Optimizing CNNs Through Binarization

Daniel Pająk, Patryk Pietranek, Przemysław Świtecki, Filip Wawrzyniak
WFiIS. Uniwersytet Łódzki
Łódź, Poland

***Abstract*—While Convolutional Neural Networks are used successfully in many different branches of engineering and science, their high accuracy is a result of processing large data sets and lengthy training times. The sheer amount of computing power and storage needed to operate algorithms using such convolutional neural networks excludes their use in embedded systems, wearable technology or other compact devices. One way of solving this issue, that we will implement and evaluate in this project, is to binarize. This way we can not only downscale hardware requirements, but also introduce new hardware altogether, one that is specifically built to operate on binary datasets, allowing further optimization and lower production costs. The aim of this paper is to explore the effects of binarization of the input layer. In our case we are using images that are already in greyscale and we threshold them. This way we significantly downscale the amount of data we need to process. To test our approach we used the Sing Language MNIST dataset. During our research we have found out that the accuracy of our binarized neural network not only closely matches the one achieved by using a traditional approach, but also in some certain cases, it exceeds it.**

I. Introduction

Currently machine learning is one of the leading fields in computer science. In the last decade it has experienced tremendous growth, this topic fascinates and intrigues many scholars and hobbyists. One of the reasons why it is the case is that the options of using this technology are almost limitless. However, while the accuracy of Convolutional Neural Networks is near perfect, the hardware limitations are still a concern, hence the need to optimize and speed up the processes. One of the most common ways to do so is implementing binarized versions of existing structures. There are numerous ways of approaching this subject. Some papers propose binarizing the input layer, some binarize the weights.

To achieve binarization we decided to binarize the input layer. First, we threshold the image, so instead of the full range of values from 0 to 255, we can get either 0 or 255. Then we split our data into thresholds and merge them into one image of n channels, where n represents the number of thresholds. After processing our data, we can proceed and start training the network and then, in the end, to evaluate it and test the accuracy loss.

The aim of our project is to evaluate the losses in accuracy and the possible gains in performance using our binarized convolutional neural network. During our tests we have decided to use a popular dataset Sing Language MNIST. It provides countless images to test and analyze, a perfect fit for our purpose. We explored how not only the number of intervals impacts the accuracy of our model, but also the ranges themselves. We found out that our approach works very well on the dataset chosen by us. If set up correctly, our binarized version closely follows the results we got from a traditional neural network and sometimes achieves better accuracy. We also found out that the intervals shouldn't be too wide. For example, If the first interval is too wide, we lose too much information during thresholding and the accuracy plummets.

This paper is organized in the following way, in Section II we explore approaches proposed by other scholars and how they relate to our implementation. In Section III we describe our way of implementing the binarized convolutional neural network and present its diagram. Section IV includes a detailed description of all the tests we've conducted and the analysis of thereof. The final Section V summarizes our work, describes problems and challenges we've faced during our work and outlines possible options for further research.

II. Alternative approaches

Overall the research and development of binary Convolutional Neural Networks can be split into multiple categories. While each is different, all methods still follow the same goal of making the Convolutional Neural Networks less power hungry and more compact. In the end, all those methods can be mixed and matched. The categories include the following types but are not limited to them.

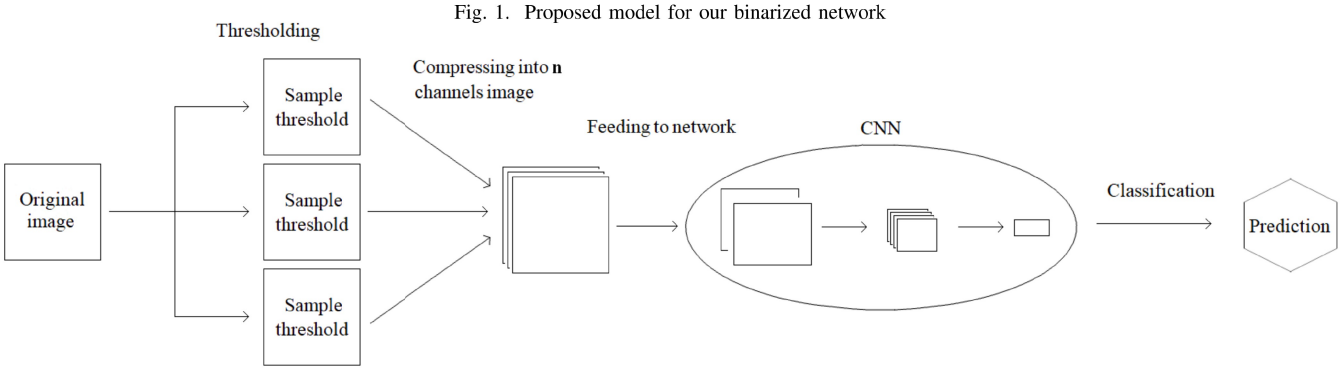
A. Binarization of Input Data

For the sake of simplicity, we will be discussing neural networks working on images. In this case, the binarization of the input comes down to simplifying the image we want to analyze. Instead of using the whole range of values for each pixel, for example from 0 to 255, we use either 1 or 0. This way we can limit the amount of data we need to process. In case of images saved in RGB format we can binarize a single channel, we can threshold each channel separately or convert the image to grayscale. In our project we decided to use the third option.

B. Binarization of Weights

It's a similar concept to the previous method, but this time, instead of binarizing an array of pixels, we binarize weights of each layer in our neural network. Weights are a critical part of the process of forming a CNN. In the process of training, they have to be changed and adjusted numerous times. By simplifying them the process can be sped up significantly.

