Coordinates to Color Quaternion Neural Network (CoCoQNN) and Coordinates to Color Quaternion Convolutional Neural Network (CoCoQCNN) for Image Processing

Guillermo Altamirano-Escobedo[a] and **Eduardo Bayro-Corrochano**[b]
[a]Department of Electrical Engineering and Computer Science
CINVESTAV, Guadalajara, México
guillermo.altamirano@cinvestav.mx

[b]Institute of Automatic Control and Robotics
Poznan University of Technology, Poznan, Poland
edubayy@gmail.com

Abstract

In this work, we propose three architectures for mapping pixel coordinates in an image to their corresponding RGB (or grayscale) values. These three architectures are based on the CocoNet (coordinates-to-color network) model: the Coordinates to Color Quaternion Neural Network (CoCoQNN), the Coordinates to Color Convolutional Neural Network (CoCoCNN) and Coordinates to Color Quaternion Convolutional Neural Network (CoCoQCNN). The first model, is a modified CocoNet model using quaternion fully connected layers whereas the last model incorporates quaternion convolutional layers. The proposed quaternion-valued architectures have the advantage of requiring only 25% of trainable parameters when compared to their real-valued counterparts. During the training process, these architectures learn to encode the input image within their layers. At test time, when providing normalized coordinates as input, these architectures will output the approximate RGB (or grayscale) values reconstructing the entire learned image. We conducted experiments using images from the CIFAR10, Set5 and the UCSD retinal OCT datasets, in order to test the proposed models, showing a competitive performance when compared to the baseline CocoNet architecture.

CoCoQNN

We propose a quaternion neural network that learns a mapping function f from the pixel coordinates in an image to the corresponding RGB (or grayscale) values. The input layer consists of a single quaternion neuron which is constructed from two pairs of Cartesian coordinates (x_1, y_1) and (x_2, y_2). The origin of one set of Cartesian coordinates is situated in the bottom-right corner of the image, diagonally opposite from the corner (top-left) that serves as the origin for the other set of Cartesian coordinates (see Fig. 1). We are using normalized coordinates in the interval $[0, 1]$ and the pixel values are normalized within the range $[0, 1]$ (normalized pixel value = pixel value/255). For and RGB image, the mapping function f learned by the proposed quaternion neural network is defined as follows:

$$f : \mathbb{H} \longrightarrow \mathbb{H}, \tag{1}$$

that is, for an input quaternion $q = x_1 + y_1\mathbf{i} + x_2\mathbf{j} + y_2\mathbf{k}$ we have its corresponding output $\mathrm{GR} + \mathrm{R}\mathbf{i} + \mathrm{G}\mathbf{j} + \mathrm{B}\mathbf{k}$, where GR, R, G and B denote the grayscale, red, green, and blue

values of the pixel, respectively. For a grayscale image, the map is given by:

$$f : \mathbb{H} \longrightarrow [0, 1]. \tag{2}$$

CoCoQCNN

We proposed a modification of the previous architecture by using quaternion convolutional layers [6] instead of quaternion fully-connected layers. In this new scenario we use four input channels constructed from the x_1, y_1, x_2 and y_2 coordinates (see Fig. 2) obtained from two cartesian systems.

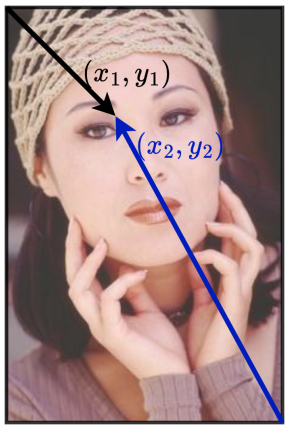


Figure 1: Sample image from the Set5 [4] dataset where the coordinates are obtained from two cartesian systems.

