Inference at the edge: the impact of compression on performance

Deliverable 1: Final year Dissertation

Bsc Computer Science: Artificial Intelligence

Sam Fay-Hunt — sf52@hw.ac.uk

Supervisor: Rob Stewart — R.Stewart@hw.ac.uk

November 2, 2020

DECLARATION

I, Sam Fay-Hunt confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: ....

Date: ....

Abstract: a short description of the project and the main work to be carried out – probably between one and two hundred words

Contents

1 Introduction

Summarising the context of the dissertation project, stating the aim and objectives of the project, identifying the problems to be solved to achieve the objectives, and sketching the organisation of the dissertation.

With the revolution of AI technologies a greater need to perform inference at the edge is becoming ever more prevalent. The argument for localising inference is only becoming stronger with the ever increasing avaliablilty of computation power alongside new and constatnly evolving AI applications, inference at the edge can provide better privacy and latency than the remote datacenter alternatives. This dissertation will focus on methodologys for improving inference performance with preexisting compression techniques.

These models can have a huge number of parameters so inference can sometimes be impractical. [1] - "see Table 1"

Issues with limited resource computation [2]

This dissertation will study the effect of pruning algorithms exposed by the Intel distiller framework on inference.

outline the document: We start with ..., then we cover x, y, and z ...

2 Background

Discussing related work found in the technical literature and its relevance to your project.

This Section will be split into 4 subsections:

Section 2.1 - **Deep Learning**: An overview of the basic components of a neural network and the CNN & RNN models.

Section 2.2 - **Compression Types**: ...

Section 2.3 - **Edge Computing**: stuff about edge comp

Section 2.4 - **Hardware Memory architectures**: brief stuff about this section

2.1 Deep Neural Networks

2.1.1 Neural Networks & Deep Learning

- Summary of NN

- Structure of NN

- Training & Inference stages

- weight update methodologies

- Feed Forwards

- Feedback Nerual Network

- Self-organizing Neural Network

- Weight parameters updated using back-propagation

Deep learning is a subcategory of machine learning techniques where a hierarchy of layers perform some manner of information processsing with the goal of computing high level abstractions of the data by utilising low level abstractions identified in the early layers [3].

Neural networks fundamental purpose is to transform an input vector commonly re-

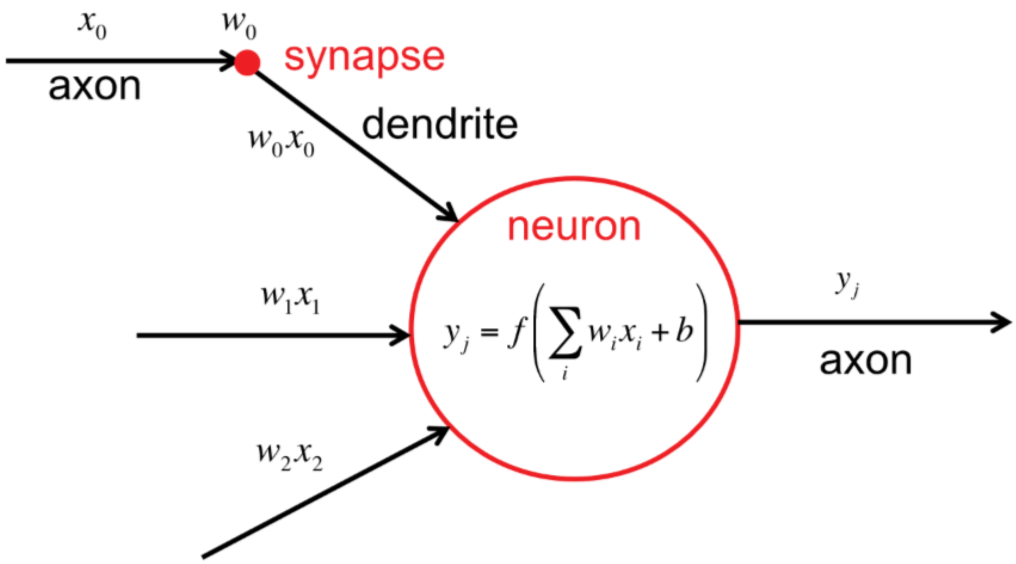

Figure 1: Neuron with corresponding biologically inspired labels.
(Adopted figure from [2])

