Large Neural Networks at a Fraction

Aritra Mukhopadhyay[1], Adhilsha Ansad[1], and Subhankar Mishra*[2]

[1]School of Physical Sciences, NISER
[2]School of Computer Sciences, NISER
{aritra.mukhopadhyay, adhilsha.a, smishra}@niser.ac.in

Abstract

Large-scale deep learning models are known for their large amount of parameters, weighing down on the computational resources. The core of the Lottery Ticket Hypothesis showed us the potential of pruning to reduce such parameters without a significant drop in accuracy. Quaternion neural networks achieve comparable accuracy to equivalent real-valued networks on multi-dimensional prediction tasks. In our work, we implement pruning on real and quaternion-valued implementations of large-scale networks in the task of image recognition. For instance, our implementation of the ResNet-101 architecture on the CIFAR-100 and ImageNet64x64 datasets resulted in pruned quaternion models outperforming their real-valued counterparts by 4% and 7% in accuracy at sparsities of about 6% and 0.4%, respectively. We also got quaternion implementations of ResNet-101 and ResNet-152 on CIFAR-100 with steady Lottery tickets, whereas the Real counterpart failed to train at the same sparsity. Our experiments show that the pruned quaternion implementations perform better at higher sparsity than the corresponding real-valued counterpart, even in some larger neural networks.

1 Introduction

In the realm of large-scale and complex learning tasks, the presence of a higher number of trainable parameters within deep learning networks is highly sought after. Notably, numerous cutting-edge deep learning networks, as evidenced by works of Nitta [1], Devlin et al. [2], and Li et al. [3], boast millions of parameters, if not more. However, it is crucial to acknowledge that such parameter-rich networks also impose significant strain on computational resources and expenses. Consequently, we have been pursuing the ideal of developing low-parameter efficient models that incur minimal costs.

Pruning done on neural networks has been focused on maintaining accuracy on par with the original network as their core objective [4]. It is essential to understand that the pruned model can lose accuracy after extreme levels of sparsity [5, 6]. Still, the models pruned to higher sparsities as much as 90-98% is also interesting as they can outperform smaller dense models with comparable size (same number of parameters) [7–9]. This motivates the pruning of large-scale models regardless of the dropping accuracies. The quaternion-valued implementation of neural networks used in hyperdimensional data embeddings [10] has been found to have as much as 75% parameter reduction (explained in section 3.2) without significant loss of accuracy [10–13]. Quaternion networks for various ML tasks already exist [11, 14–17] and have proved to be a worthy competitor.

*Corresponding Author.

2 Background

2.1 Pruning

Modern neural network architectures often use an excessive parameter count [6], leading to increased computational and memory demands during inference. Weight pruning has gained attention as a means to reduce model complexity without sacrificing accuracy [4, 18, 19], with successful applications across various network architectures like VGG [20] and ResNet [6, 9, 21]. Weight magnitude pruning [5], a prominent approach, removes redundant and less influential connections, resulting in streamlined models. However, careful reinitialization techniques are required to mitigate initial performance degradation after pruning [6].

Pruning techniques can be categorized into structured and unstructured pruning. Structured pruning removes entire structured components like filters, channels, or layers for higher inference speed, while unstructured pruning eliminates individual weights, often outperforming structured pruning in reducing model complexity. Advancements in sparse operations [22] and specialized hardware for sparse multiplication [23] have improved the computational efficiency of pruned models.

One compelling aspect of pruned networks is their ability to outperform small dense models with the same number of weights [9]. This makes them preferable under resource constraints, even if their performance doesn't match with the unpruned model across various model architectures and tasks [7–9, 24]. This highlights the potential of pruned models to balance model size and performance.