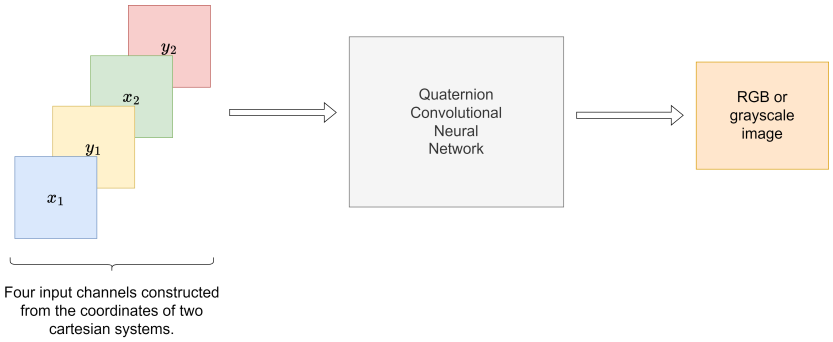


Figure 2: General architecture of the CoCoQCNN: Four input channels are constructed from the coordinates of two cartesian systems. Then the channels are processed by means of a QCNN.

This CoCoQCNN aims to learn the following mapping function g:

$$g : \mathbb{H}^{m \times n} \longrightarrow \mathbb{R}^{m \times n \times 3}. \tag{3}$$

for an RGB image and the following mapping for a grayscale image:

$$g : \mathbb{H}^{m \times n} \longrightarrow \mathbb{R}^{m \times n}. \tag{4}$$

The coordinates can be mapped to a grayscale and an RGB image at the same time with the following mapping:

$$g : \mathbb{H}^{m \times n} \longrightarrow \mathbb{H}^{m \times n}. \tag{5}$$

where the output quaternion image can be constructed as follows: $\mathrm{GR}_{\mathrm{ch}} + \mathrm{R}_{\mathrm{ch}}\mathbf{i} + \mathrm{G}_{\mathrm{ch}}\mathbf{j} + \mathrm{B}_{\mathrm{ch}}\mathbf{k}$, where $\mathrm{GR}_{\mathrm{ch}}$, R_{ch}, G_{ch} and B_{ch} denote the grayscale image, red channel, green channel and blue channel, respectively..

Experimental Results

We carried out experiments focusing on three main tasks: image reconstruction, image upsampling and image denoising. Additionally, for all the experiments in this work, we used the mean squared error (MSE) loss function.

CIFAR10: Reconstruction and Upsampling

The reconstruction and upsampling capabilities of the CoCoQNN were tested using an image of size 32×32 from the CIFAR10 dataset [3]. The architecture used for this task consists of 6 quaternion fully-connected layers having 64 quaternion neurons with the exception of the last layer, that has only 1 quaternion neuron.
We trained the CocoNet and CoCoQNN with learning rate equal to 5×10^{-4} and 10 000 epochs with a batch size equal to 256. Additionally, we utilized the Adam Optimization algorithm.
All layers have tanh as activation function with the exception of the last layer that has the sigmoid activation function. From Figs 4 and 5, we can observe that the learned (reconstructed) image in both CoCoQNN and its real-valued counterpart keep the high level features of the original image. Regarding upsampling, we tested an upscale factor equal to 20, see Figs. 6 and 7. Apart from upscaling the image, both models exhibit image smoothing.

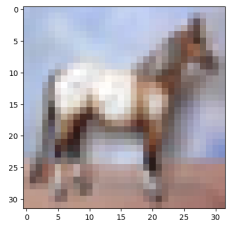


Figure 3: Original sample image from CIFAR10 dataset.

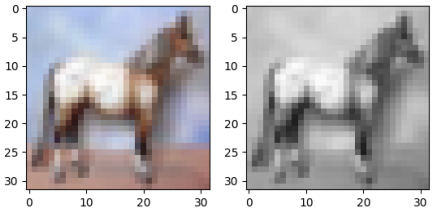


Figure 4: Reconstructed/learned image using CoCoQNN with MSE loss equal to 2.9385×10^{-5} and 67 332 trainable parameters.

