2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP) | 979-8-3503-2411-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/MLSP55844.2023.10285926

PHYDI: INITIALIZING PARAMETERIZED HYPERCOMPLEX NEURAL NETWORKS AS IDENTITY FUNCTIONS

Matteo Mancanelli, Eleonora Grassucci, Aurelio Uncini, and Danilo Comminiello

Dept. of Information Engineering, Electronics, and Telecomm., Sapienza University of Rome, Italy

ABSTRACT

Neural models based on hypercomplex algebra systems are growing and prolificating for a plethora of applications, ranging from computer vision to natural language processing. Hand in hand with their adoption, parameterized hypercomplex neural networks (PHNNs) are growing in size and no techniques have been adopted so far to control their convergence at a large scale. In this paper, we study PHNNs convergence and propose parameterized hypercomplex identity initialization (PHYDI), a method to improve their convergence at different scales, leading to more robust performance when the number of layers scales up, while also reaching the same performance with fewer iterations. We show the effectiveness of this approach in different benchmarks and with common PHNNs with ResNets- and Transformer-based architecture. The code is available at `https://github.com/ispamm/PHYDI`.

***Index Terms*—** Hypercomplex neural networks, identity initialization, residual connections, hypercomplex algebra, neural networks convergence

1. INTRODUCTION

Although the largest part of deep learning models is defined following the rules of real-valued numbers and algebra $\mathbb{R}$, such a choice is not always the best one for multidimensional data. Therefore, an increasing number of works are exploring the possibility of defining models with different underlying algebras that better fit the problem of study. Among these, Clifford, Cayley-Dickson, and hypercomplex algebras have been widely adopted. More recently, Parameterized hypercomplex neural networks (PHNNs) have transformed the wide research field of neural models defined over such hypercomplex algebras. This happened thanks to their flexibility and malleability that allow users to make use of hypercomplex-based networks without developing ad-hoc architectures or seeking the proper algebra domain that best fits the specific task. Indeed, PHNNs grasp algebra rules directly from data and therefore they can be employed for any n-dimensional data without modifications to architectural layers. For that reason, the already ample field of hypercomplex models based on complex [1], quaternion [2], dual quaternion [3, 4], and octonion [1] numbers has been permeated by PHNNs. These networks have been defined with different known backbones such as ResNets [5, 6], GANs [7, 8], graph neural networks [9], and Transformers [10], among others [11, 12]. So as with any neural model, their expressiveness increases with deeper representations, also due to the fact that for high values of the hyperparameter n, which also affects the number of parameters reducing them to $1/n$, PHNNs may be defined with very few parameters even if the number of layers is large.

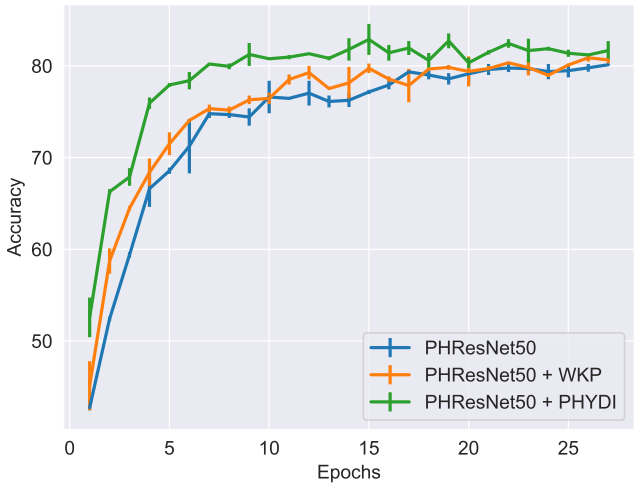

Fig. 1. The proposed PHYDI initialization speeds up the convergence of parameterized hypercomplex neural networks in which it is employed.

