Lightweight and Efficient Neural Natural Language Processing with Quaternion Networks

[1]Yi Tay, [2]Aston Zhang, [3]Luu Anh Tuan, [4]Jinfeng Rao,* [5]Shuai Zhang
[6]Shuohang Wang, [7]Jie Fu, [8]Siu Cheung Hui
[1,8]Nanyang Technological University, [2]Amazon AI, [3]MIT CSAIL
[4]Facebook AI, [5]UNSW, [6]Singapore Management University,
[7]Mila and Polytechnique Montréal

Abstract

Many state-of-the-art neural models for NLP are heavily parameterized and thus memory inefficient. This paper proposes a series of lightweight and memory efficient neural architectures for a potpourri of natural language processing (NLP) tasks. To this end, our models exploit computation using Quaternion algebra and hypercomplex spaces, enabling not only expressive inter-component interactions but also significantly (75%) reduced parameter size due to lesser degrees of freedom in the Hamilton product. We propose Quaternion variants of models, giving rise to new architectures such as the Quaternion attention Model and Quaternion Transformer. Extensive experiments on a battery of NLP tasks demonstrates the utility of proposed Quaternion-inspired models, enabling up to 75% reduction in parameter size without significant loss in performance.

1 Introduction

Neural network architectures such as Transformers (Vaswani et al., 2017; Dehghani et al., 2018) and attention networks (Parikh et al., 2016; Seo et al., 2016; Bahdanau et al., 2014) are dominant solutions in natural language processing (NLP) research today. Many of these architectures are primarily concerned with learning useful feature representations from data in which providing a strong architectural inductive bias is known to be extremely helpful for obtaining stellar results.

Unfortunately, many of these models are known to be heavily parameterized, with state-of-the-art models easily containing millions or billions of parameters (Vaswani et al., 2017; Radford et al., 2018; Devlin et al., 2018; Radford et al., 2019). This renders practical deployment challenging. As such, the enabling of efficient and lightweight adaptations of these models, without significantly degrading performance, would certainly have a positive impact on many real world applications.

To this end, this paper explores a new way to improve/maintain the performance of these neural architectures while substantially reducing the parameter cost (compression of up to 75%). In order to achieve this, we move beyond real space, exploring computation in Quaternion space (*i.e.*, hypercomplex numbers) as an inductive bias. Hypercomplex numbers comprise of a real and three imaginary components (*e.g.*, i, j, k) in which interdependencies between these components are encoded naturally during training via the Hamilton product $\otimes$. Hamilton products have fewer degrees of freedom, enabling up to four times compression of model size. Technical details are deferred to subsequent sections.

While Quaternion connectionist architectures have been considered in various deep learning application areas such as speech recognition (Parcollet et al., 2018b), kinematics/human motion (Pavllo et al., 2018) and computer vision (Gaudet and Maida, 2017), our work is the first hypercomplex inductive bias designed for a wide spread of NLP tasks. Other fields have motivated the usage of Quaternions primarily due to their natural 3 or 4 dimensional input features (*e.g.*, RGB scenes or 3D human poses) (Parcollet et al., 2018b; Pavllo et al., 2018). In a similar vein, we can similarly motivate this by considering the multi-sense nature of natural language (Li and Jurafsky, 2015; Neelakantan et al., 2015; Huang et al., 2012). In this case, having multiple embeddings or components per token is well-aligned with this motivation.

Latent interactions between components may also enjoy additional benefits, especially pertaining to applications which require learning pairwise affinity scores (Parikh et al., 2016; Seo

* Work done while at University of Maryland.

et al., 2016). Intuitively, instead of regular (real) dot products, Hamilton products $\otimes$ extensively learn representations by matching across multiple (inter-latent) components in hypercomplex space. Alternatively, the effectiveness of multi-view and multi-headed (Vaswani et al., 2017) approaches may also explain the suitability of Quaternion spaces in NLP models. The added advantage to multi-headed approaches is that Quaternion spaces explicitly encodes latent interactions between these components or heads via the Hamilton product which intuitively increases the expressiveness of the model. Conversely, multi-headed embeddings are generally independently produced.

To this end, we propose two Quaternion-inspired neural architectures, namely, the Quaternion attention model and the Quaternion Transformer. In this paper, we devise and formulate a new attention (and self-attention) mechanism in Quaternion space using Hamilton products. Transformation layers are aptly replaced with Quaternion feed-forward networks, yielding substantial improvements in parameter size (of up to 75% compression) while achieving comparable (and occasionally better) performance.

