Large Neural Networks at a Fraction

LTH Experiments on Quaternion Neural Networks

Aritra Mukhopadhyay[1] Adhilsha A[1] Dr. Subhankar Mishra[2]

[1]School of Physical Sciences
NISER Bhubaneswar and HBNI Mumbai, India

[2]School of Computer Sciences
NISER Bhubaneswar and HBNI Mumbai, India

Northern Lights Deep Learning Conference, January 2024

Motivation

In our quest for more efficient neural networks on basic devices, quaternion networks prove handy by cutting down on parameters. This means big models can run smoothly even on low-end gadgets without compromising much on their efficiency, making technology more accessible to everyone.

Quaternion Neural Networks

Quaternion Neural Networks are those that use quaternions ($q = a + bi + cj + dk$) instead of Real numbers.

Quaternion Neural Networks

Quaternion Neural Networks are those that use quaternions ($q = a + bi + cj + dk$) instead of Real numbers. The Hamilton product of two Quaternions is defined as:

$$\begin{aligned} q_1 \otimes q_2 = &(a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2) \\ &(a_1 b_2 + b_1 a_2 + c_1 d_2 - d_1 c_2) i \\ &(a_1 c_2 - b_1 d_2 + c_1 a_2 + d_1 b_2) j \\ &(a_1 d_2 + b_1 c_2 - c_1 b_2 + d_1 a_2) k \end{aligned} \tag{1}$$

where $q_1 = a_1 + b_1 i + c_1 j + d_1 k$ and $q_2 = a_2 + b_2 i + c_2 j + d_2 k$.

Quaternion Neural Networks

Quaternion Neural Networks are those that use quaternions ($q = a + bi + cj + dk$) instead of Real numbers. The Hamilton product of two Quaternions is defined as:

$$\begin{aligned} q_1 \otimes q_2 = &(a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2) \\ &(a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2)i \\ &(a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2)j \\ &(a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2)k \end{aligned} \tag{1}$$

where $q_1 = a_1 + b_1i + c_1j + d_1k$ and $q_2 = a_2 + b_2i + c_2j + d_2k$. Furthermore, the matrix representation of such a Quaternion can be formed, making sure the matrix multiplication of such is consistent with the Hamilton product.

$$q = \begin{bmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{bmatrix} \tag{2}$$

Quaternion Neural Networks

Quaternion Neural Networks have one-fourth the number of parameters compared to their Real versions.

Quaternion Neural Networks

Quaternion Neural Networks have one-fourth the number of parameters compared to their Real versions.

How does that happen?

Quaternion Neural Networks

Quaternion Neural Networks have one-fourth the number of parameters compared to their Real versions.

How does that happen?

Suppose in some **Real** layer, we have a weight matrix of shape ‘$m \times n$’ where $m, n \in \mathbb{N}$; The number of parameters is ‘mn’.

Quaternion Neural Networks

Quaternion Neural Networks have one-fourth the number of parameters compared to their Real versions.

How does that happen?

Suppose in some **Real** layer, we have a weight matrix of shape '$m \times n$' where $m, n \in \mathbb{N}$; The number of parameters is 'mn'.

Then the **quaternion** version will have **four** '$m/4 \times n/4$' matrices; number of parameters is '$mn/4$' (4 times lesser than Real). From these four matrices, we reconstruct the '$m \times n$' weight matrix as explained before. This process is explained in detail in Figure 1.

Reduction in Quaternion

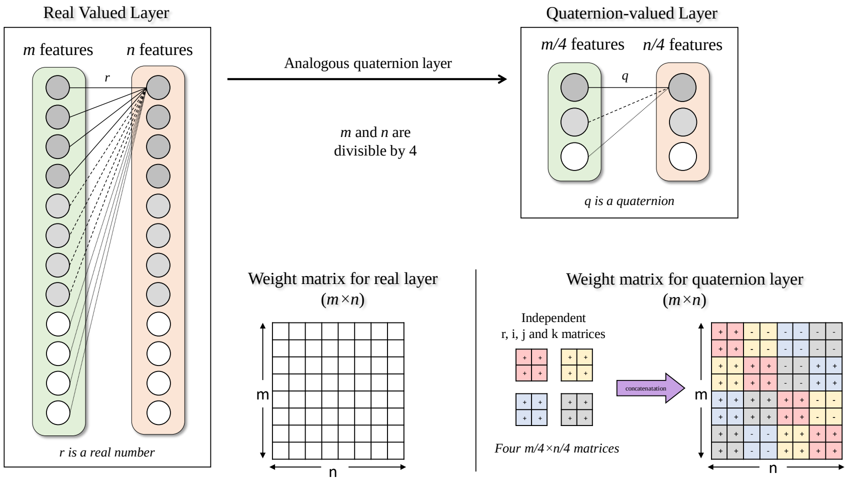


Figure: How Quaternion structure helps in weight reduction

LTH

The Lottery Ticket Hypothesis

‘A randomly-initialized, dense neural network contains a subnetwork that is initialized such that — when trained in isolation — it can match the test accuracy of the original network after training for at most the same number of iterations.’