However, any convergence study or regularization and normalization strategies have been proposed for improving PHNNs training stability and convergence when the number of layers increases. Indeed, PHNNs behavior with a large number of layers is still unknown, as well as it is not clear how the parameters reduction driven by the hyperparameter n affects the learning of very deep networks and whether intra-layer parameters and overall parameters are balanced

Corresponding author's email: eleonora.grassucci@uniroma1.it. We acknowledge financial support from PNRR MUR project PE0000013-FAIR.

979-8-3503-2411-2/23/$31.00 ©2023 IEEE

during training.

In this paper, therefore, we first conduct a study on PHNNs convergence in large-scale training, discovering that very deep architectures have convergence issues, and founding that the hyperparameter n is related to the convergence. In order to address these issues, we propose parameterized hypercomplex identity initialization (PHYDI), a method to help PHNNs converge fast and improve large-scale model learning motivated by the dynamical isometry that has been proved a clear indicator of trainability [13]. We propose to initialize each parameterized hypercomplex multiplication (PHM) or convolution (PHC) layer, that represent the core of PHNNs, as an identity function. The proposed initialization is carried out by adding a residual connection and a trainable zero-initialized parameter that multiplies the actual layer parameters, as introduced for real-valued models [14].

We prove that PHYDI improves very deep ResNets- and Transformer-based PHNNs convergence in different benchmarks even when standard PHNNs diverge, therefore improving the learning ability of large architectures. Furthermore, the proposed method leads to faster convergence of each PHNN we test, both in image and language datasets.

In summary, our contributions are: i) We conduct, to the best of our knowledge, the first study on the convergence of PHNNs in large-scale training, showing that this is also related to the key hyperparameter n. ii) We propose PHYDI, a method to avoid divergence of very deep PHNNs, which also fastens the convergence of convolutional- and attention-based PHNNs in multiple benchmarks, allowing the learning of large-scale networks with fewer iterations.

The rest of the paper is organized as follows. The background on PHNNs is developed in Section 2, the proposed method is presented in Section 3, while the experimental evaluation is performed in Section 4. Finally, conclusions are drawn in Section 5.

2. PARAMETERIZED HYPERCOMPLEX NEURAL NETWORKS

Although the largest part of neural models is defined over the set of real numbers $\mathbb{R}$, this configuration does not always perfectly fit every task. In the case of multidimensional data, such as objects in the 3D space, multichannel audio signals, or color images, real-valued models break the multidimensional nature of the input and their learning may fail. For this reason, recently, several works are considering other numerical domains with stronger algebraic and geometry properties that better model the multidimensional structure of the data. Among these domains, the complex (one imaginary unit) and the quaternion (three imaginary units) ones have been the most cited due to their 2D and 4D nature and their algebraic properties that can be leveraged during the learning phase. Indeed, the so-called Hamilton products of quaternions allow a parameter reduction up to the 75% and, most importantly,

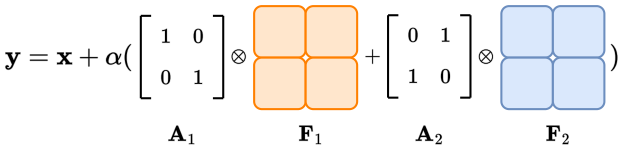

Fig. 2. A simple PHM formulation for the proposed method with $n = 2$. The PHM layer builds the weights matrix $\mathbf{H}$ as a sum of 2 Kronecker products that are multiplied by the PHYDI learnable parameter α, initialized to 0 so as to ensure the identity function brought by the inserted residual connection. Note that the matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are learnable during training, exactly as the classical weights matrix $\mathbf{F}_1$ and $\mathbf{F}_2$.