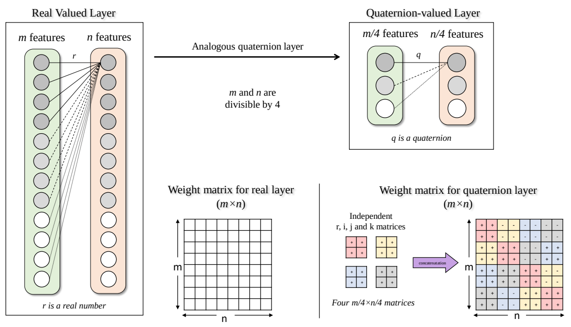


Figure 1. Quaternion matrix representation and parameter reduction illustration

2.2 Quaternion

The realm of multidimensional data embeddings starts with complex numbers and has been applied in music transcription tasks [12]. However, higher-dimensional data types are needed in scenarios involving data types like images with three channels. The question of whether Quaternion data embedding is optimal or not has been explored by Parcollet et al. [10], shedding light on the potential benefits of employing such techniques. Zhu et al. [25] compared real and quaternion neural networks with similar architectures and the same parameter count on the CIFAR10 dataset, which showed the superiority of quaternion neural networks in both training loss convergence and performance. Quaternion embeddings have also shown promise in various domains, including image classification [11] and speech recognition [14], demonstrating their potential for effectively capturing and representing complex data structures. A quaternion q can be defined as $a + bi + cj + dk$. 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.

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

Conventional neural networks are built using real numbers, where each neuron and weight are represented as real values. In these networks, each layer needs $p \times n$ weights, where p denotes the number of weights in the preceding layer, and n represents the number of weights in the subsequent layer. In the case of Quaternion networks, four neurons can be represented using one quaternion represented in Figure 1. Therefore, only $\frac{p}{4} \times \frac{n}{4}$ quaternion weights are left. It should be noted that all of these weights must be Quaternions rather than real numbers. Thus, four numbers are required, resulting in a total of of $4 \times \frac{p}{4} \times \frac{n}{4} = \frac{p \times n}{4}$ number of weights. Thus, quaternion offers a 75% reduction in weight. Even though the parameters get reduced, the number of calculations remains the same due to the properties of the matrix representation.

3 Method

3.1 Terminology

Throughout this paper, we adopt some abbreviations and terms to refer to different aspects of our work. 'Quat' or '$\boldsymbol{Q}$' denotes quaternion-valued implementations of neural networks, while 'Real' or '$\boldsymbol{R}$' denotes real-valued implementations. 'ImageNet64' or just 'ImageNet' to refer to the downsized 64x64 version of the ImageNet dataset mentioned below, and 'ILSVRC' to refer to the full version of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset. 'ResNet' or 'RN' is short for Residual Network, a deep convolutional neural network architecture used in our experiments. 'BN' refers to Batch Normalization, and 'n% sparsity' refers to weight matrix sparsity at which n% trainable weights are left.

3.2 Datasets and models

For the experiments, we used the CIFAR100 [26] and the ImageNet64 [27] datasets. ImageNet64 was constructed by applying cropping and other transformations to the original images from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [28] dataset, resulting in a dataset of 64x64 colored images. The training set consists of approximately 1.28 million images of various sizes and resolutions, labeled over 1,000 classes, and there are 50,000 images in the validation set.

To ensure compatibility with quaternion layers, it is necessary to have an image with a channel count that is a multiple of 4. Previous attempts to address this requirement have involved various approaches, such as setting the last channel to all zeros or incorporating dedicated layers to learn the content of the last channel [11]. In our work, we adopted the utilization of grayscale values as the fourth channel, as also used by Iqbal and Mishra [29]. However, it is worth noting that they exclusively used 4-channel images for quaternion models, whereas we extended the use of 4-channel images to both quaternion and real models for easier comparison.

Table 1. Hyperparameters for the experiment

Datasets Models	CIFAR100 All ResNet models	ImageNet64 ResNet18, ResNet34, ResNet50	ImageNet64 ResNet101, Resnet152
Training epochs/batch size	100/100	15/256	15/256
Optimizer	SGD	SGD	SGD
Learning Rate Schedule (epochs)	0.1 (1-100)	0.1 (1-10), 0.01 (11-12), 0.001 (13-15)	0.1 (1-10), 0.01 (11-12), 0.001 (13-15)
Momentum / Weight decay	0.9/0.0001	0.9/0.0001	0.9/0.0001
% Pruned per pruning iteration	70%	70%	50%

In terms of architecture, the Real ResNet architectures from the repository of Liu et al. [30] was utilized, with a modification to accommodate 4-channel input and included an adaptive average pooling layer before the final linear layer. This pooling layer served the purpose of handling varying resolutions across datasets, namely 32x32 CIFAR100 and 64x64 ImageNet64. To create the Quaternion models, the real layers, both linear and convolutional, were replaced with quaternion layers. We retained the BN (Batch Normalization) layers in real value, as implementing BN in quaternion form didn't reduce parameter count but had a massive negative effect on training speed. Our experiments employed standard architectures, including ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152, with the number of trainable parameters detailed in Table 2.

