and scaling factor for filter maps might differ greatly between different data sets which results in a lot of experiments being necessary to apply the WRPN technique effectively.

4 Other Quantisation techniques

The idea of reducing precision of a neural network by quantizing it is not unique to WRPN. This section will try to give an overview of some of the techniques on which WRPN are based. These techniques are similar to WRPN since they limit the precision of network weights by quantizing them, however WRPN uniquely also quantize the input layer, and include extra layers to compensate for that.

4.1 Binary connect

Similar to WRPN, neural networks using BinaryConnect(BC)[7] also reduce the precision of the calculations. However BC's main focus is decreasing computational cost and storage requirements for the model when running it - not during training. BC abuses the fact that the main computationally costly operations during both backward and forward propagation of neural networks are convolutions and matrix multiplications, both of which require a lot of multiply-accumulate operations[7].

BC achieves a significant reduction of these costs by limiting the space of possible weight values of the DNN to {-1,1}. Which makes it possible to replace a lot of multiplications with simple additions and subtractions and weights can now be stored in just a single bit per weight. This enables the use of simpler and easier to scale hardware to compute the neural network. The paper[7] mentions the use of fixed-point adders instead of multiply-accumulators as an example for this.

The major benefit of the simpler hardware required to compute the neural networks using BC is the fact that it is cheaper to run in a deployment. Or a larger model can be run at the same cost as a neural network not running BC. However these benefits come at a significant loss of precision of the neural network.

Quantisation Method: The binary weights for BC are computed from the original weights of a neural network by applying:

$$w_b = \begin{cases} +1 \text{ with probability } p = \sigma(w) \\ -1 \text{ with probability } 1 - p \end{cases}$$

with σ being defined as:

$$\sigma(x) = clip(\frac{x+1}{2}, 0, 1)$$

and clip being the clipping function (1)

4.2 Ternary-weight networks

Ternary-weight Networks (TWN)[3] are based on neural networks using BC, and share its goal of decreasing computational cost and storage requirements of running the model. Similar to Networks using BC this is also achieved by limiting the space of available values that are used as weights in the neural network.

TWN's try to reduce the significant precision loss that occurs in neural networks using BC by also including 0 as an available value for weights. The

main idea here is that this enables Networks to be(have) more similar to conventional neural networks while retaining a similar factor of reduction in computational costs and storage consumption as BC based Networks. TWN's inherit the capability of being computed on easier to scale and cheaper hardware from BC based Networks - since the extra zero states only eliminates some of the values that are added up in a node. And therefore does not significantly affect the computational costs of running the neural network compared to BC.

Since the main goal of TWN's is to more accurately depict the original weights off a DNN then BC, Weights should be chosen by quantising as optimal as possible. Therefore the Ternary Function used to quantize the weights is more complicated then the one used by BC. The paper on TWN's explains how they "solved" this optimisation Problem by modelling it in way that makes it possible to transform to the Problem of optimising a single threshold value which is then used to calculate the ternary weights by applying the following formula:

$$w_t = \begin{cases} +1 \text{ if } w > \Delta \\ -1 \text{ if } w < -\Delta \\ 0 \quad \text{else} \end{cases}$$

Where w is the original weight, w_t the ternary weight and Δ is the threshold value. The threshold value is the solution to the optimisation problem:

$$\Delta = \arg\min_{\Delta>0} \frac{1}{I_\Delta}(\sum_{i \in I_\Delta} |w_i|)^2 \tag{2}$$

Where $I_\Delta := \{i||w_i| > \Delta, w_i \in W\}$ and W is the ordered set of all original weights. The paper mentions that even though there is no straightforward solution to this problem, under the assumption that weights are normal or uniformly distributed the following is a very good approximation [3]:

$$\Delta \approx \frac{0.75}{|W|} * \sum_{w \in \mathrm{W}} |w| \tag{3}$$

4.3 Neural Networks with fixed point weights

Another approach to reduce the resource requirements of neural networks is quantizing their weights by storing them as fixed point numbers with limited precision. This works because "high-precision computation in the context of [machine] Learning is rather unnecessary"[8] since neural networks possess a certain "algorithm-level noise-tolerance"[8]. So we can save a lot of resources when not performing computations at an unnecessary high floating point precision over traditional non-quantized neural networks.

Because using fixed point numbers limits the precision over floating point numbers we need to round to perform the conversion. [8] results show that

choosing the right rounding function has a large influence on how accurate and similar to the non-quantized neural network the quantized neural network performs. They experimentally determined that stochastic rounding significantly outperforms the naive approach of rounding to the nearest number, unless a large amount of fractional bits is used for the fixed point representation. This means if we want to achieve a significant reduction in computational and storage requirements we want to use stochastic rounding. Which results in the following quantisation scheme:

Given a chosen fixed point representation $\langle I, F\rangle$ with I being the number of integer bits and F being the number of fractional bits they define:

$$Round(x, \langle I, F\rangle) = \begin{cases} \lfloor x \rfloor & \text{with probability } 1 - \frac{x - \lfloor x \rfloor}{\epsilon} \\ \lfloor x \rfloor + \epsilon & \text{with probability } \frac{x - \lfloor x \rfloor}{\epsilon} \end{cases}$$

where $\epsilon = 2^{-F}$ is the smallest difference between two values in the chosen fixed point representation.

They then use *Round* for the quantisation:

$$w_{\langle I,F\rangle} = Round(clip(w, 2^{I-1} - 2^{-F}, -2^{I-1}), \langle I, F\rangle)$$

Where $w_{\langle I,F\rangle}$ is the quantized weight, w the original weight and *clip* the clipping function (1).