the parameters sharing within the layer. This feature enables the model equipped with such algebra to preserve local relations and correlations among the input dimensions, building a more accurate view of the multidimensional data. However, such models are limited to domain dimensionality and therefore arduous to apply when data does not fit the domain-specific dimensions, such as 2 for the complex domain or 4 for the quaternion one. For this reason, parameterized hypercomplex neural networks (PHNNs) have been introduced in the literature [5]. The family of PHNNs comprises neural models defined over various domains by means of hypercomplex layers parameterized by the user. The core operation of these networks is the parameterized hypercomplex (PH) layer that builds the weight matrix as a sum of Kronecker products as follows:

$$\mathbf{H} = \sum_{i=1}^{n} \mathbf{A}_i \otimes \mathbf{F}_i, \tag{1}$$

where the hyperparameter n defines the domain in which the model operates (e.g., $n = 2$ complex domain, $n = 4$ quaternion domain, and so on), and, most importantly, it can be set to any integer value even though the algebra regulations are not known. That is possible thanks to the matrices $\mathbf{A}_i$ that learn the algebra rules directly from data, and the matrices $\mathbf{F}_i$ representing the parameters (or filters for convolutional layers). Thanks to their flexibility, PH layers can then be involved in several common neural architectures such as ResNets [5], transformers [10], and graph neural networks [9], just as a replacement for standard real-valued layers. Although PHNNs show outstanding results in several tasks, such models are tremendously new in the literature and their convergence behavior has not been studied yet.

3. PHYDI: PARAMETERIZED HYPERCOMPLEX IDENTITY INITIALIZATION

The proposed parameterized hypercomplex identity initialization (PHYDI) for PHNNs is easy to be involved in any pre-existing PHNN as it consists of a slight modification to the

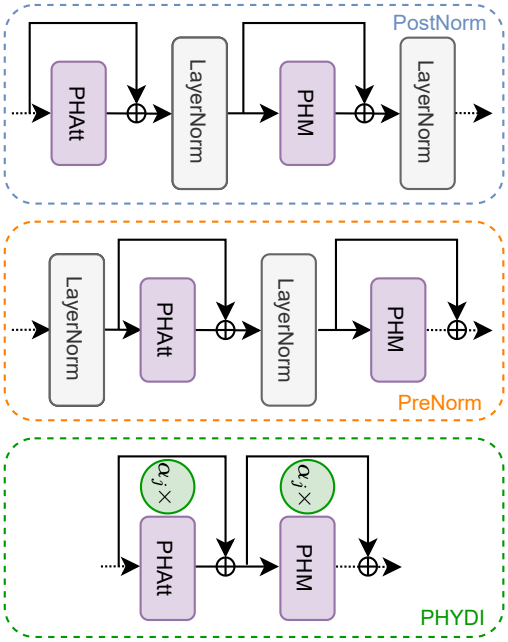

Fig. 3. The proposed PHYDI formulation for PHTransformer layers compared with common PostNorm [15] and PreNorm [16] architectures.

hypercomplex architecture to perform the identity operation. Through identity, we can ensure initial dynamical isometry that has been proven to be a clear aspect of well-trainable networks [13]. In practice, to simplify the gradient propagation at the initialization, the signal should not propagate on the PH layer $\mathcal{F}\{\mathbf{H}\}$, but rather on its residual connection $\mathbf{x}$. To do that, a parameter α is set to multiply the PH layer and initialized to 0, so that only the residual connection remains active during the first iteration. More formally, the $j+1$-th PH layer with PHYDI formulation becomes:

$$\mathbf{x}_{j+1} = \mathbf{x} + \alpha_j \mathcal{F}\{\mathbf{H}\}(\mathbf{x}), \tag{2}$$

whereby α is a learnable parameter initialized to 0 at the beginning of the training. A visual representation is shown in Fig. 2, where the Kronecker product blocks that build the PH layer are then summed and multiplied by the PHYDI learnable parameter α. In the next subsections, we formally define two of the most common vision and language PH architectures with the proposed PHYDI method, which are PHResNets and PHTransformers.