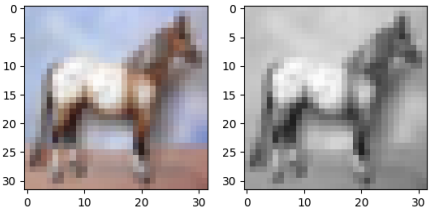


Figure 5: Reconstructed/learned image using CocoNet with MSE loss equal to 8.0297×10^{-6} and 265 476 trainable parameters.

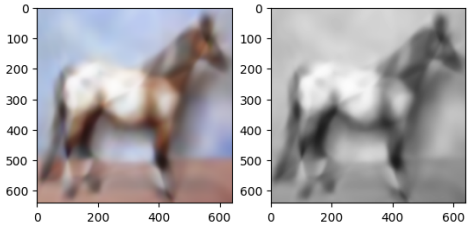


Figure 6: Upsampled image using the CoCoQNN model.

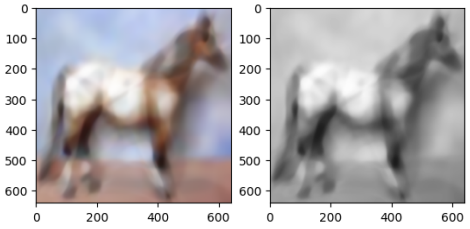


Figure 7: Upsampled image using the CocoNet model.

Set5: Reconstruction and Upsampling

In order to provide a metric for upsampling, we use an image from the Set5 dataset [4]. First we downscale the image by a factor of 3. Then, the downscaled image is learned by the model. Finally, the image upscaled to its original size (upscaling factor of 3) is obtained from the models by using its corresponding normalized coordinates as input, see Figs. 12 and 13. The architectures of the CocoNet and CoCoQNN consist of 10 hidden layers using the ReLU activation function, with 200 real-valued and 100

quaternion-valued output neurons, respectively. We used the same learning rate, batch size and optimization algorithm as with CIFAR10. The number of epochs during training was set equal to 2000.

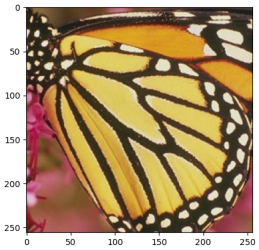


Figure 8: Sample image from the Set5 dataset.

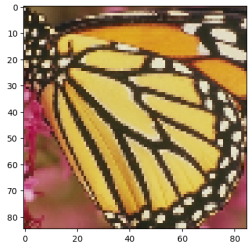


Figure 9: Downsampled image by a factor of 3.

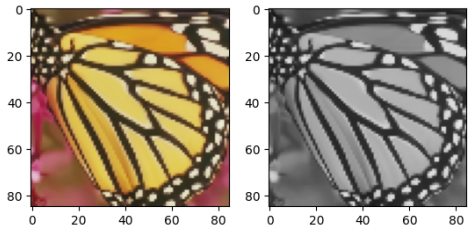


Figure 10: Learned (reconstructed) image by the CocoNet model with MSE equal to 1.1590×10^{-3} and 363 604 trainable parameters.

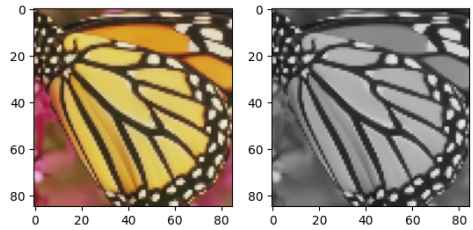


Figure 11: Learned (reconstructed) image by the CoCoQNN model with MSE equal to 9.8506×10^{-4} and 364 804 trainable parameters.

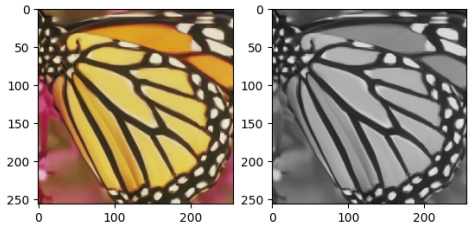


Figure 12: Learned image upsampled to its original size by the CocoNet model with MSE equal to 5.5591×10^{-3}.