Fig. 1. Proposed model for our binarized network



C. Binarization of Activations

Another concept is based on the principle of limiting the activation function to either 0 or 1. In most papers it is achieved in a similar way to binarizing weights. While it may seem redundant at a first glance, it's another critical part of developing a fully binary neural network.

III. Our model

We have decided that the best approach for our project will be the binarization of the input data. In our case it allows for way more interesting and varied ways of testing our approach. Additionally, our dataset is focused on a very particular subject, each image depicts a hand gesture, a single letter in the sign language, against a light grey background. Combined with the fact that the images are already in greyscale when imported we were excited to explore the effects thresholding will have on our neural network.

As you can see in Fig. 1, the first step in our binarization is thresholding the image. We need to downscale the amount of data that has to be processed. We can achieve it by reducing data saved in each pixel to just 0 or 255, instead of a full range from 0 to 255. This way each pixel of a layer can hold only two values, hence the term binarization. After splitting our data by thresholding it, we merge all of our thresholds into a single image consisting of n channels, where n represents how many thresholds we have generated.

At this step the process of binarization if finished, we have a properly prepared input layer that we can use to train our network. After training we can proceed and start the process of testing and evaluating. In this step we can finally find out, whether the parameters we have chosen are optimal.

IV. Tests

We tested our network with the Sing Language MNIST dataset that contains 27,455 training cases and 7172 test cases. First of all, we tested the original network to see the original accuracy rate. After that we compared it with our network that includes binarized input with different threshold ranges. For this dataset original network did really well with the 94,9% accuracy. In the next step we had to come up with some threshold ranges. We decided to test the network with just two threshold ranges and then increase the number of ranges in the subsequent trials. We have conducted the test with two threshold ranges two times, using different threshold ranges each time.

TABLE I
Accuracy and threshold ranges of the first two test

Threshold ranges	Accuracy
[0,150,255]	90,3%
[0,200,255]	62,5%

As we can see in the Table 1, the value that impacts the accuracy the most is the middle one. If set too high the accuracy is severely degraded, compared to the test ran with lower values. The loss is significant, around 30%.

Next, we tested our network once with three threshold ranges [0, 150, 170, 255]. The accuracy of the network was 92,7%, a result we deemed satisfactory. Then the network was tested two times with four different threshold ranges each time.

TABLE II
Accuracy and threshold ranges of the two tests with four threshold ranges

Threshold ranges	Accuracy
[0,150,170,200,255]	91,7%
[0,90,150,170,255]	94,8%

As shown in the Table 2, the accuracy was better when the first threshold range ended with a smaller value.

Afterwards we increased the number of our threshold ranges to five [0, 90, 150, 170, 210, 255]. This time after testing it we achieved the accuracy of 95,6%, which is the best score so far. Second to last test included six threshold ranges [0, 150, 170, 190, 210, 230, 255]. Contrary to our predictions, the accuracy fell off, the accuracy was 93,4%, which is worse than in the test with five threshold ranges.

Our last test was done with seven threshold ranges [0, 90, 120, 150, 180, 210, 230, 255] and the accuracy was 94,7%, which places this configuration right in the middle. To sum it all up in the Table 3 we showcase the three best threshold ranges and their accuracy.

TABLE III
THREE BEST THRESHOLD RANGES AND THEIR ACCURACY

Threshold ranges	Accuracy
[0,90,150,170,210,255]	95,6%
[0,90,150,170,255]	94,8%
[0,90,120,150,180,210,230,255]	94,7%

V. CONCLUSION

In this project, we set out to show differences between neural networks trained using the traditional approach and the ones utilizing binarization. We examined how networks work with data binarized by thresholding input. Our network, after being trained with the original set of data, achieved 94,9 % accuracy. To compare the initial results with the ones obtained by binarization, network was tested with different numbers of thresholds, from 2 to 7, and with different endings of each one.

Network trained with only 2 threshold intervals achieved the worst results. We observed that the first threshold range could not end with a value that is too high, in that case too much information is lost. When the thresholds were set to [0, 200, 255] the accuracy has suffered greatly compared to all the other tests with a measly value of 60,2%. When it comes to the category of just two intervals, we have reached the conclusion that the optimal range for the first interval is from 0 to 150, for this interval we've managed to achieve the accuracy of 90.3

Hoping to achieve even better accuracy, we increased number of thresholds up to 5. For five intervals the result was the highest thus far, surpassing the original approach and was equal to 95,6 %. It is surprising that for the 6 thresholds the accuracy was slightly worse and was 93,4%. In the case of 7 intervals, we observed a rebound of accuracy back to 94,7%.

To reach the best results, we observed that the network should not be trained with too many thresholds and the ranges should be quite small. To sum it all up, binarization by thresholding could improve speed of training network and so far the results are really promising. Accuracy was very close to the original one and in one case it was even higher.

During our work we've encountered only a few problems, we were able to remedy most of them relatively quickly. The biggest obstacle we have encountered was the lack of proper hardware needed to train our neural network in a graphics processor unit mode. Only the newest models of Nvidia graphics cards support this mode, and those are prohibitively expensive and infamously hard to obtain nowadays, mostly due to the bitcoin gold rush and scalpers. Thankfully we have found a solution in the form of Google Colab. One of the services it provides is cloud computing, GPU included. Thanks to this service we were able to proceed without any further interruptions.

In our future work we will try to examine how different types of binarization impact the training and the accuracy of networks. We are also interested in seeing how binarized neural networks will behave when faced with bigger and more intricate input data sets. The effect they might have on binarized neural networks is well worth investigating.