

**University of Warsaw**  
Faculty of Mathematics, Informatics and Mechanics

**Krystyna Gajczyk, Jakub Pierewoj, Przemysław  
Przybyszewski, Adam Starak**

Student number: 332118, 360641, 332493, 361021

# **Inference in neural networks using low-precision arithmetic**

**Bachelor thesis in COMPUTER SCIENCE**

Supervised by:  
**Dr Konrad Durnoga**

March 16, 2017

## **Oświadczenie kierującego pracą**

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data

Podpis kierującego pracą

## **Oświadczenie autora (autorów) pracy**

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data

Podpis autora (autorów) pracy

## **Abstract**

TODO

## **Keywords**

binarized neural network, XORNET

## **Subject Area (Socrates/Erasmus code)**

11.3 Informatics, Computer Science

## **Categories and Subject Descriptors**

???

## **Thesis title in Polish**

Inferencja w sieciach neuronowych przy użyciu arytmetyki niskiej precyzji



# Contents

- 1. Introduction . . . . . 5**
  - 1.1. What is deep learning and why is it interesting? . . . . . 5
  - 1.2. What is the goal of the project? . . . . . 6
- 2. Architecture overview . . . . . 9**
  - 2.1. Used framework . . . . . 9
  - 2.2. Our implementation . . . . . 9
- Bibliography . . . . . 11**



# Chapter 1

## Introduction

### 1.1. What is deep learning and why is it interesting?

Deep learning is branch of machine learning which tries to model high level abstractions in data (like understanding words or finding objects on image) using deep graph of simple elements - neurons, connected with specific activations and weights to others. It has recently gained a lot of interest from the industry, especially after success of Alexet in Imagenet large scale visual recognition competition in 2010. The most important part of deep learning is training net based on the samples that the result is known. After each training step one can change weights and activation in model to make output of the network as close as possible to expected value.

Currently deep neural networks get better results than state-of-the-art algorithms in many areas, such as computer vision and speech recognition. This breakthrough was possible because computing power of modern computers is high enough to handle intensive workload required to train large artificial neural networks.

Deep neural networks (DNN) usually use high precision (floating point) numbers to represent weights and activations. Computations using that arithmetic are usually handled by graphic cards. This is the factor, which is limiting use of deep neural networks on devices with low computing power, such as mobile phones.

One of the possible solutions to that problem is to represent activations and weights as binary numbers and use low precision arithmetic to perform inference. That kind of computations could be optimized to be handled efficiently by CPUs or on potential new devices, specially optimized for that purpose.

Researchers tried to address this problem in the number of papers in the recent years. Couple of main approaches have been investigated in order to reduce inefficient computation and memory usage in deep neural networks while maintaining the classification accuracy. Those approaches can be summarized as:

- Shallow networks - it has been proved, that every neural network can be replaced with a corresponding neural network with a single (possibly large) hidden layer. One of the main problems with this approach is, that in order to achieve similar accuracy as the original neural network, the shallow networks need approximately the same number of parameters. Second problem is that empirical test have shown, that while it shows

good results for relatively small datasets, it underperforms for larger datasets (such as ImageNet).