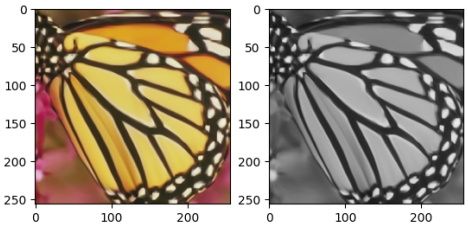


Figure 13: Learned image upsampled to its original size by the CoCoQNN model with MSE equal to 5.0564×10^{-3}.

Retinal OCT Denoising

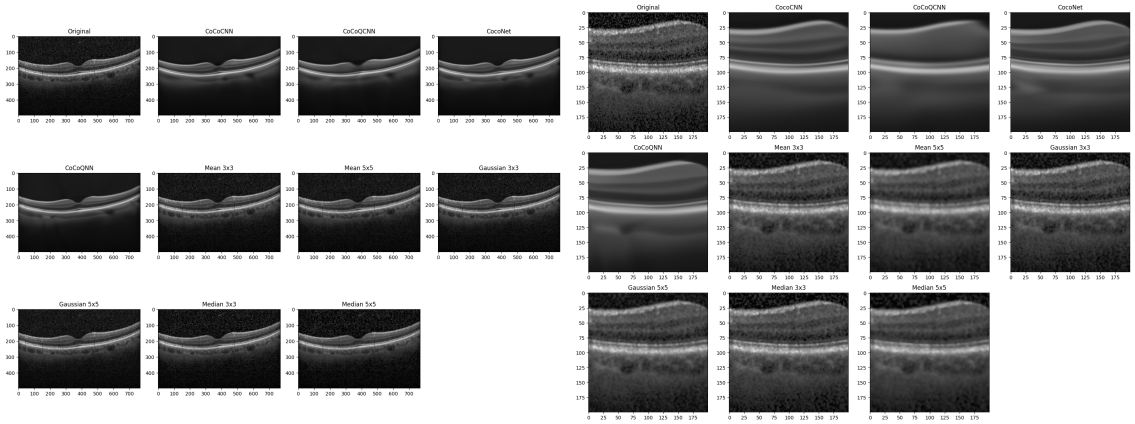
The analysis of OCT images and their diagnostic effectiveness is impacted by the presence of speckle noise [2]. We test the proposed CoCoQNN, CoCoCNN and CoCoQCNN using the UCSD retinal OCT dataset [5] for the denoising task. We present the denoising results for an OCT image and a random patch in Fig. 14 compared with denoising by applying mean, median and Gaussian filters of size 3×3 and 5×5.

The CocoNet and CoCoQNN consist of 10 hidden layers with 200 real-valued and 50 quaternion-valued output units, respectively. The number of trainable parameters of the CocoNet and CoCoQNN is 363 001 and 92 401, respectively. For both models the last layer is a fully connected layer with one output unit, used to provide the grayscale value of the pixel. We trained both models a with learning rate equal to 5×10^{-4} and 100 epochs with a batch size equal to 1024. Additionally, we utilized the AdamW optimization algorithm. At the end of the training the obtained MSE losses for the CocoNet and CoCoQNN were 5.5256×10^{-3} and 5.8363×10^{-3}, respectively.
The architecture for the proposed convolutional models are as follows:

- **CoCoCNN**: Six convolutional layers. The first and last convolutional layers had 64 and 1 filter, respectively. The rest of the layers use 64 filters. The activation function for the first five convolutional layers is the tanh. The activation function for the final convolutional layer is the sigmoid. The number of trainable parameters for this model is equal to 417 921.

- **CoCoQCNN**: Same architecture replacing convolutional layers by quaternion convolutional layers with the exception of the last layer. The first and last convolutional layers had 16 quaternion-valued and 1 real-valued filter, respectively. The number of trainable parameters for this model is equal to 105 921.

We trained both models a with learning rate equal to 1×10^{-4} and 5000 epochs with a batch size equal to 1. Additionally, we utilized the AdamW optimization algorithm. At the end of the training the obtained MSE losses for the CoCoCNN and CoCoCQNN were 5.3815×10^{-3} and 5.6179×10^{-3}, respectively.