5 Comparison with other techniques

When trying to compare the different quantisation techniques we need to compare their effectiveness both in reducing computation costs and memory requirements as well as maintaining an accuracy as close as possible to the original non-quantized neural network. For this we first look at the results of prior research ([7],[3],[8]) and then compare it to the results that were found in [1] Results with wide reduced precision Networks. Luckily for us [10] already compares these 3:

Table 2 from [10]: "A qualitative comparison of binary, ternary, integer/fixed point, and mixed precision quantisation schemes"

Quantisation Scheme	**Accuracy Loss**	**Advantages**	**Disadvantages**
Binary	High	**Low cost.** All arithmetic done via binary operations. 32× size compression rate.	**High accuracy loss.** Binary networks often incur around 10% accuracy reductions
Ternary	Low - Moderate	**High compression rate.** Multiplications done via binary operations or capped at two multiplications per activation if using asymmetric scaling factors. 16× compression rate.	**Floating point arithmetic.** For negligible accuracy loss, ternary networks use asymmetric floating point scaling factors, so they need to perform two floating point multiplications per activation
Integer/Fixed Point	Low	**Integer arithmetic.** All arithmetic done via integer arithmetic, which is much cheaper than floating point arithmetic.	**Uniform precision.** For minimal accuracy loss, the networks are limited to the bit width of most sensitive layer, which is often 8 bits, so the compression rates are at most 4×.
Mixed Precision	Low	**Custom precision.** Quantisation scheme for each layer or even row of weights is tailored to their precision sensitivity, reaping the benefits of binary, ternary, and integer quantisation.	**Large search space.** The search space for which quantisation scheme to use for each layer or weight row is exponential in the number of layers or weight rows, respectively

[1] finds that WRPN with double the amount of filter maps based on AlexNet and the ILSVRC-12 dataset "with 4-bits weights and 2-bits activations [exhibit] accuracy at-par with full-precision networks. Operating with 4-bits weights and [more then] 4-bits [for] activations surpasses the baseline accuracy"[1]. The paper also finds that "with 4b operands for weights and activations [...] reduced-precision AlexNet is just 49% of the total compute cost of the full-precision baseline"[1].

[1] also includes results of applying WRPN techniques to ResNet. They found that when doubling the amount of filter maps 4bit activations and 2 bit weights were sufficient to be on-par with the baseline non-quantized network, while using more than 2 bit weights outperforms the baseline network and using 2 bit weights and activations only caused a 0.2% loss in accuracy compared to baseline.

This means that when using enough filter layers the accuracy loss of WRPN is small to negligible. So in terms of accuracy loss WRPN seem superior to techniques that apply quantisation just to the weights of a neural network. Also WRPN are computationally cheaper like traditional quantisation techniques when compared to conventional neural networks. However the paper also finds that this reduction in computational cost is only possible when balancing the precision reduction of weights and activations with the increase in filter maps. Failing to do so properly actually results in an increased computational cost.

6 Conclusion

As we have seen WRPN can be an effective solution to the main issue of quantisation, the Tradeoff between accuracy and meaningful reduction in resource requirements. Common previous techniques either achieved a high resource reduction or a minimal loss in accuracy - not both. This advantage of WRPN however comes at the disadvantage that WRPN now have another trade off instead which needs to be optimised for each dataset: The optimal precision reductions (bit-lengths of the weights and activations) and scaling factor of filter maps. If not optimised properly in the worst case the performance of the WRPN can be even worse then the non-quantized network while having higher resource requirements as well. This means WRPN require a lot of experimental data on a specific data set to be applied to it effectively.

Nevertheless in our opinion the fact that these additional experiments can be seen as part of the "training" or the quantisation process means that WRPN can be a great solution to reducing resource requirements of running a convolutional neural network after training.

References

1. Asit Mishra, Eriko Nurvitadhi, J.J.C.D.M.: Wrpn: Wide reduced-precision networks (Sep 2017), https://arxiv.org/abs/1709.01134v1
2. Damelin, S.B., Miller, W.: The Mathematics of Signal Processing. Cambridge University Press (2011)
3. Fengfu Li, Bin Liu, X.W.B.Z.J.Y.: Ternary weight networks (Nov 2022), https://arxiv.org/abs/1605.04711
4. Hardesty, L.: Explained: Neural networks (2017), https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (Dec 2015), https://arxiv.org/abs/1512.03385
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012), https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
7. Matthieu Courbariaux, Yoshua Bengio, J.P.D.: Binaryconnect: Training deep neural networks with binary weights during propagations (Apr 2016), https://arxiv.org/abs/1511.00363
8. Suyog Gupta, Ankur Agrawal, K.G.P.N.: Deep learning with limited numerical precision (Feb 2015), https://arxiv.org/abs/1502.02551
9. Valueva, M., Nagornov, N., Lyakhov, P., Valuev, G., Chervyakov, N.: Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. Mathematics and Computers in Simulation **177**, 232–243 (2020). https://doi.org/https://doi.org/10.1016/j.matcom.2020.04.031, https://www.sciencedirect.com/science/article/pii/S0378475420301580
10. Weng, O.: Neural network quantization for efficient inference: A survey (Jan 2023), https://arxiv.org/abs/2112.06126
11. Yoshua Bengio, Nicholas Léonard, A.C.: Estimating or propagating gradients through stochastic neurons for conditional computation (Aug 2013), https://arxiv.org/abs/1308.3432