# Real to H–Space Encoder for Speech Recognition

**4 authors:**

Titouan Parcollet
Université d´Avignon et des Pays du Vaucluse
**26** PUBLICATIONS   **71** CITATIONS

SEE PROFILE

Georges Linarès
Université d´Avignon et des Pays du Vaucluse
**208** PUBLICATIONS   **948** CITATIONS

SEE PROFILE

Mohamed Morchid
Avignon Université
**66** PUBLICATIONS   **242** CITATIONS

SEE PROFILE

Renato De Mori
McGill University and University of Avignon
**322** PUBLICATIONS   **4,657** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project   National Research Agency (France) - Galerie des Festivals (GaFes) View project

Project   Interspeech 2017 View project

# Real to H-space Encoder for Speech Recognition

*Titouan Parcollet*[1,2], *Mohamed Morchid*[1]*, Georges Linarès*[1]*, Renato De Mori*[3]

[1]Avignon Université, LIA, France
[2]ORKIS, Aix-en-provence, France
[3]McGill University, Montréal, QC, Canada

titouan.parcollet@alumni.univ-avignon.fr, {firstname.lastname}@univ-avignon.fr

## Abstract

Deep neural networks (DNNs) and more precisely recurrent neural networks (RNNs) are at the core of modern automatic speech recognition systems, due to their efficiency to process input sequences. Recently, it has been shown that different input representations, based on multidimensional algebras, such as complex and quaternion numbers, are able to bring to neural networks a more natural, compressive and powerful representation of the input signal by outperforming common real-valued NNs. Indeed, quaternion-valued neural networks (QNNs) better learn both internal dependencies, such as the relation between the Mel-filter-bank value of a specific time frame and its time derivatives, and global dependencies, describing the relations that exist between time frames. Nonetheless, QNNs are limited to quaternion-valued input signals, and it is difficult to benefit from this powerful representation with real-valued input data. This paper proposes to tackle this weakness by introducing a real-to-quaternion encoder that allows QNNs to process any one dimensional input features, such as traditional Mel-filter-banks for automatic speech recognition.

**Index Terms**: quaternion neural networks, recurrent neural networks, speech recognition

## 1. Introduction

Automatic speech recognition (ASR) systems have been widely impacted by machine learning, and more precisely by the resurgence of deep neural networks (DNNs). In particular, recurrent neural networks (RNNs) have been designed to learn parameters of sequence to sequence mapping, and various models were successfully applied to ASR with a remarkable increasing in the ASR system performance. In order to avoid parameter estimation problems, RNNs with long short-term memory [1, 2], and gated recurrent unit (GRU) [3] have been proposed to mitigate vanishing and exploding gradients when learning long input sequences. Nevertheless, less attention has been paid to model input features with multiple views of speech spectral tokens.

A noticeable exception is the use of complex-valued numbers in neural networks (CVNNs) to jointly represent amplitude and phase of spectral samples [4]. More recently, quaternion-valued neural networks (QNNs) have been investigated to process the traditional Mel-frequency cepstral coefficients (MFCCs), or Mel-filter-banks plus time derivatives [5, 6, 7] as composed entities. Superior accuracy, with up to four times less model parameters, has been observed with quaternion-valued models compared to results obtained with real-valued equivalent models. In fact, common real-valued neural networks process energies and time derivatives independently, learning both global dependencies between multiple time frames, and local values in a specific time frame, without considering the relations between a value and its derivatives. In-

stead, the quaternion algebra allows QNNs [5, 6, 8, 9, 10, 11] to process time frames as composed entities, with internal relations learned within the specific algebra, and global dependencies learned with the neural network architecture, while reducing by an important factor the number of neural parameters needed. As a side effect, with quaternion algebra, the number of neural parameters to be estimated is reduced by an important factor. Nonetheless, QNNs input features must be encoded as quaternion numbers, requiring a preliminary definition of input views that cannot be modified by the learning process. In many cases, it looks advantageous to have multiple input feature views, but there may be different choices of them and it is not clear how to make a selection. Examples could be views based on temporal or spectral relations. In this paper, a real-to-quaternion encoder (R2H) is proposed to let a quaternion-valued neural architecture learn hidden representations of input feature views. The R2H layer acts as an encoder to train QNNs with any real-valued input vector. Indeed, this encoder allows the model to learn in an end-to-end architecture, a latent and quaternion-valued representation of the input data. This representation is then used as an input to a quaternion-valued classifier, to exploit the capabilities of quaternion neural networks. For achieving this objective, the contributions of this paper are:

- Investigate different real-to-quaternion (R2H) encoders to learn an internal representation of any real-valued input data (Section 4).

