# Approximation of Weight Matrices using Hierarchical Matrices

Till Hülder
Chair of Data Processing, Technical University of Munich
till.huelder@tum.de

*Abstract*—**Deep neural networks have shown their high performance accuracy in many areas such as image and speech recognition. Further they are also associated with high computational costs. In this paper, the approximation of a weight matrices is investigated in terms of time and prediction accuracy. Hierarchical matrices ($\mathcal{H}$-matrices) are used for the approximation. In order to find a suitable $\mathcal{H}$-matrix approximation, submatrices which are approximately low rank must be found in the original matrix. Various variations in algorithm such as the low-rank approximation method and the rank were investigated. From Pre-trained Pytorch models such as ResNet, GoogLeNet and MobileNetV2 the last layers were extracted and approximated. The ImageNet dataset was used for testing. For all tested models it was shown that the time required for a matrix vector operation is be significantly smaller for an approximated matrix. By using GoogLeNet with an approximation rank of 30, 43.89 % of the computing time was saved with a percentage accuracy loss of 5.69 %.**

***Keywords*—*Neural Network, Deep Learning, Hierarchical Matrices, Weight Matrices, Compression***

## I. Introduction

Over the past few years, machine learning has become a powerful tool in various application areas such as image and speech recognition. Since the possibilities of this technology are promising, many researchers worked on developing better algorithms [5],[6]. Deep neural networks and overparameterised networks have commonly played a major role in this process, as they are of great use for diverse applications [7], [8]. As a result of this trend, weight matrices have also become larger. For computing the output of a layer, the matrix-vector product between the weight matrix of the layer with the outputs of the previous layer must be computed. Since a matrix-vector multiplication of quadratic $n \times n$ matrices requires a computational cost of $\mathcal{O}(n^2)$ and this is very laborious with large matrices, solutions are sought to reduce the computational effort. There are some approaches to tackle this problem. Among others, weight matrices with low displacement rank [9] or block-circulant matrices [10] were used to reduce the computational effort. The concept of approximating matrices using hierarchical matrices ($\mathcal{H}$-matrices) was introduced by Hackbusch [11], [12]. This method allows matrix operations to be performed in $\mathcal{O}(n \log(n))$ operation effort with an approximation error and to reduce the memory space. Due to these properties, hierarchical matrices have been applied in some areas such as solving the helmholtz equation [13] or the numerical analysis of multi-crack large-scale plane problems [14]. The aim of this method is to divide a matrix into submatrices and, if possible, to approximate them with matrices of lower rank. If this is not possible, the submatrix is further subdivided or, if the submatrices are small enough, the submatrices are left in full rank. This paper investigates how far this method is applicable to weight matrices of deep neural networks. In addition, it will be investigated how the trade-off between reduction of computational operations versus loss of accuracy is when using the approximated weight matrix. The method is tested on known weight matrices from trained neural networks such as ResNet [1], GoogLeNet [2] or MobileNetV2 [3], which are provided by Torchvision [1].
The paper is organized as follows. First, I give an overview over the state of the art in literature. The following chapter presents problems and questions that will be answered in the course of this paper. The subsequent chapter presents the approach used to answer the questions. In the final chapters, the experiments are presented, the results are discussed and a Conclusion is given.

## II. State of the art – Previous work

Since large neural networks are used in a wide range of applications and are often accompanied by a large amount of data, it is important to work on efficiency in terms of time and computing capacity. There are several approaches to compress neural networks such as pruning [15] or quantization [16]. Nevertheless, large neural networks are often used with correspondingly large weight matrices. These matrices should be represented compactly in order to keep the computational effort as low as possible. A popular method for the compression of such matrices is a low-rank approximation. This can be done in various ways such as singular value decomposition (SVD) [17] or sparse low-rank factorization [18]. A combination of the different methods mentioned can also be used. Depending on the application, different methods come into account and have different characteristics. Since the use of hierarchical matrices ($\mathcal{H}$-matrices) in the approximation of weight matrices has not yet been investigated in detail, this paper will discuss whether this method is an alternative to the existing methods.

## III. Research Questions

Since $\mathcal{H}$-matrices can reduce the complexity of matrix vector multiplication, the question arises to what extent this method can help to approximate weight matrices of deep

[1]Torchvision, Software Libary, https://pytorch.org/vision/stable/models.html

neural networks and how accurate such approximated weight matrices are. In the following it will be explored which methods are available to represent a weight matrix by an $\mathcal{H}$-matrix and see how these methods work in this specific case. The Neuronal Network of Pre-trained pytorch models (ResNet [1], GoogLeNet [2], MobileNetV2 [3]) were used as weight matrices. From these models, the last dense layer of the neural network was extracted and used. In addition, it should be determined how computationally intensive and accurate the matrix vector multiplication is.

