Enhancing Image Classification Capabilities In The Vision Transformer Network Model With Quaternion Algebra

Minh Tuan Pham[1 [0000−0001−9843−9676]] and An Hung Nguyen[1 [0009−0009−6164−1745]]

The University of Da Nang, University of Science and Technology, Danang City, Vietnam
pmtuan@dut.udn.vn, 102210009@sv1.dut.udn.vn

Abstract. Vision Transformer is a novel approach in artificial intelligence, focusing on image classification. Despite its potential, ViT's emphasis on global data processing presents accuracy challenges compared to local data processing methods like Convolutional Neural Networks (CNN). To address this, we propose two methods. The first integrates a portion of the Residual Network to replace token transformation layers, allowing for local data feature extraction and improved relationship learning between tokens. The second solution suggests transforming layers in the bottleneck component into types that process in the Quaternion hypercomplex domain, enhancing the multidimensional representation of data. Both solutions aim to leverage the strengths of CNN and ViT, thereby indirectly improving image classification accuracy.

Keywords: Image classification · Deep learning · Vision Transformer · Quaternion Algebra · Multilayer Perceptron Algebra.

1 Introduction

Over recent years, artificial intelligence has seen significant advancements in machine learning models. Among these models is the Vision Transformer (ViT), a new network model primarily used for computer vision tasks such as image classification.

However, ViT's focus on global data processing rather than local data processing has been a weakness, leading to subpar performance compared to convolutional neural network models when applied in a time domain.

This issue has attracted attention from researchers worldwide, leading to various proposed solutions such as Sharpness-Aware Minimization (SAM) [12] and Orthogonalization-Guided Feature Fusion [13][14] to improve the accuracy of the Vision Transformer model.

In this research, we propose two different approaches to enhance the ViT model. These approaches can be seen as a combination of the CNN model with ViT, along with the conversion of real numbers to Quaternion numbers with

 ISBN: 978-604-80-9774-5

adjusted layers. To evaluate the effectiveness of the proposed methods, we conducted tests on three well-known datasets: CIFAR10, CIFAR100, and GTSRB.

The research is divided into five parts. The second part provides background information on related studies such as Vision Transformer, mathematical transformations on Quaternion numbers, setting up a convolutional neural network in the Quaternion domain, a brief introduction to Residual Networks, and weight initialization in the Quaternion number domain. The third part discusses the research method, the process of constructing the proposed solutions, including Quaternion Batch Normalization, Quaternion Fully Connected (QFC) layer, and the implementation of the proposed solutions. The fourth part presents the experimental setup, evaluation method, and research results. Finally, the fifth part concludes the research by summarizing the main contributions and future research plans.

2 Related Methods

2.1 Vision Transformer

The Vision Transformer (ViT) is a type of transformer designed for computer vision. Transformers were introduced in 2017 [5] and widely applied in natural language processing. In 2020, they were adapted, replacing some components to learn relationships between image patches rather than words, making it partially resemble Convolutional Neural Networks (CNNs) to better suit computer vision. This modified transformer is referred to as ViT.

The basic structure of ViT involves dividing the input image into a sequence of patches and encoding these patches into tokens before applying them to a standard Transformer architecture. ViT's attention mechanism transforms the representation vectors of image patches, capturing semantic relationships between patches. This is analogous to natural language processing (NLP), where transformer-encoded vectors combine semantic relationships between words, from syntax to meaning.

ViT has found applications in image recognition, image segmentation, etc. The foundational architecture, inspired by a 2020 paper [7], draws inspiration from the BERT (Bidirectional Encoder Representations from Transformers) model introduced in October 2018 [8]. The input image is divided into equally-sized patches, each passing through a linear operator to convert it into a vector, known as "patch embedding." The position of each patch is also transformed into a vector using "position encoding." These two vectors are then added and processed through a series of transformer encoder layers.

In essence, the architecture transforms an image into a sequence of vector representations. To utilize these vector representations for downstream applications, additional network modules are added. For image classification, a block of MLP (Multilayer Perceptron) [9] is stacked on top (output of the Transformer Encoder) to produce a probability distribution for labels. In the author's paper, a linear-GeLU-linear-softmax MLP block is used.