Contributions All in all, we make the following major contributions:

- We propose Quaternion neural models for NLP. More concretely, we propose a novel Quaternion attention model and Quaternion Transformer for a wide range of NLP tasks. To the best of our knowledge, this is the first formulation of hypercomplex Attention and Quaternion models for NLP.

- We evaluate our Quaternion NLP models on a wide range of diverse NLP tasks such as pairwise text classification (natural language inference, question answering, paraphrase identification, dialogue prediction), neural machine translation (NMT), sentiment analysis, mathematical language understanding (MLU), and subject-verb agreement (SVA).

- Our experimental results show that Quaternion models achieve comparable or better performance to their real-valued counterparts with up to a 75% reduction in parameter costs. The key advantage is that these models are expressive (due to Hamiltons) and also parameter efficient. Moreover, our Quaternion components are self-contained and play well with real-valued counterparts.

2 Background on Quaternion Algebra

This section introduces the necessary background for this paper. We introduce Quaternion algebra along with Hamilton products, which form the crux of our proposed approaches.

Quaternion A Quaternion $Q \in \mathbb{H}$ is a hypercomplex number with three imaginary components as follows:

$$Q = r + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}, \tag{1}$$

where $\mathbf{ijk} = \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$ and noncommutative multiplication rules apply: $\mathbf{ij} = \mathbf{k}, \mathbf{jk} = \mathbf{i}, \mathbf{ki} = \mathbf{j}, \mathbf{ji} = -\mathbf{k}, \mathbf{kj} = -\mathbf{i}, \mathbf{ik} = -\mathbf{j}$. In (1), r is the real value and similarly, x, y, z are real numbers that represent the imaginary components of the Quaternion vector Q. Operations on Quaternions are defined in the following.

Addition The addition of two Quaternions is defined as:

$$\begin{aligned} Q + P = Q_r + P_r + (Q_x + P_x)\mathbf{i} \\ +(Q_y + P_y)\mathbf{j} + (Q_z + P_z)\mathbf{k}, \end{aligned}$$

where Q and P with subscripts denote the real value and imaginary components of Quaternion Q and P. Subtraction follows this same principle analogously but flipping $+$ with $-$.

Scalar Multiplication Scalar α multiplies across all components, *i.e.*,

$$\alpha Q = \alpha r + \alpha x\mathbf{i} + \alpha y\mathbf{j} + \alpha z\mathbf{k}.$$

Conjugate The conjugate of Q is defined as:

$$Q^* = r - x\mathbf{i} - y\mathbf{j} - z\mathbf{k}.$$

Norm The unit Quaternion $Q^\triangleleft$ is defined as:

$$Q^\triangleleft = \frac{Q}{\sqrt{r^2 + x^2 + y^2 + z^2}}.$$

Hamilton Product The Hamilton product, which represents the multiplication of two Quaternions Q and P, is defined as:

$$\begin{aligned} Q \otimes P = (Q_r P_r - Q_x P_x - Q_y P_y - Q_z P_z) \\ + (Q_x P_r + Q_r P_x - Q_z P_y + Q_y P_z)\,\mathbf{i} \\ + (Q_y P_r + Q_z P_x + Q_r P_y - Q_x P_z)\,\mathbf{j} \\ + (Q_z P_r - Q_y P_x + Q_x P_y + Q_r P_z)\,\mathbf{k}, \end{aligned} \tag{2}$$

which intuitively encourages inter-latent interaction between all the four components of Q and P. In this work, we use Hamilton products extensively for vector and matrix transformations that live at the heart of attention models for NLP.

3 Quaternion Models of Language

In this section, we propose Quaternion neural models for language processing tasks. We begin by introducing the building blocks, such as Quaternion feed-forward, Quaternion attention, and Quaternion Transformers.

3.1 Quaternion Feed-Forward

A Quaternion feed-forward layer is similar to a feed-forward layer in real space, while the former operates in hypercomplex space where Hamilton product is used. Denote by $W \in \mathbb{H}$ the weight parameter of a Quaternion feed-forward layer and let $Q \in \mathbb{H}$ be the layer input. The linear output of the layer is the Hamilton product of two Quaternions: $W \otimes Q$.

Saving Parameters? How and Why In lieu of the fact that it might not be completely obvious at first glance why Quaternion models result in models with smaller parameterization, we dedicate the following to address this.