3.1. Initialization of Hypercomplex ResNets

PHResNets have already been defined with residual connections, so no architectural changes are needed except for the insertion of the PHYDI parameter α that initializes the network to perform the identity operation. Therefore, the PHResNet layer will be defined exactly as (2), where instead of generic layers, the function $\mathcal{F}(\mathbf{x}, \{\mathbf{H}_j\})$ comprises parameterized hypercomplex convolutional (PHC) layers:

$$\mathcal{F}(\mathbf{x}, \{\mathbf{H}_j\}) = \text{PHC}\left(\text{ReLU}\left(\text{PHC}(\mathbf{x})\right)\right). \tag{3}$$

Although this formulation is already based on residual connections, we will demonstrate that the identity initialization due to the α parameter gives PHResNets a faster convergence, especially in very deep networks.

3.2. Initialization of Hypercomplex Transformers

PHTransformers also have a residual connection structure inside their layers. However, its composition is more complex than PHResNets one and a more detailed study has to be performed for the identity initialization. Indeed, a standard PHTransformer layer can be described by

$$\begin{aligned}\mathbf{x}_{j+1} = \text{LayerNorm}\{\mathbf{x}_j + \text{PHM}(\\ \text{LayerNorm}(\mathbf{x}_j + \text{PHAtt}(\mathbf{x}_j))\},\end{aligned} \tag{4}$$

that is, a post-normalization of the sub-layer modules (Parameterized hypercomplex multi-head attention (PHAtt) and Parameterized hypercomplex multiplication (PHM)). Following the literature suggestion [14], in order to initialize the layer as the identity function, we can remove the layer normalization and insert the PHYDI parameters as multipliers for the sub-layers as follows:

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \text{PHM}(\mathbf{x}_j + \alpha_j \text{Att}(\mathbf{x}_j)). \tag{5}$$

Note that the learnable parameter α_j is the same within the PHTransformer layer and it is initialized to $\alpha_j = 0$ to ensure the identity operation. Figure 3 shows the three different PHTransformer configurations we test in our experiments, namely PostNorm [15], PreNorm [16], and PHYDI (proposed).

4. EXPERIMENTS

In this Section, we present the experimental validation of our theoretical claims. We conduct experiments on two common tasks such as image classification and sequence-to-sequence prediction. Therefore, we involve different ResNets backbones (ResNet18, ResNet50, and ResNet152) for the first task, and a Transformer-based model varying the number of layers for the second task.

4.1. Comparison methods

For convolutional networks, no previous initialization or fast-convergence method have been proposed for PHHNs, so we compare our proposal with standard PHResNets with classical initialization [5]. Additionally, we experiment with

Table 1. PHResNets with standard, WKP, and PHYIDI initialization for different values of the hyperparameter n in the CIFAR10 dataset. Metrics M1: Epochs to 80% Acc, M2: # Epochs to beat one w/ PHYDI. The uncertainties correspond to standard error.

Model	n	M1↓	M2
PHResNet18	2	6.00 ± 0.58	2
+ WKP	2	**5.75** ± 0.25	2
+ PHYDI	2	6.00 ± 0.00	-
PHResNet50	2	10.67 ± 1.20	3
+ WKP	2	10.67 ± 0.67	2
+ PHYDI	2	**7.00** ± 0.58	-
PHResNet152	2	32.67 ± 2.03	4
+ WKP	2	29.80 ± 3.68	4
+ PHYDI	2	**6.33** ± 1.33	-
PHResNet18	3	6.33 ± 0.33	2
+ WKP	3	6.00 ± 0.00	1
+ PHYDI	3	**5.00** ± 0.58	-
PHResNet50	3	8.67 ± 1.20	2
+ WKP	3	9.0 ± 0.58	2
+ PHYDI	3	**6.33** ± 0.67	-
PHResNet152	3	26.67 ± 1.76	4
+ WKP	3	20.00 ± 2.12	4
+ PHYDI	3	**4.67** ± 0.33	-
PHResNet18	4	**3.75** ± 0.48	1
+ WKP	4	5.67 ± 0.67	2
+ PHYDI	4	4.50 ± 0.50	-
PHResNet50	4	8.33 ± 0.88	2
+ WKP	4	9.33 ± 0.88	2
+ PHYDI	4	**7.00** ± 1.15	-
PHResNet152	4	22.67 ± 3.71	3
+ WKP	4	20.60 ± 1.60	3
+ PHYDI	4	**5.33** ± 0.33	-