ferred to as X into an output vector $\hat{Y}$. The output vector $\hat{Y}$ is some form of classification such as a binary classification or a probability distribution over multiple classes [4]. Between the input layer (X) and the output layer ($\hat{Y}$) there exists some number of interior layers that are referred to as hidden layers, the hidden and output layers are composed of neurons that pass signals derived from weights through the network, this model of computing was inspired by connectionism and our understanding of the human brain, see Fig. 1 for labels of the analagous biological components. Weights in a neural network effectively correspond to the synapses in the brain and the output of the neruon is modelled as the axon. All neruons in a Neural network have weights corresponding to their inputs, these weights are are intended to mirror the value scaling effect of a synapse by performing a weighted sum operation [2].

Neural networks and deep neural networks are often reffered to interchangably, they are primarily distinguished by the number of layers, there is no hard rule indicating when a neural network is considered deep but generally a network with more than 3 hidden layers is considered a deep neural network, the rest of this dissertaion will refer to DNNs for consistency. Each neuron in a DNN applies an non-linear activation function to the result of its weighted sum of inputs and weights, without which a DNN would just be a linear algebra operation [2], the cumulative effect of the activations in each layer results in elabourate causal chains of transormations than influence the aggregate activation of the network.

Backpropagation Although not the first to propose this approach [5] the 1986 paper Learning representations by back-propagating errors [6] popularised back-propagation for updating weights during training multi-layer networks.

DNNs can be categorised as feedforwards, feedback, and self-organising networks depending on their processing method [1].

There are many popular deep neural network architectures, this document will continue to outline the CNN & RNN architectures because these provide a high level overview of the type of models that will be used for the research posed in this dissertation.

2.1.2 Convolutional Neural Networks

Convolutional Neural Network (CNN)

- A class of DNN

- CNN consist of: Convolutional Layers, Pooling layers & fully connected layers.

- Convolutional Layers contain sets of filters/kernels

Much like traditional nerual networks the CNN architecture was inspired by human and animal brains, the concept of processing the input with local receptive fields is conceptually similar some functionality of the cat's visual cortex [7]–[9]. The influential paper by Hubel & Weisel [7] ultimately had a significant influence on the design of CNNs via the Neocognitron, as proposed by Fukushima in [10] and again evaluated in [11]**(provide some comment on these papers)**.

A critical aspect of image recognition is robustness to input shift and distortion, this robustness was indicated as one of the primary achivements of the Neocognitron in Fukushima's paper [10]. LeCunn and Bengio provide comprehensive explainations of how traditional DNNs are so inefficient for these tasks

The local receptive fields enable neurons to extract low level features such as edges, corners, and end-points with respect to their orientation. CNNs are robust to input shift or distortion by using receptive fields to identify these low level features across the entire input space, performing local averaging and downsampling in the layers following convolution layers means the absolute location of the features is less important than the position of the features relative to the position of other identified features [8]. Each layer produces higher degrees of abstraction from the input layer, in doing so these abstractions retain important information about the input, these abstractions are referred to as feature maps. The layers performing downsampling are known as pooling layers, they reduce the resolution or dimensions of the feature map which reduces overfitting and speeds up training by reducing the number of parameters in the network [9].

Convolutional Networks for Images, Speech, and Time-Series by LeCunn & Bengio

CNNs have been found to be effective in many different AI domains, popular applications include: computer vision, NLP, and speech processing.

2.1.3 Recurrent Neural Netowrks

Recurrent Neural Network (RNN)

RNNs are deep learning models that use loops in their layer connections to make predictions

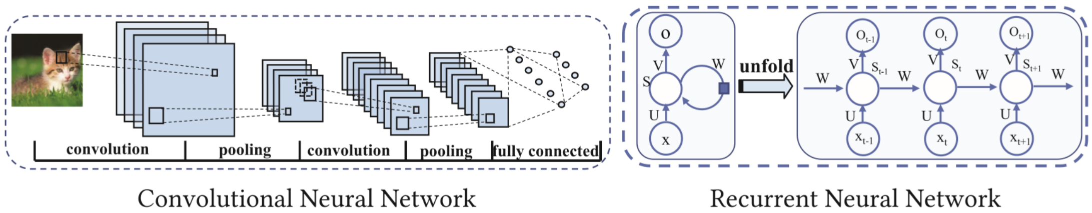

Figure 2: A typical example of a CNN (left) and RNN (right)
(Adopted figure from [1])

with sequential inputs and maintain state over those inputs, this architecture is designed specifically for time series predictions [12].