For the sake of parameterization comparison, let us express the Hamilton product $W \otimes Q$ in a Quaternion feed-forward layer in the form of matrix multiplication, which is used in real-space feed-forward. Recall the definition of Hamilton product in (2). Putting aside the Quaterion unit basis $[1, \mathbf{i}, \mathbf{j}, \mathbf{k}]^\top$, $W \otimes Q$ can be expressed as:

$$\begin{bmatrix} W_r & -W_x & -W_y & -W_z \\ W_x & W_r & -W_z & W_y \\ W_y & W_z & W_r & -W_x \\ W_z & -W_y & W_x & W_r \end{bmatrix} \begin{bmatrix} r \\ x \\ y \\ z \end{bmatrix}, \quad (3)$$

where $W = W_r + W_x\mathbf{i} + W_y\mathbf{j} + W_z\mathbf{k}$ and Q is defined in (1).

We highlight that, there are only 4 distinct parameter variable elements (4 degrees of freedom), namely W_r, W_x, W_y, W_z, in the weight matrix (left) of (3), as illustrated by Figure 1; while in real-space feed-forward, all the elements of the weight matrix are different parameter variables ($4 \times 4 = 16$ degrees of freedom). In other words, the degrees of freedom in Quaternion feed-forward is only a quarter of those in its real-space counterpart, resulting in a 75% reduction in parameterization. Such a parameterization reduction can also be explained by *weight sharing* (Parcollet et al., 2018b,a).

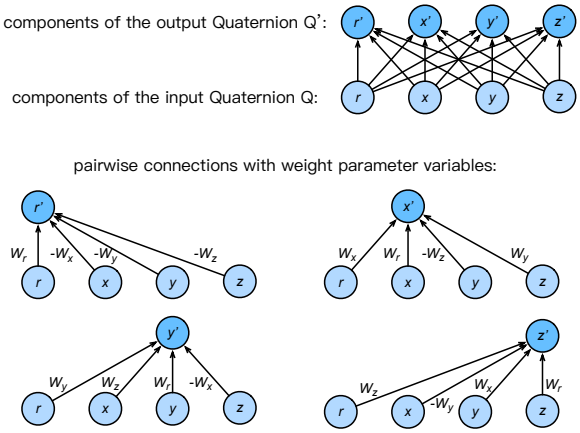

Figure 1: 4 weight parameter variables (W_r, W_x, W_y, W_z) are used in 16 pairwise connections between components of the input and output Quaternions.

Nonlinearity Nonlinearity can be added to a Quaternion feed-forward layer and component-wise activation is adopted (Parcollet et al., 2018a):

$$\alpha(Q) = \alpha(r) + \alpha(x)\mathbf{i} + \alpha(y)\mathbf{j} + +\alpha(z)\mathbf{k},$$

where Q is defined in (1) and $\alpha(.)$ is a nonlinear function such as tanh or ReLU.

3.2 Quaternion Attention

Next, we propose a Quaternion attention model to compute attention and alignment between two sequences. Let $A \in \mathbb{H}^{\ell_a \times d}$ and $B \in \mathbb{H}^{\ell_b \times d}$ be input word sequences, where ℓ_a, ℓ_b are numbers of tokens in each sequence and d is the dimension of each input vector. We first compute:

$$E = A \otimes B^\top,$$

where $E \in \mathbb{H}^{\ell_a \times \ell_b}$. We apply Softmax(.) to E component-wise:

$$G = \text{ComponentSoftmax}(E)$$
$$B' = G_R B_R + G_X B_X \mathbf{i} + G_Y B_Y \mathbf{j} + G_Z B_Z \mathbf{k},$$

where G and B with subscripts represent the real and imaginary components of G and B. Similarly, we perform the same on A which is described as follows:

$$F = \text{ComponentSoftmax}(E^\top)$$
$$A' = F_R A_R + F_X A_X \mathbf{i} + F_Y A_Y \mathbf{j} + F_Z A_Z \mathbf{k},$$

where A' is the aligned representation of B and B' is the aligned representation of A. Next, given $A' \in \mathbb{R}^{\ell_b \times d}, B' \in \mathbb{R}^{\ell_A \times d}$ we then compute and **compare** the learned alignments:

$$C_1 = \sum \text{QFFN}([A'_i; B_i, A'_i \otimes B_i; A'_i - B_i])$$
$$C_2 = \sum \text{QFFN}([B'_i; A_i, B'_i \otimes A_i; B'_i - A_i]),$$

where QFFN(.) is a Quaternion feed-forward layer with nonlinearity and $[;]$ is the component-wise contatentation operator. i refers to word positional indices and $\sum$ over words in the sequence. Both outputs C_1, C_2 are then passed

$$Y = \text{QFFN}([C_1; C_2; C_1 \otimes C_2; C_1 - C_2]),$$

where $Y \in \mathbb{H}$ is a Quaternion valued output. In order to train our model end-to-end with real-valued losses, we concatenate each component and pass into a final linear layer for classification.