3.3 Setup

For the CIFAR100 experiments,a computer with one NVIDIA A100 Server Grade Tensor Core GPU with 80GB VRAM and 512GB RAM was used. All the ImageNet (except the RN101) experiments were run on a computer with four NVIDIA A100 80GB GPUs and 1TB RAM. The ResNet101 ImageNet experiments were run on a computer with 1 NVIDIA A6000 48GB GPU and 125GB RAM. All the experiments were run on a Linux operating system. The code was written using `PyTorch` library [31] and ran in Python 3.10.6. No sort of multi-GPU or multi-node parallelization was used for the experiments. For the quaternion layers, the `hTorch` library[1] has been referred and used. Our codes are available at our Github repository[2].

3.4 Experiments

We used different hyperparameters over different datasets and models, but we always used the same hyperparameters for the Real and Quat versions of the same model-dataset pair. The hyperparameters for the experiment are summarized in Table 1.

[1]https://github.com/ispamm/hTorch
[2]https://github.com/smlab-niser/quatLT23

Table 2. Table showing trainable parameter count for different models

Model arch	Real Models	Quat models
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

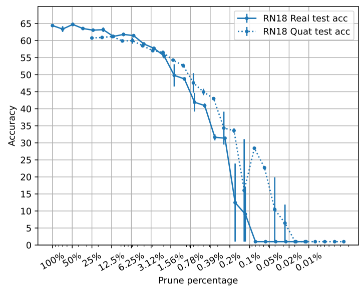
3.4.1 Training

For the most part, we stuck to the widely used, popular set of hyperparameters used for ResNet architectures. For the CIFAR100 dataset, we utilized a learning rate scheduler with a cosine decay strategy, incorporating a warmup period of 10 epochs. For the ImageNet one, we reduced the learning rate by a factor of 10 at epochs 11 and 13 each time, as it was advantageous in performance. This is similar to the strategy employed in the original ResNet paper [21]. This choice was provoked by our observations that the accuracy of the model started to plateau around the 10th epoch and these "shocks" helped prevent significant overfitting and resulted in an additional increase of approximately 11% in accuracy. Following these adjustments, we concluded our training process at the 15th epoch.

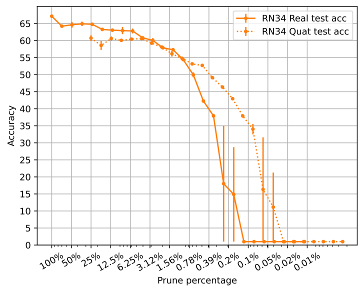
3.4.2 Pruning

Our method of pruning can be briefly summarized as 'rest-train-prune' loops. After making the model instance, we initialize the weights with uniform Xavier initialization. Then, we train the full model as mentioned in Section 3.4.1. Then, we use L1 unstructured pruning to prune the weights of the linear and the convolutional layers in the model. We didn't prune the biases or the BN layers. This is because they all combined constituted less than 0.1% of the weights. Pruning is done by the creation and application of a mask matrix as the same size as the weight matrix, but kept separate.

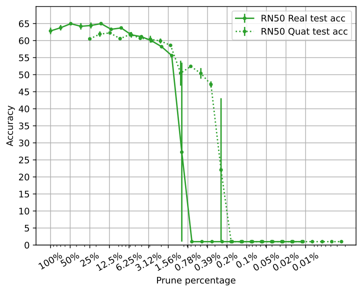
In the next iteration, we reinitialize the weights