- Merge the R2H with the previously introduced quaternion long short-term memory neural network (QLSTM, Section 3)[6][1].

- Evaluate this approach on the TIMIT, and Librispeech speech recognition tasks (Section 5).

Results improvements on both TIMIT and Librispeech speech recognition tasks are reported with the introduction of a R2H encoder in a QLSTM architecture, with input made of 40 Mel-filter-bank coefficients, and with more than three times fewer neural parameters than with real-valued LSTMs.

## 2. Quaternion algebra

The quaternion algebra $\mathbb{H}$ defines operations between quaternion numbers. A quaternion Q is an extension of a complex number to the hyper-complex plane defined in a four dimensional space as:

$$Q = r1 + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}, \tag{1}$$

where $r$, $x$, $y$, and $z$ are real numbers, and 1, $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the quaternion unit basis. In a quaternion, $r$ is the real part, while

---

[1]Code is available at: https://github.com/Orkis-Research/Pytorch-Quaternion-Neural-Networks

$x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ with $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$ is the imaginary part, or the vector part. Such a definition can be used to describe spatial rotations. A quaternion $Q$ can also be summarized into the following matrix of real numbers, that turns out to be more suitable for computations:

$$Q_{mat} = \begin{bmatrix} r & -x & -y & -z \\ x & r & -z & y \\ y & z & r & -x \\ z & -y & x & r \end{bmatrix}. \tag{2}$$

The conjugate $Q^*$ of $Q$ is defined as:

$$Q^* = r1 - x\mathbf{i} - y\mathbf{j} - z\mathbf{k}. \tag{3}$$

Then, a normalized or unit quaternion $Q^{\triangleleft}$ is expressed as:

$$Q^{\triangleleft} = \frac{Q}{|Q|}, \tag{4}$$

with $|Q|$ the norm of Q defined as:

$$|Q| = \sqrt{r^2 + x^2 + y^2 + z^2}. \tag{5}$$

Finally, the Hamilton product $\otimes$ between two quaternions $Q_1$ and $Q_2$ is computed as follows:

$$\begin{aligned} Q_1 \otimes Q_2 =& (r_1 r_2 - x_1 x_2 - y_1 y_2 - z_1 z_2) + \\ & (r_1 x_2 + x_1 r_2 + y_1 z_2 - z_1 y_2)\boldsymbol{i} + \\ & (r_1 y_2 - x_1 z_2 + y_1 r_2 + z_1 x_2)\boldsymbol{j} + \\ & (r_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 r_2)\boldsymbol{k}. \end{aligned} \tag{6}$$

The Hamilton product is used in QNNs to perform transformations of vectors representing quaternions, as well as scaling and interpolation between two rotations following a geodesic over a sphere in the $\mathbb{R}^3$ space as shown in [12].

## 3. Quaternion long short-term memory neural networks

Long short-term memory neural networks (LSTM) are a well-known and investigated extension of recurrent neural networks [1, 13]. LSTMs offer an elegant solution to the vanishing and exploding gradient problems, alongside with a stronger learning capability of long and short-term dependencies within sequence. Following these strengths, a quaternion long short-term memory neural network (QLSTM) has been proposed [5].

In a quaternion-valued layer, all parameters are quaternions, including inputs, outputs, weights and biases. The quaternion algebra is ensured by manipulating matrices of real numbers [7, 11] to reconstruct the *Hamilton product* from quaternion algebra. Consequently, for each input vector of size $N$, output vector of size $M$, dimensions are split in four parts: the first one equals to $r$, the second to $x\mathbf{i}$, the third one is $y\mathbf{j}$, and the last one equals to $z\mathbf{k}$. The inference process of a fully-connected layer is defined in the real-valued space by the dot product between an input vector and a real-valued $M \times N$ weight matrix. In a QLSTM, this operation is replaced with the *Hamilton product* '$\otimes$' (Eq. 6) with quaternion-valued matrices (*i.e.* each entry in the weight matrix is a quaternion).