3.3 Quaternion Transformer

This section describes our Quaternion adaptation of Transformer networks. Transformer (Vaswani et al., 2017) can be considered state-of-the-art across many NLP tasks. Transformer networks are characterized by stacked layers of linear transforms along with its signature self-attention mechanism. For the sake of brevity, we outline the specific changes we make to the Transformer model.

Quaternion Self-Attention The standard self-attention mechanism considers the following:

$$A = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V,$$

where Q, K, V are traditionally learned via linear transforms from the input X. The key idea here is that we replace this linear transform with a Quaternion transform.

$$Q = W_q \otimes X; K = W_k \otimes X; V = W_v \otimes X,$$

where $\otimes$ is the Hamilton product and X is the input Quaternion representation of the layer. In this case, since computation is performed in Quaternion space, the parameters of W is effectively reduced by 75%. Similarly, the computation of self-attention also relies on Hamilton products. The revised Quaternion self-attention is defined as follows:

$$A = \text{ComponentSoftmax}(\frac{Q \otimes K}{\sqrt{d_k}})V. \quad (4)$$

Note that in (4), $Q \otimes K$ returns four $\ell \times \ell$ matrices (attention weights) for each component (r, i, j, k). Softmax is applied component-wise, along with multiplication with V which is multiplied in similar fashion to the Quaternion attention model. Note that the Hamilton product in the self-attention itself does not change the parameter size of the network.

Quaternion Transformer Block Aside from the linear transformations for forming query, key, and values. Tranformers also contain position feed-forward networks with ReLU activations. Similarly, we replace the feed-forward connections (FFNs) with Quaternion FFNs. We denote this as Quaternion Transformer (*full*) while denoting the model that only uses Quaternion FFNs in the self-attention as (*partial*). Finally, the remainder of the Transformer networks remain identical to the original design (Vaswani et al., 2017) in the sense that component-wise functions are applied unless specified above.

3.4 Embedding Layers

In the case where the word embedding layer is trained from scratch (*i.e.*, using Byte-pair encoding in machine translation), we treat each embedding to be the concatenation of its four components. In the case where pre-trained embeddings such as GloVe (Pennington et al., 2014) are used, a nonlinear transform is used to project the embeddings into Quaternion space.

3.5 Connection to Real Components

A vast majority of neural components in the deep learning arsenal operate in real space. As such, it would be beneficial for our Quaternion-inspired components to interface seamlessly with these components. If input to a Quaternion module (such as Quaternion FFN or attention modules), we simply treat the real-valued input as a concatenation of components r, x, y, z. Similarly, the output of the Quaternion module, if passed to a real-valued layer, is treated as a $[r; x; y; z]$, where $[;]$ is the concatenation operator.

Output layer and Loss Functions To train our model, we simply concatenate all r, i, j, k components into a single vector at the final output layer. For example, for classification, the final Softmax output is defined as following:

$$Y = \text{Softmax}(W([r; x; y; z]) + b),$$

where $Y \in \mathbb{R}^{|C|}$ where $|C|$ is the number of classes and x, y, z are the imaginary components. Similarly for sequence loss (for sequence transduction problems), the same can be also done.

Parameter Initialization It is intuitive that specialized initialization schemes ought to be devised for Quaternion representations and their modules (Parcollet et al., 2018b,a).

$$w = |w|(\cos(\theta) + q^{\triangleleft}_{imag} \sin(\theta),$$

where $q^{\triangleleft}_{imag}$ is the normalized imaginary constructed from uniform randomly sampling from $[0, 1]$. θ is randomly and uniformly sampled from $[-\pi, \pi]$. However, our early experiments show that, at least within the context of NLP applications, this initialization performed comparable or worse than the standard Glorot initialization. Hence, we opt to initialize all components independently with Glorot initialization.

4 Experiments

This section describes our experimental setup across multiple diverse NLP tasks. All experiments were run on NVIDIA Titan X hardware.