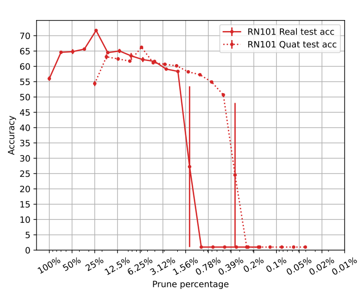
(a) ResNet18 on CIFAR100



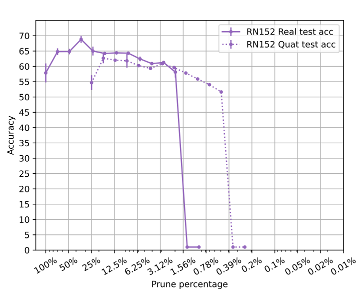
(b) ResNet34 on CIFAR100



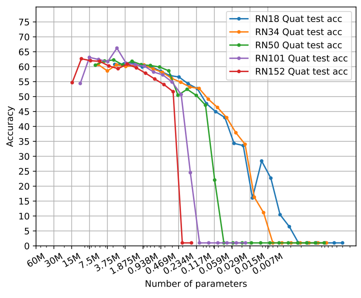
(c) ResNet50 on CIFAR100



(d) ResNet101 on CIFAR100

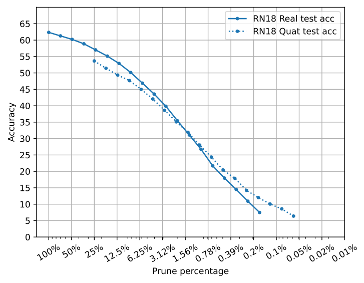


(e) ResNet152 on CIFAR100

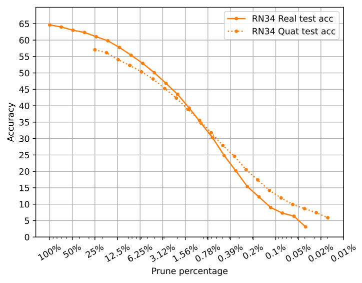


(f) All Quat models on CIFAR100

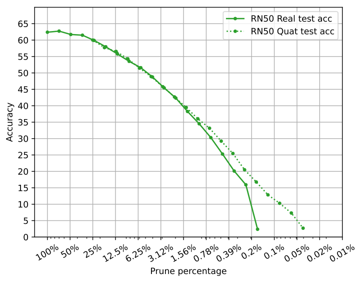
Figure 2. Validation (top 1) accuracy vs pruning results of all ResNet models on CIFAR100 dataset.



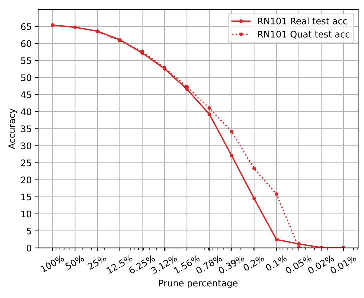
(a) ResNet18 on ImageNet64



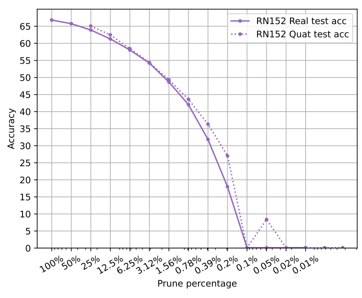
(b) ResNet34 on ImageNet64



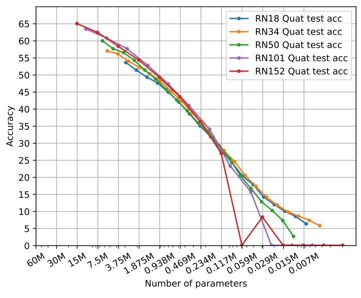
(c) ResNet50 on ImageNet64



(d) ResNet101 on ImageNet64



(e) ResNet152 on ImageNet64



(f) All Quat models on ImageNet64

Figure 3. Validation (top 1) accuracy vs pruning results of all ResNet models on ImageNet64 dataset.

Table 3. Complete pruning results of experiments with pruning proportions mentioned in Table 1 per iteration i. The percentage of weights left can be found as $N \times (\text{pruning proportion})^i$ where N = 100 for Real and 25 for Quat (Quat models start with 25% weights). This table contains the detailed values depicted in Figures 2 and 3.