Fixup initialization [17], but we note that it required a senseless number of epochs for reaching the 80% of accuracy with PHResNets18 and almost damaged the final accuracy performances for deeper PHResNets. Moreover, identity initialization methods have been already proven to be better than Fixup for real-valued models [14]. For these reasons, we do not report Fixup scores in Table 1. We further validate PHYDI against a weighted non-identity initialization of the Kronecker products. We insert a learnable weight α initialized to 1 in each PH layer product in order to let the network learn the proper weighting mechanism for each Kronecker multiplication, and therefore, for each input dimension. However, the weighted Kronecker product (WKP) does not ensure identity initialization.

Regarding transformers, we compare PHTransformers with different layer normalization positions, such as the original real-valued Transformer PostNorm [15], the newer PreNorm [16], and the proposed PHYDI initialization without normalizations.

4.2. Results on faster convergence

Table 1 shows the image classification results for different PHResNets with and without the proposed PHYDI method. We conduct experiments with three different ResNets backbones with an increasing number of layers (18, 50, 152) and for different configurations of the hyperparameter n. We train all the models following the recipe of the original real-valued ResNet paper [18] that the PHResNet paper has proved to work well also in hypercomplex domains [5]. We just edit the batch size, using a value of 64. We experiment on the image classification task with the CIFAR10 dataset in order to test the method on a standard benchmark. We evaluate the performances of our initialization method according to two metrics, namely the number of epochs the models require to reach the 80% of accuracy (M1), and the number of epochs to beat a model with the proposed approach (M2). PHYDI ensures faster convergence in every test we conduct, lowering the number of epochs to obtain the 80% of accuracy. Furthermore, the advantages of our approach are shared among the most common choices of the hyperparameters n, proving that PHYDI PHResNets are robust across architectural changes. Moreover, the effect of the proposed method is more evident with the increase of model layers. Indeed, while PHResNets152 suffers from slow convergence, endowing such networks with PHYDI initialization drastically fastens the convergence, lowering the number of epochs by more than 20 in some cases. This is clear evidence of the incoming results regarding PHYDI ability in allowing improved learning of large-scale models.

4.3. Results on large-scale models

The signal and gradient propagation in real- as well as in hypercomplex-valued neural networks becomes much more difficult as the model depth increases. Therefore, proper forward and backward flows are crucial for the effective learning of a neural model. In this subsection, we focus our experiment on the sequence-to-sequence PHTransformers [10] on the WikiText2 dataset. As in the previous tests, we experiment PHNNs with different values of the hyperparameter $n = 2, 3, 4$. We train all models for 50 epochs with a batch size of 64 and optimize them with Adagrad and a step learning rate scheduler that progressively reduces it from its original value of 0.01. We experiment with different encoder depths and a number of layers in the set $[12, 16, 24, 32, 48, 64, 96]$. Similar to previous experiments, we set up a metric to prove the fast convergence: we consider the number of epochs to reach a value of perplexity equal to 200.

Figure 4 shows the results of our experiments for different n settings. The first consideration we can draw is that the original Transformer architecture defined in hypercomplex do-