## IV. Methodology

The scope of this paper is approximating weight matrices with $\mathcal{H}$-matrices. To evaluate the method the accuracy and computational cost (time) of matrix vector multiplication are used.

### A. Weight Matrix Approximation

The HLIBpro C++ software library[2] is used to approximate the weight matrices. It is free available for academic purposes. To represent a matrix by an H-matrix, the following main steps are necessary:

- Clustering
  The matrix is to be decomposed into subareas that are suitable for the approximation. To define sub-areas, tree structures are used for reasons of efficiency. The tree structure can be determined by different methods like the geometrically balanced method or cardinality balanced method. In this paper the method TAutoBSPPartStrat, that analyses the coordinates data and selects an appropriate strategy is used. Separate cluster trees must be built for rows ($T_\mathrm{i}$) and columns ($T_\mathrm{j}$). The leaves of the tree are therefore row or column index sets. It divid the index sets in a way that the index set is halved. In addition, it is possible to define how small the minimum size of a leaf of the cluster tree can be.

- Blockclustering
  During block clustering, the sub matrices are formed from the columns and row indices. For these sub areas it is checked whether they can be approximated by a low rank matrix. For this purpose, the admissibility condition is introduced:

  $$\max(diam(s), diam(t)) \leq \eta \cdot dist(s,t) \qquad s,t \in T_\mathrm{j}, T_\mathrm{i}$$

  The method $diam()$ indicates the diameter of a cluster and $dist()$ the distance between two clusters [11]. The parameter $\eta$ determines the structure of the $\mathcal{H}$-matrices. The more the parameter approaches 0, the smaller the blocks become. If the admissibility condition is fulfilled, the matrix can be approximated. If this is not fulfilled, the block is further subdivided and if the size of the block is equal to the minimum leaf size, it is saved completely.

- Matrix Construction
  Here a matrix is formed based on the block cluster tree. Various low-rank approximation methods for the permissible blocks can be used. For example, singular value decomposition (SVD) [19], adaptive cross approximation (ACA) [20] or rank revealing QR-Decomposition (QR) [21] can be used, as well as variations of these. It is important to note that each method has its own characteristics. Additionally, the desired rank of the approximated matrix must be specified. The specified rank has a direct influence on how accurately the submatrices are approximated and how computationally intensive the further use of the H-matrix is. The higher the rank, the more accurately the submatrix can be approximated and the more computationally intensive is the further use of the matrix. An example illustration of an approximated matrix with hierarchical structure is depicted in Fig. 1.

[2]Max Planck Institute for Mathematics in the Sciences (MiS) in Leipzig, Software Libary for $\mathcal{H}$-matrices, https://www.hlibpro.com/doc/2.9/index.html

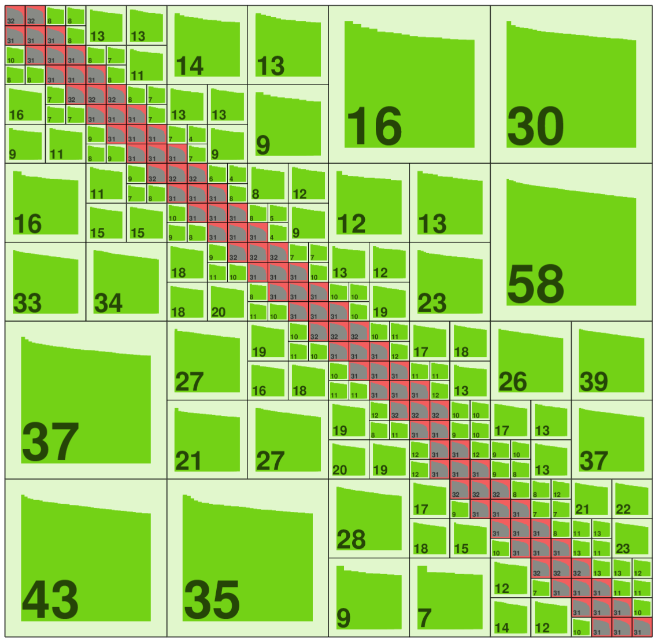

Fig. 1. Weight matrix of GoogLeNet [2] approximated by SVD and accuracy=0.5 . The green blocks represent the approximated low rank submatrices and the red blocks represent the non-approximated, full-rank submatrices. The numbers in the green submatrices indicate the rank of the approximated submatrix.

