Optimization of Neural Networks

Konrad Dziadek, Justyna Kołodziej, Mateusz Borowiec, Michał Boduła
Faculty of Physics and Applied Informatics
University of Lodz, Poland
{konrad.dziadek, justyna.kolodziej, mateusz.borowiec, michal.bodula}@edu.uni.lodz.pl

***Abstract*—Convolutional Neural Networks (CNN), used extensively in various information processing problems, are most likely the best type of neural networks for any computer vision application and have achieved state-of-the-art results on different tasks. Due to substantial power and memory consumption, replacing complex (CNN) models with their binarized equivalent becomes a satisfactory method of making them less computationally intensive. However, this process significantly reduces accuracy of the network. The aim of the paper is to present results we obtained for fashionMNIST dataset by performing binarization procedures just on the classification stage of LeNet network as well as prove that having suitable parameters chosen we can reach satisfying accuracy and also greatly decrease model size.**

***Index Terms*—Convolutional Neural Network, Binarized Neural Networks, binarized classifier, CNN optimization**

I. Introduction

CNN is a type of neural network used mainly for image-related algorithms, where a special technique called convolution is performed. Convolution means applying convoluted kernels or filters that are slid over the input data. As a result, this operation provides translation equivariant responses known as feature maps. This process is considered highly effective in order to detect specific features and patterns on smaller data slices as opposed to looking at the whole image at once. However, the CNNs' architecture requires a huge number of computational operations leading to excessive memory and power consumption. This issue can be diminished by performing a binarization, which essentially provides reduction of memory usage and increase power efficiency by replacing most multiply-accumulate operations by simple accumulations. Since binarization is not a fresh topic, we can already observe few different approaches on it, although typically, this operation cause noticeable decrease in model's accuracy.

In the project, we focused on binarization performed on the LeNet network classification layers for FashionMNIST dataset. In order to achieve this, we replaced final normal dense layers with binary dense ones. We include more information about it and the dataset in section II. Therefore, the results obtained after applying a binarization were compared with the standard LeNet results. What is more, we performed more experiments with different parameters. Sections III and IV contain the details about results we managed to achieve. Presenting these data, we make an attempt to analyze it as well as deduce whether binarization approach we undertook can be considered reasonable and provide sufficient outcome. Section V is a short overview on our future plans. In the end, in section VI we make a final conclusion.

II. Experimental setup

A. Dataset

The dataset used for the project is Fashion-MNIST from TensorFlow Datasets. This is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes demonstrated below. [1] [2]

TABLE I
FASHION-MNIST DATASET CLASSES

Label	*Description*
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

B. Implementation

We decided to benefit from a robust processing power of a web IDE for python called Google Colaboratory. Each model is created using TensorFlow [3] library. Furthermore, a QuantDense [5] layer used in our binarized networks comes from the open-source deep learning library Larq. [4]

The quantization function SteSign [6] used by Larq for binarization looks as following:

$$q(x) = \begin{cases} -1 & x < 0 \\ 1 & x \geq 0 \end{cases} \tag{1}$$

The gradient is estimated using the Straight-Through Estimator:

$$\frac{\partial q(x)}{\partial x} = \begin{cases} 1 & |x| \leq 1 \\ 0 & |x| > 1 \end{cases} \tag{2}$$

III. Results

As we mentioned in section I, we built LeNet CNN with and without binarization layers. The results are as following.

A. About models we considering

1) Pure LeNet: Base LeNet CNN is built with two groups of Convolutional and MaxPooling2D layer, flatten layer and three dense layers. The last dense layer is an output of the model. The model summary looks as following:

+sequential stats----------

Layer	Input prec. (bit)	Outputs	# 32-bit x 1	Memory (kB)	32-bit MACs
conv2d	-	(-1, 28, 28, 6)	156	0.61	117600
max_pooling2d	-	(-1, 14, 14, 6)	0	0	0
conv2d_1	-	(-1, 14, 14, 16)	2416	9.44	470400
max_pooling2d_1	-	(-1, 7, 7, 16)	0	0	0
flatten	-	(-1, 784)	0	0	0
dense	-	(-1, 120)	94200	367.97	94080
dense_1	-	(-1, 84)	10164	39.70	10080
dense_2	-	(-1, 10)	850	3.32	840
Total			107786	421.04	693000