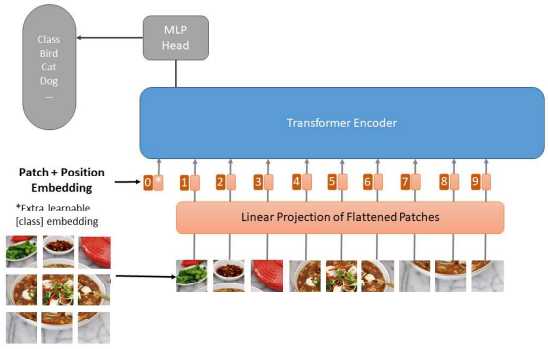

Fig. 1. Vision Transformer

2.2 Quaternion algebra

In the research on applying Quaternions to depth prediction in images [4], the study discussed the representation of Quaternions and basic operations such as addition and multiplication between two Quaternions. Specifically, as outlined below.

$$q = q_0 + \overrightarrow{\mathbf{v}} = q_0 + q_1 i + q_2 j + q_3 k \tag{1}$$

Addition operation of two Quaternions p and q.

$$\begin{aligned} p + q &= (p_0 + q_0) + \left(\overrightarrow{\mathbf{v}_p} + \overrightarrow{\mathbf{v}_q}\right) \\ &= (p_0 + q_0) + (p_1 + q_1)i + (p_2 + q_2)j + (p_3 + q_3)k \end{aligned} \tag{2}$$

Multiplication operation of two Quaternions p and q.

$$\begin{aligned} pq &= p_0 q_0 - \overrightarrow{\mathbf{v}_p}.\overrightarrow{\mathbf{v}_q} + (p_0 \overrightarrow{\mathbf{v}_p} + q_0 \overrightarrow{\mathbf{v}_q} + \overrightarrow{\mathbf{v}_p} \otimes \overrightarrow{\mathbf{v}_q}) \\ &= (p_0 q_0 - p_1 q_0 - p_2 q_2 - p_3 q_3) \\ &+ (p_0 q_1 + p_1 q_0 + p_2 q_3 - p_3 q_2) i \\ &+ (p_0 q_2 - p_1 q_3 + p_2 q_0 + p_3 q_1) j \\ &+ (p_0 q_3 + p_1 q_2 - p_2 q_1 + p_3 q_0) k \end{aligned} \tag{3}$$

2.3 Quaternion-valued Convolutional Neural Networks

A neural network utilizing quaternion algebra number representation can be constructed by employing quaternion operations (Section 2.2). In a neural network using simple quaternion numbers with weights $W = W_0 + \overrightarrow{\mathbf{v}_W}, \overrightarrow{\mathbf{v}_W} = (W_1, W_2, W_3)$, and input $x = x_0 + \overrightarrow{\mathbf{v}_x}$, $\overrightarrow{\mathbf{v}_x} = (x_1, x_2, x_3)$ the forward multiplication is performed using quaternion multiplication as follows:

$$Wx = W_0 x_0 - \overrightarrow{\mathbf{v}_W} \cdot \overrightarrow{\mathbf{v}_x} + \left(W_0 \overrightarrow{\mathbf{v}_x} + x_0 \overrightarrow{\mathbf{v}_x} + \mathbf{v}_W \times \overrightarrow{\mathbf{v}_x}\right) \tag{4}$$

 ISBN: 978-604-80-9774-5

Although current machine learning platforms do not support the use of quaternion number representations, operations on quaternions can still be expressed using real numbers. Specifically, this can be constructed as follows:

$$\begin{bmatrix} \mathcal{R}[\mathbf{W} * \mathbf{x}] \\ I[\mathbf{W} * \mathbf{x}] \\ \mathcal{J}[\mathbf{W} * \mathbf{x}] \\ \mathcal{K}[\mathbf{W} * \mathbf{x}] \end{bmatrix} = \begin{bmatrix} W_0 & -W_3 & -W_1 & -W_2 \\ -W_3 & W_0 & W_2 & -W_1 \\ W_1 & -W_2 & W_0 & -W_3 \\ W_2 & W_1 & W_3 & W_0 \end{bmatrix} * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{5}$$