– Jonathan Frankle and Michael Carbin. *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*. 2019. arXiv: 1803.03635 [cs.LG]

This ‘subnetwork’ (the ‘Lottery Ticket’) has just a subset of the weights of the original network. Thus reducing the number of parameter count of the model; hence increasing prediction speed.

Iterative Pruning

Given a model, we find the lottery ticket following the *prune-reset-train* loop:

- Initiate the model (say m_1) with some random weights.

Iterative Pruning

Given a model, we find the lottery ticket following the *prune-reset-train* loop:

- Initiate the model (say m_1) with some random weights.
- (train) Train m_1

Iterative Pruning

Given a model, we find the lottery ticket following the *prune-reset-train* loop:

- Initiate the model (say m_1) with some random weights.
- (train) Train m_1
- (prune) Prune trained m_1 to get m_2 (say after removing, p% weights remain)

Iterative Pruning

Given a model, we find the lottery ticket following the *prune-reset-train* loop:

- Initiate the model (say m_1) with some random weights.
- (train) Train m_1
- (prune) Prune trained m_1 to get m_2 (say after removing, p% weights remain)
- (reset) We set the remaining weights in m_2 to the weights in untrained m_1

Iterative Pruning

Given a model, we find the lottery ticket following the *prune-reset-train* loop:

- Initiate the model (say m_1) with some random weights.
- (train) Train m_1
- (prune) Prune trained m_1 to get m_2 (say after removing, p% weights remain)
- (reset) We set the remaining weights in m_2 to the weights in untrained m_1
- (repeat) We train this model, and repeat the the process.

Iterative Pruning (Some Information)

As we prune more and more, the model has a lesser number of weights, and a point comes when the accuracy decreases and becomes ‘too low’ (for us, a few iterations after accuracy dropped below 20%). We stop the process there.

Iterative Pruning (Some Information)

As we prune more and more, the model has a lesser number of weights, and a point comes when the accuracy decreases and becomes ‘too low’ (for us, a few iterations after accuracy dropped below 20%). We stop the process there.

After the i^{th} pruning iteration, the percentage of weights remaining in the model is $(100 \times p^i)\%$

Our Work

The previous paper[1] compared the accuracy vs number of weights on pruning for both Real and quaternion models and found that in quaternion models, the accuracy remains comparable to the full model till higher sparsities than that in the Real version.

The said paper used small models and small datasets for the observations. In this paper, we work towards validating the same results for bigger models and larger datasets.

[1] Sahel Mohammad Iqbal and Subhankar Mishra. "Neural Networks at a Fraction with Pruned Quaternions". In: *Proceedings of the 6th Joint International Conference on Data Science amp; Management of Data (10th ACM IKDD CODS and 28th COMAD)*. CODS-COMAD 2023. ACM, Jan. 2023. DOI: 10.1145/3570991.3570997. URL: http://dx.doi.org/10.1145/3570991.3570997.

Model Statistics

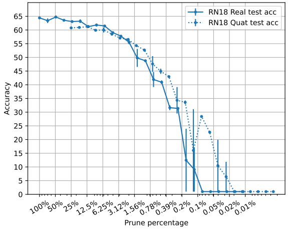
Model	Number of Parameters	
	Real Version	Quaternion Version
ResNet18	11.68M	2.93M
ResNet34	21.79M	5.46M
ResNet50	25.55M	6.43M
ResNet101	44.54M	11.22M
ResNet152	60.19M	15.16M

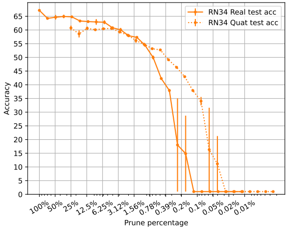
Dataset Statistics

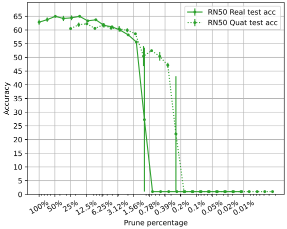
Dataset	Images	Resolution	Classes
CIFAR100	60K	32x32 (coloured)	100
ImageNet64	1.28M	64x64 (coloured)	1000