pruning	CIFAR100 on Real					CIFAR100 on Quat					ImageNet64 on Real			ImageNet64 on Quat			ImageNet64 on Real		ImageNet64 on Quat	
iteration	RN18	RN34	RN50	RN101	RN152	RN18	RN34	RN50	RN101	RN152	RN18	RN34	RN50	RN18	RN34	RN50	RN101	RN152	RN101	RN152
0	64.42	67.16	62.84	56	57.86	60.77	60.74	60.51	54.39	54.67	62.35	64.63	62.4	53.64	57.06	60.01	65.42	66.82	63.48	65.06
1	63.4	64.24	63.78	64.6	64.81	60.93	58.58	61.92	63.12	62.67	61.28	63.99	62.74	51.42	56.16	57.71	64.76	65.76	60.86	62.48
2	64.75	64.65	64.97	64.81	64.83	61.22	60.61	62.26	62.42	62.01	60.21	63.01	61.71	49.34	54.04	56.54	63.63	63.9	57.7	58.44
3	63.56	64.94	64.2	65.64	68.82	59.9	60.08	60.58	61.72	61.8	58.87	62.33	61.5	47.66	52.3	54.3	61.1	61.29	52.82	54.38
4	63.08	64.8	64.44	71.72	65	60	60.44	61.82	66.22	60.24	57.02	61.03	59.91	44.99	50.35	51.46	57.23	58.05	47.35	49.3
5	63.21	63.28	64.99	64.48	64.21	58.49	60.58	60.69	61.18	59.34	55.12	59.8	57.97	42.06	48.16	48.89	52.6	54.21	41.05	43.61
6	61.24	63.07	63.32	65.03	64.4	57.06	59.26	60.41	60.68	60.88	52.89	57.76	55.77	38.61	45.26	45.8	46.58	48.64	34.11	36.31
7	61.82	62.94	63.72	63.43	64.31	56.54	58.1	59.9	60.18	59.62	50.12	55.4	53.53	35.09	42.35	42.68	39.26	42.04	23.34	27.04
8	61.47	62.82	61.7	62.22	62.43	54.28	56.14	58.61	58.22	57.82	46.87	52.91	51.59	31.91	38.88	39.49	27.08	31.84	15.81	0.1
9	59.04	60.81	61.16	61.61	60.9	52.66	54.8	50.44	57.26	55.88	43.56	50.05	48.77	28.01	35.55	36.03	14.51	17.99	0.1	8.37
10	57.76	60.06	59.96	59.11	61.26	47.6	53.17	52.44	54.89	54.02	39.85	46.82	45.44	24.35	31.79	33.15	2.48	0.1	-	-
11	55.52	57.89	58.22	58.36	58.12	44.93	52.72	50.38	50.7	51.64	35.38	43.46	42.34	20.42	27.88	29.22	1.18	0.1	-	-
12	49.8	57.32	55.62	27.24	1	42.97	49.13	47.13	24.53	1	31.09	39.28	38.26	17.88	24.6	25.46	-	-	-	-
13	48.77	54.43	27.27	1	1	34.33	46.36	22.05	1	1	26.73	34.74	34.51	14.22	20.53	20.49	-	-	-	-
14	41.92	49.96	1	1	-	33.6	42.96	1	1	-	21.68	30.27	30.29	12.02	17.39	16.77	-	-	-	-
15	40.94	42.28	1	1	-	16.04	37.89	1	1	-	17.96	24.85	25.28	10.05	14.21	12.85	-	-	-	-
16	31.63	37.9	1	1	-	28.42	34.02	1	1	-	14.5	20.17	20.1	8.58	11.91	10.28	-	-	-	-
17	31.37	18.03	1	1	-	22.69	16.31	1	1	-	10.92	15.33	15.92	6.38	9.92	7.31	-	-	-	-
18	12.46	14.86	1	1	-	22.69	16.31	1	1	-	-	-	-	-	-	-	-	-	-	-
19	9.07	1	1	-	-	10.44	11.14	1	-	-	-	-	-	-	-	-	-	-	-	-
20	1	1	1	-	-	6.44	1	1	-	-	-	-	-	-	-	-	-	-	-	-

with uniform Xavier initialization using the same seed and train the model with the mask matrix applied over the weight matrix. The pruning applied in this iteration is also added to the mask matrix. This process of 'reset-train-prune' is repeated till the model accuracy reduces to 20% and a few more iteration to observe any further significant changes in the curve.

4 Results

4.1 CIFAR100

When applying ResNet to the CIFAR100 dataset, we obtained similar results to those reported by Sahel et al. [29] when they used Conv-6 models on the same dataset. However, due to ResNet's larger size compared to Conv-6 models, we had to perform more aggressive pruning to achieve a similar level of accuracy degradation. The results obtained are given in Figure 2, and detailed values on our pruning results are compiled in Table 3. Our Quat models performed almost as well as our original unpruned Real counterparts (slightly 2-3% less). Upon analyzing the graphs in Figures 2(a) through 2(e), we observed that the difference between the highest accuracy obtained by the Real and Quat models decreased as we moved to higher-capacity models. For the comparison of the original model's performance both on CIFAR100 and ImageNet64, we used the works by Shahadat and Maida [32] as a reference point.