Our Models On pairwise text classification, we benchmark Quaternion attention model (Q-Att), testing the ability of Quaternion models on pairwise representation learning. On all the other tasks, such as machine translation and subject-verb agreement, we evaluate Quaternion Transformers. We evaluate two variations of Transformers, *full* and *partial*. The *full* setting converts all linear transformations into Quaternion space and is approximately 25% of the actual Transformer size. The second setting (*partial*) only reduces the linear transforms at the self-attention mechanism. Tensor2Tensor[1] is used for Transformer benchmarks, which uses its default Hyperparameters and encoding for all experiments.

4.1 Pairwise Text Classification

We evaluate our proposed Quaternion attention (Q-Att) model on pairwise text classification tasks. This task involves predicting a label or ranking score for sentence pairs. We use a total of seven data sets from problem domains such as:

- **Natural language inference (NLI)** - This task is concerned with determining if two sentences entail or contradict each other. We use SNLI (Bowman et al., 2015), SciTail (Khot et al., 2018), MNLI (Williams et al., 2017) as benchmark data sets.

- **Question answering (QA)** - This task involves learning to rank question-answer pairs. We use WikiQA (Yang et al., 2015) which comprises of QA pairs from Bing Search.

- **Paraphrase detection** - This task involves detecting if two sentences are paraphrases of each other. We use Tweets (Lan et al., 2017) data set and the Quora paraphrase data set (Wang et al., 2017).

- **Dialogue response selection** - This is a response selection (RS) task that tries to select the best response given a message. We use the Ubuntu dialogue corpus, UDC (Lowe et al., 2015).

Implementation Details We implement Q-Att in TensorFlow (Abadi et al., 2016), along with the Decomposable Attention baseline (Parikh et al., 2016). Both models optimize the cross entropy loss (*e.g.*, binary cross entropy for ranking tasks such as WikiQA and Ubuntu). Models are optimized with Adam with the learning rate tuned amongst $\{0.001, 0.0003\}$ and the batch size tuned amongst $\{32, 64\}$. Embeddings are initialized with GloVe (Pennington et al., 2014). For Q-Att, we use an additional transform layer to project the pre-trained embeddings into Quaternion space. The measures used are generally the accuracy measure (for NLI and Paraphrase tasks) and ranking measures (MAP/MRR/Top-1) for ranking tasks (WikiQA and Ubuntu).

Baselines and Comparison We use the Decomposable Attention model as a baseline, adding $[a_i; b_i; a_i \odot b_i; a_i - b_i]$ before the compare[2] layers since we found this simple modification to increase performance. This also enables fair comparison with our variation of Quaternion attention which uses Hamilton product over Element-wise multiplication. We denote this as DeAtt. We evaluate at a fixed representation size of $d = 200$

[1] https://github.com/tensorflow/tensor2tensor.

[2] This follows the matching function of (Chen et al., 2016).

Task	NLI			QA	Paraphrase		RS	
Measure	Accuracy			MAP/MRR	Accuracy		Top-1	
Model	SNLI	SciTail	MNLI	WikiQA	Tweet	Quora	UDC	# Params
DeAtt ($d = 50$)	83.4	73.8	69.9/70.9	66.0/67.1	77.8	82.2	48.7	200K
DeAtt ($d = 200$)	**86.2**	79.0	**73.6/73.9**	**67.2/68.3**	80.0	**85.4**	**51.8**	700K
Q-Att ($d = 50$)	85.4	**79.6**	72.3/72.9	66.2/68.1	**80.1**	84.1	51.5	200K (-71%)

Table 1: Experimental results on pairwise text classification and ranking tasks. Q-Att achieves comparable or competitive results compared with DeAtt with approximately one third of the parameter cost.

Model	IMDb	SST	# Params
Transformer	82.6	78.9	400K
Quaternion Transformer (*full*)	**83.9** (+1.3%)	80.5 (+1.6%)	100K (-75.0%)
Quaternion Transformer (*partial*)	83.6 (+1.0%)	**81.4** (+2.5%)	300K (-25.0%)

Table 2: Experimental results on sentiment analysis on IMDb and Stanford Sentiment Treebank (SST) data sets. Evaluation measure is accuracy.

(equivalent to $d = 50$ in Quaternion space). We also include comparisons at equal parameterization ($d = 50$ and approximately $200K$ parameters) to observe the effect of Quaternion representations. We selection of DeAtt is owing to simplicity and ease of comparison. We defer the prospect of Quaternion variations of more advanced models (Chen et al., 2016; Tay et al., 2017b) to future work.