2.4 Residual Network

Residual Network (ResNet) is a unique type of neural network introduced in 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their paper "Deep Residual Learning for Image Recognition"[1]. ResNet models have shown remarkable success, securing the top position in the ILSVRC 2015 classification competition with an impressive top-5 error rate of 3.57%. To overcome the challenges of vanishing/exploding gradients during the training of deep neural networks, ResNet introduced an innovative architecture known as the Residual Network. This architecture incorporates the concept of a Residual Block, using skip connections to link the activations of consecutive layers by bypassing certain intermediate layers. This technique allows the network to learn residual mappings instead of fundamental mappings, facilitating the training of deep neural networks without encountering the issues associated with vanishing/exploding gradients.

2.5 Quaternion Weight initialization

In the research paper "Quaternion Convolutional Neural Networks for Depth Estimation" [4], a weight initialization method in the Quaternion domain was developed, inspired by the methods proposed by He [2] and Glorot [3].

The weight Glorot and He weight initialization for Quaternion equations are followed as:

$$\begin{aligned} \mathbf{D}(W) &= \frac{2}{n_{in} + n_{out}} = 4\sigma^2_{Glorot_{Quaternion}} \\ \Rightarrow \sigma_{Glorot_{Quaternion}} &= \frac{1}{\sqrt{2(n_{in} + n_{out})}} \end{aligned} \tag{6}$$

$$\begin{aligned} \mathbf{D}(W) &= \frac{2}{n_{in}} = 4\sigma^2_{He_{Quaternion}} \\ \Rightarrow \sigma_{He_{Quaternion}} &= \frac{1}{\sqrt{2n_{in}}} \end{aligned} \tag{7}$$

3 Methodology

3.1 Quaternion Batch Normlization

Normalization is an essential component in the process of training neural networks, bringing several benefits such as normalizing the activations of hidden layers to expedite the training process, and smoothing the loss function, thereby aiding in optimizing the trained network model. In 2015, the authors Sergey Ioffe and Christian Szegedy proposed a normalization solution called 'Batch Normalization [10],' which has been applied in subsequent convolutional neural networks. The implementation formula is as follows:

$$x = \frac{(x - E[x])}{\sqrt{Var[x] + \epsilon}} * \gamma + \beta \tag{8}$$

In this equation, where x represents the input data, $Var[x] = \frac{m}{m-1}.E_B[{\sigma_B}^2]$ (with m being mini-batches and ${\sigma_B}^2$ being their sample variance) is the estimated variance, $\epsilon = 1e-5$ is the epsilon parameter designed to prevent division errors when $Var[x] = 0$, which would result in division by zero. γ and β are learnable parameter vectors with a size of C (where C is the size of the input data). However, in default cases, we often set the parameters of γ to 1 and the parameters of β to 0 for simplicity. Therefore, the formula is as follows:

$$x = \frac{(x - E[x])}{\sqrt{Var[x] + \epsilon}} \tag{9}$$

However, the normalization method mentioned above is suitable only for processing in the real-number domain and platforms that support Quaternion. For the Quaternion domain, which involves representations from various Quaternion spaces, this method is not applicable. Therefore, we need to adjust some operations of the method to make it usable for Quaternions on platforms that do not support it.

Firstly, we set the parameter $\epsilon = 1e-5$ and two vectors with parameters $\gamma_{(r,i,j,k)}$ as 1 and $\beta_{(r,i,j,k)}$ as 0, where (r, i, j, k) represents the Quaternion domains (r for the real domain, and i, j, k for the three imaginary domains). This is done to simplify the formula setup for the method. Then, we consider an input Q in the form of a Quaternion:

$$Q = Q_r + Q_i + Q_j + Q_k \tag{10}$$

We will convert that to a vector including the Quaternion elements.

$$Q = [Q_r Q_i Q_j Q_k]^T \tag{11}$$

The elements can be set as follows:

 ISBN: 978-604-80-9774-5

$$Q - E\left[Q\right] = \begin{bmatrix} Q_r - E\left[Q_r\right] \\ Q_i - E\left[Q_i\right] \\ Q_j - E\left[Q_j\right] \\ Q_k - E\left[Q_k\right] \end{bmatrix} \tag{12}$$

The variance is set up as follows:

$$Var\left[Q\right] = \ E\left[\left|Q\right|^2\right] - E\left[Q\right]^2 \tag{13}$$

However, We know $E[Q]^2 = 0$, so the variance has the following formula:

$$Var\left[Q\right] = \ E\left[\left|Q\right|^2\right] \tag{14}$$

$$Var\left[Q\right] = \ E\left[\left|Q\right|^2\right] = \int_{-\infty}^{+\infty} x^2 f\left(x\right) dx \tag{15}$$

However, to simplify the problem, we perform the modulus operation on the input Q beforehand. The formula is as follows:

$$\left|Q\right| = \ \sqrt{Q_r^2 + Q_i^2 + Q_j^2 + Q_k^2} \tag{16}$$

From 15 and 16 we have:

$$\begin{aligned} Var[Q] &= E[|Q^2|] \\ &= E[Q_r^2 + Q_i^2 + Q_j^2 + Q_k^2] \\ &= E[Q_r^2] + E[Q_i^2] + E[Q_j^2] + E[Q_k^2] \end{aligned} \tag{17}$$

Considering formulas 12 and 17, we can establish the normalization formula for the Quaternion domain as follows:

$$\hat{Q} = \frac{\begin{bmatrix} Q_r - E\left[Q_r\right] \\ Q_i - E\left[Q_i\right] \\ Q_j - E\left[Q_j\right] \\ Q_k - E\left[Q_k\right] \end{bmatrix}}{\sqrt{E\left[Q_r^2\right] + \ E\left[Q_i^2\right] + \ E\left[Q_j^2\right] + \ E\left[Q_k^2\right] + \ \epsilon}} \tag{18}$$

Afterward, we need to modify formula 17 to adapt it for handling batch sizes larger than 1 (where the number of input parameters is m Quaternion inputs with m $>$ 1). We have:

$$\hat{Q} = \frac{\left(Q_r - E\left[Q_r\right]\right) + \left(Q_i - E\left[Q_i\right]\right) + \left(Q_j - E\left[Q_j\right]\right) + \left(Q_k - E\left[Q_k\right]\right)}{\sqrt{\frac{m}{m-1} \cdot \left(E\left[Q_r^2\right] + \ E\left[Q_i^2\right] + \ E\left[Q_j^2\right] + \ E\left[Q_k^2\right]\right) + \ \epsilon}} \tag{19}$$

Combining formula above with the two parameters $\gamma_{(r,i,j,k)}$ and $\beta_{(r,i,j,k)}$, we obtain the complete formula as follows:

$$\hat{Q} = \frac{(Q_r - E[Q_r]) + (Q_i - E[Q_i]) + (Q_j - E[Q_j]) + (Q_k - E[Q_k])}{\sqrt{\frac{m}{m-1}.\left(E[Q_r^2] + E[Q_i^2] + E[Q_j^2] + E[Q_k^2]\right) + \epsilon}} . \gamma_{(r,i,j,k)} + \beta_{(r,i,j,k)} \tag{20}$$

From the above formula, we can construct the batch normalization method in the Quaternion number domain.

3.2 Quaternion Fully connected

The fully connected operation is commonly used in neural networks. Its formula represents a function that reshapes the input to the desired size:

$$\omega\left(\overrightarrow{V_x}\right) = W_x \cdot {V_x}^T + b_x \tag{21}$$

In which, $\omega\left(\overrightarrow{V_x}\right)$ is a linear function transforming the input data V_x, W_x is the weight that transforms V_x into a different size, and b_x is the bias parameter. The Quaternion version simply applies the multiplication operation established in section 2.3 to formula 21