Both LSTM and QLSTM networks rely on a gate action [14], that allows the cell-state to retain or discard information from the past, and the future in the case of a bidirectional (Q)LSTM. Gates are defined in the quaternion space following [5]. Indeed, the gate mechanism implies a component-wise product of the components of the quaternion-valued signal with the gate potential in a split manner [15]. Let $f_t$, $i_t$, $o_t$, $c_t$, and $h_t$ be the forget, input, output gates, cell states and the hidden state of a QLSTM cell at time-step $t$. QLSTM equations are defined as:

$$f_t = \sigma(W_f \otimes x_t + R_f \otimes h_{t-1} + b_f), \tag{7}$$

$$i_t = \sigma(W_i \otimes x_t + R_i \otimes h_{t-1} + b_i), \tag{8}$$

$$c_t = f_t \times c_{t-1} + i_t \times \alpha(W_c \otimes x_t + R_c \otimes h_{t-1} + b_c), \tag{9}$$

$$o_t = \sigma(W_o \otimes x_t + R_o \otimes h_{t-1} + b_o), \tag{10}$$

$$h_t = o_t \times \alpha(c_t), \tag{11}$$

with $\sigma$ and $\alpha$ the Sigmoid and Tanh *quaternion split activations* [15, 9].

Bidirectional connections allow (Q)LSTM networks to consider the past and future information at a specific time step, enabling the model to capture a more global context [2]. Quaternion bidirectional connections are identical to real-valued ones. Indeed, past and future contexts are added together component-wise at each time step.

An adapted scheme initialization for quaternion neural networks parameters has been proposed in [6]. In practice, biases are set to zero while weights are sampled following a Chi-distribution with four degrees of freedom. Finally, QRNNs require a specific backpropagation algorithm detailed in [6].

## 4. R2H encoder

As mentioned in the introduction, having input features represented by quaternions requires to have predefined a number of views for the same input token. This prevents the use of quaternion networks when prior knowledge suggests to use multiple views whose number and type cannot be exactly defined. For example, it is known that time relations between a speech spectrum and its neighbor spectra may improve the classification of the phoneme whose utterance produced the spectrum. Nevertheless, these relations may not be limited to time derivatives of all spectra samples in a spoken sentence. To overcome this limitation, a new method is proposed. It consists in introducing a real-valued encoder directly connected to the real-valued input signal. The real-to-quaternion (R2H) encoder is trained jointly to the rest of the model in an end-to-end manner, such as any other layer. After training, the encoder is expected to allow a mapping from the real space of the input features, to a latent internal representation meaningful for the following quaternion layers. The trained model is thus able to directly deal with real-valued input features, while internally processing quaternion numbers. The R2H encoder is a traditional dense layer followed by a quaternion activation function and normalization. The number of neurons contained in the layer must be a multiple of four for the quaternion representation. Let $W$, $X$ and $B$ be the weight matrix, the real-valued input and the bias vectors respectively. $Q_{out}^{\triangleleft}$ is the unit quaternion vector obtained at the output of the projection layer and is expressed as:

$$Q_{out}^{\triangleleft} = \frac{Q_{out}}{|Q_{out}|}, \tag{12}$$

with,

$$Q_{out} = \alpha(W.X + B), \tag{13}$$

and $\alpha$ any quaternion split activation function. In practice, $Q_{out}$ and $Q_{out}^{\triangleleft}$ follow the quaternion internal representation defined
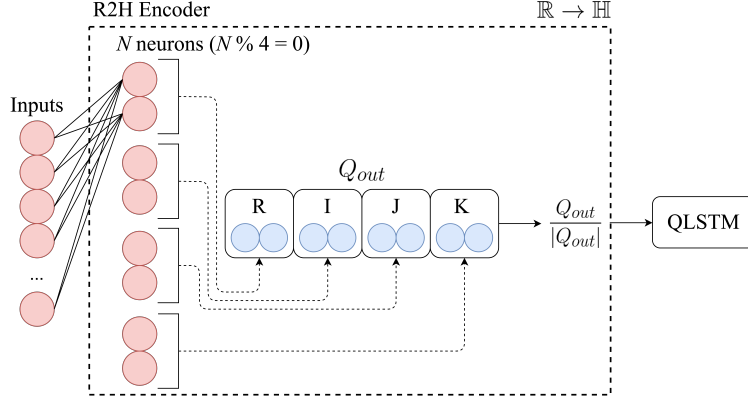
Figure 1: *Illustration of the R2H encoder, used as an input layer to a QLSTM. Inputs are real, before being turned into quaternions, and finally unitary quaternions within the R2H encoder.*