Results Table 1 reports results on seven different and diverse data sets. We observe that a tiny Q-Att model ($d = 50$) achieves comparable (or occasionally marginally better or worse) performance compared to DeAtt ($d = 200$), gaining a 68% parameter savings. The results actually improve on certain data sets (2/7) and are comparable (often less than a percentage point difference) compared with the $d = 200$ DeAtt model. Moreover, we scaled the parameter size of the DeAtt model to be similar to the Q-Att model and found that the performance degrades quite significantly (about $2\% - 3\%$ lower on all data sets). This demonstrates the quality and benefit of learning with Quaternion space.

4.2 Sentiment Analysis

We evaluate on the task of document-level sentiment analysis which is a binary classification problem.

Implementation Details We compare our proposed Quaternion Transformer against the vanilla Transformer. In this experiment, we use the *tiny* Transformer setting in Tensor2Tensor with a vocab size of $8K$. We use two data sets, namely IMDb (Maas et al., 2011) and Stanford Sentiment Treebank (SST) (Socher et al., 2013).

Results Table 2 reports results the sentiment classification task on IMDb and SST. We observe that both the *full* and *partial* variation of Quaternion Transformers outperform the base Transformer. We observe that Quaternion Transformer (*partial*) obtains a $+1.0\%$ lead over the vanilla Transformer on IMDb and $+2.5\%$ on SST. This is while having a 24.5% saving in parameter cost. Finally the *full* Quaternion version leads by $+1.3\%/1.6\%$ gains on IMDb and SST respectively while maintaining a 75% reduction in parameter cost. This supports our core hypothesis of improving accuracy while saving parameter costs.

4.3 Neural Machine Translation

We evaluate our proposed Quaternion Transformer against vanilla Transformer on three data sets on this neural machine translation (NMT) task. More concretely, we evaluate on IWSLT 2015 English Vietnamese (*En-Vi*), WMT 2016 English-Romanian (*En-Ro*) and WMT 2018 English-Estonian (*En-Et*). We also include results on the standard WMT EN-DE English-German results.

Implementation Details We implement models in Tensor2Tensor and trained for $50k$ steps for both models. We use the default base single GPU hyperparameter setting for both models and average checkpointing. Note that our goal is not to obtain state-of-the-art models but to fairly and systematically evaluate both vanilla and Quaternion

	BLEU			
Model	IWSLT'15 *En-Vi*	WMT'16 *En-Ro*	WMT'18 *En-Et*	# Params
Transformer Base	28.4	**22.8**	14.1	44M
Quaternion Transformer (*full*)	28.0	18.5	13.1	11M (-75%)
Quaternion Transformer (*partial*)	**30.9**	22.7	**14.2**	29M (-32%)

Table 3: Experimental results on neural machine translation (NMT). Results of Transformer Base on EN-VI (IWSLT 2015), EN-RO (WMT 2016) and EN-ET (WMT 2018). Parameter size excludes word embeddings. Our proposed Quaternion Transformer achieves comparable or higher performance with only 67.9% parameter costs of the base Transformer model.

Transformers.

Results Table 3 reports the results on neural machine translation. On the IWSLT'15 En-Vi data set, the *partial* adaptation of the Quaternion Transformer outperforms (+2.5%) the base Transformer with a 32% reduction in parameter cost. On the other hand, the *full* adaptation comes close (-0.4%) with a 75% reduction in paramter cost. On the WMT'16 En-Ro data set, Quaternion Transformers do not outperform the base Transformer. We observe a -0.1% degrade in performance on the *partial* adaptation and -4.3% degrade on the *full* adaptation of the Quaternion Transformer. However, we note that the drop in performance with respect to parameter savings is still quite decent, *e.g.*, saving 32% parameters for a drop of only 0.1 BLEU points. The *full* adaptation loses out comparatively. On the WMT'18 En-Et dataset, the *partial* adaptation achieves the best result with 32% less parameters. The *full* adaptation, comparatively, only loses by 1.0 BLEU score from the original Transformer yet saving 75% parameters.

WMT English-German Notably, Quaternion Transformer achieves a BLEU score of 26.42/25.14 for partial/full settings respectively on the standard WMT 2014 En-De benchmark. This is using a single GPU trained for $1M$ steps with a batch size of 8192. We note that results do not differ much from other single GPU runs (i.e., 26.07 BLEU) on this dataset (Nguyen and Joty, 2019).