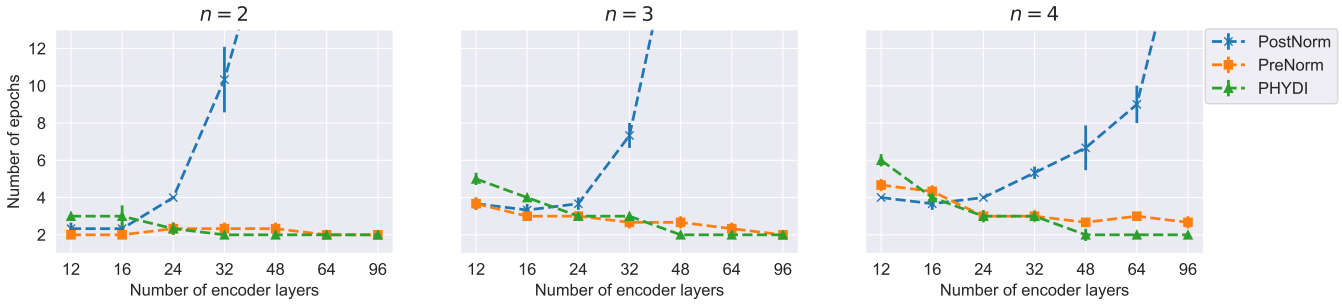

Fig. 4. Number of epochs to reach a perplexity value ≥ 200 in the WikiText2 dataset for PH Transformers with increasing depth of the encoder model from 12 to 96. The three plots refer to different values of the hyperparameter $n = 2, 3, 4$. Vertical standard error bars are shown too, computed over multiple runs. The PH Transformer equipped with PHYDI preserves the performance even in large-scale models while standard and PostNorm models diverge.

mains with PostNorm diverges as the number of encoder layers increases. This result extends real-valued results already proven in [14]. However, this configuration seems linked to the number of parameters of the model and, consequently, to the choice of the hyperparameter n in PHNNs. Indeed, it is interesting to note that while for $n = 2$ the PostNorm configuration diverges with just 48 layers, acceptable results are instead obtained up to 64 layers when setting $n = 4$. Therefore, the gradient is better propagated with fewer parameters and model convergence does not depend only on the depth of the model itself.

The PreNorm method performs better than the original PostNorm, providing stabler performances even when the depth increases. Additionally, this method performs best for small encoders with 12 or 16 layers, exceeding also the proposed approach. However, PHYDI clearly improves PH-Transformer convergence and robustness when the architecture becomes very deep, especially from 32 layers up to the maximum of our experiments. The PHTransformer equipped with PHYDI initialization still requires just 2 epochs to reach a perplexity value of 200 even with 96 encoder layers, proving that the proposed approach improves the learning of large-scale PHNNs. Overall, it is important to note that PHYDI improves large models stability and it also provides robust results also for unfavorable configurations, e.g. small networks.

4.4. Robustness to hyperparameters settings

We perform additional experiments with different hyperparameters settings further to validate PHYDI and its robustness to various configurations. In detail, we conduct tests varying the learning rate, considering both 0.01 and 0.1, and changing the batch size (from 64 to 128), since this usually affects convergence speed. Table 2 shows the results of the experiments with PHResNets18 and PHResNets50 with the same evaluation metrics as Table 1. It is evident that PHYDI improves results under different settings too, being robust to various hyperparameters configurations.

Table 2. PHResNets results with different initialization methods and learning rate configurations. The batch size is different from previous experiments. Results show that PHYIDI is robust across various settings, improving convergence speed.

Model	n	btc-sz	lr	M1↓	M2
PHResNet18	2	128	0.1	21.86 ± 3.14	3
+ WKP	2	128	0.1	17.17 ± 3.35	3
+ PHYDI	2	128	0.1	**10.00** ± 1.22	-
PHResNet50	2	128	0.01	24.33 ± 2.19	3
+ WKP	2	128	0.01	12.67 ± 1.67	2
+ PHYDI	2	128	0.01	**8.67** ± 0.88	-
PHResNet18	3	128	0.1	20.50 ± 5.08	3
+ WKP	3	128	0.1	12.83 ± 1.68	3
+ PHYDI	3	128	0.1	**8.00** ± 0.58	-
PHResNet50	3	128	0.01	11.25 ± 2.56	2
+ WKP	3	128	0.01	12.80 ± 3.89	2
+ PHYDI	3	128	0.01	**10.75** ± 2.25	-
PHResNet18	4	128	0.1	21.40 ± 3.22	3
+ WKP	4	128	0.1	20.67 ± 2.78	2
+ PHYDI	4	128	0.1	**8.80** ± 1.24	-
PHResNet50	4	128	0.01	22.33 ± 8.67	2
+ WKP	4	128	0.01	11.00 ± 0.58	2
+ PHYDI	4	128	0.01	**10.00** ± 1.00	-

5. CONCLUSION

In this paper, we study the convergence of common PHNNs, proposing parameterized hypercomplex identity initialization (PHYDI), a method that, with very few architectural changes, improves convergence in speed and learning depth. We experiment with PH convolutional and transformer models on known benchmarks and we prove that PHNNs endowed with PHYDI gain convergence speed and a better gradient propagation when the number of layers increases.