3.3 Proposed Methods

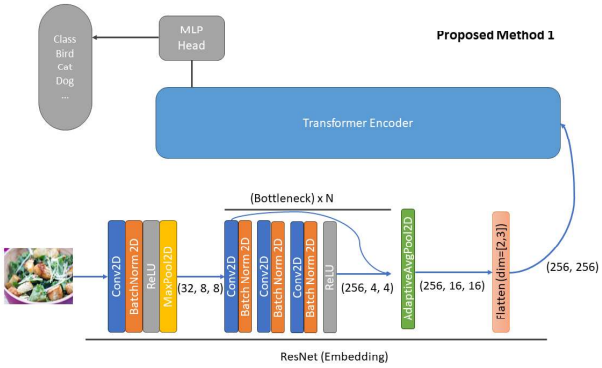

Fig. 2. Illustration of Proposed Method 1. The original image is divided into B-patches, which are then encoded into token codes. These token codes are fed into a modified ResNet block to encode the input image. The generated tokens are input into the Transformer Encoder to learn relationships between them.

Proposed Method 1: The proposed method is illustrated in Fig.2 and detailed as follows: Initially, the research team will proceed to eliminate the Patch

 ISBN: 978-604-80-9774-5

Embedding and Position Embedding components within the Vision Transformer network.

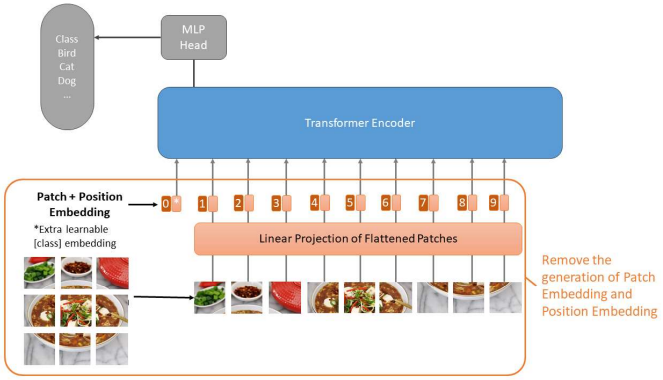

Fig. 3. The process of replacing the Patch + Position Embedding in the model.

Subsequently, construct a CNN network resembling ResNet and fine-tune certain aspects to align with the input requirements of the Transformer, as depicted in Fig.4.

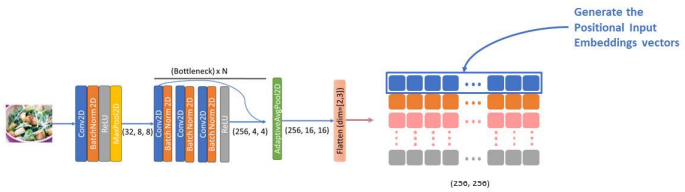

Fig. 4. Construction of a ResNet neural network to generate Positional Input Embeddings. These embeddings contain both positional information and embedded data.

Once a network model capable of generating embedding vectors containing position information is constructed, simulating the operation depicted as the removed component (replacement) in Fig. 3, we can use this network to generate Positional Input Embeddings. These embeddings perform the transformation of the input data into a sequence of token embeddings, which are then fed into the Transformer Encoder.

Proposed Method 2: The second method is constructed similarly to the first method (section 3.3.1); however, there are some changes described in Fig.5, as detailed below:

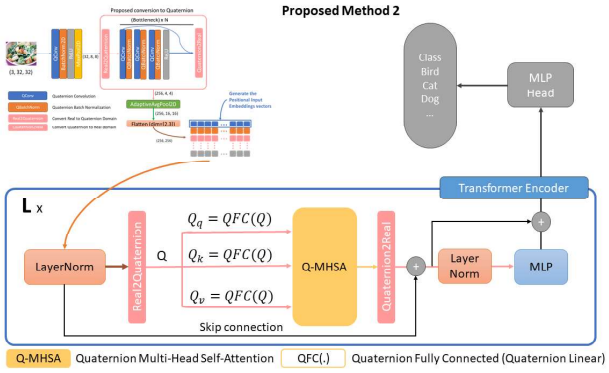

Fig. 5. Illustration of Proposed Method 2. The method involves changing the components of the layers to Quaternion. Note that the Quaternion layers reduce the filter size by a factor of 4 to ensure a fair speed comparison.