in Section 3. Consequently, the input is split in four features from a latent sub-space, that are interpreted as quaternion components: the first one equals to $r$, the second to $x\mathbf{i}$, the third one is $y\mathbf{j}$, and the last one equals to $z\mathbf{k}$, making it possible to apply the quaternion normalization, and the activation function. At the end of training $Q_{out}^\triangleleft$ capture an internal latent mapping of the real-valued input signal $X$ trough a vector of unit quaternions. Adding the R2H encoder as an input layer to QLSTMs or to any other QNNs allow the model to deal with real-valued inputs, while taking the strengths of QNNs (Figure 1).

# 5. Experiments

Model architectures used for the experiments are presented in Section 5.1. Then, the R2H encoder is compared to the traditional and naive quaternion representation on the TIMIT and Librispeech speech recognition tasks (Section 5.2)

## 5.1. Model architectures

QLSTMs have already been investigated for speech recognition in [5] and [6]. Consequently, and based on these previous researches, QLSTMs are composed with four bidirectional QLSTM layers with an internal real-valued size of $1,024$, equivalent to 256 quaternion neurons. Indeed, $256 \times 4 = 1,024$ real numbers. The R2H encoder size varies from $256$ to $1,024$ to explore the best latent quaternion representation. Tanh, HardTanh and ReLU activation functions are investigated to compare the impact of bounded (Tanh, Hardtanh) and unbounded (ReLU) R2H encoders. In fact, the quaternion normalization allows a numerical reduction of the internal representation, but the ReLU counteracts the latter effect by integrating high real and positive values to the encoding. The final layer is real-valued and corresponds to the HMM states obtained with the Kaldi [16] toolkit. A dropout of $0.2$ is applied across all the layers except the output. The Adam optimizer [17] is used to train the models with vanilla hyperparameters. The learning rate is halved every-time the loss on the validation set is below a certain threshold fixed to $0.001$ to avoid overfitting. Finally, models are implemented with the Pytorch-Kaldi toolkit [18]. While the effectiveness of QLSTM over LSTM has been demonstrated, an LSTM network trained in the same conditions and based on [5] is considered as a baseline. All the models are trained during 30 epochs, and the results on both the validation and test sets are saved at this point.

## 5.2. Phoneme recognition with the TIMIT corpus

The training process is based on the standard $3,696$ sentences uttered by $462$ speakers, while testing is conducted on $192$ sentences uttered by $24$ speakers of the TIMIT [19] dataset. A validation set composed of $400$ sentences uttered by $50$ speakers is used for hyper-parameter tuning. The raw audio is processed with a window of 25ms and an overlap of 10ms. Then, 40-dimensional log Mel-filter-bank coefficients are extracted with the Kaldi toolkit. In previous work with QLSTMs [6, 5], first, second and third time order derivatives were composed with spectral energies to build a multidimensional input representation with quaternions. In this paper, the time derivatives are no longer used. Instead, latent representations are directly learned from the R2H encoder, fed with the 40 log Mel-filter bank coefficients. For the sake of comparison, an input quaternion is naively composed with input features from four consecutive Mel-filter-bank coefficients, before being fed to a standard QLSTM.

Figure 2 reports the results obtained for the investigation of the R2H encoder size and the impact of the activation layer. Results are from an average of three runs and are not obtained w.r.t to the validation set. Indeed, performances on the test set are evaluated only once at the end of the training phase. It is first interesting to note that a layer of $1,024$ neurons always gives better results than a layer of size $256$ or $512$, without even considering activation functions. In the same manner, the Tanh activation outperforms both ReLU and Hardtanh activation function with all the layer size, with an average phoneme error rate (PER) on the TIMIT test set of $15.6\%$ compared to $16.7$ and $16\%$ for the ReLU and HardTanh activations. It is important to note that the ReLU activation gives the worst results. An explanation of such phenomenon is the definition interval of the ReLU function. When dealing with ReLU, outputs of the R2H layer are not bounded in the positive domain before being normalised. Therefore, the dense layer can output large values that are then squashed by the quaternion normalization, and it can be hard for the neural network to learn such mapping. Conversely, both Hardtanh and Tanh functions are bounded by $-1$ and $1$, making it easier to learn, since values of the R2H layer before and after normalization vary on the same range. The HardTanh function also hardly saturates at $-1$ and $1$ in the same manner as the ReLU activation for negative numbers, while the Tanh smoothly tends to these bounds. Consequently, the HardTanh gives slightly worst results than the Tanh. Finally, a best