(a) (b)

Figure 14: Comparison of denoising results for (a) the complete image and (b) a random patch: original image, CoCoCNN, CoCoQCNN, CocoNet, CoCoQNN, mean 3×3, mean 5×5, Gaussian 3×3, Gaussian 5×5, median 3×3 and median 5×5. The proposed models remove the Speckle noise effectively and show defined borders. Filtering results with mean, median and Gaussian filters do not remove all the speckle noise.

Conclusions

The proposed models are constructed from the CocoNet architecture by replacing the fully connected layers by quaternion fully connected layers, convolutional layers and

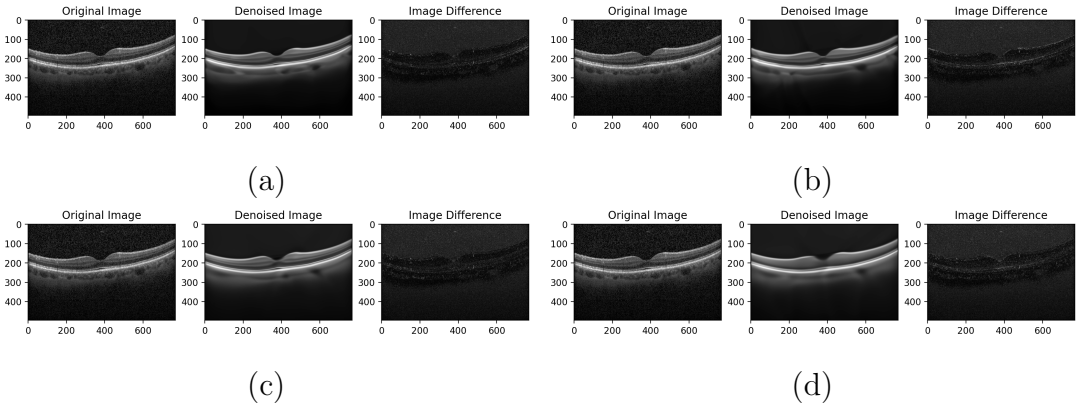


Figure 15: Comparison of the difference between the original image and denoised image (a) CocoNet (b) CoCoQNN (c) CoCoCNN and (d) CoCoQCNN.

quaternion convolutional layers resulting in the CoCoQNN, CoCoCNN and CoCoQCNN, respectively. We conducted a series of experiments including reconstruction, upsampling and denoising of images in order to test the CoCoQNN, CoCoCNN and CoCoQCNN models.

The proposed CoCoCNN showed a competitive performance when compared to the Coconet model in the OCT denoising task. Additionally, the proposed quaternion-valued architectures have the advantage of requiring only 25% of trainable parameters when compared to their real-valued counterparts while having a competitive performance.

The proposed models show some advantages regarding denoising of OCT images (effective Speckle noise removal and defined borders). Future work will involve classification and segmentation of OCT images applying denoising with the proposed models as a pre-processing step.

References

[1] Bricman, P. A., & Ionescu, R. T. (2018). CocoNet: A deep neural network for mapping pixel coordinates to color values. In Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part II 25 (pp. 64-76). Springer International Publishing.

[2] Muxingzi Li, Ramzi Idoughi, Biswarup Choudhury, and Wolfgang Heidrich, "Statistical model for OCT image denoising," Biomed. Opt. Express 8, 3903-3917 (2017).

[3] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[4] Bevilacqua, M., Roumy, A., Guillemot, C., & Alberi-Morel, M. L. (2012). Low-complexity single-image super-resolution based on nonnegative neighbor embedding.

[5] Kermany, Daniel S., et al. "Identifying medical diagnoses and treatable diseases by image-based deep learning." cell 172.5 (2018): 1122-1131.

[6] Parcollet, T., Morchid, M., & Linarès, G. (2020). A survey of quaternion neural networks. Artificial Intelligence Review, 53(4), 2957-2982.