2.2 Compression types

pruning

distillation

Quantization

Network design strategies

low-rank factorization

2.3 Computing at the edge

Some background on edge computing - maybe a detailed definition

- Challenges of resource bound deep learning

- Online vs offline learning

2.4 Memory factors in Deep Neural Networks

- Discuss VPU/TPU/APU/GPU/FPGA/ASIC memory arcitecture and how it handles matrix sparsity

- *Show ineffectivity of pruning on hardware without optimisations for sparse matrices*

2.4.1 Memory Allocation

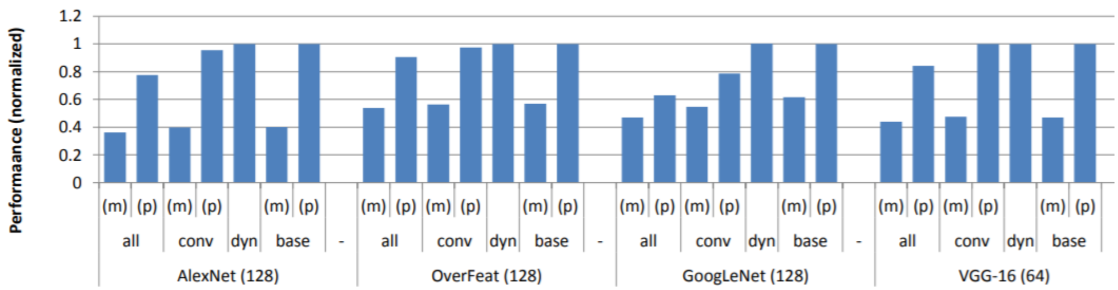

Figure 3: vDNN performance, showing the throughput using various memory allocation strategies.
(Adopted figure from [13])

While designed specifically for training networks that would otherwise be to large for a GPU, the memory manager vDNN proposed in [13] does provide some insight into the importance of memory locality to neural network throughput. Fig. 3 summarizes the performance of vDNN policies compared to a baseline memory management policy (*base*), the vDNN policies include: static policies (denoted as *all* and *conv*) and a dynamic policy (*dyn*). *base* simply loads the full model into the GPU memory consequently providing optimal memory locality. *all* refers to a policy of moving all Xs out of GPU memory, and *conv* only offloads Xs from convolutional layers, Xs are the input matrices to each layer, denoted by the red arrows in Fig. 4. Each of *base*, *conv* and *all* are evaluated using two distinct convolutional algorithms - memory-optimal (m) and performance-optimal (p). Finally the *dyn* allocation policy chooses (m) and (p) dynamically at runtime. Fig. 3 communicates a significant (58% and 55%) performance loss compared to baseline when no effort is made to optimise the memory locality of parameters in the network