### B. Analysis of the approximated Matrix

In my experiments, I compared the approximated matrices with the original matrices in terms of the time needed for performing a matrix-vector product and the prediction accuracy of the model in which the matrix was embedded. For the measurment, 100000 matrix-vector multipilations were performed in the script for different approximation levels of the

submatrices and the time was stopped. These measurements were carried out using different approximation methods (SVD, ACA, QR) and different weight matrices (ResNet, GoogLeNet, MobileNetV2). Alternating with the measurements of the H-matrices vector- Matrix multiplication, the non-approximated weight matrices were multiplied 100000 times and the time was measured. This value was used as a comparison value. In order to produce comparable results, all measurements were carried out on the same computer (Intel(R) Core(TM) i7-8550U CPU).
For the accuracy analysis, I embedded the approximatd $\mathcal{H}$-matrices into the original pre-trained neural network. This Model was tested using the validation data set of the ImageNet [4].This process was again repeated out with different weight matrices (ResNet, GoogLeNet, MobileNetV2), different approximation methods (SVD, ACA, QR) and various approximations ranks. The error of the original pre-trained neural network over the same data set was used as a comparison value for the accuracy.

## V. Experiments

In order to obtain different parameters for the analysis of the $\mathcal{H}$-matrix, different low rank approximation algorithms (SVD, QR, ACA) of the subblocks were considered. These methods were each measured with the same rank approximation of the sub-blocks to obtain a comparable result.
The time measurement Figure 2 carried out for matrix vector multiplications concluded that the SVD method (dotted green line) is the most time-efficient method, followed by the QR method (dashed orange line) and the ACA method (dashdoted blue line). The red line in Figure 2 indicates the reference and is the multiplication with the same vector and the unapproximated matrix. Another effect is that the

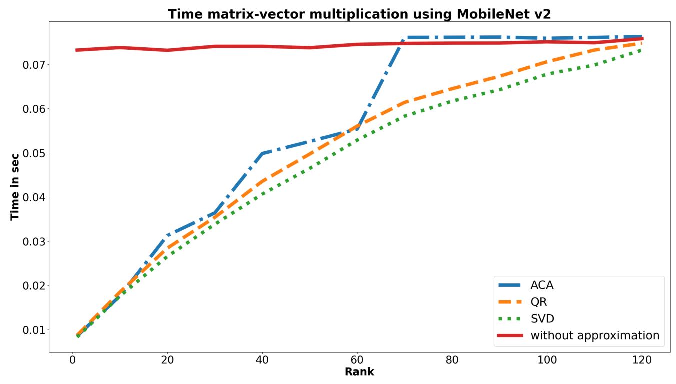

Fig. 2. Time measurement 100 000 matrix vector mulitplications with different approximation methods (SVD, QR, ACA) using MobileNetV2. [3]

memory requirements for the matrices become smaller with the approximation, as illustrated in Figure 3. The SVD- and QR method can compress the matrix to the same memory size. The ACA method (dashdoted blue line) provides the smallest compression. Afer a rank of 70, it has no memory advantages compared to the non-approximation (red line).

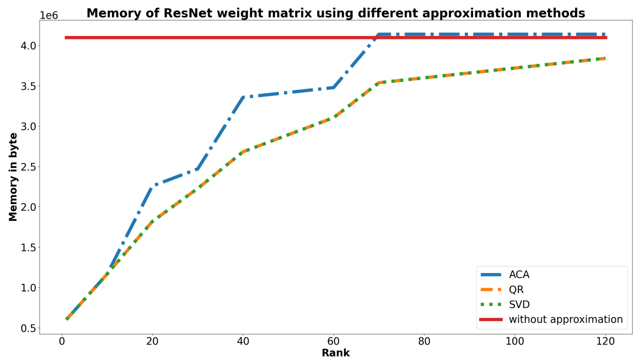

Fig. 3. Storage capacity for different approximation for ResNet [1] and various approximation methods (SVD, QR, ACA).