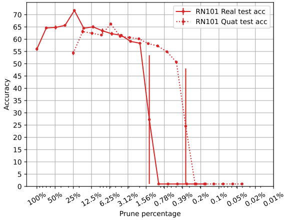
Results

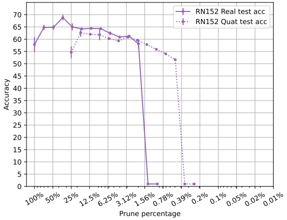
Validation (top 1) accuracy vs pruning results of all ResNet models on CIFAR100 dataset.

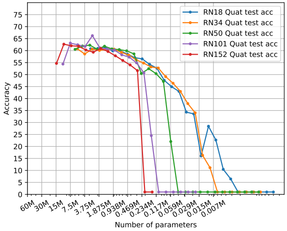






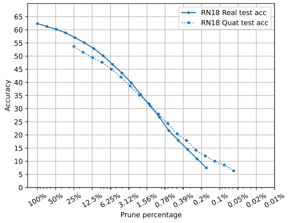


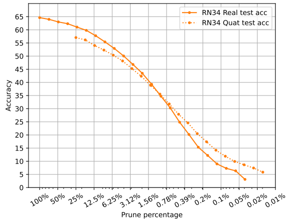


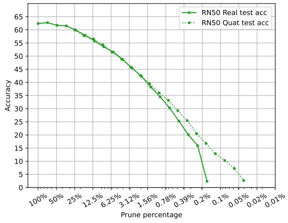


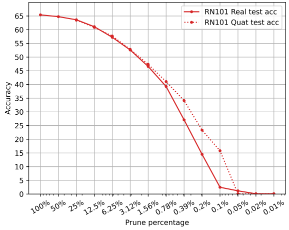
Results

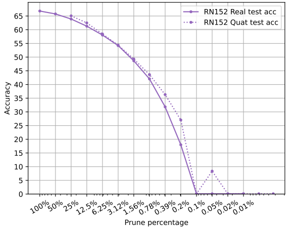
Validation (top 1) accuracy vs pruning results of all ResNet models on ImageNet64 dataset.

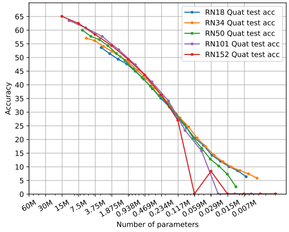












Discussion

- The observation that quaternion models achieve comparable accuracies to Real models with only one-fourth of the parameters challenges conventional understanding of model complexity and parameter count.
- This suggests that Real models in our study were potentially over-parameterized, surpassing the required capacity for learning the dataset, highlighting the efficiency and effectiveness of quaternion models.

Discussion: CIFAR100 and ImageNet64 Comparisons

CIFAR100 Real vs. Quat:

- For CIFAR100, parameter count of the smallest Real ResNet (ResNet18) is near the quaternion ResNet101; a much larger model. Quaternion models outperform Real models, emphasizing this advantage primarily in scenarios with smaller datasets relative to the model's capacity.

ImageNet64 Real vs. Quat:

- Results for ImageNet64 demonstrate that quaternion models, despite having sufficient parameters, do not yield satisfactory accuracy. This could be due to a mismatch in hyperparameters favoring Real models, indicating the need for further investigation.

Discussion: Accuracy Trends on ImageNet Dataset

- The declining accuracies imply that the models were inadequately equipped with parameters. They failed to achieve optimal over-parameterization necessary for comprehending the complexities of the ImageNet dataset, suggesting a potential need for more parameters.
- With larger models (RN101 and RN152), the initial comparable accuracies indicate the possibility of a Lottery Ticket phenomenon under the right hyperparameter tuning, emphasizing the importance of exploring the parameter space for optimal model performance.

Future Works

- Acknowledging some challenges, our next steps involve fine-tuning quaternion models to perform even better than Real models by finding the right settings.
- We also plan to test these models on bigger datasets and explore their use in different tasks like object detection, NLP, and more.
- Additionally, we want to see how well quaternion models work in various types of neural networks, going beyond just image-related tasks.

Thank You!