As seen in Figure 2(d),all the Quat models pruned to about 6.25% sparsity have similar or significantly higher accuracies than the full model, i.e., they contain Lottery Tickets. For example, Quat RN101 had an accuracy of 66.22%, whereas its unpruned version had 54.4%, a rise of 11.8% in accuracy. Similar lottery tickets can be seen in RN50, and RN152, with an accuracy bump of 1.3%, and 5.6% respectively, at the same sparsity. This confirms the existence of lottery tickets even in large quaternion networks.

On further pruning, we found that at 1% sparsity, RN101 Real failed to train and gave 1% accuracy, but Quat counterpart gave 57.26%. We can also see similar trends in the other models where in Real RN50 and Real RN152, the accuracy started dropping steeply after 1.4% and 2% sparsities, their Quat counterparts retained accuracies till 0.3% and 0.5% sparsities respectively. This empirically validates that the quaternion consistently performs better than the real at high sparsities.

4.2 ImageNet64

In this particular phase of the experiment, we observed distinct differences in the graphs compared to the CIFAR100 dataset. From the very first pruning iteration, the ImageNet networks exhibited an immediate and consistent decline in accuracy, as shown in Figures 3(a) through 3(e). Further discussion on this is given in the next section. The graphs produced with the Real and Quat models are given in Figure 3. Specifically, in models such as RN18 and RN34, the Quat model's accuracy was consistently lower than the Real model's at lower sparsity levels.

However, as we progressed to higher-capacity models such as RN50 and RN101, the Quat graph closely overlapped with the Real graph at higher levels of accuracy, especially at RN152; the quaternion model consistently outperformed the real model at all sparsity levels, indicating a clear advantage.

Unfortunately, we didn't find any lottery tickets for this due to the aforementioned consistent performance decline from the initial pruning. On pruning, we found that at 0.39% sparsity, RN101 Real gave 27.08% accuracy, and Quat gave 34.11% accuracy, which is 7% higher. We can also see similar trends in the other models where in RN18, we have a gain of about 3.4%, RN34 a gain of 4.4%, RN50 a gain of 5.4%, and so on.

If we compare the unpruned Real and Quat models, we can see that in RN18 and RN34, the Real outperforms Quat with a very high accuracy difference (8.71% and 7.57%, respectively). But, the performance gap decreased in the rest of RN50, RN101, and RN152 (the difference is 2.04%, 1.94%, and 1.76%, respectively), and thus, Quat comes close to Real in terms of performance in higher complexities.

The accuracy obtained at different pruning iterations is given in detail in Table 3.

5 Discussions

The observation that quaternion models, despite having one-fourth of the parameters compared to real models, achieved comparable accuracies challenges the conventional understanding of model complexity and parameter count. It suggests that the real models in this study were over-parameterized, exceeding the necessary capacity for learning the dataset.

CIFAR100 Real vs. Quat: The CIFAR100 results indicate that in the case of RN18 real and RN101 quaternion models, their parameter counts are nearly equal, and the corresponding graphs exhibit significant overlap. This observation suggests that the claim of Quaternion models outperforming Real models holds true primarily when the dataset size is relatively small compared to the model's capacity.

ImageNet64 Real vs. Quat: Figure 3(a) and 3(b) provide clear evidence that quaternion models, despite having a sufficient number of parameters, do not yield satisfactory accuracy. This discrepancy might be attributed to the phenomenon where larger models tend to perform well across a wide range of hyperparameters, whereas smaller models excel within a narrow set of values, as demonstrated by Taylor et al. [33]. It is possible that the hyperparameters used in our experiments were more suitable for real models rather than quaternion models. However, to ensure a fair comparison, we did not conduct specific hyperparameter tuning experiments for quaternion models.

Why do the accuracies for ImageNet dataset decline from the beginning?: This observation, made in Figures 3(a) through 3(e), leads us to believe that the models were inadequately equipped with parameters, failing to achieve an optimal level of over-parameterization necessary for effectively comprehending and mastering the intricacies of the ImageNet dataset containing a rich source of information. With more parameters (in RN101 and RN152), we can see that the accuracies are almost same in the first one or two iterations, indicating a possibility of Lottery Ticket under the right hyperparameter tuning.