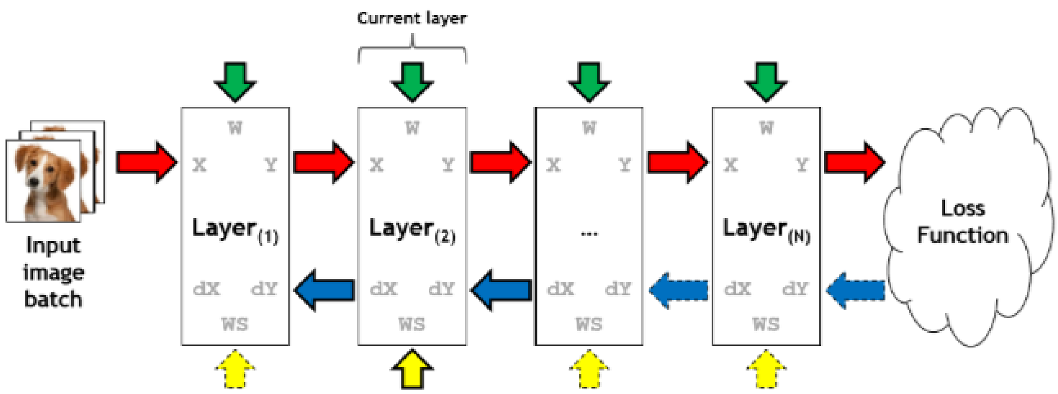

Figure 4: Memory allocations required for linear networks. All green (W) and red (X) arrows are allocated during inference, the blue and yellow arrows are allocated during training.
(Adopted figure from [13])

Justifies the need for compression ... pruning

2.4.2 Memory Access

A significant portion of DNN computation is matrix-vector multiplication, the primary bottleneck of matrix-vector multiplication is memory access [14]. When the cache capacity is insufficient for the input matrix size this bottleneck is expounded, as a consequence memory access is neccessary for every operation since there is no reuse of the input matrix [15]. As observed in 2.4.1 this indicates that compression techniques should alliviate this bottleneck, and while compression doese reduce the total number of operations, the irregular memory access pattern caused by compression hinders effective acceleration see Fig. 5.

Platform	Batch Size	Matrix Type	AlexNet			VGG16			NT-		
			FC6	FC7	FC8	FC6	FC7	FC8	We	Wd	LSTM
CPU (Core i7-5930k)	1	dense	7516.2	6187.1	1134.9	35022.8	5372.8	774.2	605.0	1361.4	470.5
		sparse	3066.5	1282.1	890.5	3774.3	545.1	777.3	261.2	437.4	260.0
	64	dense	318.4	188.9	45.8	1056.0	188.3	45.7	28.7	69.0	28.8
		sparse	1417.6	682.1	407.7	1780.3	274.9	363.1	117.7	176.4	107.4
GPU (Titan X)	1	dense	541.5	243.0	80.5	1467.8	243.0	80.5	65	90.1	51.9
		sparse	134.8	65.8	54.6	167.0	39.8	48.0	17.7	41.1	18.5
	64	dense	19.8	8.9	5.9	53.6	8.9	5.9	3.2	2.3	2.5
		sparse	94.6	51.5	23.2	121.5	24.4	22.0	10.9	11.0	9.0
mGPU (Tegra K1)	1	dense	12437.2	5765.0	2252.1	35427.0	5544.3	2243.1	1316	2565.5	956.9
		sparse	2879.3	1256.5	837.0	4377.2	626.3	745.1	240.6	570.6	315
	64	dense	1663.6	2056.8	298.0	2001.4	2050.7	483.9	87.8	956.3	95.2
		sparse	4003.9	1372.8	576.7	8024.8	660.2	544.1	236.3	187.7	186.5
EIE	**Theoretical Time**		**28.1**	**11.7**	**8.9**	**28.1**	**7.9**	**7.3**	**5.2**	**13.0**	**6.5**
	Actual Time		**30.3**	**12.2**	**9.9**	**34.4**	**8.7**	**8.4**	**8.0**	**13.9**	**7.5**

Figure 5: Wall clock time comparison for sparse and dense matrices between CPU, GPU, mGPU and EIE (an FPGA custom accelerator)
(Adopted figure from [15])

The paper proposing the EIE inference engine [15] provides a clear description of a technique for exploiting the sparity of activations by storing an encoded sparse weight matrix in a variant of compressed sparse column format [16].

3 Research Methodology

This is required for research projects and should be linked back to the project aim and objectives. It should describe the research methods that will be employed in the project and the research questions that will be investigated.