- Compression of pre-trained DNN - it is possible to prune a DNN at inference time by discarding the weights, that don't bring any information to the classification process. It has been also further proposed to reduce the number of parameters and/or activations in order to increase acceleration, compression. This reduces the main blockers for using DNN on small, embedded devices, such as memory usage and energy. Memory usage can be achieved through quantizing and compressing weights with Huffman coding or hashing weights, such that the weights assigned to one hash bucket share the same parameter.
- Compact layers - this approach also reduces the memory usage and computation time. This approach replaces parts of the DNN structure with corresponding elements, which are smaller in size and bring nearly as much information. A few techniques have been examined, such as replacing a fully connected layer with global average pooling and replacing a convolution with a corresponding one requiring smaller amount of parameters.
- Quantizing parameters - this technique aims to replace the floating-point parameters of the neural network with the quantized values (through vector quantization methods), which require smaller number of bits of memory and need simpler arithmetic operations for computation. Number of DNN with quantization have been designed: 8-bit integer instead of 32-bit floating point activations, ternary weights and 3-bit activations instead of floating points, etc. It was shown, that this approach can lead to a DNN representation, which accuracy is not very far off from the state-of-the-art results.
- Network binarization - this is the extreme case of the parameter quantization technique. Due to new learning algorithm, which is Expected Back Propagation, which bases on inferring network with binary weights and neurons through a variational Bayesian approach. Techniques using this approach mainly use the real-valued weights as a reference for their binarization. This idea was further extended to binarize both weights and activations, which was implemented in networks such as BinaryNet and XNOR-Net.

## 1.2. What is the goal of the project?

We would like to create network XNORNET which will be as efficient as ALEXNET. The goal is to find optimal size of network and it's depth to get same results as in classical DNN. We will achieve this goal by modifying existing libraries for DNN.

The main goal is to analyze the results of inference, depending on 3 properties:

- topology
- weights
- feature maps

Each DNN has it's own tolerance of inference precision. There are numerous topologies in neural networks. It indicates the number layers and defines how the neurons are connected. The aim is to study the structures and predict which one will behave the best in our environment. Dealing with 1-bit integer weights is going to strongly affect the complexity



of computational algorithms and the size of data. The whole workflow is going to be much faster. 1-bit operations are a way simpler than the 32-bit ones. Furthermore, the size of the inputs will require less memory. That is the perfect solution for less efficient devices.

Unfortunately, those properties will negatively affect on the quality of prediction. Thus, the increase of the depth of chosen net should be also taken into consideration. A feature map is an input for the next level neurons. The loss of data and weights precision can be alleviated by the increased number of the feature maps. After the whole research the question "How many times the number of feature maps in low arithmetic precision in chosen net should be increased, in order to achieve the similar quality of the prediction as the basic DNN?" should be clearly known. To sum up, the research may bear out, that BDNNs have some great properties, which should be investigated much deeper. It may also find out, that they achieve better results in some cases and the present solutions should be replaced with BDNN.



## Chapter 2

# Architecture overview

Our goal is to create a binarized neural network - first to repeat experiments done by researchers and then to test out our ideas and hypothesis.

### 2.1. Used framework

We decided to work with one of open source framework that can be used for neural networks. The structure of BNN proposed in Binary Connect [3] and XORNET [2] is very similar to standard convolutional network so there is no need to write BNN from scratch.

From available frameworks we decided to choose TensorFlow. It is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays transported between them. In case on of neural networks, nodes will represent layers of network.

All frameworks that we analyse (TensorFlow, Torch, Caffe) have similar functionalities, for example they have already implemented pooling and affine layers so we can easily reuse them.

The advantage of TensorFlow over other frameworks is really good community support and solid documentation.

### 2.2. Our implementation

To create BNN we need to implement in c++ a new computation node (operation) which will extend standard convolutional layer to perform binary convolution. This operation will be added in tensorflow/core/kernels folder where other popular function are. We plan to write unit tests to test correctness of our implementation. Using this operation we will be able to create BNN using tensor flow python API.

We are going to implement algorithms:

- Binary Connect [3]
- XORNET [2]
- Binary-Weight-Networks [2]



# Bibliography

- [1] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, Oriol Vinyals, *Understanding deep learning requires rethinking generalization*, <https://arxiv.org/abs/1611.03530> (2016)
- [2] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, Ali Farhadi, *XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks*, <https://arxiv.org/abs/1603.05279> (2016)
- [3] Matthieu Courbariaux, Yoshua Bengio, Jean-Pierre David, *BinaryConnect: Training Deep Neural Networks with binary weights during propagations*, <https://arxiv.org/abs/1511.00363> (2015)