6 Conclusion

Our study sought to determine the suitability of pruned quaternion networks over pruned real models in resource-constrained settings. Through the successful training of pruned quaternion networks to achieve accuracy levels comparable to the original models, We validated the Lottery Ticket Hypothesis in large-scale quaternion networks, specifically exploring five ResNet model depth variations. While our findings are significant, further research is essential to generalize across a broader range of large-scale models. Our experiments involving pruning large network models across both small and large datasets further confirmed that pruned quaternion models consistently outperform their real counterparts under conditions of high sparsity.

7 Limitations and Future Work

In our study, we acknowledge several limitations and propose potential areas for future exploration. First, comprehensive hyperparameter optimization tailored to quaternion models is needed to enhance their performance and potentially surpass real models in accuracy. Additionally, while experiments were conducted on ImageNet, further research with larger datasets like the original ILSVRC is essential to unlock quaternion models' full potential, especially in sparse conditions. Furthermore, extending quaternion models beyond image classification to tasks like object detection, semantic segmentation, and natural language processing is valuable. Lastly, exploring quaternion models' performance in various neural network architectures beyond ResNet, such as Transformers, GNNs, and RNNs, could reveal novel applications and advantages, offering valuable insights into their broader applicability.

Acknowledgments

This work is partially supported by RIN4001-Basic Research in Computer Sciences, NISER.

References

[1] T. Nitta. "A quaternary version of the backpropagation algorithm". In: *Proceedings of ICNN'95 - International Conference on Neural Networks* 5 (1995), 2753–2756 vol.5. DOI: 10.1109/icnn.1995.488166. URL: https://doi.org/10.1109%2Ficnn.1995.488166.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[3] J. Li, Y. Wang, C. Wang, Y. Tai, J. Qian, J. Yang, C. Wang, J. Li, and F. Huang. “DSFD: Dual Shot Face Detector”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019, pp. 5060–5069. DOI: 10.1109/cvpr.2019.00520. URL: https://doi.org/10.1109%2Fcvpr.2019.00520.

[4] Y. LeCun, J. S. Denker, and S. A. Solla. “Optimal Brain Damage”. In: *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*. Ed. by D. S. Touretzky. Morgan Kaufmann, 1989, pp. 598–605. URL: http://papers.nips.cc/paper/250-optimal-brain-damage.

[5] S. Han, J. Pool, J. Tran, and W. Dally. “Learning both Weights and Connections for Efficient Neural Network”. In: *Advances in Neural Information Processing Systems (Montreal, Canada)*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015, pp. 1135–1143. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf.

[6] J. Frankle and M. Carbin. “The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks”. In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=rJl-b3RcF7.

[7] M. Zhu and S. Gupta. *To prune, or not to prune: exploring the efficacy of pruning for model compression*. 2017. arXiv: 1710.01878 [stat.ML].

[8] N. Lee, T. Ajanthan, S. Gould, and P. H. S. Torr. “A Signal Propagation Perspective for Pruning Neural Networks at Initialization”. In: *CoRR* abs/1906.06307 (2019). arXiv: 1906.06307. URL: http://arxiv.org/abs/1906.06307.

[9] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag. “What is the State of Neural Network Pruning?” In: *Proceedings of Machine Learning and Systems*. Ed. by I. Dhillon, D. Papailiopoulos, and V. Sze. Vol. 2. 2020, pp. 129–146. URL: https://proceedings.mlsys.org/paper_files/paper/2020/file/6c44dc73014d66ba49b28d483a8f8b0d-Paper.pdf.

[10] T. Parcollet, M. Morchid, and G. Linarès. “A survey of quaternion neural networks”. In: *Artificial Intelligence Review* 53.6 (2020), pp. 2957–2982. DOI: 10.1007/s10462-019-09752-1. URL: https://doi.org/10.1007/s10462-019-09752-1.

[11] C. J. Gaudet and A. S. Maida. “Deep Quaternion Networks”. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489651.

[12] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal. “Deep complex networks”. In: *International Conference on Learning Representations*. 2018, pp. 1–19. URL: https://arxiv.org/abs/1705.09792.

[13] T. Tran, Y. Hu, C. Hu, K. Yen, F. Tan, K. Lee, and S. R. Park. “HABERTOR: An Efficient and Effective Deep Hatespeech Detector”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020. DOI: 10.18653/v1/2020.emnlp-main.606. URL: https://doi.org/10.18653%2Fv1%2F2020.emnlp-main.606.