4.4 Mathematical Language Understanding

We include evaluations on a newly released mathematical language understanding (MLU) data set (Wangperawong, 2018). This data set is a character-level transduction task that aims to test a model's the compositional reasoning capabilities. For example, given an input $x = 85, y = -523, x * y$ the model strives to decode an output of -44455. Several variations of these problems exist, mainly switching and introduction of new mathematical operators.

Implementation Details We train Quaternion Transformer for $100K$ steps using the default Tensor2Tensor setting following the original work (Wangperawong, 2018). We use the tiny hyperparameter setting. Similar to NMT, we report both *full* and *partial* adaptations of Quaternion Transformers. Baselines are reported from the original work as well, which includes comparisons from Universal Transformers (Dehghani et al., 2018) and Adaptive Computation Time (ACT) Universal Transformers. The evaluation measure is accuracy per sequence, which counts a generated sequence as correct if and only if the entire sequence is an exact match.

Results Table 4 reports our experimental results on the MLU data set. We observe a modest $+7.8\%$ accuracy gain when using the Quaternion Transformer (*partial*) while saving 24.5% parameter costs. Quaternion Transformer outperforms Universal Transformer and marginally is outperformed by Adaptive Computation Universal Transformer (ACT U-Transformer) by 0.5%. On the other hand, a *full* Quaternion Transformer still outperforms the base Transformer $(+2.8\%)$ with 75% parameter saving.

4.5 Subject Verb Agreement

Additionally, we compare our Quaternion Transformer on the subject-verb agreement task (Linzen et al., 2016). The task is a binary classification problem, determining if a sentence, *e.g.*, *'The keys to the cabinet _____.'* follows by a plural/singular.

Implementation We use the Tensor2Tensor framework, training Transformer and Quaternion Transformer with the tiny hyperparameter setting

Model	Acc / Seq	# Params
Universal Transformer	78.8	-
ACT U-Transformer	84.9	-
Transformer	76.1	400K
Quaternion Transformer (*full*)	78.9 (+2.8%)	100K (-75%)
Quaternion Transformer (*partial*)	**84.4** (+8.3%)	300K (-25%)

Table 4: Experimental results on mathematical language understanding (MLU). Both Quaternion models outperform the base Transformer model with up to 75% parameter savings.

with $10k$ steps.

Results Table 5 reports the results on the SVA task. Results show that Quaternion Transformers perform equally (or better) than vanilla Transformers. On this task, the partial adaptation performs better, improving Transformers by $+0.7\%$ accuracy while saving 25% parameters.

Model	Acc	Params
Transformer	94.8	400K
Quaternion (*full*)	94.7	100K
Quaternion (*partial*)	**95.5**	300K

Table 5: Experimental results on subject-verb agreement (SVA) number prediction task.

5 Related Work

The goal of learning effective representations lives at the heart of deep learning research. While most neural architectures for NLP have mainly explored the usage of real-valued representations (Vaswani et al., 2017; Bahdanau et al., 2014; Parikh et al., 2016), there have also been emerging interest in complex (Danihelka et al., 2016; Arjovsky et al., 2016; Gaudet and Maida, 2017) and hypercomplex representations (Parcollet et al., 2018b,a; Gaudet and Maida, 2017).

Notably, progress on Quaternion and hypercomplex representations for deep learning is still in its infancy and consequently, most works on this topic are very recent. Gaudet and Maida proposed deep Quaternion networks for image classification, introducing basic tools such as Quaternion batch normalization or Quaternion initialization (Gaudet and Maida, 2017). In a similar vein, Quaternion RNNs and CNNs were proposed for speech recognition (Parcollet et al., 2018a,b). In parallel Zhu *et al.* proposed Quaternion CNNs and applied them to image classification and denoising tasks (Zhu et al., 2018). Comminiello *et al.* proposed Quaternion CNNs for sound detection (Comminiello et al., 2018). (Zhang et al., 2019) proposed Quaternion embeddings of knowledge graphs. A common theme is that Quaternion representations are helpful and provide utility over real-valued representations.

The interest in non-real spaces can be attributed to several factors. Firstly, complex weight matrices used to parameterize RNNs help to combat vanishing gradients (Arjovsky et al., 2016). On the other hand, complex spaces are also intuitively linked to associative composition, along with holographic reduced representations (Plate, 1991; Nickel et al., 2016; Tay et al., 2017a). Asymmetry has also demonstrated utility in domains such as relational learning (Trouillon et al., 2016; Nickel et al., 2016) and question answering (Tay et al., 2018). Complex networks (Trabelsi et al., 2017), in general, have also demonstrated promise over real networks.