A difference between the chosen methods is also visible in the accuracy analysis using the GoogLeNet Fig.4. In this case the SVD turns out to be the most accurate method, followed by the QR method. This behavior was also seen with the other models (ResNet, MobileNetV2), as shown in Table 1.
In terms of accuracy and time analysis, it can be seen that

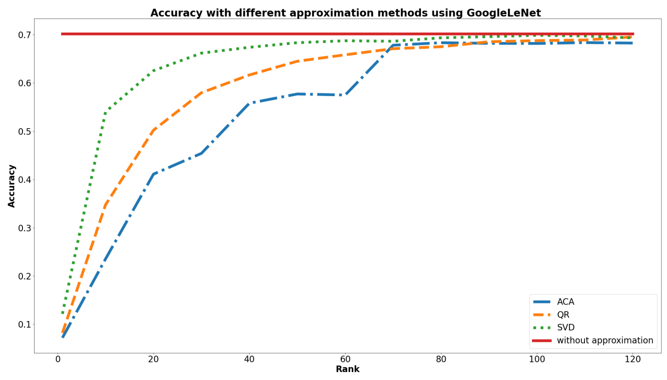

Fig. 4. Accuracy measurement using pre-trained neural network GoogLeNet [2] and various approximation methods (SVD, QR, ACA). The validation dataset of ImageNet [4] was used.

the SVD performs better than the other methods (QR, ACA) . The performance of the SVD with different models (ResNet, GoogLeNet, MobileNetV2) is shown in Table I. The tendency for lower rank to result in higher time savings and greater accuracy loss is confirmed here for the various models. it can also be seen that the time savings and accuracy loss can vary depending on the model.

## VI. Discussion

From the experiment results in Section 5 it can be seen that an approximation of the weight matrix of a neural network with an $\mathcal{H}$-matrix can save a significant time in matrix-vector multiplication. This could be demonstrated in all three weight matrices used, as shown in Table 1. How much time is saved depends on the type of low rank approximation of the submatrices, the chosen rank of the approximation and the underlying

| modell | rank | time saving in % | accuracy loss in % |
|---|---|---|---|
| MobileNet V2 | 30 | 54,56 % | 8,56 % |
| MobileNet V2 | 50 | 37,54 % | 4,27 % |
| MobileNet V2 | 70 | 21,64 % | 2,55 % |
| GoogLeNet | 30 | 43,89 % | 5,69 % |
| GoogLeNet | 50 | 26,09 % | 2,62 % |
| GoogLeNet | 70 | 13,72 % | 2,19 % |
| ResNet | 30 | 36,43 % | 16,46 % |
| ResNet | 50 | 19,11 % | 7,43 % |
| ResNet | 70 | 4,02 % | 3,85 % |

TABLE I. TIME SAVING AND ACCURACY PERFORMANCE OF THE SVD WITH DIFFERENT MODELS (RESNET [1], GOOGLENET [2], MOBILENETV2 [3])

matrix. The smaller the low rank approximation chosen, the more time could be saved. In terms of approximation, the QR and SVD methods were found to be close in terms of time savings. SVD proved to be slightly faster. In contrast to the other methods, the ACA method had a rather choppy curve showing a tendency to save less time. It can be seen in the Fig. 2 that all three approximation algorithms are very close to each other up to approximation rank 20 and only then the different trend of time saving become visible. In general, one can clearly see that the lower the aproximation rank was chosen, the more time could be saved.
Similar findings to the time savings can be derived from the examination of the storage capacitz in Fig. 2. Here, however, it can be seen that the SVD and QR achieve equal results. As with the time analysis, the ACA method has a similarly choppy course and, from a rank of 70, virtually no more advantages over the non-approximated matrix.
In the accuracy analysis, the low rank approximation method SVD turned out to be the most accurate of the examined methods. The second most accurate was the QR method followed by the ACA method. With all methods it can clearly be seen that with a lower approximation rank the accuracy decreases strongly see Fig.4. With a higher approximation rank the accuracy remains relatively constant and is only slightly worse than without approximation. Each approximation curve begins to fall off at a different rank order. The SVD manages to remain stable for the longest time and provides more accurate results than QR and ACA methods. This might be due to the the Eckart-Young-Mirsky theorem [22], which states that the best possible low rank approximation takes place using SVD. The results of the time and accuracy analysis show that the time savings and loss of accuracy are proportional to each other. The lower the rank of the submatrices, the more time is saved and the accuracy decreases. This leads to a trade off in which, depending on the application, one has to decide how much decrease in the accuracy can be tolerated.
Furthermore, the experiments show that it is difficult to make general statements. Table I shows that the approximation results in different amounts of time gain and accuracy loss depending on the neural network used. This could have to do with the fact that the size of the weight matrix can vary depending on the hyperparameter of the neural network.