PER of 15.4% is obtained with a normalised R2H encoder of size $1,024$ based on the Tanh activation function, compared to 16.5% and 15.9% with ReLU and Hardtanh functions.
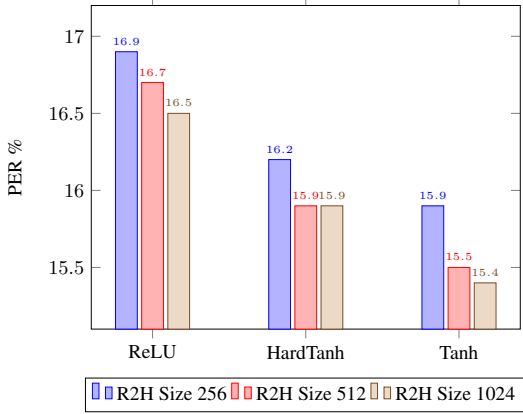


*Figure 2: Phoneme Error Rate (PER %) obtained on the test set of the TIMIT corpus with different activation functions, and different R2H encoder size for a QLSTM. Results are from an average of three runs.*

Table 1 presents a summary of the results observed on the TIMIT phoneme recognition task with a QLSTM and basic quaternion features, compared to the proposed QLSTM coupled with the best R2H encoder from Fig. 2. For fair comparison, a real-valued LSTM is also tested. As highlighted in [6], QL-STMs models require less neural parameters than LSTMs due to their internal quaternion algebra. Therefore, an LSTM with $1,024$ neurons per layer is composed of $46.0$ million parameters, while corresponding QLSTMs only need $15.5M$ parameters. It is first interesting to note that the R2H encoder helps the QLSTM to obtains the same PER as the real-valued LSTM, while dealing with a real-valued input signal. Indeed, both models performed at 15.4% on the test set, while the QLSTM still requires more than three times fewer neural parameters.

*Table 1: Phoneme error rate (PER%) of the models on the development and test sets of the TIMIT dataset. "Params" stands for the total number of trainable parameters. "R2H-Norm" and "R2H" correspond to R2H encoders with and without normalization. Results are from an average of 3 runs.*

| Models | Dev. | Test | Params |
|---|---|---|---|
| LSTM | 14.5 | 15.4 | 46.0M |
| QLSTM | 14.9 | 15.9 | 15.5M |
| R2H-QLSTM | 14.7 | 15.7 | 15.5M |
| R2H-Norm-QLSTM | 14.4 | 15.4 | 15.5M |

Then, it is worth underlining that the basic QLSTM without R2H layer obtains the worst PER of all models with 15.9% on the test set, due to the inappropriate input representation. Then, the impact of the quaternion normalization process is investigated by comparing a R2H encoder without normalization, to a normalised one. As expected, the quaternion normalization helps the input to fit the quaternion representation, and thus gives better results with 15.4% of PER compared to 15.7% for the non-normalized R2H encoder. It is important to mention that such results are obtained without batch-normalization, speaker adaptation or rescoring methods.

## 5.3. Speech recognition with the Librispeech corpus

The experiments are extended to the larger Librispeech dataset [20]. Librispeech is composed of three distinct training subsets of 100, 360 and 500 hours of speech respectively, representing a total training set of 960 hours of read English speech. In our experiments, the models are trained following the setups described in Section 5.1, and based on the *train_clean_100* subset containing 100 hours. Results are reported on the *test_clean* set. Input features are the same as for the TIMIT experiments, and the best activation function reported in Figure 2 is used (Tanh). No regularization techniques such as batch-normalization are used, and no rescoring methods are applied at testing time.