6. REFERENCES

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

[2] Z. Jia, Q. Jin, M. K. Ng, and X.-L. Zhao, "Non-local robust quaternion matrix completion for large-scale color image and video inpainting," *IEEE Trans. on Image Process.*, vol. 31, pp. 3868–3883, 2022.

[3] J. Pöppelbaum and A. Schwung, "Predicting rigid body dynamics using dual quaternion recurrent neural networks with quaternion attention," *IEEE Access*, vol. 10, pp. 82 923–82 943, 2022.

[4] E. Grassucci, G. Mancini, C. Brignone, A. Uncini, and D. Comminiello, "Dual quaternion ambisonics array for six-degree-of-freedom acoustic representation," *Pattern Recognition Letters*, vol. 166, pp. 24–30, Feb 2023.

[5] E. Grassucci, A. Zhang, and D. Comminiello, "PHNNs: Lightweight neural networks via parameterized hypercomplex convolutions," *IEEE Trans. on Neural Netw. and Learning Systems*, pp. 1–13, dec 2022.

[6] E. Lopez, E. Grassucci, M. Valleriani, and D. Comminiello, "Hypercomplex neural architectures for multiview breast cancer classification," *arXiv preprint arXiv:2204.05798*, 2022.

[7] G. Sfikas, G. Retsinas, B. Gatos, and C. Nikou, "Hypercomplex generative adversarial networks for lightweight semantic labeling," in *Pattern Recognition and Artificial Intelligence*. Springer International Publishing, 2022, pp. 251–262.

[8] E. Grassucci, L. Sigillo, A. Uncini, and D. Comminiello, "Hypercomplex image-to-image translation," in *Int. Joint Conf. on Neural Netw. (IJCNN)*, 2022.

[9] T. Le, M. Bertolini, F. No'e, and D. A. Clevert, "Parameterized hypercomplex graph neural networks for graph classification," in *Int. Conf. on Artificial Neural Netw. (ICANN)*, 2021.

[10] A. Zhang, Y. Tay, S. Zhang, A. Chan, A. T. Luu, S. C. Hui, and J. Fu, "Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters," in *Int. Conf. on Machine Learning (ICML)*, 2021.

[11] L. Basso, Z. Ren, and W. Nejdl, "Towards efficient ecg-based atrial fibrillation detection via parameterised hypercomplex neural networks," *ArXiv preprint arXiv:2211.02678*, 2022.

[12] I. I. Panagos, G. Sfikas, and C. Nikou, "Compressing audio visual speech recognition models with parameterized hypercomplex layers," in *Hellenic Conference on Artificial Intelligence*, ser. SETN '22, 2022.

[13] J. Pennington, S. Schoenholz, and S. Ganguli, "Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice," in *Advances in Neural Information Processing (NIPS)*, 2017.

[14] T. Bachlechner, B. P. Majumder, H. Mao, G. Cottrell, and J. McAuley, "Rezero is all you need: fast convergence at large depth," in *Conf. on Uncertainty in Artificial Intelligence*, ser. Proceedings of Machine Learning Research, vol. 161, 2021, pp. 1352–1361.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[16] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T.-Y. Liu, "On layer normalization in the transformer architecture," in *Int. Conf. on Machine Learning (ICML)*, 2020.

[17] H. Zhang, Y. N. Dauphin, and T. Ma, "Fixup initialization: Residual learning without normalization." in *Int. Conf. on Learning Representations (ICLR)*, 2019.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778.