Firstly, modify the layers in the Bottleneck block to transition to the Quaternion domain, with components described and constructed as follows: "Quaternion Domain Convolutional Layer - Quaternion Convolution" and "Quaternion Domain Normalization Layer - Quaternion Batch Normalization". Additionally, two additional components, "Real2Quaternion" and "Quaternion2Real", are introduced to transform data between the real domain and the hypercomplex Quaternion domain.

The conversion formula from the real number domain to the Quaternion number domain can be constructed as follows:

$$R2Q\,(X) = \begin{cases} Q_r = X_{4.i,\ *} \\ Q_i = X_{4.i+1,*} \\ Q_j = X_{4.i+2,\ *} \\ Q_k = X_{4.i+3,*} \end{cases},\ i\ \in \{0, 1, 2, \ldots\} \tag{22}$$

In which, X is the input data, R2Q is the function transforming data from the real number domain to the hypercomplex Quaternion number domain, and $*$ symbolizes the remaining dimensions of the data. Here, i is considered as the index variable for the Quaternion set at the $i-th$ position.

Secondly, modify the layers of the Multi-Head Self-Attention component [5] in the Transformer Encoder to a version that operates in the Quaternion domain. We can refer to it as Quaternion Multi-Head Self-Attention (Q-MHSA).

In Q-MHSA, the research team first employs a set of three layers QFC(.) to transform the segment q into quaternion query Q_q, quaternion key Q_k, and

 ISBN: 978-604-80-9774-5

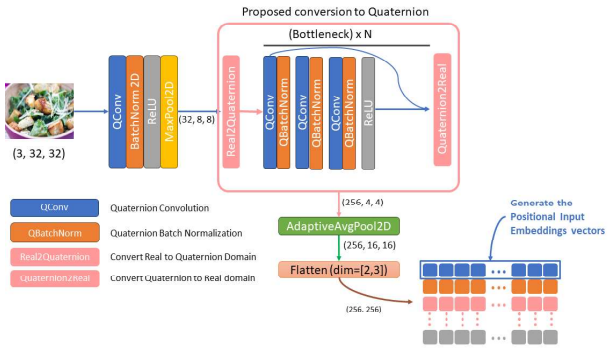

Fig. 6. Modification of the bottleneck block components to layers that operate in the Quaternion domain.

quaternion value Q_v as described in formula (23) below.

$$Q_q = QFC\left(q\right), \quad Q_k = QFC\left(q\right), \; Q_v = QFC\left(q\right) \tag{23}$$

$$head_i = \sigma((Q_q \otimes \; Q_k^T) \; / \sqrt{d}) \otimes , \; Q_v \tag{24}$$

$$MultiHead = QFC(Concat(head_1 \ldots head_i)) \tag{25}$$

In which, $\sigma\left(\overrightarrow{z}\right) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$ is the Softmax function, $\otimes$ denotes the vector multiplication, d is the dimension of the head, and $Concat(.)$ is the function concatenating the individual $head_i$.

4 Experimental Results

4.1 Experimental Setup

In this study, the effectiveness of the proposed method is assessed using the accuracy evaluation method. The models are implemented on the PyTorch platform, utilizing the Adam optimizer (lr = 1e-3), Cross-Entropy as the loss function (26), and accuracy evaluation as the experimental evaluation method (27).

$$Loss_{CE}\left(\theta\right) = -\sum_{i=1}^{m} t_i log\left(p_i\right) \tag{26}$$

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \tag{27}$$

The aforementioned models are trained on an Nvidia Tesla V100 GPU (Google Colab Pro) with 16GB of memory. To demonstrate their impact, the research team utilizes various datasets, including CIFAR10 [15], CIFAR100 [15], GTSRB [16], and BIRDS 525 SPECIES [17] for evaluation purposes.

4.2 Experimental Results

After the training and testing processes on various datasets such as CIFAR10, CIFAR100, and GTSRB, the results are aggregated in Tables 1 and 2, demonstrating the effectiveness of the two proposed methods by our research team.