*Table 2: Word error rate (WER%) of the models on test_clean set of the Librispeech dataset with a training on the train_clean_100 subset. "Params" stands for the total number of trainable parameters. "R2H-Norm" and "R2H" correspond to R2H encoders with and without normalization. No rescoring technique is applied.*

| Models | Test | Params |
|---|---|---|
| LSTM | 8.1 | 49.0M |
| QLSTM | 8.5 | 17.7M |
| R2H-QLSTM | 8.3 | 17.7M |
| R2H-Norm-QLSTM | 8.0 | 17.7M |

The total number of neural parameters is slightly different when compared with the TIMIT experiments due to the increased number of HMM states, and therefore neurons, of the output layer for the Librispeech task. Nonetheless, the number of parameters is still lowered by a factor of 3 when using QL-STM networks, compared to the real-valued LSTM. Similarly to the TIMIT experiments, the QLSTM with a normalized R2H layer reaches slightly better performances in term of word error rate (WER), with 8.0% compared to 8.1% for the LSTM. Moreover, the R2H encoder allows the QLSTM WER to decrease from 8.5% to 8.0%, representing an absolute gain of 0.5%. The reported results on the larger Librispeech dataset demonstrate that the R2H encoder solution scales well with more realistic speech recognition tasks.

## 6. Conclusions

**Summary.** This paper addresses one of the major weakness of quaternion-valued neural networks known as the inability to process non quaternion-valued input signal. A new real-to-quaternion (R2H) encoder is introduced, making it possible to learn in a end-to-end manner a latent quaternion representation from any real-valued input data. Such representation is then processed with QNNs such as a quaternion LSTM. The experiments conduced on the TIMIT phoneme recognition task demonstrate that this new approach outperforms a naive quaternion representation of the input signal, enabling the use of QNNs with any type of inputs.

**Future work.** Split activation functions and current quaternion gate mechanisms do not fully respect the quaternion algebra by considering each elements as uncorrelated components. A future work will consist on the investigation of purely quaternion recurrent neural networks, involving well-adapted activation functions, and proper quaternion gates.

# 7. References

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[3] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015, pp. 577–585.

[4] C. Trabelsi, O. Bilaniuk, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," *arXiv preprint arXiv:1705.09792*, 2017.

[5] T. Parcollet, M. Morchid, G. Linarès, and R. De Mori, "Bidirectional quaternion long short-term memory recurrent neural networks for speech recognition," *arXiv preprint arXiv:1811.02566*, 2018.

[6] T. Parcollet, M. Ravanelli, M. Morchid, G. Linarès, C. Trabelsi, R. D. Mori, and Y. Bengio, "Quaternion recurrent neural networks," *arXiv:1806.04418v2*, 2018.

[7] T. Parcollet, Y. Zhang, M. Morchid, C. Trabelsi, G. Linarès, R. de Mori, and Y. Bengio, "Quaternion convolutional neural networks for end-to-end automatic speech recognition," in *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018.*, 2018, pp. 22–26. [Online].

[8] T. Nitta, "A quaternary version of the back-propagation algorithm," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 5. IEEE, 1995, pp. 2753–2756.

[9] P. Arena, L. Fortuna, G. Muscato, and M. G. Xibilia, "Multilayer perceptrons to approximate quaternion valued functions," *Neural Networks*, vol. 10, no. 2, pp. 335–342, 1997.

[10] T. Isokawa, N. Matsui, and H. Nishimura, "Quaternionic neural networks: Fundamental properties and applications," *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*, pp. 411–439, 2009.

[11] C. J. Gaudet and A. S. Maida, "Deep quaternion networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[12] T. Minemoto, T. Isokawa, H. Nishimura, and N. Matsui, "Feed forward neural network with random quaternionic neurons," *Signal Processing*, vol. 136, pp. 59–68, 2017.

[13] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[14] I. Danihelka, G. Wayne, B. Uria, N. Kalchbrenner, and A. Graves, "Associative long short-term memory," *arXiv preprint arXiv:1602.03032*, 2016.

[15] D. Xu, L. Zhang, and H. Zhang, "Learning alogrithms in quaternion neural networks using ghr calculus," *Neural Network World*, vol. 27, no. 3, p. 271, 2017.

[16] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011, iEEE Catalog No.: CFP11SRW-USB.

[17] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] M. Ravanelli, T. Parcollet, and Y. Bengio, "The pytorch-kaldi speech recognition toolkit," in *In Proc. of ICASSP*, 2019.

[19] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, 1993.

[20] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.