## VII. CONCLUSION

The test results showed that the approximation of hierarchical matrices is suitable for approximating weight matrices of neural networks. It can lead to a reduction of computational cost. However, accuracy is always decreasing with an approximation. The gain in computational speed and the loss in accuracy cannot be expressed in general terms. They vary depending on the weight matrix. It is therefore important to take a close look at the application. In the experiments, the SVD turned out to be the most suitable low rank approximation of the subblocks. The QR method follows the SVD with a little performance loss. The low rank approximation method ACA can only reduce the computational effort in places with low performance compared to SVD. There is a trade-off between the time gain and the loss of accuracy. The lower the approximated rank of the submatrix, the higher the time gain and the lower the accuracy.

## REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” 2014.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” 2019.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[5] C. Langlotz, B. Allen, B. Erickson, J. Kalpathy-Cramer, K. Bigelow, T. Cook, A. Flanders, M. Lungren, D. Mendelson, J. Rudie, G. Wang, and K. Kandarpa, “A roadmap for foundational research on artificial intelligence in medical imaging: From the 2018 nih/rsna/acr/the academy workshop,” 2019.

[6] Z. Tüske, G. Saon, and B. Kingsbury, “On the limit of english conversational speech recognition,” 2021.

[7] M. Belkin, D. Hsu, S. Ma, and S. Mandal, “Reconciling modern machine learning practice and the bias variance trade off,” 2019.

[8] Z. Allen-Zhu, Y. Li, and Y. Liang, “Learning and generalization in overparameterized neural networks, going beyond two layers,” 2020.

[9] L. Zhao, S. Liao, Y. Wang, J. Tang, and B. Yuan, “Theoretical properties for neural networks with weight matrices of low displacement rank,” 2017.

[10] C. Ding, S. Liao, Y. Wang, Z. Li, N. Liu, Y. Zhuo, C. Wang, X. Qian, Y. Bai, G. Yuan, X. Ma, Y. Zhang, J. Tang, Q. Qiu, X. Lin, and B. Yuan, “Circnn: Accelerating and compressing deep neural networks using block-circulantweight matrices,” *CoRR*, vol. abs/1708.08917, 2017.

[11] W. Hackbusch, *Hierarchische Matrizen : Algorithmen und Analysis*, 2009.

[12] W. Hackbusch, L. Grasedyck, and S. Börm, “An introduction to hierarchical matrices,” *Mathematica Bohemica, v.127, 229-241 (2002)*, vol. 127, 2002.

[13] O. Z. Boris Dilba, Otto von Estorff, “Hierarchische matrizen für die helmholtzgleichung,” *DAGA*, 2014.

[14] T. Grytsenko and A. Galybin, “Numerical analysis of multi-crack large-scale plane problems with adaptive cross approximation and hierarchical matrices,” *Engineering Analysis with Boundary Elements*, vol. 34, no. 5, pp. 501–510, 2010.

[15] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” 2016.

[16] J. O’Neill, “An overview of neural network compression,” *CoRR*, vol. abs/2006.03669, 2020.

[17] J. Xue, J. Li, and Y. Gong, “Restructuring of deep neural network acoustic models with singular value decomposition,” in *INTERSPEECH*, 2013.

[18] S. Swaminathan, D. Garg, R. Kannan, and F. Andres, “Sparse low rank factorization for deep neural network compression,” *Neurocomputing*, vol. 398, pp. 185–196, 2020.

[19] K. Fernando and B. Parlett, “Accurate singular values and differential qd algorithms,” *Numerische Mathematik*, vol. 67, pp. 191–229, 1994.

[20] Y. Liu, W. Sid-Lakhdar, E. Rebrova, P. Ghysels, and X. S. Li, “A parallel hierarchical blocked adaptive cross approximation algorithm,” 2019.

[21] J. J. Dongarra, M. Faverge, T. Hérault, J. Langou, and Y. Robert, “Hierarchical QR factorization algorithms for multi-core cluster systems,” 2011.

[22] G. Golub, A. Hoffman, and G. Stewart, “A generalization of the eckart-young-mirsky matrix approximation theorem,” *Linear Algebra and its Applications*, vol. 88-89, pp. 317–327, 1987.

**Till Hülder** recieved his Bachelor’s degree in Electrical and Computer Engineering from the Karlsruhe Institute of Technology (KIT), Germany, in April 2020. He is currently pursing his Master’s degree in Electrical and Computer Engineering at the Technical University of Munich (TUM), Germany, with a specialization in communications engineering and data science.