[14] T. Parcollet, M. Ravanelli, M. Morchid, G. Linarès, and R. D. Mori. *Speech recognition with quaternion neural networks*. 2018. arXiv: 1811.09678 [eess.AS].

[15] D. Comminiello, M. Lella, S. Scardapane, and A. Uncini. “Quaternion Convolutional Neural Networks for Detection and Localization of 3D Sound Events”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 8533–8537. DOI: 10.1109/ICASSP.2019.8682711.

[16] T. Parcollet, M. Ravanelli, M. Morchid, G. Linarès, C. Trabelsi, R. D. Mori, and Y. Bengio. *Quaternion Recurrent Neural Networks*. 2019. arXiv: 1806.04418 [stat.ML].

[17] D. Pavllo, C. Feichtenhofer, M. Auli, and D. Grangier. “Modeling Human Motion with Quaternion-Based Neural Networks”. In: *International Journal of Computer Vision* 128.4

(Oct. 2019), pp. 855–872. DOI: 10.1007/s11263-019-01245-6. URL: https://doi.org/10.1007%2Fs11263-019-01245-6.

[18] B. Hassibi and D. Stork. "Second order derivatives for network pruning: Optimal Brain Surgeon". In: *Advances in Neural Information Processing Systems*. Ed. by S. Hanson, J. Cowan, and C. Giles. Vol. 5. Morgan-Kaufmann, 1992, pp. 164–171. URL: https://proceedings.neurips.cc/paper_files/paper/1992/file/303ed4c69846ab36c2904d3ba8573050-Paper.pdf.

[19] R. Reed. "Pruning algorithms - a survey". In: *IEEE Transactions on Neural Networks* 4.5 (1993), pp. 740–747. DOI: 10.1109/72.248452. URL: https://doi.org/10.1109%2F72.248452.

[20] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[21] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 770–778.

[22] E. Elsen, M. Dukhan, T. Gale, and K. Simonyan. "Fast Sparse ConvNets". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. DOI: 10.1109/cvpr42600.2020.01464. URL: https://doi.org/10.1109%2Fcvpr42600.2020.01464.

[23] J. Pool, A. Sawarkar, and J. Rodge. *Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT*. Technical Walkthrough. Available at: https://developer.nvidia.com/blog/accelerating-inference-with-sparsityusing-ampere-and-tensorrt/. 2021.

[24] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu. "Efficient Neural Audio Synthesis". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 2410–2419. URL: https://proceedings.mlr.press/v80/kalchbrenner18a.html.

[25] X. Zhu, Y. Xu, H. Xu, and C. Chen. "Quaternion Convolutional Neural Networks". In: *Lecture Notes in Computer Science*. Vol. 11212. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2018, pp. 645–661. DOI: 10.1007/978-3-030-01237-3_39. eprint: 1903.00658. URL: https://doi.org/10.1007/978-3-030-01237-3_39.

[26] A. Krizhevsky and G. Hinton. "Learning multiple layers of features from tiny images". In: (2009). URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[27] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. "Downsampled ImageNet 64x64". In: (). URL: http://image-net.org/small/download.php.

[28] 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". In: *International Journal of Computer Vision* 115.3 (Apr. 2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y. URL: https://doi.org/10.1007%2Fs11263-015-0816-y.

[29] S. M. Iqbal and S. Mishra. "Neural Networks at a Fraction with Pruned Quaternions". In: *Proceedings of the 6th Joint International Conference on Data Science Management of Data (10th ACM IKDD CODS and 28th COMAD)*. CODS-COMAD '23. Mumbai, India: Association for Computing Machinery, 2023, pp. 19–27. ISBN: 9781450397971. DOI: 10.1145/3570991.3570997. URL: https://doi.org/10.1145/3570991.3570997.

[30] K. Liu, W. Yang, Y. Peiwen, and F. Ducau. *Train CIFAR10 with PyTorch*. 2021. URL: https://github.com/kuangliu/pytorch-cifar.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019, pp. 8026–8037.

[32] N. Shahadat and A. S. Maida. "Enhancing ResNet image classification performance by using parameterized hypercomplex multiplication". In: (Jan. 2023). arXiv: 2301.04623 [cs.CV].

[33] R. Taylor, V. Ojha, I. Martino, and G. Nicosia. "Sensitivity Analysis for Deep Learning: Ranking Hyper-parameter Influence". In: *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. 2021, pp. 512–516. DOI: 10.1109/ICTAI52525.2021.00083.