Table 1. Predicted Results on Training Dataset

	Params	**CIFAR10**	**CIFAR100**	**GTSRB**
ViT	1.52M	70.5%	70.5%	98.6%
ViT + ResNet	1.69M	**92.3%**	**90.6%**	**100%**
ViT + QResNet + Q-MHSA	**0.53M**	85.1%	75.2%	100%

Table 2. Predicted Results on Testing Dataset

	Params	**CIFAR10**	**CIFAR100**	**GTSRB**	**Bird525**
ViT	1.52M	59.27%	32.35%	87.23%	65.62%
ViT + ResNet	1.69M	68.62%	38.70%	89.94%	75.03%
ViT + QResNet + Q-MHSA	**0.53M**	**69.66%**	**39.60%**	**91.62%**	**76.12%**

The results from the prediction table on the training dataset (Table 1) indicate that the **ViT + ResNet** combination method yields the best prediction results across all three training datasets, followed by the **ViT + QResNet + Q-MHSA** model. However, upon examining Table 2, it is evident that the **ViT + QResNet + Q-MHSA**'method performs even better, demonstrating the potential of this approach to achieve accurate results in subsequent training sessions.

5 Conclusion and Future Works

In conclusion, the results have demonstrated the influence of Quaternion components and the potential of combining the Convolutional Neural Network (CNN) with the Vision Transformer model by replacing its embedding components. However, this is just one approach to constructing two different solutions aimed at improving the accuracy of the Vision Transformer with a small parameter count. These solutions have not been tested on larger datasets with higher resolution and strong correlations. Therefore, in future research, our team aims to experiment with these solutions on larger datasets, using a larger parameter count, possibly with the ViT-Large model, to comprehensively demonstrate the impact of the proposed solutions.

 ISBN: 978-604-80-9774-5

References

1. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.
2. X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256.
3. K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” 2015, cite arxiv:1502.01852. . .
4. A. H. Nguyen, C. D. Hoang, D. H. P. Phan and M. T. Pham, "Quaternion Convolutional Neural Networks for Depth Estimation," 2022 Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCIS&ISIS), Ise, Japan, 2022, pp. 1-6, doi: 10.1109/SCISISIS55246.2022.10002074
5. A. Vaswani et al., “Attention Is All You Need”. 2017, cite arxiv:1706.03762
6. K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks”. 2015, cite arxiv:1511.08458
7. Dosovitskiy et al., “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” CoRR, vol. abs/2010.11929, 2020, [Online]. Available: https://arxiv.org/abs/2010.11929
8. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” CoRR, vol. abs/1810.04805, 2018, [Online]. Available: http://arxiv.org/abs/1810.04805
9. Z. Peng, “Multilayer Perceptron Algebra,” arXiv e-prints, p. earXiv:1701.04968, Jan. 2017, doi: 10.48550/arXiv.1701.04968.
10. Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. CoRR, abs/1502.03167. http://arxiv.org/abs/1502.03167
11. Chen, X., Hsieh, C.-J., & Gong, B. (2021). When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations. CoRR, abs/2106.01548. https://arxiv.org/abs/2106.01548
12. Foret, P., Kleiner, A., Mobahi, H., & Neyshabur, B. (2020). Sharpness-Aware Minimization for Efficiently Improving Generalization. CoRR, abs/2010.01412. https://arxiv.org/abs/2010.01412
13. S. Lin, M. Bai, F. Liu, L. Shen and Y. Zhou, "Orthogonalization-Guided Feature Fusion Network for Multimodal 2D+3D Facial Expression Recognition," in IEEE Transactions on Multimedia, vol. 23, pp. 1581-1591, 2021, doi: 10.1109/TMM.2020.3001497.
14. Zhou, Yu et al. “Quaternion Orthogonal Transformer for Facial Expression Recognition in the Wild.” ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2023): 1-5.
15. A. Krizhevsky, “Learning multiple layers of features from tiny images,” techreport, 2009.
16. J. Stallkamp, M. Schlipsing, J. Salmen and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," The 2011 International Joint Conference on Neural Networks, San Jose, CA, USA, 2011, pp. 1453-1460, doi: 10.1109/IJCNN.2011.6033395.
17. Gerry, “Birds 525 species- image classification,” https://www.kaggle.com/datasets/gpiosenka/100-bird-species, 2020.