In a similar vein, the hypercomplex Hamilton product provides a greater extent of expressiveness, similar to the complex Hermitian product, albeit with a 4-fold increase in interactions between real and imaginary components. In the case of Quaternion representations, due to parameter saving in the Hamilton product, models also enjoy a 75% reduction in parameter size.

Our work draws important links to multi-head (Vaswani et al., 2017) or multi-sense (Li and Jurafsky, 2015; Neelakantan et al., 2015) representations that are highly popular in NLP research. Intuitively, the four-component structure of Quaternion representations can also be interpreted as some kind of multi-headed architecture. The key difference is that the basic operators (*e.g.*, Hamilton product) provides an inductive bias that encourages interactions between these components. Notably, the idea of splitting vectors has also been explored (Daniluk et al., 2017), which is in similar spirit to breaking a vector into four components.

6 Conclusion

This paper advocates for lightweight and efficient neural NLP via Quaternion representations. More concretely, we proposed two models - Quaternion attention model and Quaternion Transformer. We evaluate these models on eight different NLP tasks and a total of *thirteen* data sets. Across all data sets the Quaternion model achieves comparable performance while reducing parameter size. All in all, we demonstrated the utility and benefits of incorporating Quaternion algebra in state-of-the-art neural models. We believe that this direction paves the way for more efficient and effective representation learning in NLP. Our Tensor2Tensor implementation of Quaternion Transformers will be released at https://github.com/vanzytay/QuaternionTransformers.

7 Acknowledgements

The authors thank the anonymous reviewers of ACL 2019 for their time, feedback and comments.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. pages 265–283.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. 2016. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*. pages 1120–1128.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* .

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038* .

Danilo Comminiello, Marco Lella, Simone Scardapane, and Aurelio Uncini. 2018. Quaternion convolutional neural networks for detection and localization of 3d sound events. *arXiv preprint arXiv:1812.06811* .

Ivo Danihelka, Greg Wayne, Benigno Uria, Nal Kalchbrenner, and Alex Graves. 2016. Associative long short-term memory. *arXiv preprint arXiv:1602.03032* .

Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling. *arXiv preprint arXiv:1702.04521* .

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819* .

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Chase Gaudet and Anthony Maida. 2017. Deep quaternion networks. *arXiv preprint arXiv:1712.04604* .

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 873–882.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. *arXiv preprint arXiv:1708.00391* .

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? *arXiv preprint arXiv:1506.01070* .

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics* 4:521–535.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909* .

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 142–150. http://www.aclweb.org/anthology/P11-1015.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654* .

Phi Xuan Nguyen and Shafiq Joty. 2019. Phrase-based attentions. https://openreview.net/forum?id=r1xN5oA5tm.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Thirtieth Aaai conference on artificial intelligence*.

Titouan Parcollet, Mirco Ravanelli, Mohamed Morchid, Georges Linarès, Chiheb Trabelsi, Renato De Mori, and Yoshua Bengio. 2018a. Quaternion recurrent neural networks. *arXiv preprint arXiv:1806.04418* .

Titouan Parcollet, Ying Zhang, Mohamed Morchid, Chiheb Trabelsi, Georges Linarès, Renato De Mori, and Yoshua Bengio. 2018b. Quaternion convolutional neural networks for end-to-end automatic speech recognition. *arXiv preprint arXiv:1806.07789* .

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933* .

Dario Pavllo, David Grangier, and Michael Auli. 2018. Quaternet: A quaternion-based recurrent model for human motion. *arXiv preprint arXiv:1805.06485* .

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Tony Plate. 1991. Holographic reduced representations: Convolution algebra for compositional distributed representations.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training .

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners .

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. pages 1631–1642.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Hermitian co-attention networks for text matching in asymmetrical domains.

Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017a. Learning to rank question answer pairs with holographic dual lstm architecture. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 695–704.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017b. Compare, compress and propagate: Enhancing neural architectures with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102* .

Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. 2017. Deep complex networks. *arXiv preprint arXiv:1705.09792* .

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*. pages 2071–2080.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 5998–6008.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814* .

Artit Wangperawong. 2018. Attending to mathematical language with transformers. *CoRR* abs/1812.02825. http://arxiv.org/abs/1812.02825.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426* .

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 2013–2018.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. *arXiv preprint arXiv:1904.10281* .

Xuanyu Zhu, Yi Xu, Hongteng Xu, and Changjian Chen. 2018. Quaternion convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. pages 631–647.