Fig. 1. Base LeNet CNN summary

After learning this model, we reached 94.23% accuracy. Size of the model is 421.04 kB

2) LeNet with one layer binarized: We binarized one layer of CNN by adding a binarization layer and normalization layer. Summary of the model looks as following:

+sequential_1 stats----------

Layer	Input prec. (bit)	Outputs	# 1-bit x 1	# 32-bit x 1	Memory (kB)	1-bit MACs	32-bit MACs
conv2d_2	-	(-1, 28, 28, 6)	0	156	0.61	0	117600
max_pooling2d_2	-	(-1, 14, 14, 6)	0	0	0	0	0
conv2d_3	-	(-1, 14, 14, 16)	0	2416	9.44	0	470400
max_pooling2d_3	-	(-1, 7, 7, 16)	0	0	0	0	0
flatten_1	-	(-1, 784)	0	0	0	0	0
dense_3	-	(-1, 120)	0	94200	367.97	0	94080
dense_4	-	(-1, 84)	0	10164	39.70	0	10080
batch_normalization	-	(-1, 84)	0	168	0.66	0	0
flatten_2	-	(-1, 84)	0	0	0	0	0
quant_dense	1	(-1, 10)	840	0	0.10	840	0
batch_normalization_1	-	(-1, 10)	0	20	0.08	0	0
activation	-	(-1, 10)	0	0	0	?	?
Total			840	107124	418.56	840	692160

Fig. 2. LeNet with one binarized layer summary

After learning CNN, we achieved 93,89% accuracy and 418.56 kB memory usage. This is 0.4% less accuracy, but memory usage also decreased – it is 0.6% less memory usage than in pure LeNet.

3) LeNet with 2 layers binarized: LeNet with binarized two layers reaches worst accuracy result (92.21%) than the previous ones, but memory usage of the Net fall about 9% in comparison to the one layer binarized. Everything is visible in the model's summary:

+sequential_7 stats----------

Layer	Input prec. (bit)	Outputs	# 1-bit x 1	# 32-bit x 1	Memory (kB)	1-bit MACs	32-bit MACs
conv2d_14	-	(-1, 28, 28, 6)	0	156	0.61	0	117600
max_pooling2d_14	-	(-1, 14, 14, 6)	0	0	0	0	0
conv2d_15	-	(-1, 14, 14, 16)	0	2416	9.44	0	470400
max_pooling2d_15	-	(-1, 7, 7, 16)	0	0	0	0	0
flatten_13	-	(-1, 784)	0	0	0	0	0
dense_15	-	(-1, 120)	0	94200	367.97	0	94080
batch_normalization_12	-	(-1, 120)	0	240	0.94	0	0
flatten_14	-	(-1, 120)	0	0	0	0	0
quant_dense_6	1	(-1, 84)	10080	0	1.23	10080	0
batch_normalization_13	-	(-1, 84)	0	168	0.66	0	0
quant_dense_7	1	(-1, 10)	840	0	0.10	840	0
batch_normalization_14	-	(-1, 10)	0	20	0.08	0	0
activation_6	-	(-1, 10)	0	0	0	?	?
Total			10920	97200	381.02	10920	682080

Fig. 3. LeNet with two binarized layers summary

If we decrease batch size, which in the previous model was 64, the accuracy decreases. Above result is the best result we achieved with two binarized layers.

4) LeNet with 3 layers binarized: Binarizing three layers ends with 90.52% of accuracy but memory usage is extremely reduced and equals 30.66 kB This is much less memory (about 92% less) than previous example with two layers binarized. Of course, accuracy falls, but in comparison to memory usage we have excellent accuracy/memory ratio. Everything is visible in the model's summary:

+sequential_19 stats----------

Layer	Input prec. (bit)	Outputs	# 1-bit x 1	# 32-bit x 1	Memory (kB)	1-bit MACs	32-bit MACs
conv2d_38	-	(-1, 28, 28, 6)	0	156	0.61	0	117600
max_pooling2d_38	-	(-1, 14, 14, 6)	0	0	0	0	0
conv2d_39	-	(-1, 14, 14, 16)	0	2416	9.44	0	470400
max_pooling2d_39	-	(-1, 7, 7, 16)	0	0	0	0	0
flatten_37	-	(-1, 784)	0	0	0	0	0
batch_normalization_48	-	(-1, 784)	0	1568	6.12	0	0
flatten_38	-	(-1, 784)	0	0	0	0	0
quant_dense_30	1	(-1, 120)	94080	0	11.48	94080	0
batch_normalization_49	-	(-1, 120)	0	240	0.94	0	0
quant_dense_31	1	(-1, 84)	10080	0	1.23	10080	0
batch_normalization_50	-	(-1, 84)	0	168	0.66	0	0
quant_dense_32	1	(-1, 10)	840	0	0.10	840	0
batch_normalization_51	-	(-1, 10)	0	20	0.08	0	0
activation_18	-	(-1, 10)	0	0	0	?	?
Total			105000	4568	30.66	105000	588000

Fig. 4. LeNet with three binarized layers summary

B. Results comparison

As we see pure LeNet has the highest accuracy and the highest memory usage. On the other side we have LeNet with three layers binarized that has lower accuracy but extremely lower memory consumption. All four examples are compared in table:

TABLE II
Accuracy and memory consumption

Net type	*Accuracy(%)*	*Memory(kB)*	*Normalised memory(%)*	*Normalised accuracy(%)*
Pure LeNet	94.23	421.04	100	100
1 binary layer	93.89	418.56	99.6	99.4
2 binary layers	92.21	381.02	97.86	90
3 binary layers	90.52	30.66	96	7.2

As we see the best option considering accuracy and memory is option with three layers binarized. Memory usage is only 30.66 kB and the accuracy is quite high (90.52%). Reducing

memory consumption is important to implement CNNs in small devices with low memory capacity.

IV. Diagram comparisons with others model's settings

During the project we checked many different CNNs parameters. Below we present diagrams comparing accuracy with different parameter's values. In section III we discussed the best accuracy results we obtained.

A. Pure LeNet and LeNet with one layer binarized comparison

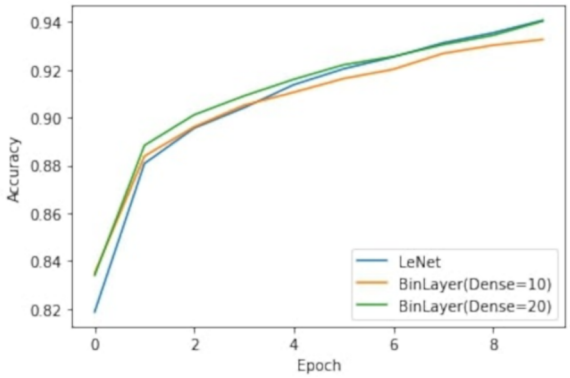

Fig. 5. LeNet with/without binarization layer (Dense = 10, Batch = 64, Epochs = 10, unless otherwise stated)

In the figure we can LeNet without binarization and with binarization layer. Pure LeNet accuracy and LeNet with BinLayer(Dense = 20) accuracy is the same. The LeNet with BinLayer(Dense = 10) accuracy is a little worse.

B. LeNet with one layer binarized different model's options comparison

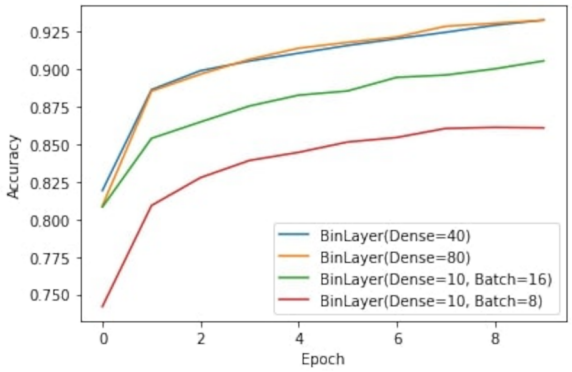

Fig. 6. LeNet with binarization layer (Dense = 10, Batch = 64, Epochs = 10, unless otherwise stated)

As we can see the best result we achieve with BinLayer(Dense = 40) and BinLayer(Dense = 80) – it is about 92%. If we decrease Dense and Batch parameter accuracy extremely falls.