1. build dataset of benchmarks from my systematic benchmark framework from models
2. perform pruning on models
3. run benchmark again with pruning
4. make adjustments to underlaying mechanism of parameter storage
5. verify adjustments do not break the model
5. run benchmarks again
6. draw conclusions

Find baselines/benchmarks

How to perform pruneing

Look at underlaying storage mechanism of parameters in Network

- provide some plots visualising the sparsity of the weights for the pruned matrix

perform some engineering of refactoring/altering these mechanisms

rerun systematic benchmarking framework

4 Design

Initial software design/sketch of research Methodology

5 Evaluation Strategy

Details of the evaluation and analysis to be conducted

6 Project Management

6.1 Plan

6.2 Risk Analysis

mention benchmarking NLP/NLG/Audio - text/text - audio models as a risk to the project

6.3 Professional, Legal & Ethical issues

A Back matter

A.1 References

References

[1] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, and Q. Zhang, "Deep Learning on Mobile and Embedded Devices: State-of-the-art, Challenges, and Future Directions," *ACM Computing Surveys*, vol. 53, no. 4, pp. 1–37, Sep. 26, 2020, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3398209. [Online]. Available: https://dl.acm.org/doi/10.1145/3398209 (visited on 10/01/2020).

[2] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017, ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2017.2761740. [Online]. Available: http://ieeexplore.ieee.org/document/8114708/ (visited on 10/01/2020).

[3] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing*, vol. 3, e2, 2014, ISSN: 2048-7703. DOI: 10.1017/atsip.2013.9. [Online]. Available: https://www.cambridge.org/core/product/identifier/S2048770313000097/type/journal_article (visited on 10/16/2020).

[4] J. Thierry-Mieg, "How the fundamental concepts of mathematics and physics explain deep learning.," p. 16,

[5] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015, ISSN: 08936080. DOI: 10.1016/j.neunet.2014.

09.003. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0893608014002135 (visited on 10/21/2020).

[6] G. Hinton, D. Rumelhart, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088,

[7] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, no. 1, pp. 106–154, Jan. 1, 1962, ISSN: 00223751. DOI: 10.1113/jphysiol.1962.sp006837. [Online]. Available: http://doi.wiley.com/10.1113/jphysiol.1962.sp006837 (visited on 10/15/2020).

[8] Y. LeCun, Y. Bengio, and T. B. Laboratories, "Convolutional Networks for Images, Speech, and Time-Series," *The handbook of brain theory and neural networks MIT Press*, p. 15,

[9] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A Survey on Deep Learning: Algorithms, Techniques, and Applications," *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–36, Jan. 23, 2019, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3234150. [Online]. Available: https://dl.acm.org/doi/10.1145/3234150 (visited on 10/15/2020).

[10] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, Apr. 1980, ISSN: 0340-1200, 1432-0770. DOI: 10.1007/BF00344251. [Online]. Available: http://link.springer.com/10.1007/BF00344251 (visited on 10/18/2020).

[11] ——, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, no. 2, pp. 119–130, Jan. 1988, ISSN: 08936080. DOI:

10.1016/0893-6080(88)90014-7. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0893608088900147 (visited on 10/18/2020).

[12] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019, ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2019.2921977. [Online]. Available: https://ieeexplore.ieee.org/document/8763885/ (visited on 10/01/2020).

[13] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar, and S. W. Keckler. (Jul. 28, 2016). "vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design." arXiv: 1602.08124 [cs], [Online]. Available: http://arxiv.org/abs/1602.08124 (visited on 10/30/2020).

[14] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, "Going Deeper with Embedded FPGA Platform for Convolutional Neural Network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '16, New York, NY, USA: Association for Computing Machinery, Feb. 21, 2016, pp. 26–35, ISBN: 978-1-4503-3856-1. DOI: 10.1145/2847263.2847265. [Online]. Available: https://doi.org/10.1145/2847263.2847265 (visited on 11/02/2020).

[15] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, Seoul, South Korea: IEEE, Jun. 2016, pp. 243–254, ISBN: 978-1-4673-8947-1. DOI: 10.1109/ISCA.2016.30. [Online]. Available: http://ieeexplore.ieee.org/document/7551397/ (visited on 11/02/2020).

[16] R. Vuduc, "Automatic Performance Tuning of Sparse Matrix Kernels," p. 455,

A.2 Appendices

to include additional material, consult with your supervisor.

Acronyms

CNN Convolutional Neural Network. 2, 4, 5, 6

DNN Deep Neural Network. 4, 5

NLP Natural Language Processing. 5

RNN Recurrent Neural Network. 2, 4, 5, 6