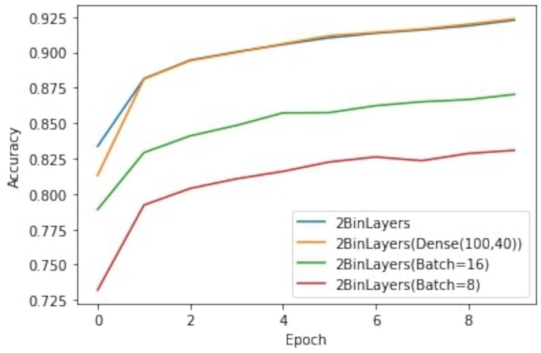

Fig. 7. LeNet with two binarization layers (Dense = (100,10), Batch = 64, Epochs = 10, unless otherwise stated)

C. LeNet with two layers binarized different model's options comparison

In this case we have the same situation as in the previous point. The best accuracy is for parameters equals Dense = (100, 10) or Dense = (100, 40) and Batch = 64. If we take lower Batch parameter, the accuracy decreases to about 85%.

D. LeNet with three layers binarized different model's options comparison

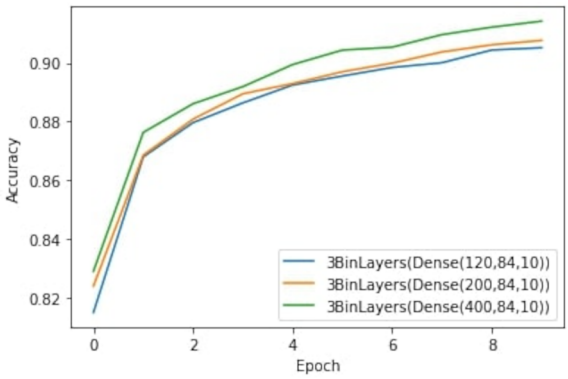

Fig. 8. LeNet with three binarization layers (Dense = (120, 84, 10), Batch = 64, Epochs = 10, unless otherwise stated)

In this case we changed only density of the binarization layers. All three results are very similar but the best result (about 90% accuracy) has the net with highest layers density. The worst accuracy has the net with the lowest density parameter.

E. Comparison of chosen models

On the figure we can see that the best accuracy gets pure LeNet. In this graph we do not consider memory consumption. If we consider memory usage, we can say that accuracy decrease is not so high (see table II).

V. Future work

In the nearest future we are planning more deep research in CNNs. We also want to check different values of parameters (such as Dense, Batch etc.). We are also considering increasing

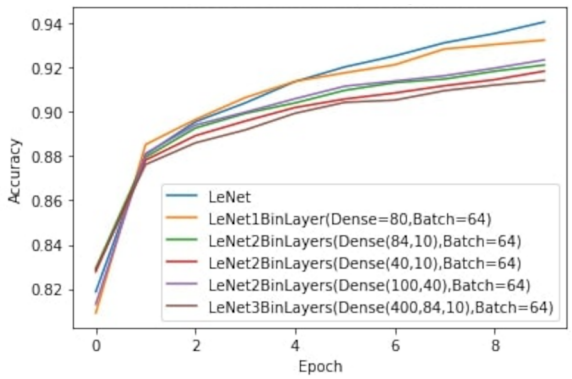

Fig. 9. Best scores from each comparison

number of binarized layers in LeNet and see how accuracy and memory usage is going to change.

VI. Consideration

Considering all results binarization could reduce memory consumption and do not break the accuracy so much. We achieved about 90% of accuracy with three binarized layers (see table II). Net's memory usage was only about 30 kB. This is 7% of the pure LeNet memory consumption. Reducing memory is important to implement CNNs to embedded systems such as cars, smartphones, smartwatch etc. Reduced memory helps to speed up calculations without losing too much accuracy. That is why CNNs binarization is so important these days.

References

[1] TensorFlow resources, fashion-MNIST dataset description https://www.tensorflow.org/datasets/catalog/fashion_mnist
[2] Keras API reference, classes of fashion-MNIST https://keras.io/api/datasets/fashion_mnist/
[3] TensorFlow library https://www.tensorflow.org/overview
[4] Larq open-source library documentation https://docs.larq.dev/larq/
[5] Larq.layers documentation https://docs.larq.dev/larq/api/layers/#quantdense
[6] Larq.quantizers documentation https://docs.larq.dev/larq/api/quantizers/#stesign